Protection Profile for USB Flash Drives

Mitigating the Risk of a Manipulated, Misplaced, or Stolen USB Flash Drive

Information Assurance Directorate

01 December 2011

Version 1.0

Table of Contents

List of Tables

List of Figures

Revision History

Version	Date	Description
1.0	01 December 2011	Initial release

1 Introduction to the PP

1.1 PP Overview of the TOE

1 This is the first generation of a Standard Protection Profile (PP) for USB flash drives. The two usage scenarios for USB Flash Drives addressed by this PP are 1) the use by an organization to transfer information between two devices, and 2) the use to store files for a device or set of devices for a single user. This PP addresses the primary threats that an adversary will obtain a misplaced or stolen USB flash drive and extract the sensitive data or will attempt to place malicious system files on the device that can be used to compromise host environments. The Target of Evaluation (TOE) defined in this Protection Profile (PP) is a USB flash drive and any associated software used to access the data on that drive and manage the drive.

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

2 USB flash drives that conform to this PP must encrypt the user data stored on the device with a processor on the device. The TOE includes software provided by the vendor to enable the user to interact with the USB flash drive. This software will reside on the host device and potentially on the USB flash drive itself, and the amount of interaction between the two environments will vary based on the implementation. All cryptographic operations (encryption and decryption of user data stored on the device, formation of Key Encryption Key (KEK) used to mask the Data Encryption Key (DEK), the operations used to mask the DEK) must be performed by a processor on the USB Flash Drive and not on the host.

3 The data that are to be stored by the USB flash drive is encrypted using a DEK. The DEK is masked using a key encryption key (KEK). The KEK can be derived from multiple components (referred to as submasks, which are derived from authorization factors) or obtained from a single submask. The foremost security objective of USB flash drive 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 USB flash drive does not carelessly store an authorization factor with the USB flash drive.

4 The device contains a protected area for system files. Known attacks using USB flash drives as an attack vector depend on replacing system files on the USB device that are automatically run when the USB flash drive is inserted into a host. USB flash drives conformant to this PP are required to only install updates to system files on the USB flash drives that can be checked through means of a digital signature to ensure that those updates came from an authorized source. Further, conformant devices will not replace any system file without first checking a digital signature. Users are provided an interface that can be used to determine the version of the product on the USB flash drive that can aid in ensuring the most up-to-date firmware is used.

5 The vendor is required to provide configuration guidance (AGD_PRE, AGD_OPR) to correctly install and administer the TOE for every operational environment supported (for example, for every O/S supported by the product) and for the two use cases of local file storage and file transfer.

1.1.2 Authorization

6 After a USB flash drive has been initialized and encryption has been enabled, one or more authorization factors must be established before data can be transferred from a host to the USB flash drive. This authorization factor(s) must be presented to the USB flash drive in order for the user to request that the USB flash drive decrypt and transfer the data to the user's host. Authorization factors are not required

to be unique to individual users. In other words, authorization factors are only required to establish that the possessor is in the community of users authorized to request data stored on the USB flash drive. Each authorization factor is used to derive a submask; the submasks are used to derive the KEK.

7 All compliant TOEs must support a password authorization factor. This is conditioned to produce a submask. Compliant TOEs also can optionally support:

- A randomly-generated bit string stored on the host called a "host split authorization factor". In this case, the authorization factor also serves directly as the submask.
- A randomly-generated bit string in tamper protected storage on the USB flash drive, which is only released if a user provides the correct PIN. In this case, the PIN is the authorization factor and the randomly-generated bit string is the submask. For these authorization factors, the PIN is subject to anti-hammer provisions where if an incorrect PIN is entered a (configurable) number of times consecutively, the device is locked.

The submasks produced from these optional authorization factors are then combined with the password-based submask, and the KEK is then derived from this combination.

The host split authorization factor will usually be supported when the USB flash drive is used as external storage for a particular device. Ideally, the host split authorization factor will be protected by a Trusted Platform Module (TPM) on the host device. Regardless, though, disk encryption should still be used on the host, which would protect the split, its remnants, or the remnants of any additional authorization factors.

8 If the ST author defines additional authorization factors and means by which submasks are produced from them, they must be fully documented and cannot diminish the strength of the submasks produced from the password and/or the other authorization factors. All authorization factors must be conditioned such that they produce submasks that are the same size (bit length) as the DEK they are masking, and must be combined using an XOR function to produce the KEK.

9 To reflect current practice, the vendor shall support the ability to have a password of at least 32 characters that can be generated externally by an administrator. Once the password is entered by the user, it is conditioned by the TSF on the USB Flash Drive prior to being provided as an input for the KEK. Large passphrases (256 characters or more) are preferred. If supported, the requirements in Appendix C for Passphrase Authorization Factors should be included in the ST instead. Future generations of this PP will require support for passphrases 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.

10 If one of the other optional authorization factors is used, then the ST author will include the appropriate material from Appendix C. The host split authorization factor must be generated by a RBG that resides on the USB flash drive. The PIN-protected submask that resides in memory on the USB flash drive must be generated by a RBG located on the USB flash drive. The PIN authorization factor can be set and chosen in the operational environment, but the submask must be generated by a randomizer on the USB flash drive. As noted below, the DEK must be generated by a RBG that resides on the USB flash drive.

1.1.3 Encryption

11 A USB flash drive may not have the encryption enabled when it leaves the manufacturing facility. Once a user or administrator initializes the drive and turns on the encryption, the data on the device must stay encrypted. Further, the initial user of the device must be able to generate the DEK when they initialize the device. Data may not be stored on the device until the encryption is enabled and the DEK is masked with a KEK derived from authorization factors for a user, i.e., not until a user takes ownership of the

device. Although it is desirable, it is not required to be able to re-generate the DEK after the initial user takes ownership and initializes the device.

12 Two keys (both a DEK and a KEK) are required in order to facilitate password changes by the user, since changing the password authorization factor is a required capability. Using the KEK to mask the DEK means that a user can change the password authorization factor (thus deriving a new KEK) without having to re-encrypt the data on the device.

13 If the cryptography used to generate, handle, and protect keys, submasks, or authorization factors is sufficiently robust and if the implementation has no critical mistakes, an adversary who obtains a misplaced or stolen USB flash drive without the authorization factors, submasks, or KEK has to exhaust the key space of the KEK or DEK to obtain the data. Note that the password might offer less strength than exhausting over the potential number of keys for data encryption algorithm (AES). Furthermore, if the password is the only authorization factor unknown to the adversary, then the keyspace is the minimum of the work needed to exhaust AES or to exhaust the number of possible passwords. As a consequence, the next generation of this Protection Profile will require support for passphrases and other more robust authorization factors/submasks.

14 The data transferred to the USB flash drive is encrypted using the DEK. The DEK is masked by the KEK (through an XOR operation or by using AES). The DEK will be generated using a Deterministic Random Bit Generator (DRBG) contained on the USB flash drive and will be either 128 or 256 bits. A properly seeded DRBG ensures a sample of noise at least equal to the key size of the DEK. The entropy used as input to the DRBG algorithm must be provided by at least one hardware-based source.

15 At any time the USB flash drive is extracted from the device (shutdown), the unencrypted keying material (with the exception of the PIN-protected submask protected by the tamper proof module) or user data cannot reside in persistent memory. Persistent memory is defined as memory that holds state after the USB flash drive loses power. There are several scenarios that define a shutdown: when a USB flash drive is suddenly removed from the host computer, when a USB flash drive is removed using the Operating System to safely remove the hardware, when the host computer is turned off (powered down, sleep mode, hibernate mode). Since not all of the cases are controlled, the implications are that 1) key material is erased as soon as it is used, or kept in volatile memory, and 2) user data are never stored on the device in unencrypted form.

1.1.4 Updates to TOE Firmware/Software

16 USB flash drives conformant to the PP will have the capability to cryptographically verify updates to TOE firmware and software (if present) have originated from a trusted source (the product vendor, in most cases) using digital signatures as specified in the requirements. The public key used for verification of the update must be contained on the USB Flash Drive to limit "chain-of-trust" checks necessary in performing the verification of the signature on the update. This certificate is treated as a "system file" and may be updated, but only after being verified by the then-current certificate. In order to support such updates any user will be able to query the USB flash drive for the current version of firmware/software it is using, and to initiate an update to the firmware/software. The USB flash drive will reject any changes for which it cannot validate the signature. The cryptographic operations associated with the checks will be performed by the cryptographic functions on the USB flash drive itself; it will not rely on cryptographic functions that run on the host device.

1.1.5 Administration

17 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 who have specific responsibilities and also have greater "trust" than the general user population.). Typically, administrators possess privilege to invoke functionality on the TOE that is not available to general users. For USB flash drives, however, once the device is put into use there should be no need for administrative involvement; even updates to the TOE should be able to be performed by the (non-administrative) user in possession of the device.

18 However, in the baseline requirements of this PP, the only required authorization factor is password-based. Untrusted users have historically been shown to create easily-guessable passwords with low entropy, even though the requirements of this PP allow for hard-to-guess passwords with over 150-bits of entropy. Organizations wishing to ensure that the protections afforded by this PP are not trivially circumvented should ensure their users understand the importance of choosing good passwords, and have policies in place that users do so. Having passwords created by administrators (and then provided to the users) is an alternative approach, but as most USB flash drives provide the ability for the user to change their authorization factor, unless the users are convinced of the need for strong passwords, they will simply change the administrator-provided password to something they prefer once they obtain possession of the USB flash drive.

19 A solution to trusting all users to choose good authorization factors, while still allowing the user some discretion in the matter, is to have the TOE enforce a password metric. As this requirement is not widely implemented in current USB flash drives, it is an optional requirement contained in Appendix C.

1.1.6 Authorized Users

20 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 unlock the protected USB flash drive. It is the responsibility of the authorized users of the USB flash drive to secure and protect the USB flash drive and authorization factors for the TOE while it is officially in their possession. Authorized users shall not leave/store the password and/or unprotected authorization factors with the USB flash drive or if multiple factors are used, with each other. The user will be provided appropriate guidance to maintain a secure TOE.

1.1.7 Available non-TOE Hardware/Software/Firmware

21 The TOE relies on the operational environment (host device) for providing an interface for the password-based authorization factor; an interface for the trusted update function; and a communications mechanism between the host and the USB device. The vendor is expected to provide sufficient installation and configuration instructions to identify an operational environment with the necessary features and to provide instructions for how to configure it and update it in a correct and secure manner.

22 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. Such guidance is considered operational guidance for the TOE and should be provided as part of the documentation available for end users of the TOE.

2 Security Problem Definition

23 The primary asset that is being protected is the data stored on the USB flash drive. The threat model thus focuses on how data can be inadvertently or intentionally compromised when the USB flash drive is used to store or transfer data.

2.1 Threats

24 A threat consists of a threat agent, an asset and an adverse action of that threat agent on that asset.

25 The primary threat agents are the entities that put the assets at risk if that threat agent steals or otherwise obtains a misplaced USB flash drive. For instance, the chart below contains T.UNAUTHORIZED_ACCESS. The threat agent is the possessor (unauthorized user) of a misplaced or stolen USB flash drive. The asset is the user data. The adverse action is to attempt to obtain user data from the USB flash drive. This threat drives the first functional requirements for the USB flash drive (TOE) and puts the user in possession of a key or key split that is needed to decrypt the data. Since possession of the KEK, DEK, authorization factors, submasks, and random numbers or any other values that contribute to the creation of keys or authorization factors could allow an unauthorized user to defeat the encryption, keying material is considered equivalent to the user data in importance and is the other asset addressed in the threat table.

26 For this PP, the TOE is not expected to defend against all threats related to malicious software that may reside in user data files on the USB flash drive. For instance, the TOE is not responsible for detecting malware in the data selected by the user for encryption and transport (that is a responsibility of the host environment). Once the USB flash drive is operational in a host system, the threats against the data from potentially malicious software on the host are also not in the threat model of this PP. For example, there are no requirements in this PP that address a malicious host capturing (either through malicious action or failure to erase the data after use) the password-based authorization factor.

27 When the USB flash drive is used as external storage, full disk encryption should still be used on the host or hosts, even if the USB flash drive is being used as the sole storage repository for those data. One risk is that temporary copies are made of files that are examined or modified when the USB flash drive is connected to the host. Those remnants may persist on an unpowered host (for instance, in the operating system page file) and would be subject to disclosure if not encrypted or erased. A second risk is that the user's authentication factors might be inadvertently swapped to persistent memory when the user provides them to the USB flash drive by way of the host.

Table 1: Threats

Threat	Description of Threat
T.KEYING_MATERIAL_ COMPROMISE	An attacker can obtain unencrypted key material (the KEK, the DEK, authorization factors, submasks, and random numbers or any other values from which a key is derived) that the TOE has written to persistent memory on the USB Flash Drive, and use these values to gain access to user data.

Threat	Description of Threat
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.TSF_COMPROMISE	A malicious user or process may cause TSF data or executable code (e.g., the firmware on the USB flash drive) to be inappropriately accessed (viewed, modified, or deleted) to gain access to key material or user data.
T.MALWARE_PROPOGATION	A malicious entity on the host device places a (malicious) system file on the USB flash drive that automatically transfers itself to hosts into which the TOE is inserted, thus compromising the integrity and security features of that host.
T.UNAUTHORIZED_ACCESS	An unauthorized user that has access to the misplaced or stolen USB flash drive may gain access to the data being stored on the USB flash drive.
T.UNAUTHORIZED_UPDATE	A malicious party attempts to supply the end user with an update to the product that may compromise the security features of the TOE.
T.UNSAFE_AUTHFACTOR_VERIFICATION	An attacker can take advantage of an unsafe method for performing verification of a user-entered authorization factor, resulting in exposure of the KEK, DEK, or user data.

2.2 Assumptions

28 This section of the security problem definition shows the assumptions that are made on the operational environment in order to be able to provide security functionality. If the TOE is placed in an operational environment that does not meet these assumptions, the TOE may not be able to provide all of its security functionality. Assumptions can be for physical, personnel and connectivity aspects of the operational environment.

Table 2: TOE Assumptions

Assumption	Description of Assumption
A.AUTHORIZED_USER	Authorized users will follow all provided guidance.
A.PASSWORD_BASED_AUTH_ FACTOR	An authorized user will be responsible for ensuring that the password authorization factor has sufficient strength and entropy to reflect the sensitivity of the data being protected.

3 Security Objectives

29 The Security Objectives are the requirements for the Target of Evaluation (TOE) and for the Operational Environment derived from the threats, organizational security policies, and the assumptions in Section 2. Section 4 restates the security objectives for the TOE more formally as Security Functionality Requirements (SFR). The TOE is evaluated against the SFR.

3.1 Security Objectives for the TOE

30 Table 4 identifies the security objectives of the TOE. These security objectives reflect the stated intent to counter identified threats and/or comply with any organizational security policies identified. The TOE has to meet these objectives by satisfying the security functional requirements.

Table 3: Security Objectives for the TOE

Objective	Objective Description
O.AUTHORIZATION	The TOE must obtain the authorization factor(s) to be able to encrypt and decrypt the data on the USB flash drive.
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.ENCRYPT_ALL	The TOE will encrypt all user data that is stored on the USB flash drive.
O.DEK_SECURITY	The TOE will mask the DEK using a key encryption key (KEK) created from one or more submasks (which in turn are derived from the authorization factors) so that a threat agent who does not have authorization factor(s) will be unable to gain access to the user data by obtaining the DEK.
O.OWNERSHIP	The TOE shall ensure that ownership is taken (that is, a DEK is created, authorization factors are established, any default authorization factors are changed, a KEK is formed from the derived submasks, and the DEK is associated with the KEK) prior to any user data being stored on the TOE.
O.KEY_MATERIAL_COMPROMISE	The TOE shall ensure that no unencrypted/unmasked keys or keying material are written to persistent memory on the USB flash drive.
O.PROPOGATION_PREVENTION	The TOE shall implement mechanisms to prevent the USB flash drive from being used as a mechanism for the automated spread of malicious software.
O.SAFE_AUTHFACTOR_VERIFICATION	The TOE shall perform verification of the authorization factors in such a way that the KEK, DEK, or user data are not inadvertently exposed.
O.TRUSTED_UPDATE	The TOE shall provide users the capability to update the TOE firmware/software, and verify that updates to the product are received from the intended source.

3.2 Security Objectives for the Operational Environment

31 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.

32 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 2.3 are incorporated as security objectives for the environment. They levy additional requirements on the environment, which are largely satisfied through procedural or administrative measures. Table 5 identifies the security objectives for the Operational Environment.

Table 4: Security Objectives for the Operational Environment

Objective	Objective Description
OE.TRAINED_USERS	Authorized users will be properly trained and follow all user guidance, including the creation of strong passwords.

3.3 Security objective rationale

33 This section describes the rationale for the Security Objectives as defined in the previous section. Table 5 illustrates the mapping from Security Objectives to the Threats.

Table 5: Security Objectives to Threats Mappings

Threat/Policy	Objectives Addressing the Threat and Policies	Rationale
T.KEYING_MATERIAL_ COMPROMISE An attacker can obtain unencrypted key material (the KEK, the DEK, authorization factors and random numbers or any other values from which a key is derived) that the TOE has written to persistent memory on the USB Flash Drive, and use these values to gain access to user data.	O.DEK_SECURITY The TOE will mask the DEK using a key encryption key (KEK) created from one or more submasks (which in turn are derived from the authorization factors) so that a threat agent who does not have authorization factor(s) will be unable to gain access to the user data by obtaining the DEK. O.KEY_MATERIAL_COMPROMISE The TOE shall ensure that no unencrypted/unmasked keys or keying material are written to persistent memory on the USB flash drive.	O.DEK_SECURITY mitigates this threat by ensuring that the DEK is protected by the KEK, so the DEK cannot be directly compromised. Additionally, O.KEY_MATERIAL_COMPROMISE ensures that since no plaintext key materials are written to persistent memory on the USB flash drive, a malicious entity that recovers the USB flash drive will not be able to obtain material that can be used to compromise the DEK.
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.	O.DEK_SECURITY The TOE will mask the DEK using a key encryption key (KEK) created from one or more submasks (which in turn are derived from the authorization factors) so that a threat agent who does not have authorization factor(s) will be unable to gain access to the user data by obtaining the DEK.	O.DEK_SECURITY counters this threat by ensuring that the threat agent will be required to exhaust the key or password space to acquire access to the TOE.
T.TSF_COMPROMISE A malicious user or process may cause TSF data or executable code (e.g., the firmware on the USB flash drive) to be inappropriately accessed (viewed, modified, or deleted) to gain access to key material or user data.	O.CORRECT_TSF_OPERATION The TOE shall provide the capability to verify correct operation of the TSF in the operational environment. O.TRUSTED_UPDATE The TOE shall verify that updates to the product are received from the intended source.	O.CORRECT_TSF_OPERATION ensures that the TOE verifies the correct operation of its components; incorrect operation of the components could lead to information being accessible when it should not be. O.PROPOGATION_PREVENTIO N and O.TRUSTED_UPDATE

Threat/Policy	Objectives Addressing the Threat and Policies	Rationale
	O.PROPOGATION_PREVENTION The TOE shall implement mechanisms to prevent the USB flash drive from being used as a mechanism for the automated spread of malicious software.	ensure that attempts by unauthorized users to modify the TSF are not successful.
T.MALWARE_PROPOGATION An unauthorized user places a malicious system file on the USB flash drive that automatically transfers itself to hosts into which the TOE is inserted, thus compromising the integrity and security features of that host.	O.PROPOGATION_PREVENTION The TOE shall implement mechanisms to prevent the USB flash drive from being used as a mechanism for the automated spread of malicious software.	O.PROPOGATION_PREVENTION counters this threat by making sure there is no means by which a malicious file can be introduced to the USB flash drive that will automatically execute on the host.
T.UNAUTHORIZED_ACCESS An unauthorized user that has access to the misplaced or stolen USB flash drive may gain access to the data being transferred by the USB flash drive.	O.AUTHORIZATION The TOE must obtain the authorization factor(s) from a user to be able to decrypt the data on the USB flash drive. O.ENCRYPT_ALL The TOE will encrypt all user data which is stored on the USB flash drive. O.DEK_SECURITY The TOE will mask the DEK using a key encryption key (KEK) created from one or more submasks (which in turn are derived from the authorization factors) so that a threat agent who does not have authorization factor(s) will be unable to gain access to the user data by obtaining the DEK. O.OWNERSHIP The TOE shall ensure that ownership is taken (that is, a DEK is created, authorization factors are established, any default authorization factors are changed, a KEK is formed from the derived	O.AUTHORIZATION counters this threat by using authorization factors from an authorized user of the USB flash drive that are necessary to decrypt the data. O.ENCRYPT_ALL counters this threat by ensuring that all user data is encrypted when written to the USB flash drive. O.DEK_SECURITY counters this threat by ensuring that the data encryption key is masked with a KEK that is derived from authorization factors that are not stored on the USB flash drive. This objective also ensures that an unauthorized user does not have access to any keying material used to generate the KEK. O.OWNERSHIP addresses the threat by ensuring that there is no window in which encryption is not established on the USB flash drive and user data can be written to

Threat/Policy	Objectives Addressing the Threat and Policies	Rationale
	submasks, and the DEK is associated with the KEK) prior to any user data being stored on the TOE.	that drive. Additionally, if default authorization factors exist, there is a mechanism and appropriate guidance so that a user can change these authorization factors, thus preventing a trivial compromise of the data.
T.UNAUTHORIZED_UPDATE A malicious party attempts to supply the end user with an update to the product that may compromise the security features of the TOE.	O.TRUSTED_UPDATE The TOE shall provide users the capability to update the TOE firmware/software, and verify that updates to the product are received from the intended source.	O.TRUSTED_UPDATE counters the threat by ensuring that any updates received from a third party can be verified as coming from a source identified to the end user.
T.UNSAFE_AUTHFACTOR_ VERIFICATION An attacker can take advantage of an unsafe method for performing verification of a user-entered authorization factor, resulting in exposure of the KEK, DEK, or user data.	O. SAFE_AUTHFACTOR_ VERIFICATION The TOE shall perform verification of the authorization factors in such a way that the KEK, DEK, or user data are not inadvertently exposed	O.SAFE_AUTHFACTOR_ VERIFICATION counters the threat by ensuring that the verification of authorization factors is done without exposing the KEK, DEK, or user data; this includes countermeasures to attacks aimed at reducing the effective strength of the KEK (for example, exposing one of the authorization factors in a multi-authorization-factor use case).

34 Table 6 illustrates the mapping from Security Objectives to Assumptions.

Table 6: Security Objectives to Assumptions Mappings

Assumption	Objectives Addressing the Assumption	Rationale
A.AUTHORIZED_USER Authorized users will follow all provided user guidance.	OE.TRAINED_USERS Authorized users will be properly trained and follow all user guidance, including the creation of strong passwords.	OE.TRAINED_USERS ensures that users will be trained how to use the TOE correctly according to the user guidance.
A.PASSWORD_BASED_AUTH_FACTOR An authorized user will be responsible	OE.TRAINED_USERS	OE.TRAINED_USERS satisfies this policy by ensuring that the user

Assumption	Objectives Addressing the Assumption	Rationale
for ensuring that the password authorization factor conforms to password policy and has sufficient strength and entropy to reflect the sensitivity of the data being protected.	Authorized users will be properly trained and follow all user guidance, including the creation of strong passwords.	will create passwords with sufficient strength and entropy to reflect the sensitivity of the data being protected.

4 Security Requirements and Rationale

35 The Security Requirements are divided into functional requirements and assurance requirements. The Security Functional Requirements (SFRs) are a formal instantiation of the Security Objectives and are provided with application notes in Section 4.1. Section 4.2 is the required tracing of the SFRs to the Security Objectives.

36 The Security Assurance Requirements (SARs) are typically inserted into a PP and listed separately from the SFRs; the CEM is then consulted during the evaluation based on the SARs chosen. Because of the nature of the Common Criteria Security Assurance Requirements and the specific technology identified as the TOE, a more tailored approach is taken in this PP. While the SARs are still listed for context and completeness in Section 4.3, the majority of the activities that an evaluator needs to perform for this TOE with respect to each SFR and SAR are detailed in "Assurance Activities" paragraphs. Assurance Activities are normative descriptions of activities that must take place in order for the evaluation to be complete. Assurance Activities are located in two places in this PP; those that are associated with specific SFRs are located in Section 4.1, while those that are independent of the SFRs are detailed in Section 4.3. Note that the Assurance Activities are in fact a tailored evaluation methodology, presented in-line for readability, comprehension, and convenience.

37 For the activities associated directly with SFRs, after each SFR one or more Assurance Activities is listed detailing the activities that need to be performed to achieve the assurance required for conformant devices.

38 For the SARs that require activities that are independent of the SFRs, Section 4.3 indicates the additional Assurance Activities that need to be accomplished, along with pointers to the SFRs for which specific Assurance Activities associated with the SAR have been written.

39 Future iterations of the Protection Profile may provide more detailed Assurance Activities based on lessons learned from actual product evaluations.

4.1 Security Functional Requirements

40 The Security Functional Requirements (SFRs) are a translation of the security objectives for the TOE. They are usually at a more detailed level of abstraction, but they have to be a complete translation (the security objectives must be completely addressed). The CC requires this translation into a standardized language for several reasons:

- To provide an exact description of what is to be evaluated. As security objectives for the TOE are usually formulated in natural language, translation into a standardized language enforces a more exact description of the functionality of the TOE.
- To allow comparison between two STs. As different ST authors may use different terminology in describing their security objectives, the standardized language enforces using the same terminology and concepts. This allows easy comparison.

4.1.1 Class: Cryptographic Support (FCS)

41 The primary threats addressed by these functional requirements are a brute force attack against the key space and the failure of a cryptographic component.

42 The cryptographic requirements make reference to standards describing the algorithms; most of these standards are available from NIST are Special Publications (800-xxx) or Federal Information Processing Standards (FIPS). The assurance requirements detail how the implementation of these requirements are to be verified. Each scheme has the option of specifying the process under which the cryptographic assurance activities may be considered satisfied. All cryptographic functionality specified below must be implemented on the USB Flash Drive.

Cryptographic Key Management (FCS_CKM)

43 Conformant implementations will contain at least two keys: a Key Encryption Key (KEK) and a Data Encryption Key (DEK). The following requirements specify how the keys are produced. The generation of the DEK is specified in FCS_CKM.1(1), while the KEK is created from submasks derived from one or more authorization factors as described in FCS_CKM.1(2). The authorization factors consist of a required password-based authorization factor (conditioning to produce the submask is specified in FCS_CKM.1(3)) and (optionally) a host split authorization factor (FCS_CKM.1(X1)) and or a pin-protected submask (FCS_CKM.1(X2)). Other authorization factors from which submasks that contribute to the formation of the KEK are allowed as long as they are combined (using the XOR function) with submasks derived from the previously-mentioned authorization factors.

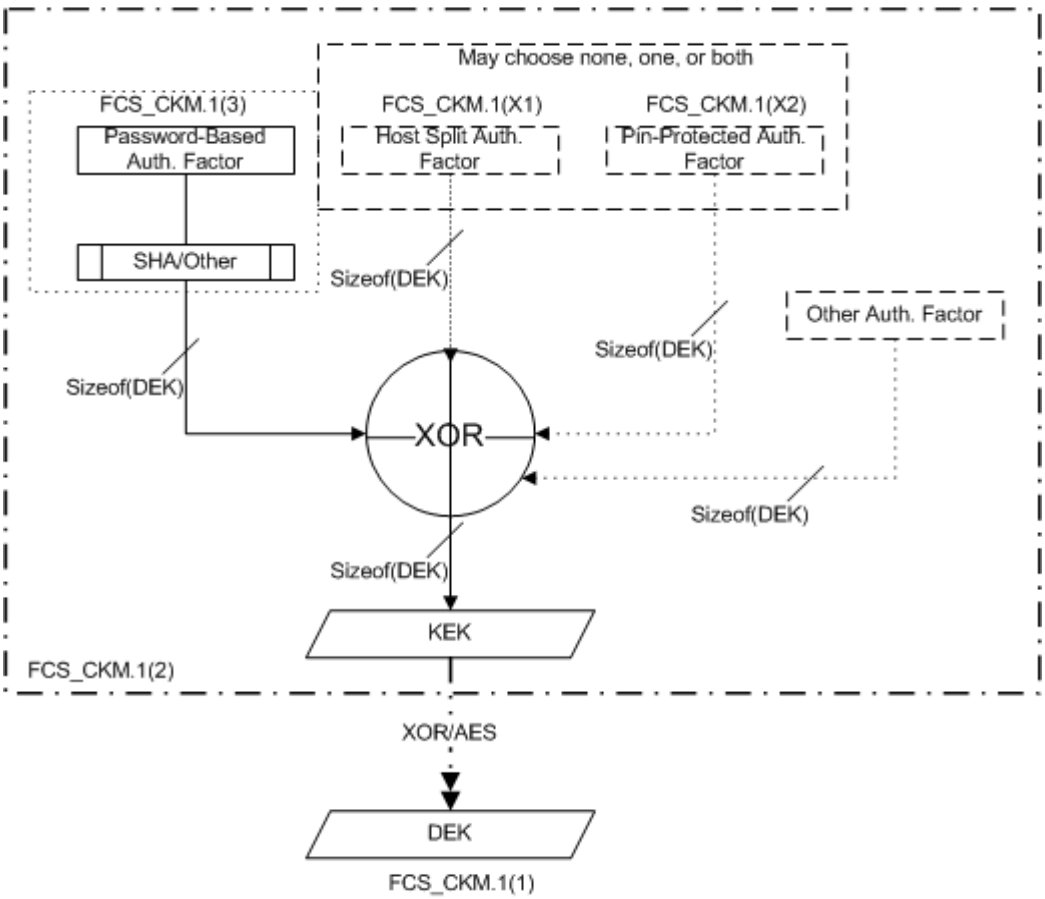

Figure 1: KEK Derivation Options

The use of the authorization factors to derive submasks used to form the KEK is covered in FCS_CKM.1(2). The TOE may use (in addition to the password) the Host Split or Pin authorization factor(s). If these authorization factors are used by the TOE, the ST should use the appropriate set of requirements from Appendix C, "Authorization Factor Generation" in the body of the ST.

The requirements pertaining to the production of the KEK and DEK are presented below.

FCS_CKM.1(1)	**Cryptographic key generation (DEK)**
FCS_CKM.1.1(1)	**Refinement:** The TSF shall generate **DEK** cryptographic keys **using a Random Bit Generator as specified in FCS_RBG_EXT.1 and** specified cryptographic key sizes **[selection: 128 bit, 256 bit]** that meet the following: [**No Standard**].

Application Note:

44 The intent of this requirement is to ensure that the DEK cannot be recovered with less work than a full exhaust of the key space for AES. The key generation capability of the TOE uses a RBG implemented on the TOE device. Either 128-bit or 256-bit (or both) are allowed; the ST author makes the selection appropriate for the device. A DEK is used in addition to the KEK so that authorization factors (especially the password authorization factor) can be changed without having to re-encrypt all of the user data on the device.

Assurance Activity:

45 The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. To the extent possible from the description of the RBG functionality in FCS_RBG_EXT, the evaluator shall determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data (FCS_COP.1(1)).

FCS_CKM.1(2)	**Cryptographic key generation (KEK)**
FCS_CKM.1.1(2)	**Refinement:** The TSF shall **derive KEK** cryptographic keys in accordance with a specified cryptographic key **derivation** algorithm **[selection: None, exclusive OR (XOR)] with the following inputs**: **a submask derived from a password authorization factor conditioned as defined in FCS_CKM.1(3),** **[selection, one or more of:** **none,** **a host split authorization factor (which is itself a submask),** **a pin-protected submask,** **[selection, choose one of: no other inputs, [assignment: list of other authorization factors and associated submask derivation methods] that produce submasks that are the same size as the DEK as specified in FCS_CKM.1(1)]]** **maintaining the effective strength of each authorization factor** and specified cryptographic key sizes **[selection: 128 bits, 256 bits]** that meet the following: **[No standard].**
Application Note:	These requirements are intended to define how the authorization factors are used to derive submasks, which in turn are used to generate the KEK. While specific guidance to the ST author is provided below for each assignment and selection, the following is a high-level description of the point of this component. The ST author chooses any additional (to the password) authorization factors supported by the TOE, including the option of defining additional authorization factors. If any additional authorization factors are defined, then the method by which submasks are produced from these authorization factors must also be described. The only condition levied on such an assignment is that the submasks produced be the same size as the DEK. Use

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

For the first selection, the ST author chooses “None” if only the password authorization factor is used. If more than one authorization factor is used, then the ST author chooses “XOR” (no other combining methods are conformant with this PP). Note that the XOR function must be performed on the USB flash drive.

For the second selection, the ST author selects any optional authorization factors used. Note that more than one can be selected. If additional authorization factors beyond a conditioned password are used, then the ST author uses the assignment within the selection within this second selection to specify these factors (or selects “no other inputs”). If either the host split or pin authorization factor selections are chosen, then the ST author must also include in the ST the appropriate requirements from Appendix C concerning the generation of the selected authorization factor(s)/submasks.

For the cryptographic key size selection, the size of the KEK that is produced is chosen; this must be the same bit length as was specified for the DEK in FCS_CKM.1(1).

Note that “No Standard” is needed because either there is only one authorization factor used to form the KEK (whose composition is specified elsewhere), or the KEK is formed using the XOR function.

Assurance Activity: 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-based factors the examination of the TSS section is performed as part of FCS_CKM.1(3) assurance activities. For the host split and pin authorization factors, if these are included their or their submask generation may be examined as part of the assurance activities taken from Appendix C for those requirements if such generation is performed by the TOE. If not, only their storage and retrieval shall be described, and will be examined as part of the assurance activities for those requirements.

If other authorization factors are specified, then for each factor, the TSS specifies how the factors are input into the TOE; how a submask is produced from the authorization factor (including any associated standards to which this process might conform), and verification performed to ensure the length of the submask meets the required size (as specified in this requirement).

If there is only one authorization factor, it naturally is not combined with anything and so no assurance activity is associated with this case. If the submasks produced from the authorization factors are XORed together to form the KEK, 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 the same as that of the DEK.

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

FCS_CKM.1(3) **Cryptographic key generation (Password conditioning)**

FCS_CKM.1.1(3) **Refinement: A password used to generate a submask shall contain up to [assignment: positive integer of 32 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 be conditioned [selection:**

- **using [selection: SHA-1, SHA-256, SHA-512] for 128-bit DEKs;**
- **using [selection: SHA-256, SHA-512] for 256-bit DEKs;**
- **using NIST SP 800-132 with a salt generated using a Random Bit Generator as specified in FCS_RBG_EXT.1, an iteration count of [assignment: *number of iterations*], and HMAC using [selection: SHA-1, SHA-256, SHA-512];**

] such that the submask that is output from the conditioning function is equal to the size (in number of bits) of the DEK.

Application Note: The password 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 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. If 800-132 conditioning is specified, then the ST author will fill in the number of iterations (C) that are performed; this value must be at least 10000. 800-132 also 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 from Appendix C.
Passwords of shorter lengths that are able to be memorized by users typically are a poor source of entropy for the KEK, as they are not long enough to provide the same "strength" as the underlying KEK/DEK. However, because this is the most convenient authorization factor at the time of this writing, passwords will be allowed as the sole authorization factor. In future versions of this PP, it is likely that passphrases that do contain sufficient entropy to be commensurate with the underlying key strength will be required.

It should be noted that the hash function required by the key derivation function (or any cryptographic operations required by the conditioning specified in 800-132) must be implemented by USB flash drive.

In subsequent publications of this PP, it is likely that SHA-1 will no longer be an approved algorithm for cryptographic hashing, and that conditioning using SP 800-132 will be required.

Assurance Activity: There are two aspects of this component that require evaluation: passwords of at least 32 characters are supported, the characters that are input are subject to the selected conditioning function. These activities are separately addressed in the text below.

Support for password lengths of 32 characters

The evaluators shall check the TSS section to determine that it specifies that a capability exists to accept passwords with the maximum number of characters specified in the ST in this assignment statement, and that the number specified is at least 32. The evaluators shall also check the operational guidance to determine that there are instructions for administrators generating such passwords, and that guidance indicates how the passwords are entered into the TOE.

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 having a character length that is equal to the maximum of 32 and the number specified in the ST for the first assignment.
- Test 2: Ensure that the TOE supports passwords of shorter lengths, consistent with what is specified in the operational guidance supplied by the vendor (for instance, if the guidance specifies that passwords have a minimum length of 16 characters, this test would minimally determine that 16-character passwords were accepted by the TOE).
- Test 3: Ensure that the TOE contains support for passwords composed as specified in guidance contained in the AGD_OPR or AGD_PRE guidance. For instance, if the guidance specifies that passwords must contain a special character, this test would fail if the TOE only supported letters and numbers.

Password Conditioning

For SHA-based conditioning of the password, the evaluator performs the following activities. The evaluator shall check that the TSS describes the method by which the password 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 in FCS_COP.1(3) 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 described in FCS_CKM.1(2), and is the same length as the DEK as specified in FCS_CKM.1(1).

For 800-132-based conditioning of the password, the required assurance activities will be performed when doing the assurance activities for the appropriate Appendix C requirements. If any manipulation of the master key is performed in forming the submask that will be used to form the 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.2	**Cryptographic key distribution (Trusted Update)**
FCS_CKM.2.1	**Refinement:** The TSF shall distribute **the public key used to verify updates to TOE system files** in accordance with a specified cryptographic key distribution method *system file on USB Flash Drive* that meets the following: *No Standard*.
Application Note:	The public key that is used for the verification of updates during the system file update process must be delivered on the USB Flash Drive.
Assurance Activity:	Activities relating to the verification of this requirement are contained in the assurance activities for FPT_SFP_EXT.1 and FPT_TUD_EXT.1. No further assurance activities are required.

Cryptographic Operation (FCS_COP)

FCS_COP.1(1)	**Cryptographic operation (Data Encryption)**
FCS_COP.1.1(1)	**Refinement:** The TSF shall perform **data encryption and decryption** in accordance with a specified cryptographic algorithm **AES used in [selection: CBC, CCM, XTS] mode** and cryptographic key sizes **[selection: 128 bits, 256 bits]** that meet the following: **FIPS PUB 197, "Advanced Encryption Standard (AES)" and [selection: NIST SP 800-38A, NIST SP 800-38C, NIST SP 800-38E].**

Application Notes

46 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 USB flash drive. For the first selection, the ST author should indicate the mode or modes supported by the TOE implementation. The second selection indicates the key size to be used, which is identical to that specified for FCS_CKM.1(1). The third 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.

47 Future versions of this PP may include new cryptographic modes as they are reviewed and approved by NIST.

Assurance Activities:

48 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. Note that some of these tests will require a reference implementation of the algorithms that is acceptable to the evaluation facility's Scheme.

CBC Mode

49 The CBC mode tests reference *The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)* [AESAVS], available from http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf.

50 The evaluators shall run a set of known answer tests for each key size supported by the TSF. Inputs are a key, IV, and either plaintext to be encrypted or ciphertext to be decrypted. All of the test vectors (both encrypt and decrypt) for CBC mode in the supported key lengths from http://csrc.nist.gov/groups/STM/cavp/documents/aes/KAT_AES.zip shall be used to perform these tests

51 The evaluators shall perform a multi-block message test for each key length supported. To perform this test, the evaluators generated 10 data sets for encryption and 10 data sets for decryption. Each data set consists of key, an IV, and plaintext (for encryption) or ciphertext (for decryption). The length of a block shall be 128 bits; the length of the plaintext/ ciphertext shall be block length * i, where i indicates the data set number and i ranges from 1 to 10 (so messages will range from 128 bits to 1280 bits).

52 The evaluators shall perform a Monte Carlo test. The evaluators shall generate 10 sets of starting values for encryption (values for the key, IV, and plaintext) and 10 sets of starting values for decryption (values for the key, IV, and ciphertext). The length of the plaintext/ciphertext shall be 128 bits. Each set of starting values is used to generate and perform 100 tests; the algorithm for generating the 100 test values (per set of starting values) is contained in section 6.4.2 of [AESAVS].

CCM Mode

53 The CCM mode tests reference *The CCM Validation System (CCMVS)* [CCMVS], available from http://csrc.nist.gov/groups/STM/cavp/documents/mac/CCMVS.pdf.

54 The evaluators shall examine the TSS to ensure the lengths for the payload, associated data, nonce, and tags (as well as the key length) are specified. These values shall be used in constructing the tests described in the next section. If multiple values are supported, then the evaluator shall examine the operational guidance to determine how the values are selected by the user.

55 The evaluator shall perform the following five tests for each key length supported by the USB flash drive.

56 The evaluator shall perform a variable associated data test. For each associated data length supported, the evaluators shall devise 10 sets of input data. Each set of input data shall use the same key and nonce, and have the same tag (MAC) length. For each of the 10 sets, a unique string of associated data and payload data shall be used. The evaluators shall calculate the correct ciphertext for the inputs, and then ensure that the TSF calculates the same value for all input sets for all supported associated data lengths. An example of the input sets (for a 256-bit key) can be found in the VADT256.txt file from the archive http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip.

57 The evaluator shall perform a variable payload test. For each payload length supported, the evaluators shall devise 10 sets of input data. Each set of input data shall use the same key and nonce, and have the same tag (MAC) length. For each of the 10 sets, a unique string of associated data and payload data shall be used. The evaluators shall calculate the correct ciphertext for the inputs, and then ensure that the TSF calculates the same value for all input sets for all supported payload lengths. An example of the input sets (for a 256-bit key) can be found in the VPT256.txt file from the archive http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip.

58 The evaluator shall perform a variable nonce test. For each nonce length supported, the evaluators shall devise 10 sets of input data. Each set of input data shall use the same key and have the same tag (MAC) length. For each of the 10 sets, a unique nonce and unique strings of associated data and payload data shall be used. The evaluators shall calculate the correct ciphertext for the inputs, and then ensure that the TSF calculates the same value for all input sets for all supported nonce lengths. An example of the input sets (for a 256-bit key) can be found in the VNT256.txt file from the archive http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip.

59 The evaluator shall perform a variable tag test. For each tag length supported, the evaluators shall devise 10 sets of input data. Each set of input data shall use the same key and nonce. For each of the 10 sets, a unique string of associated data and payload data shall be used. The evaluators shall calculate the correct ciphertext for the inputs, and then ensure that the TSF calculates the same value for all input sets for all supported tag lengths. An example of the input sets (for a 256-bit key) can be found in the VTT256.txt file from the archive http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip.

60 The final test the evaluators shall perform is the decryption-verification process test. This test is performed for each combination of associated data length, payload length, nonce length, and tag length supported by the TSF. For each combination, 15 sets of input data are provided to the TSF. The input data consists of a key, associated data, payload data, nonce, and ciphertext. The evaluators should ensure that between 1/3 and 2/3 of the ciphertext values do not pass the MAC check for a variety of error types. The inputs are supplied to the TSF and the evaluators verify that the TSF correctly identifies erroneous MAC values as well as passing values. An example of the input sets (for a 256-bit key) can be found in the VTT256.txt file from the archive http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip.

XTS Mode

61 The XTS mode tests reference *The XTS-AES Validation System (XTSVS)* [XTSVS], available from http://csrc.nist.gov/groups/STM/cavp/documents/aes/XTSVS.pdf.

62 The evaluator first conducts the tests identified in the *CBC Mode* section above. After completing those tests, the evaluator examines the TSS to ensure that the range of data lengths supported in XTS mode is identified, and the format of the tweak value (either a 128-bit string or a data unit sequence number).

63 The evaluator then devises test sets for each key length supported. For a given key length, the evaluator chooses a sample of at least 5 data lengths to test. For each data length, the evaluator devises 100 encryption tests and 100 decryption tests. Each test is performed with a unique key, tweak, and plaintext (for encryption) or ciphertext (for decryption) value. Examples of test sets can be seen in http://csrc.nist.gov/groups/STM/cavp/documents/aes/XTSTestVectors.zip.

FCS_COP.1(2) Cryptographic operation (Signature Verification)

FCS_COP.1.1(2)

Refinement: The TSF shall perform **cryptographic signature verification for TOE updates** in accordance with a ***[selection:***

(1) Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater,

(2) RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of 2048 bits or greater, or

(3) Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater]

that meets the following:

Case: Digital Signature Algorithm

- **[FIPS PUB 186-3, "Digital Signature Standard"**

Case: RSA Digital Signature Algorithm

- **FIPS PUB 186-3, "Digital Signature Standard"**

Case: Elliptic Curve Digital Signature Algorithm

- **FIPS PUB 186-3, "Digital Signature Standard"**
- **The TSF shall implement "NIST curves" P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-3, "Digital Signature Standard").**

Application Note

64 The ST Author should choose the algorithm implemented to perform digital signatures; if more than one algorithm is available, this requirement (and the corresponding FCS_CKM.1(2) requirement) should be iterated to specify the functionality. For the algorithm chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.

65 For elliptic curve-based schemes, the key size refers to the $\log_2$ of the order of the base point. As the preferred approach for digital signatures, elliptic curves will be required after all the necessary standards and other supporting information are fully established.

Assurance Activities:

66 This requirement is used to verify digital signatures attached to updates from the TOE manufacturer before installing those updates on the TOE. Because this component is to be used in the update function, additional assurance activities to those listed below are covered in other assurance activities section in this document. The following assurance requirements deal only with the implementation for the digital signature algorithm; the evaluator performs the testing appropriate for the algorithm(s) selected in the component.

67 Since any hash functions or random number generation required by these algorithms is contained on the USB Flash Drive, the assurance activities associated with those functions is contained in the Cryptographic Hashing and Random Bit Generation sections that follow this component. Additionally, the only function required by the TOE is the verification of digital signatures. If the TOE generates digital signatures to support the implementation of any functionality required by this PP, then the cognizant evaluation and validation scheme must be consulted to determine the required assurance activities.

68 For any algorithm, the evaluators check the TSS to ensure that it describes the overall flow of the signature verification. This should at least include identification of the format and general location (e.g., "firmware on the USB device" vice “memory location 0x00007A4B") of the data to be used in verifying the digital signature; how the data received from the operational environment are brought on to the device; and any processing that is performed that is not part of the digital signature algorithm (for instance, checking of certificate revocation lists).

69 Each section below contains the tests the evaluators must perform for each type of digital signature scheme. Based on the assignments and selections in the requirement, the evaluators choose the specific activities that correspond to those selections. For each type of digital signature scheme, a test section is provided for conformance to FIPS 186-3 and FIPS 186-2.

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

71 Similarly, it is not anticipated that signature generation will be required to meet the baseline requirements of this PP. If signature generation is required, then the evaluation and validation scheme must be consulted to ensure the correct specification of the required assurance activities and any additional components.

RSA

72 There are two options for implementing the signature generation/verification function: ANSI X9.31 and PKCS #1 (Version 1.5 and/or Version PSS). At least one of these options must be implemented. Each implemented version must be tested as indicated below. If PKCS #1 Version PSS is chosen, then the evaluator shall check the TSS to ensure the length of the salt is specified.

73 If the TOE supports more than one modulus size, then the evaluator shall perform the following test for all modulus sizes. If the TOE supports more than one hash algorithm, the evaluator shall perform the following test for all hash algorithms. This means that if the implementation allows the choice of 2 modulus sizes and 2 hash algorithms, the evaluator would perform the following test 4 times.

74 The evaluator shall generate three groups of data. Each group of data consists of a modulus and 4 sets of test vectors consist with the modulus. The test vectors consist of a public exponent *e*; a pseudorandomly-generated message; and a signature for the message using the associated private key (consistent with *e* and the modulus *n*). This means that there will be a minimum of 12 test vectors for each modulus size/hash algorithm supported by the TSF.

75 In 3/4s of the test vectors after the correct signature has been generated (but not "fed" to the TSF), the evaluators will alter the public key, message, or signature (making sure to do at least 2 of each) so

that the signature verification failure function will be tested. The evaluators shall then run the test vectors through the TSF and verify that the results are correct.

76 In addition, if the algorithm implemented is RSASSA-PKCS1-v1_5, as specified in *Public Key Cryptography Standards (PKCS) #1 v2.1: RSA Cryptography Standard-2002*, or the RSA algorithm described in X9.31, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, the appropriate additional test vectors from http://csrc.nist.gov/groups/STM/cavp/documents/dss/SigVer15EMTest.zip (for PKCS #1 Version 1.5 implementations) or http://csrc.nist.gov/groups/STM/cavp/documents/dss/SigVer931IRTest.zip (for X9.31 implementations) shall be used by the evaluators to verify that the implementation successfully passes these tests.

DSA

77 The evaluator examines the TSS to ensure that the values used for (L, N) are given, and the hash algorithm(s) used are specified. The evaluator verifies that the hash algorithm used for a specific (L,N) provides the requisite strength, as specified in Tables 2 and 3, Section 5.6.1 of SP 800-57, *Recommendation for Key Management --Part I: General (Revised)*. The evaluators shall also ensure that the (L,N) selected has comparable strength to the symmetric (data) encryption algorithm used on the USB Flash Drive (e.g., if 128-bit AES is used to encrypt the user data, then an (L,N) of at least (3072, 256) is required).

78 The evaluator performs the following test for each (L,N) and hash combination supported. The evaluator shall generate a key pair. The evaluators will then pseudorandomly generate 15 1024-bit messages and sign them with the private key. For about half of the messages--after the correct signature has been generated (but not "fed" to the TSF)--the evaluators will alter the public key, message, or signature (making sure to do at least 2 of each) so that the signature verification failure function will be tested. The evaluators shall then run the test vectors through the TSF and verify that the results are correct.

ECDSA

79 The evaluators shall examine the TSS to determine the curve or curves used in the implementation are specified, and the hash or hashes supported are specified. The evaluators shall conduct the following test for each curve, hash pair implemented by the TSF.

80 The evaluator generates 15 sets of data. Each dataset consists of a pseudorandom message; a public/private key pair (d,Q), and a signature (r,s). For about half of the messages--after the correct signature has been generated (but not "fed" to the TSF)--the evaluators will alter the public key, message, or signature (making sure to do at least 2 of each) so that the signature verification failure function will be tested. The evaluators shall then run the data through the TSF and verify that the results are correct.

FCS_COP.1(3)	**Cryptographic operation (Cryptographic Hashing)**
FCS_COP.1.1(3)	**Refinement:** The TSF shall perform **cryptographic hashing services** in accordance with **[selection: SHA-1, SHA 256, SHA 384, SHA 512]** and **message digest sizes [selection: 160, 256, 384, and 512] bits** that meet the following: **FIPS Pub 180-3, "Secure Hash Standard".**

Application Note:

81 The intent of this requirement is to specify the hashing function used for the digital signature checking associated with trusted updates and system file protection, and for use in conditioning the password (see FCS_CKM.1(3)). The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1) and FCS_COP.1(2) (SHA 256 for 128-bit keys, SHA 512 for 256-bit keys). In subsequent publications of this PP, it is likely that SHA-1 will no longer be an approved algorithm for cryptographic hashing.

Assurance Activities:

82 The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present. The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

83 The cryptographic hashing tests reference *The Secure Hash Algorithm Validation System (SHAVS)* [SHAVS], available from http://csrc.nist.gov/groups/STM/cavp/documents/shs/SHAVS.pdf.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

88 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is $512 + 99*i$, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

FCS_COP.1(4)	**Cryptographic operation (Key Masking)**
FCS_COP.1.1(4)	**Refinement:** The TSF shall perform **key masking** in accordance with a specified cryptographic algorithm **[selection: XOR; AES used in ECB mode]** and the cryptographic key size **[selection: 128 bits, 256 bits]** that meet the following: [**selection: "None" for XOR; "FIPS PUB 197, *Advanced Encryption Standard (AES)* and NIST SP 800-38A" for AES].**

Application Note:

91 In the first selection, the ST author chooses the method by which the KEK is used to mask the DEK: either XORing the KEK and the DEK, or using AES in ECB mode. If XOR is chosen, then the ST author chooses "None" in the last selection; otherwise they choose the reference to FIPS 197 and SP 800-38A. The second selection should be made to reflect the size of the KEK.

Assurance Activities:

92 If the DEK masking method is using "XOR", the evaluators shall verify that the use of XOR is stated in the TSS. If AES is used, then the following assurance activities will be performed.

93 The evaluator shall ensure that the vendor has described the method/algorithm by which the KEK is used to mask the DEK using AES (for example, any options specified by the FIPS documents are identified, methods of padding the input, truncating the output, etc.).

94 The evaluator shall perform the following tests. If multiple modes are supported, the evaluator examines the TSS and guidance documentation to determine how ECB and the specified key-size is chosen by the end user. The evaluator then tests each key size in the manner found in the following sections, as appropriate. Note that some of these tests will require a reference implementation of the algorithms that is acceptable to the evaluation facility's Scheme.

ECB Mode

95 The ECB mode tests reference *The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)* [AESAVS], available from http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf.

96 The evaluators shall run a set of known answer tests for each key size supported by the TSF. Inputs are a key and either plaintext to be encrypted or ciphertext to be decrypted. All of the test vectors (both encrypt and decrypt) for ECB mode in the supported key lengths from http://csrc.nist.gov/groups/STM/cavp/documents/aes/KAT_AES.zip shall be used to perform these tests

97 The evaluators shall perform a multi-block message test for each key length supported. To perform this test, the evaluators generated 10 data sets for encryption and 10 data sets for decryption. Each data set consists of key and plaintext (for encryption) or ciphertext (for decryption). The length of a block shall be 128 bits; the length of the plaintext/ ciphertext shall be block length * i, where i indicates the data set number and i ranges from 1 to 10 (so messages will range from 128 bits to 1280 bits).

98 The evaluators shall perform a Monte Carlo test. The evaluators shall generate 10 sets of starting values for encryption (values for the key and plaintext) and 10 sets of starting values for decryption (values for the key and ciphertext). The length of the plaintext/ciphertext shall be 128 bits. Each set of starting values is used to generate and perform 100 tests; the algorithm for generating the 100 test values (per set of starting values) is contained in section 6.4.1 of [AESAVS].

Extended: Cryptographic Operation (Random Bit Generation) (FCS_RBG_EXT)

FCS_RBG_EXT.1	**Extended: Cryptographic operation (Random Bit Generation)**
FCS_RBG_EXT.1.1	The TSF shall perform all random bit generation (RBG) services in accordance with [selection, choose one of: NIST Special Publication 800-90 using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES) , Dual_EC_DRBG (any)]; FIPS Pub 140-2 Annex C; X9.31 Appendix 2.4 using AES] seeded by an entropy source that accumulates entropy from at least one independent TSF-hardware-based noise sources.
FCS_RBG_EXT.1.2	The deterministic RBG shall be seeded with a minimum of [selection, choose one of: 128 bits, 256 bits] of entropy at least equal to the greatest bit length of the keys and authorization factors that it will generate.

Application Note:

99 NIST Special Pub 800-90, Appendix C describes the minimum entropy measurement that will probably be required future versions of FIPS-140. If possible this should be used immediately and will be required in future publications of this PP.

100 For the first selection in FCS_RBG_EXT.1.1, the ST author should select the standard to which the RBG services comply (either 800-90 or 140-2 Annex C).

101 SP 800-90 contains four different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used (if 800-90 is selected), and include the specific underlying cryptographic primitives used in the requirement or in the TSS. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CT_DRBG are allowed. While any of the curves defined in 800-90 are allowed for Dual_EC_DRBG, the ST author not only must include the curve chosen, but also the hash algorithm used.

102 Note that for FIPS Pub 140-2 Annex C, currently only the method described in *NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms*, Section 3 is valid. If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1 may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2, the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

103 The ST author also ensures that any underlying functions are included in the baseline requirements for the TOE.

104 In the future, most of the requirements described in *A Method for Entropy Source Testing: Requirements and Test Suite Description* will be required by this PP. The follow Assurance Activities currently reflect only that subset of activities that are required.

Assurance Activity:

105 The evaluator shall review the TSS section to determine the version number of the product containing the RBG(s) used in the TOE. The evaluator shall also confirm that the TSS describes the hardware-based noise source from which entropy is gathered, and further confirm that this noise source is located on the USB Flash Drive. The evaluator will further verify that all of the underlying functions and parameters used in the RBG are listed in the TSS.

106 The evaluator shall verify that the TSS contains a description of the RBG model, including the method for obtaining entropy input, as well as identifying the entropy source(s) used, how entropy is produced/gathered from each source, and how much entropy is produced by each entropy source. The evaluator shall also ensure that the TSS describes the entropy source health tests, a rationale for why the health tests are sufficient to determine the health of the entropy sources, and known modes of entropy source failure. Finally, the evaluator shall ensure that the TSS contains a description of the RBG outputs in terms of the independence of the output and variance with time and/or environmental conditions.

107 Regardless of the standard to which the RBG is claiming conformance, the evaluator perform the following test:

- Test 1: The evaluator shall determine an entropy estimate for each entropy source by using the *Entropy Source Test Suite.* The evaluator shall ensure that the TSS includes an entropy estimate that is the minimum of all results obtained from all entropy sources.

108 The evaluator shall also perform the following tests, depending on the standard to which the RBG conforms.

Implementations Conforming to FIPS 140-2, Annex C

109 The reference for the tests contained in this section is *The Random Number Generator Validation System (RNGVS)* [RNGVS]. The evaluators shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

110 The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.

111 The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in *NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms*, Section 3. The evaluators ensure that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90

112 The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

113 If the RNG has prediction resistance enabled, each trial consists of (1) instantiate drbg, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. “generate one block of random bits” means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90).

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

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

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

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

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

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

4.1.2 Class: User Data Protection (FDP)

120 This family stipulates encryption of all stored user data.

Extended: Protection of Data on USB Flash Drive (FDP_DSK_EXT)

FDP_DSK_EXT.1 Extended: Protection of Data on USB Flash Drive

FDP_DSK_EXT.1.1 The TSF shall perform encryption of user data in accordance with FCS_COP.1.1(1).

FDP_DSK_EXT.1.2 The DEK shall only exist in persistent memory on the USB flash drive if it is masked with a KEK derived as specified in FCS_CKM.1(2) and FCS_COP.1(4).

FDP_DSK_EXT.1.3 The TSF shall encrypt all user data without user intervention.

FDP_DSK_EXT.1.4 With the exception of PIN-protected submasks written to a protected area of the USB Flash Drive, no plaintext keying material used by the TOE shall be written to persistent memory on the USB flash drive.

Application Note:

121 "Data Encryption" is defined in the Glossary for this PP as "the process of encrypting all user data written to a USB flash drive".

122 The intent of this requirement is to specify that encryption of any user data will not depend on a user electing to protect those data (for instance, a per-file-based encryption scheme would not be acceptable). The data encryption specified in FDP_DSK_EXT.1 occurs transparently to the user and the decision to protect the data is outside the discretion of the user.

123 Data and files that are stored in areas on the USB flash drive required for proper operation of the flash drive must also be protected; this is covered by FPT_SFP_EXT.1.

Assurance Activities:

124 The evaluator shall consult the TSS of the ST in performing the assurance activities for this requirement. The evaluator focuses on ensuring that the description is comprehensive in how the data are written to the device and the point at which the encryption function is applied.

125 In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on USB flash drive insertion relating to the loading of the software that will perform the collection of information from the user used to generate the authorization factors. The TSS should also cover the initialization of the TOE and the activities that are performed to ensure that the USB flash drive is entirely encrypted when the TOE is first established. If optional authorization factors are supported by the TOE, the evaluator shall check that the TSS describes how these authorization factors are obtained, and how the submasks are derived from these authorization factors.

126 The evaluator shall ensure that the description also covers how the cryptographic functions in the FCS requirements are being used, including how the DEK is unmasked and stored in the TOE.

127 The evaluator reviews the TSS to determine that it makes a case that key material is not written in plaintext form to the USB flash drive. Since in normal use all writes to the USB flash drive will be encrypted, one approach is to make an argument concerning the exceptional cases when data are written to the unencrypted portions of the device, then detailing why key material is not written in these areas.

128 The evaluators shall perform the following test activities:

- Test 1: Ensure that following the initialization activities results in the USB flash drive being encrypted. For areas of the device(s) that are found to be unencrypted, ensure justification

is provided (in the TSS or operational guidance, for instance) that no user data can be written to these areas.

- Test 2: Ensure that user data are encrypted when written to the USB flash drive. The extent that this is tested is consistent with the previous test; that is, it is acceptable to insert the USB flash drive "normally", cause user data to be written to the USB flash drive, and then ensure those data do not appear unencrypted.

4.1.3 Class: Identification and Authentication (FIA)

Extended: USB Flash Drive User Authorization (FIA_AUT_EXT)

FIA_AUT_EXT.1	**Extended: USB Flash Drive User Authorization**
FIA_AUT_EXT.1.1	The TSF shall provide a mechanism as defined in FCS_CKM.1.1(2), FCS_CKM.1.1(3), and FCS_COP.1(4) to perform user authorization.
FIA_AUT_EXT.1.2	The TSF shall perform user authorization using the mechanism provided in FIA_AUT_EXT.1.1 before allowing access to user data from the device.
FIA_AUT_EXT.1.3	The TSF shall perform user authorization using the mechanism provided in FIA_AUT_EXT.1.1 before allowing the user to change the password-based authorization factor as specified in FMT_SMF.1(c).
FIA_AUT_EXT.1.4	The TSF shall verify that the user-entered authorization factors are valid prior to decrypting user data on the USB Flash Drive.
FIA_AUT_EXT.1.5	The TSF shall ensure that the method of validation for each authorization factor does not expose or reduce the effective strength of the KEK, DEK, or CSPs used to derive the KEK or DEK.

Application Note:

129 The intent of this requirement is to specify the mechanisms by which users are authorized to decrypt the USB flash drive and thus gain access to their data. Note that this is not considered authentication of an individual user. The authorization factors can be cloned and provided to all authorized users of a USB flash drive. Or, the user could have an authorization factor that is unique to the user. However, it does require that the authorization factor is "authenticated" prior to the USB data being generally accessible to the user, and that the method used for this authentication does not compromise any keys or keying material. The means of this authentication might vary based on the authorization factor used.

130 Elements 1.4 and 1.5 deal with the validation of the authorization factors provided by the user prior to a user being able to access the information on the device. If the authorization factor is not valid, it is undesirable for the TSF to attempt to form a KEK, use it to unmask the DEK, and then 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 is typically done on the host, it may be monitored/disassembled by an attacker and so must be designed with this threat in mind.

131 User authorization only needs to be performed when the device is made accessible to the user (that is, plugged into a USB port and recognized by the underlying OS). The above requirement should not be interpreted to mean that user authorization has to take place prior to every device or file access. However, in the event that the user wishes to change their password-based authorization factor, the user authorization functionality will have to be invoked prior to the change being completed.

Assurance Activity:

The evaluator shall check the TSS section to determine it describes how the TOE is initialized; that is, the sequence of events including insertion and detection of the USB flash drive; software that is loaded from the USB flash drive (if any); the entry of the password and formation of the KEK; and the unmasking of the DEK and access to the encrypted portions of the USB flash drive. The authorization-before-change functionality (FIA_AUT_EXT.1.3) is tested under FMT_SMF.1.

The evaluator shall check that the TSS describes how the authorization factors are validated prior to allowing the user to access the data on the USB Flash Drive. This description shall be in enough detail so that the evaluator can determine that the method or methods used do not expose the DEK, KEK, or other key material. "Expose" also includes the notion of weakening the DEK 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 perform the following test:

- Test 1: Ensure that the authorization factors are prompted for (where appropriate) prior to allowing any access to unencrypted data on the USB flash drive. For each supported authorization factor, ensure that incorrect entry of an authorization factor results in a notification from the TOE that an incorrect authorization has been provided.

4.1.4 Class: Security Management (FMT)

132 The primary intent in this section is to call out critical activities that must be (or must not able to be) performed in order to use the device in a safe manner. While some of these functions might be considered “administrative” functions for other types of TOEs, for USB flash drives it is the expectation that all of these functions are able to be performed by the end user of the device. The administration model for conformant TOEs is described in Section 1.1.5 of this PP.

133

Management of TSF Data (FMT_MTD)

FMT_MTD.1	**Management of TSF Data (for reading of all symmetric keys)**
FMT_MTD.1.1	**Refinement:** The TSF shall **prevent** *reading of all key material on the USB device by users external to the device.*

Application Note:

134 The intent of the requirement is that no entity external to the USB device be able to read or view key material contained or processed on the device through the interface presented by the device; this includes the ability to read masked keys as well.

Assurance Activity:

135 The evaluator shall examine the TSS to determine that it details the interfaces to the device that are visible to the host; this can be directly specified in the TSS, or referenced as long as the reference is available to readers of the ST. The interfaces include those available to the device driver and (optionally) those that are exported by the device driver. The evaluator shall examine the TSS for an argument that none of the interfaces allow an external entity to obtain key (either masked or in plaintext) from the USB device, and validate the argument by examining the interface descriptions with respect to their claimed capability (or lack thereof).

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: a) **generate the DEK when the USB Flash Drive is initially put into use.** b) **Protect the DEK using a KEK formed from submasks derived from user-entered authorization factors; specifically a conditioned password-based authorization factor and [selection: no other inputs, PIN-protected stored submask, host split authorization factor/submask, [assignment: *other authorization factors/submasks*]].** c) **change password-based authorization factor.** d) **[selection, choose one of: no other functions, [selection: change default authorization factors, generate authorization factors, configure authorization factor inputs, configure cryptographic functionality, disable key escrow functionality, [assignment: other management functions provided by the TSF]].]**

Application Note:

136 The intent of this requirement is to express the management capabilities that the TOE possesses. This means that the TOE must be able to perform the listed functions. Items (a) and (b) establish the necessary keying material for operational use, item (c) allows the user to change their authorization factor, and Item (d) is used to specify functionality that may be included in the TOE, but is not required to conform to the PP.

137 For Item b, the appropriate authorization factors that contribute submasks to the formation of the KEK as per FCS_CKM.1(2) should be specified in the selection and assignment statements. From an implementation point of view, this binds the authorization factors to the DEK so that the factors can be provided to end users. These authorization factors must be entered or generated under user control; while an escrow capability may exist, it must be disabled (or have the ability to be disabled) such that there are no recovery or escrow keys generated when the DEK/KEK are generated.

138 For item c, only the password-based authorization factor must be changeable. If other authorization factors are allowed to be changed, then the assignment in item (d) should be used by the ST

authors, and appropriate assurance activities and rationale will need to be added as well to support this additional functionality.

139 In item d, if no other management functions are provided (or claimed), then “no other functions” should be selected. Several other common options are given:

- If the host split authorization factor or pin-protected submask are implemented by the TOE, then “generate authorization factors” must be included, along with the appropriate requirement(s) from Appendix C.
- If the TOE provides configurability of the cryptographic functions (for example, key size of the DEK), 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 escrow function, the TOE must provide the capability for the user to turn this functionality off so that no escrow key is generated.
- If “other management functions” are assigned, the National Scheme overseeing the evaluation 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 PP.

Assurance Activity:

140 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.

141 **Required Activities**.

142 *Generate DEK*

143 The evaluator reviews the AGD guidance and shall determine that the instructions for generating a DEK exist. The instructions must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate the DEK. The TSS is checked to ensure that the description of how the DEK is generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account.

144 *Protect the DEK with a KEK formed from submasks from appropriate authorization factors*

145 The ST will specify the authorization factors that are supported by the TOE, and provide requirements on how many factors—and in what combinations—are necessary to successfully use the TOE functions (this is done in the FCS_CKM requirements). This requirement deals with the initial protection of the DEK (or the protection of a new DEK) with the KEK produced from the submasks derived from those authorization factors. The evaluator shall review the AGD guidance to determine, for each supported authorization factor, that the guidance details how that factor is input into the TSF for this operation. The evaluator shall review the TSS section to determine that it describes how the submasks for the various authorization factors are derived and then combined to form the KEK, and how the KEK is used to mask the DEK; it shall also be clear whether there are any differences between this process and the process used during “normal” operations (that is, once the encryption on the TOE is established). This description could also encompass the information described in the assurance activities for the FCS_CKM.1* requirements. If

the supported authorization factors are different depending on the platform, then the AGD guidance shall be clear on the minimum requirements for each platform, and any other limitations concerning authorization factors that apply. The evaluator shall also perform the following test:

- Test 1: For each supported minimum number of authorization factors, establish a DEK and then ensure that the administrator is able to enter the authorization factors that result in the DEK being encrypted. The number of different times this test is performed is dependent on the number of platforms supported and the differences in the required authorization factors. While not all combinations of supported authorization factors and supported platforms are required to be tested, a representative sample shall be used and the evaluator shall provide justification for this sample in the test report.

146 *Change the password-based authorization factor*

147 The evaluator shall examine the operational guidance to ensure that it describes how the password-based authorization factor is to be changed. The evaluator shall also examine the TSS to ensure that it describes the sequence of activities that take place on the host and on the USB Flash Drive when this activity is performed, and ensure that the KEK and DEK are not exposed during this change. The evaluator shall also perform the following test:

- Test 2 The evaluator shall establish a password authorization factor for the USB device. The evaluators shall then transfer user data from the host to the device. They shall then use the "change authorization factor" functionality to change the password on the device, and ensure that they are prompted for the current authorization factor. They shall enter an incorrect value for the current authorization factor and observe that no change to the authorization factor is made. Upon entry of a correct value for the current authorization factor, they shall ensure that they are still able to access the user data on the device. The evaluator shall also use the old authorization factor (after successfully changing the authorization factor) to show that it no longer provides access to the user data on the device.

148 **Conditional Activities**

149 Item d in the above requirement contains several selections specifying functionality that may be provided by the TOE, but is not required to be conformant to this PP. If the functionality is provided, though, the TOE can claim conformance by including the appropriate requirements from Appendix C and making the corresponding selection above. As noted in the application note, if an assignment is made, the National Scheme overseeing the evaluation needs to be consulted to determine if compliance to the PP can be claimed.

150 It may be the case that the USB flash drive arrives with default authorization factors in place. If it does, then the selection in section d must be made so that there is a mechanism to change these authorization factors. The operational guidance shall describe the method by which the user changes these factors when they are taking ownership of the device. The TSS shall describe the default authorization factors that exist. The evaluators also perform the following test:

- Test 3 [conditional] If the TOE provides default authorization factors, the evaluator shall change these factors in the course of taking ownership of the device as described in the operational guidance. The evaluator shall then confirm that the (old) authorization factors are no longer valid for data access.

151 One of the selections concerns the generation of authorization factors: the host split authorization factor and the pin-protected submask. In each of these cases, additional requirements from Appendix C will be included in the ST; associated with these requirements are the assurance activities covering the details of how the authorization factors/submasks are generated. If the administrator has the option to determine what combination of authorization factors are valid for a particular device, then that selection should be made as well. For the FMT_SMF activities relating to the administration of these functions, the evaluator shall review the AGD information to ensure that the instructions for invoking the authorization factor mechanism are detailed and clear enough so that an authorization factor with the required characteristics can be generated. Any activities required to store or manipulate the authorization factors shall also be confirmed. Any configuration activities done by the administrator in terms of determining what authorization factors (of those supported by the TOE) are configured for use shall also be described and confirmed by the evaluation team. The tests associated with these mechanisms are specified as part of that particular mechanisms' assurance activities.

152 In some TOEs, there may be a choice with respect to the underlying cryptography that is used; for instance, the length of the DEK in bits, or the encryption mode that is used for AES. Again, this capability does not have to be offered for the TOE to claim conformance to the PP; however, if the capability is offered, it is specified in the ST and the "configure cryptographic functionality" choice is selected in the requirement above.

153 For this selection, the evaluator shall determine from the ST what portions of the cryptographic functionality are configurable. This will entail looking at the FCS requirements as well as the associated description in the TSS. Armed with this information, the evaluator shall review the AGD documentation to determine that there are instructions for manipulating all of the claimed mechanisms.

154 If the TOE supports key escrow, this must be stated in the TSS. The TSS shall also describe how to disable this functionality, including how the escrow material is provided to the escrow holder. The intent is that this description can be used by the evaluators in testing to determine whether the escrow functionality has indeed been disabled (for instance, if the TSS states that the material is sent to the third party through a network connection when a new KEK/DEK is generated, the evaluators can disable the functionality, attach a network monitor, and see whether a network connection is made when a new KEK/DEK is generated). The guidance for disabling this capability shall be described in the AGD documentation.

- Test 4 [conditional] If the TOE provides an escrow capability whose effects are visible at the TOE interface, then the evaluator shall devise a test that ensures that the escrow capability has been or can be disabled following the guidance provided by the vendor.

4.1.5 Class: Protection of the TSF (FPT)

Extended: TSF System File Protection (FPT_SFP_EXT.1)

FPT_SFP_EXT.1	**Extended: TSF System File Protection**
FPT_SFP_EXT.1.1	The TSF shall ensure that only signed and verified system files are installed on and used by the USB flash drive.

Application Note:

155 This requirement prevents users (or most often malicious programs acting on their behalf) from replacing legitimate files that are used in the operation of the USB flash drive with malicious versions, while still allowing updates to the TOE to be performed, as specified in the operational guidance and FPT_TUD_EXT.1. Vectors of delivery of malicious files can include from files automatically executed when the device is inserted (e.g., .ini files), or files that are executed when the USB device is used as a boot device.

Assurance Activities:

156 The signed update functionality is verified through assurance activities performed for the FPT_TUD_EXT.1 requirement. The other aspect of this requirement is the ability to replace system files outside of the update procedure (e.g., copying the files directly to the device, or attempting to use the interfaces provided for the update functionality to directly copy system files to the device). From FPT_TUD_EXT.1, the evaluator has information relating to the way the update functionality operates. The evaluator also examines the TSS to determine that it provides a listing of all unencrypted files on the USB Flash Drive, indicates their function, and further indicates which are system files on the USB flash drive, and where each of the files (if executable) runs (i.e., on the USB Flash Drive or on the host). The evaluator shall also determine that the TSS describes, for each file, whether or not the file can be updated/modified. If it can be updated/modified, the evaluator determines that the TSS describes the method by which the update/modification can be performed. The evaluator shall determine that the files that are not identified as system files are correctly categorized. The evaluator shall also perform the following test:

- Test 1: Using the information contained in the TSS, the evaluator shall attempt to overwrite or modify a system file such that the security properties of the USB flash drive are affected, or in a manner that would make the USB flash drive a vector for the automatic execution of malicious software (for instance, the ability to replace a Windows autorun.inf file contained in the device with a malicious version).
- Test 2: If the device is bootable (that is, if the system files supplied by the manufacturer allow the device to be used as a boot device), the evaluator shall attempt to overwrite this area with a custom boot file that, when booted, would allow them access to the user data; allow them to overwrite other system files; or allow them to infect the host with arbitrary code. If the device does not contain the appropriate system files to be bootable as delivered in the evaluated configuration, the evaluator shall attempt to place files on the device such that it can be used as a bootable device that is able to execute arbitrary code (that is, code of the evaluator's design and choosing) on the host.

Extended: USB Flash Device Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1	**Extended: USB Flash Device Trusted Update**
FPT_TUD_EXT.1.1	The TSF shall provide authorized users the ability to query the current version of the TOE firmware/software.
FPT_TUD_EXT.1.2	The TSF shall provide authorized users the ability to initiate updates to TOE firmware/software.
FPT_TUD_EXT.1.3	The TSF shall verify firmware/software updates to the TOE using a digital signature mechanism implemented on the USB flash drive prior to installing those updates.

Application Note:

157 The digital signature mechanism referenced in the third element is the one specified in FCS_COP.1(2).

Assurance Activities:

158 Updates to the USB flash drive are signed by an authorized source. The definition of an authorized source is contained in the TSS, along with a description of how the certificates used by the update verification mechanism are contained on the device. The evaluator ensures this information is contained in the TSS. The evaluator also ensures that the TSS (or the operational guidance) describes how the candidate updates are obtained; the processing associated with verifying the digital signature of the updates; and the actions that take place for successful (signature was verified) and unsuccessful (signature could not be verified) cases. The location of the software/firmware that is performing the processing must also be described in the TSS and verified by the evaluators. The evaluators shall perform the following tests:

- Test 1: The evaluator performs the version verification activity to determine the current version of the product. After the update tests described in the following tests, the evaluator performs this activity again to verify that the version correctly corresponds to that of the update.
- Test 2: The evaluator obtains a legitimate update using procedures describe in the operational guidance and verifies that it is successfully installed on the TOE. Perform a subset of other assurance activity tests to demonstrate that the update functions as expected.
- Test 3: The evaluator obtains or produces an illegitimate update, and attempts to install it on the TOE. The evaluator verifies that the TOE rejects the update.

Extended: TSF Testing (FPT_TST_EXT.1)

FPT_TST_EXT.1	**Extended: TSF Testing**
FPT_TST_EXT.1.1	The TSF shall run a suite of self tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

Assurance Activities:

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

160 The TSS shall describe the known-answer self-tests for all FCS_COP functions.

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

4.2 Rationale for Security Functional Requirements

162 This section describes the rationale for the TOE Security Functional Requirements as defined in Section 4.1. Table 8 illustrates the mapping from Security Functional Requirements to Security Objectives with a corresponding rationale that the objective is addressed by the requirement. This table should be augmented by the ST Author/vendor when they complete the selections and assignments for the requirements in Section 4.1, as well as (potentially) augment the baseline requirements with requirements from Appendix C.

Table 8: Rationale for TOE Security Functional Requirements

Objective	Requirement Addressing the Objective	Rationale
O.AUTHORIZATION The TOE must obtain the authorization factor(s) from a user to be able to encrypt and decrypt the data on the USB flash drive.	FIA_AUT_EXT.1 FCS_CKM.1(2)	FIA_AUT_EXT.1 requires that users must be authorized by the mechanisms specified in FCS_CKM.1(2) before they are allowed access to unencrypted data from the USB flash drive.
O.CORRECT_TSF_OPERATION The TOE shall provide the capability to verify correct operation of the TSF in the operational environment.	FPT_TST_EXT.1	FPT_TST_EXT.1 requires self-tests to be run on the TOE (both for the cryptographic functionality as well as for other components) prior to the TOE being put into operation, thus satisfying the objective.
O.ENCRYPT_ALL The TOE will encrypt all user data that is stored on the USB flash drive.	FDP_DSK_EXT.1 FCS_CKM.1(1) FCS_COP.1(1)	FDP_DSK_EXT.1 ensures the TOE performs encryption of USB flash drive, which includes all user data and appropriate key material. In addition to having a requirement that all the data is encrypted, FCS_CKM.1(1) and FCS_COP.1(1) specify the quality of the key used to perform the encryption, as well as the algorithm and key length that is used in the encryption operations.
O.DEK_SECURITY The TOE will mask the DEK using a key encryption key (KEK) created from one or more submasks (which in turn are derived from the authorization factors) so that a threat agent who does not have authorization factor(s) will be unable to gain access to the user data by obtaining the DEK.	FCS_CKM.1(2) FCS_CKM.1(3) FCS_RBG_EXT.1 FMT_MTD.1 FMT_SMF.1	FCS_CKM.1(2) is the requirement that specifies how the KEK will be derived, and specifies the key length of the KEK. This requirement mandates that the effective strength of each authorization factor is maintained. FCS_CKM.1(3) levies requirements on the password authorization factor so that it is conditioned such that the entropy inherent in the password is preserved

Objective	Requirement Addressing the Objective	Rationale
		and that the password is converted into a submask suitable to use in protecting the DEK. FCS_RBG_EXT.1 ensures that keying material is robustly generated. FMT_MTD.1 prevents the DEK in any form from being accessible outside of the device, preventing off-line attacks against the key. FMT_SMF.1 ensures the TSF provides the functions necessary to manage important aspects of the TOE. These include generating, protecting, and deleting a DEK, generating and configuring authorization factors, and configuring cryptographic functionality. The ST author may chose to incorporate other management functions if they chose.
O.OWNERSHIP The TOE shall ensure that ownership is taken (that is, a DEK is created, authorization factors are established, any default authorization factors are changed, a KEK is formed from the derived submasks, and the DEK is associated with the KEK) prior to any user data being stored on the TOE.	FMT_SMF.1	FMT_SMF.1 requires that a DEK be generated when the device is put into use, and that the DEK be protected with a KEK derived from submasks produced from the chosen authorization factors. Additionally, if there are default authorization factors, it requires that there be a mechanism that allows an authorized user to change these values. These requirements together achieve the objective.
O.KEY_MATERIAL_COMPROMISE The TOE shall ensure that no unencrypted/unmasked keys or keying material are written to persistent memory on the USB flash drive.	FDP_DSK_EXT.1 FMT_MTD.1	FDP_DSK_EXT.1 requires that user data be encrypted and no plaintext keying material is written in USB flash drive persistent memory. FMT_MTD.1 prevents the key material in any form from being accessible outside of the device, preventing off-line attacks against the key.
O.PROPOGATION_PREVENTION The TOE shall implement mechanisms to prevent the USB flash	FPT_SFP_EXT.1	FPT_SFP_EXT.1 requires that the TOE be incapable of allowing unauthorized users (or programs running on their behalf) to modify system files in such a way that

Objective	Requirement Addressing the Objective	Rationale
drive from being used as a mechanism for the automated spread of malicious software.		those system files could automatically transfer themselves from host-to-host, thus spreading copies of any malicious payload they might contain.
O.SAFE_AUTHFACTOR_VERIFICATION The TOE shall perform verification of the authorization factors in such a way that the KEK, DEK, or user data are not inadvertently exposed.	FIA_AUT_EXT.1	FIA_AUT_EXT.1 requires the TSF to verify the authorization factors prior to the user gaining access to the data on the USB Flash Drive. It also requires that this is performed in a manner that does not provide the attacker an advantage in guessing the DEK or KEK.
O.TRUSTED_UPDATE The TOE shall provide users the capability to update the TOE firmware/software, and verify that updates to the product are received from the intended source.	FCS_CKM.2 FCS_COP.1(2) FCS_COP.1(3) FPT_TUD_EXT.1	FPT_TUD_EXT.1 provides the required functionality by allowing users to check the version (thus allowing them to determine that an update is required); initiate the update process; and verify in a cryptographic manner that the update has not been tampered with, and that it comes from a trusted source (FCS_CKM.2, FCS_COP.1(2), FCS_COP.1(3)).

4.3 Security Assurance Requirements

163 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 4.1 are a formal instantiation of the Security Objectives.

164 As indicated in the introduction to Section 4.1, 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 section 4.1 as well as in this section.

165 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 5.1) 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 section 4.1.

166 The TOE security assurance requirements, summarized in Table 10, identify the management and evaluative activities required to address the threats identified in Section 2 of this PP. Section 4.4 provides a succinct justification for choosing the security assurance requirements in this section.

Table 5: TOE Security Assurance Requirements

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

4.3.1 Class ADV: Development

167 For TOEs conforming to this PP, 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 Section 4.1 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

4.3.1.1 ADV_FSP.1 Basic functional specification

168 The functional specification describes the TSFIs. It is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this PP will necessarily have interfaces to the Operational Environment that are not directly invokable by TOE users, 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 PP 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.

169 In understanding the interfaces to the TOE, it is important to consider that the primary threat that is to be countered is that the attacker finds a USB flash drive and attempts to probe the interface to the TOE in order to decrypt the data on the drive. Since the attacker only gets to manipulate the USB flash drive, the primary untrusted user interface is the one presented to the user when the USB flash drive is inserted into a host. In addition to these "user" interfaces, the operational interface (how the TOE is configured) also needs to be described. Another major interface is the firmware and USB middleware update interface. As described earlier, it is critical that any code or data that are replaced are properly signed and the signatures verified.

170 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:

171 There are no specific assurance activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in section 4.1, 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 the 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.

4.3.2 Class AGD: Guidance Documents

172 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 that (optionally) have responsibility for creating strong password authorization factors.

173 Guidance must include a description of how the authorized user verifies that the Operational Environment (the product that hosts the USB flash drive) can fulfill its role for the security functionality. The documentation should be in an informal style and readable by an authorized user.

174 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.

175 Guidance pertaining to particular security functionality is also provided; requirements on such guidance are contained in the assurance activities specified in section 4.1.

4.3.2.1 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:

176 Some of the contents of the operational guidance will be verified by the assurance activities in section 4.1. However, two additional warnings shall be provided in the guidance to users. The guidance shall warn authorized users that they must not let the USB flash drive leave their physical control while the USB flash drive is connected to the host and the host is powered on. Additionally, it shall state that authorized users shall not leave/store the password and/or host split authorization factors and/or the pin authorization factor with the USB flash drive or if multiple factors are used, with each other.

177 The following additional information is also required.

178 The documentation must describe the process for verifying that updates to the TOE come from the intended source (which in most cases will be the TOE vendor). This verification process is initiated by the authorized user but performed by the TSF on the USB flash drive. The evaluators shall verify that this process includes the following steps:

1. Instructions for obtaining the certificate that will be used by the FCS_COP.1(2) mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means and installed on the USB flash drive as part of its initial configuration. If not initially supplied on the USB flash drive, the guidance shall provide instructions on how to determine the obtained certificate can be trusted by the end user.
2. Instructions for obtaining the update itself. This should include instructions for making the update accessible to the USB flash drive (e.g., placement in a specific directory).
3. Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful.

4.3.2.2 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:

179 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.

180 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 user data on the USB flash drive is encrypted when setting up the product, and that this is the only allowed configuration for conformant TOEs.

- If the TOE supports host split authorization factors, the documentation shall describe the means by which the split that is to reside on the target host is installed in a manner that preserves the security of the TOE. The evaluators shall assess these procedures to determine that they are sound.

- 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.

4.3.3 Class ATE: Tests

181 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 PP, 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.

4.3.3.1 ATE_IND.1 Independent testing - Conformance

182 Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operation) documentation provided. The focus of the testing is to confirm that the requirements specified in section 5.1 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 PP.

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:

183 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 PP'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.

184 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. If all platforms claimed in the ST are tested, then no rationale is necessary.

185 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.

186 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.

4.3.4 Class AVA: Vulnerability assessment

187 For the first generation of this protection profile, the evaluation lab is expected to survey open sources to discover 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.

4.3.4.1 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:

188 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 USB flash drive 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, for example, a test would be suitable at the assurance level of this PP. If exploiting the vulnerability requires an electron microscope and liquid nitrogen, for instance, then a test would not be suitable and an appropriate justification would be formulated.

4.3.5 Class ALC: Life-cycle support

189 At the assurance level provided for TOEs conformant to this PP, 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.

4.3.5.1 ALC_CMC.1 Labeling of the TOE

190 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:

191 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.

4.3.5.2 ALC_CMS.1 TOE CM coverage

192 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.2.1D The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.2.1C The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.2.2C The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

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

Assurance Activity:

193 The "evaluation evidence required by the SARs" in this PP 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.

4.4 Rationale for Security Functional Requirements

194 The rationale for choosing these security assurance 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.

5 Conformance Claims

195 The Conformance Claim indicates the source of the collection of requirements that is met by a PP or a Security Target (ST) that passes its evaluation. Application notes are provided in the Security Functional Requirements (SFR) and Security Assurance Requirements (SAR) sections to further clarify specific requirements that must be met.

5.1 PP Conformance Claim

196 This PP is conformant to CC 3.1r3, CC Part 2 extended and CC Part 3 conformant.

197 STs that claim conformance to this PP shall meet a minimum standard of strict-PP conformance as defined in Section D3 of CC Part 1 (CCMB-2006-09-001).

198 Strict-PP conformance means the requirements in the PP are met and that the ST is an instantiation of the PP. The ST can be broader than the PP. The ST specifies that the TOE does at least the same as the PP, while the operational environment does at most the same as the PP. In this PP, application notes are provided to further clarify and explain the intent of the requirements specified and the expectation as to how the vendor will meet the requirements. It is expected that the evaluator of the ST will ensure strict-PP compliance by determining that the ST and its described TOE not only contain all the statements within this PP (and possibly more) but also met the expectations as stated by the application notes.

5.2 PP Conformance Claim rationale

199 This PP does not claim conformance to another PP.

200

Appendix A: Supporting Tables and References

[1] Common Criteria for Information Technology Security Evaluation, CCMB-2007-09, Version 3.1, September 2007.

[2] Draft Consistency Instruction Manual, for Basic Robustness Environments, Release 4.0, CC version 3.1, 2008

[3] Federal Information Processing Standard Publication (FIPS-PUB) 140-2, Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, May 25, 2001 (CHANGE NOTICES (12-03-2002)

[4] Federal Information Processing Standard Publication (FIPS-PUB) 180-2, Secure Hash Standard, August 1 2002

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

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

[7] NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, May 2004

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

[9] NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions, April 2008

[10] RFC 3394 Advanced Encryption Standard (AES) Key Wrap Algorithm, September 2002

[11] Universal Serial Bus Mass Storage Class Specification Overview, Version 1.2, June 23, 2003

Acronyms

AES	Advanced Encryption Standard
AF	Authorization factor
AS	Authorization subsystem
CAC	Common Access Card
CAVS	Cryptographic Algorithm Validation System
CC	Common Criteria
CM	Configuration management
COTS	Commercial Off-The-Shelf
CS	Configuration subsystem
DAR	Data-at-rest

DEK	Data Encryption Key
DRBG	Deterministic Random Bit Generator
DoD	Department of Defense
EAL	Evaluation Assurance Level
ES	Encryption Subsystem
FDE	Full Disk Encryption
FIPS	Federal Information Processing Standards
ISSE	Information System Security Engineers
IT	Information Technology
KEK	Key Encryption Key
MBR	Master Boot Record
OSP	Organization Security Policy
PIN	Personnel Identification Number
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
TOE	Target of Evaluation
USB	Universal Serial Bus

Appendix B: NIST SP 800-53/CNSS 1253 Mapping

Several of the NIST SP 800-53/CNSS 1253 controls are either fully or partially addressed by compliant TOEs. This section outlines the requirements that are addressed, and can be used by certification personnel to determine what, if any, additional testing is required when the TOE is incorporated into its operational configuration.

Application Note: In this version, only a simple mapping is provided. In future versions, additional narrative will be included that will provide further information for the certification team. This additional information will include details regarding the SFR to control mapping discussing what degree of compliance is provided by the TOE (e.g., fully satisfies the control, partially satisfies the control). In addition, a comprehensive review of the specified assurance activities, and those evaluation activities that occur as part of satisfying the SARs will be summarized to provide the certification team information regarding how compliance was determined (e.g., document review, vendor assertion, degree of testing/verification). This information will indicate to the certification team what, if any, additional activities they need to perform to determine the degree of compliance to specified controls.

Since the ST will make choices as far as selections, and will be filling in assignments, a final story cannot necessarily be made until the ST is complete and evaluated. Therefore, this information should be included in the ST in addition to the PP. Additionally, there may be some necessary interpretation (e.g.,"modification") to the activities performed by the evaluator based on a specific implementation. The scheme could have the oversight personnel (e.g., Validators) fill in this type of information, or could have this done by the evaluator as part of the assurance activities. The verification activities are a critical piece of information that must be provided so the certification team can determine what, if anything, they need to do in addition to the work of the evaluation team.

Identifier	Name	Applicable SFRs
CM-2	Baseline Configuration	FPT_SFP_EXT.1
CM-5	Access Restrictions for Change	FPT_TUD_EXT.1
IA-5	Authenticator Management	FCS_CKM.1(3), FIA_AUT_EXT.1, FMT_SMF.1
IA-7	Cryptographic Module Authentication	FIA_AUT_EXT.1
MP-4	Media Storage	FDP_DSK_EXT.1
MP-5	Media Transport	FDP_DSK_EXT.1
SA-7	User Installed Software	FPT_SFP_EXT.1
SC-12	Cryptographic Key Establishment and Management	FCS_CKM.1(1), FCS_CKM.1(2), FCS_CKM.2, FMT_SMF.1
SC-13	Use of Cryptography	FCS_CKM.1(3), FCS_COP.1(1), FCS_COP.1(2), FCS_COP.1(3), FCS_COP.1(4), FCS_RBG_EXT.1, FMT_SMF.1
SC-28	Protection of Information at Rest	FDP_DSK_EXT.1
SI-6	Security Functionality Verification	FPT_TST_EXT.1
SI-7	Software and Information Integrity	FPT_SFP_EXT.1

Appendix C: Additional Requirements

201 For this draft of the PP, this appendix just contains additional components without supporting threats, objectives, rationale, or assurance activities (although some guidance is given for selected components). In tandem with the current review cycle, this supporting information will be developed and incorporated into the next release of the PP. Comments on the information contained in this section (both on whether the requirements contained are applicable to the potential conformant TOEs as well as requirements that are not contained in this appendix that are widely applicable USB flash drive encryption products) are welcome and solicited.

202 As indicated in the introduction to this PP, there are several capabilities that a TOE may implement and still be conformant to this PP. These capabilities are not required, creating a dependency on the Operational Environment (for instance, identification and authentication of administrators of the TOE). However, if a TOE does implement such capabilities, the ST author will take the following information and include it in their ST. Requirements not contained in this appendix may be included in the ST, but are subject to review and acceptance by the National Scheme overseeing the evaluation before a conformance claim to this PP can be made.

C.1 FCS_RBG_EXT Supporting Requirements

203 Several selections in the baseline requirements for FCS_RBG_EXT reference standards will require that the TOE implement additional cryptographic functionality over and above what is specified in the main body of the PP. As the ST is being created, if the ST author chooses a selection that references such a standard, this section will contain the additional SFRs and associated assurance activities that will be needed in the main body of the ST.

C.1.1 Block Cipher Function

204 A block cipher function is used for implementing the NIST SP 800-90 CTR_DRBG function. It should be noted that since the RBG function is required to be implemented on the USB flash drive, the mechanism satisfying this requirement must be implemented on the USB flash drive.

Cryptographic Operation (FCS_COP)

FCS_COP.1(1)	**Cryptographic operation (Data Encryption)**
FCS_COP.1.1(1)	**Refinement:** The TSF shall perform **data encryption** in accordance with a specified cryptographic algorithm **AES used in CTR mode** and cryptographic key sizes **[selection: 128 bits, 256 bits]** that meet the following: **FIPS PUB 197, "Advanced Encryption Standard (AES)" and NIST SP 800-38A.**

Application Notes

205 For the selection, the key size to be used is chosen to be consistent with the recommendations in NIST SP 800-90.

Assurance Activities:

206 The CTR mode tests reference *The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)* [AESAVS], available from http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf.

207 The evaluator examines the TSS to confirm that it describes how the counter values for this mode are derived/obtained, and ensures that the described implementation satisfies the required property that each counter value is associated with only one plaintext block that is ever encrypted with a given key.

208 The evaluators shall run a set of known answer tests for each key size supported by the TSF. Inputs are a key, IV, and either plaintext to be encrypted or ciphertext to be decrypted. All of the test vectors (both encrypt and decrypt) for CTR mode in the supported key lengths from http://csrc.nist.gov/groups/STM/cavp/documents/aes/KAT_AES.zip shall be used to perform these tests

209 The evaluators shall perform a multi-block message test for each key length supported. To perform this test, the evaluators generated 10 data sets for encryption and 10 data sets for decryption. Each data set consists of key, an IV, and plaintext (for encryption) or ciphertext (for decryption). The length of a block shall be 128 bits; the length of the plaintext/ ciphertext shall be block length * i, where i indicates the data set number and i ranges from 1 to 10 (so messages will range from 128 bits to 1280 bits).

210 The evaluators shall perform a Monte Carlo test. The evaluators shall generate 10 sets of starting values for encryption (values for the key, IV, and plaintext) and 10 sets of starting values for decryption (values for the key, IV, and ciphertext). The length of the plaintext/ciphertext shall be 128 bits. Each set of starting values is used to generate and perform 100 tests; the algorithm for generating the 100 test values (per set of starting values) is contained in section 6.4.1 of [AESAVS].[1]

C.1.2 Hash Function

211 Hash functions are required for Hash_DRBG, HMAC_DRBG, and Dual_EC_DRBG, as well as for the PRF in NIST SP 800-132. It should be noted that since the RBG function is required to be implemented on the USB flash drive, the mechanism satisfying this requirement must be implemented on the USB flash drive as well.

FCS_COP.1(X2)	**Cryptographic operation (Cryptographic Hashing)**
FCS_COP.1.1(X2)	**Refinement:** The TSF shall perform **cryptographic hashing services** in accordance with **[selection: SHA-1, SHA 256, SHA 384, SHA 512]** and **message digest sizes [selection: 160, 256, 384, and 512] bits** that meet the following: **FIPS Pub 180-2, "Secure Hash Standard".**

Application Note:

212 The intent of this requirement is to specify the hashing function. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength

[1] CTR mode uses the ECB Monte Carlo Algorithm.

of the algorithm used for FCS_COP.1(1) and FCS_COP.1(2) (SHA 256 for 128-bit keys, SHA 512 for 256-bit keys). In subsequent publications of this PP, it is likely that SHA-1 will no longer be an approved algorithm for cryptographic hashing.

Assurance Activities:

213 The evaluator shall check the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present.

214 The evaluator shall perform the following tests. These cryptographic hashing tests reference *The Secure Hash Algorithm Validation System (SHAVS)* [SHAVS], available from http://csrc.nist.gov/groups/STM/cavp/documents/shs/SHAVS.pdf.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

219 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is $512 + 99*i$, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

220 The evaluators devise an input set consisting of $m/8$ messages, where m is the block length of the hash algorithm. The length of the ith message is $512 + 8*99*i$, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

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

C.1.3 HMAC Function

222 The HMAC function is used for implementing the NIST SP 800-90 HMAC_DRBG function as well as the PRF in NIST SP 800-132. Note that it also requires the use of a SHA function, so if this requirement is used in the ST, then the hashing requirement in C.1.2 must be included as well, with the appropriate selections. It should be noted that since the RBG function is required to be implemented on the USB flash drive, the mechanism satisfying this requirement must be implemented on the USB flash drive. It is expected that just one key-length/hash function/block size/output MAC length is used. If any of these parameters can be configured, then this requirement should be iterated in the ST to reflect this.

FCS_COP.1	**Cryptographic operation (Keyed Cryptographic Hashing)**
FCS_COP.1.1	**Refinement:** The TSF shall perform **keyed cryptographic hashing services** in accordance with **[The Keyed-Hash Message Authentication Code]** and cryptographic key size **[selection: 128 bits, 256 bits]** that meet the following: **FIPS 198-1.**

Application Note:

223 The selection in this requirement must be consistent with the key size specified for the size of the DEK.

Assurance Activities:

224 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.

225 The evaluator shall also perform a Random Message Test, referenced in *The Keyed-Hash Message Authentication Code Validation System (HMACVS)* [HMACVS] available from http://csrc.nist.gov/groups/STM/cavp/documents/mac/HMACVS.pdf.

226 For the test, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall ensure that the HMAC produced by the TSF agrees with the expected value.

C.2 Passphrase Authorization

227 As a baseline, the TOE is required to support password authorization factors that can be at least 32 characters long. These factors have less entropy than the key that they are protecting, and so it is desirable to use longer factors to protect these keys. Vendors may support significantly strong authorization factors, which will be referred to as *passphrases* to reflect their more complex nature. If passphrases are supported, the ST author should take the following actions:

228 1) Change all instances of “password” in the objective description to “passphrase.”

229 2) Replace FCS_CKM.1(3) and the associated application notes and assurance activities with the following:

FCS_CKM.1(3)	**Cryptographic key generation (Passphrase conditioning)**
FCS_CKM.1.1(3)	**Refinement: A passphrase used to generate a submask shall contain up to [assignment: positive integer of 9 or more] words of maximum length [assignment: positive integer of 8 or more] characters in the set of {upper case characters, lower case characters**, **and** [assignment: *other supported characters*]} **and shall be conditioned [selection:** • **using [selection: SHA-1, SHA-256, SHA-512] for 128-bit DEKs;** • **using [selection: SHA-256, SHA-512] for 256-bit DEKs;** • **using NIST SP 800-132 with a salt generated using a Random Bit Generator as specified in FCS_RBG_EXT.1, an iteration count of [assignment: *number greater than or equal to 1000*], and HMAC using [selection: SHA-1, SHA-256, SHA-512];** **] such that the output of the conditioning function is equal to the size (in number of bits) of the DEK.**
Application Note:	A passphrase is a sequence of words taken from a dictionary of words in a random fashion such that they provide necessary entropy to generate the submask derived from the passphrase. This requirement places requirements on the composition of the passphrase and does not require a particular method of choosing the words from the dictionary (although this is typically done in a cryptographically random fashion). The string that results consists of a sequence of characters encoded in a scheme determined by the underlying OS. This sequence 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. If 800-132 conditioning is specified, then the ST author will fill in the number of iterations (C) that are performed; this value must be at least 10000. 800-132 also 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 from Appendix C. It should be noted that the hash function required by the key derivation function (or any cryptographic operations required by the conditioning specified in 800-132) must be implemented by USB flash drive. In subsequent publications of this PP, it is likely that SHA-1 will no longer be an approved algorithm for cryptographic hashing, and that conditioning using SP 800-132 will be required.

Assurance Activity: There are two aspects of this component that require evaluation: passphrases that have at least 9 words chosen from a dictionary of words that are from 1 to 8 characters are supported, and the characters that are input are subject to the selected conditioning function. These activities are separately addressed in the text below.

Support for passphrase lengths of at least 9 words of up to at least 8 characters

The evaluators shall check the TSS section to determine that it specifies that a capability exists to accept passphrases with the maximum number of words of lengths specified in the ST in this assignment statement, and that the numbers specified are at least those indicated in the requirement. The evaluators shall also check the operational guidance to determine that there are instructions for administrators generating such passphrases, and that guidance indicates how the passphrases are entered into the TOE.

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 passphrases having at least 9 (or the number specified in the ST for the first assignment, whichever is greater) words. This test should also verify that words up to the number specified in the second assignment (or 8, whichever is greater) are supported.
- Test 2: Ensure that the TOE supports passphrases of shorter lengths, consistent with what is specified in the operational guidance supplied by the vendor (for instance, if the guidance specifies that passphrases have a minimum length of 5 words, this test would minimally determine that 5-word passphrases were accepted by the TOE).
- Test 3 [conditional]: If the ST author has filled in additional supporting characters in the 3rd assignment, ensure that the TOE contains support for passphrases composed as specified in guidance contained in the AGD_OPR or AGD_PRE guidance with respect to the specified special characters. For instance, if the guidance specifies that passphrases must contain a special character, this test would fail if the TOE only supported letters and numbers.

Passphrase Conditioning

For SHA-based conditioning of the passphrase, the evaluator performs the following activities. The evaluator shall check that the TSS describes the method by which the 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 in FCS_COP.1(3) 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 described in FCS_CKM.1(2), and is the same length as the DEK as specified in FCS_CKM.1(1).

For 800-132-based conditioning of the passphrase, the required assurance activities will be performed when doing the assurance activities for the appropriate Appendix C requirements. If any manipulation of the master key is performed in forming the submask that will be used to form the KEK, that process shall be described in the TSS.

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

C.3 Authorization Factor/Submask Generation

230 The TOE is not required to generate host split authorization factors or pin-protected submasks. However, if a TOE does offer this service then the following components need to be included in the ST for the TOE to claim credit for this capability. Requirements for other types of authorization factors/submasks not contained in this appendix are subject to review and acceptance by the National Scheme overseeing the evaluation before a conformance claim to this PP can be made.

231 The TOE must support a password authorization factor, either by itself or used in combination with another authorization factor; TOE supported authorization factors are specified in FCS_CKM.1(2). FCS_CKM.1(2) also specifies how the various submasks derived from the authorization factors are combined to form the KEK. If either the host split authorization factor or pin-protected submask are used, the appropriate requirement below must be included in the ST in order to be conformant with this PP. Host split authorization factors and pin-protected submasks must be generated by the TOE. Note that the nature of the host split authorization factor is that the split that eventually is stored on the host must be put there through external means (other than on the host supporting the TOE at the time the host split is generated).

C.3.1 Host Split Authorization Factor Generation

232 Host split authorization factors are identical to host split submasks; it is a random bit string that is stored on a host platform. The component below includes the requirements for the generation and storage of such authorization factors/submasks.

FCS_CKM_EXT.1(X1)	**Cryptographic key generation (Host Split Authorization Factor)**
FCS_CKM_EXT.1.1(X1)	The TSF shall generate a host split authorization factor using the Random Bit Generator specified in FCS_RBG_EXT.1 that produces an authorization factor of [selection: 128 bits, 256 bits] that was seeded with entropy at least equal to the size of the DEK, as specified in FCS_CKM.1(1).
FCS_CKM_EXT.1.2(X1)	The TSF shall be able to store the generated host split authorization factor on generating host.

FCS_CKM_EXT.1.3(X1)	The TSF shall clear the host split authorization factor on the USB flash drive once the host split authorization factor is stored on the generating host.

Application Note:

233 The selection should indicate an identical number of bits as specified for the DEK in FCS_CKM.1(1). The "generating host" is the host to which the USB flash drive is connected at the time the host split authorization factor is generated. The Random Bit Generator is the one on the TOE (FCS_RBG_EXT).

Assurance Activity:

234 The evaluator reviews the guidance documentation to confirm that the steps necessary for an administrator to generate the host split authorization factor are described. The evaluator reviews the TSS portion of the ST to confirm that the authorization factor generation process is described, including how the generation function uses the RBG, and how the RBG function is seeded (if this is different from other uses of the RBG). The evaluator reviews the TSS section (or administrative guidance documentation) to determine how the value generated by the RBG is established on the generating host (procedures for establishing it on other hosts are covered in the AGD_PRE assurance activities). Finally, the evaluator confirms that the TSS describes how the at what point and by what means the host split on the USB device is cleared after the split has been stored on the generating host.

235 The following tests must be performed by the evaluator:

- Test 1: Following the administrative guidance, create a host split authorization factor. If possible, confirm the # of bits the authorization factor contains. Ensure that this authorization factor can be used to access the encrypted TOE.

C.3.2 PIN-Protected Submasks

236 A PIN-protected submask is a submask that is random bit strings that is stored in a protected portion of the TOE persistent memory and are accessed by a PIN authorization factor. The component below includes the requirements for the generation and storage of such submasks. If the TOE allows for the PIN to be changed after it is initially established, the ST author will have to specify this capability in FMT_SMF.1(d). Additionally, they will need to specify requirements for establishing the authorization for the change, either by verifying the password/passphrase authorization factor (as is done for password authorization factor changes in FIA_AUT_EXT.1.3) or the existing PIN.

FCS_CKM_EXT.1(X2)	**Cryptographic key generation (TOE Stored Submasks)**
FCS_CKM_EXT.1.1(X2)	The TSF shall derive a pin-protected submask generated by a Random Bit Generator specified in FCS_RBG_EXT.1 that produces a submask of [selection: 128 bits, 256 bits] that was seeded with entropy at least equal to the size of the DEK, as specified in FCS_CKM.1(1).

FCS_CKM_EXT.1.2(X2) The TSF shall be able to store the generated submask within the USB flash drive, protected [assignment: *means by which the submask is protected*] and accessed by entering a PIN ranging from [assignment: *minimum number of digits*] to [assignment: *maximum number of digits, must be 20 or greater*] digits.

Application Note:

237 The selection for the first element should indicate an identical number of bits as specified for the DEK in FCS_CKM.1(1). In the second element, the first assignment should be filled in with the method used to protect the authorization factor, including any operations done with the PIN. The PIN-protected submask is an exception to FDP_DSK_EXT.1.4 (which requires that no unencrypted keying material is written to persistent memory), but must be the case that until the submask is written to the protected memory on the USB Flash Drive, it cannot be written to other persistent memory. The second and third assignments specify the minimum and maximum sizes (respectively) for the PIN. Conformant TOEs must be able to support PINs of at least 20 digits. The Random Bit Generator is the one on the USB Flash Drive (FCS_RBG_EXT).

Assurance Activity:

238 The evaluator reviews the guidance documentation to confirm that the steps necessary for an administrator to generate the pin-protected submask are described. The evaluator reviews the TSS portion of the ST to confirm that the authorization factor generation process is described, including how the generation function uses the RBG, and how the RBG function is seeded (if this is different from other uses of the RBG). The evaluator reviews the TSS section (or administrative guidance documentation) to determine how the value generated by the RBG is established on the generating TOE. This would include how the PIN is established and the details of how the submask is not written to persistent memory prior to be written to the protected portion of the USB flash drive, and how the submask is protected (once written) on the USB flash drive; note that any cryptography used shall be specified (if not already) in the ST using appropriate FCS_COP requirements. The interface for collecting the PIN shall be described.

239 The following tests must be performed by the evaluator:

- Test 1: Following the operational guidance, create a PIN-protected submask. If possible, confirm the # of bits the submask contains. Establish a PIN, and ensure that this PIN provides access to the submask that can be used to access the encrypted TOE. Ensure that an incorrect PIN does not provide access to the data.
- Test 2: Attempt to establish a PIN that is less than the minimum PIN length specified in the requirement (if that PIN length is greater than 1) and a PIN that is greater than the maximum length. Ensure that the PINs are not established, and do not allow access to the encrypted data.
- Test 3: Establish PINs that are exactly the minimum value specified in the assignment and the maximum value specific in the assignment. Ensure that these PINs are successfully established and allow access to the encrypted data.

C.5 PIN Authorization Factor Entry Failure Handling

240 If the TSF includes a PIN-protected submask, then the PIN authorization factor generally has much less strength (in terms of the number of bits/digits) than the submask it protects. To discourage PIN-guessing attacks, a limit can be imposed on the number of consecutive failed guesses that locks the USB device from the attacker. The following requirement should be included in the ST where this functionality is required.

241 If this component is included, the following objectives and rationale should be added to the ST.

O.ANTI_HAMMER	The TSF will implement mechanisms to mitigate brute force PIN-guessing attempts on PIN-protected submasks.

T.KEYSPACE_EXHAUST An unauthorized user may attempt a brute force attack to determine cryptographic keys or authorization factors to gain unauthorized access to data or TOE resources.	O.ANTI_HAMMER The TSF will implement mechanisms to mitigate brute force PIN-guessing attempts on PIN-protected submasks.	Implementing a mechanism that limits the effectiveness of brute force attacks on PIN-protected submasks, as called for in O.ANTI_HAMMER, will ensure that keyspace exhaustion attacks are not likely to succeed.

O.ANTI_HAMMER The TOE shall provide the capability to verify correct operation of the TSF in the operational environment.	FIA_AFL_EXT.1	FIA_AFL_EXT.1 requires a limit on the number of consecutive PIN authorization factor entry failures before locking the USB device. If the limit is small enough, it will effectively limit the number of guesses, rendering brute force attacks ineffective.

Extended: Authorization Failure Handling (FIA_AFL)

FIA_AFL_EXT.1	**Authorization Failure Handling**
FIA_AFL_EXT.1	The TSF shall zeroize the DEK when [assignment: *number of incorrect PIN entries*] consecutive unsuccessful PIN entry attempts occur.

Application Note:

242 To mitigate the threat that the relatively (to the size of the submask) short PIN is subject to a brute force attack, this component requires that the DEK be zeroized (which will result in the loss of all data on the device at that time) after the specified number of authorization failures. This number is assigned by the ST author.

Assurance Activity:

243 The evaluator shall examine the TSS to determine that it describes the method by which unsuccessful authorization attempts are detected, and how consecutive failures are tracked. This includes failures that take place after the device has been disconnected from the host, then reconnected. The evaluator shall also perform the following test:

- Test 1: The evaluator shall verify that when the indicated number of consecutive authorization failures takes place, the data on the device are no longer accessible even when the correct PIN is subsequently entered. The evaluator shall ensure that at least one test is performed when the USB flash drive is continuously plugged into the host, and one test is performed where the device is unplugged, then plugged back in.

Appendix D: Document Conventions

244 Except for replacing United Kingdom spelling with American spelling, the notation, formatting, and conventions used in this PP are consistent with version 3.1 of the Common Criteria (CC). Selected presentation choices are discussed here to aid the PP reader.

245 The notation, formatting, and conventions used in this PP are largely consistent with those used in version 3.1 of the Common Criteria (CC). Selected presentation choices are discussed here to aid the PP user. The CC allows several operations to be performed on functional and assurance requirements; *refinement*, *selection*, *assignment*, and *iteration* are defined in Appendix C4 of Part 1 of the CC 3.1. Each of these operations is used in this PP.

Refinement Convention

246 The **refinement** operation is used to add detail to a requirement, and thus further restricts a requirement. Refinement of security requirements is denoted by the word "Refinement" in **bold text** after the element number and the additional text in the requirement in bold text.

Selection Convention

247 The **selection** operation is used to select one or more options provided by the CC in stating a requirement (see appendix C.4.3 Part 1, CC 3.1). Selections that have been made by the PP authors show the selection in **bold** characters, the brackets and the word "selection" removed. Selections to be filled in by the ST author are shown in square brackets with an indication that a selection is to be made, [selection:].

Assignment Convention

248 The **assignment** operation is used to assign a specific value to an unspecified parameter, such as the length of a password (see appendix C.4.2 Part 1, CC 3.1). Showing the value in **bold** characters denotes assignments that have been made by the PP authors, the brackets and the word "assignment" are removed. Assignments to be filled in by the ST author are shown in square brackets with an indication that an assignment is to be made [assignment:].

Iteration Convention

249 The **iteration** operation is used when a component is repeated with varying operations (see appendix C.4.1 Part 1, CC 3.1). The iteration number (iteration_number) is show in parenthesis following the component identifier.

250 The iteration operation may be performed on every component. The PP/ST author performs an iteration operation by including multiple requirements based on the same component. Each iteration of a component shall be different from all other iterations of that component, which is realized by completing assignments and selections in a different way, or by applying refinements to it in a different way.

Extended Requirement Convention

251 Extended requirements are permitted if the CC does not offer suitable requirements to meet the authors' needs. **Extended requirements** must be identified and are required to use the CC class/family/component model in articulating the requirements. Extended requirements will be indicated with the "EXT" inserted within the component.

Application Notes

252 Application notes contain additional supporting information that is considered relevant or useful for the construction of security targets for conformant TOEs, as well as general information for developers, evaluators, and ISSEs. Application notes also contain advice relating to the permitted operations of the component.

Assurance Activities

253 Assurance activities serve as a Common Evaluation Methodology for the functional requirements levied on the TOE to mitigate the threat. The activities include instructions for evaluators to analyze specific aspects of the TOE as documented in the TSS, thus levying implicit requirements on the ST author to include this information in the TSS section. In this version of the PP these activities are directly associated with the functional and assurance components, although future versions may move these requirements to a separate appendix or document.

Appendix E: Glossary of Terms

Administrator – a user that has administrative privilege to configure or update the TOE.

Authorization factor (AF) – a value submitted by the user or present on the host used to establish that the user (and potentially the host) is in the community authorized to use the USB flash drive. The authorization factors produce submasks that are used to form the KEK. In some cases the authorization factor is identical to the submasks. 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 a USB flash drive.

Data Encryption Key (DEK) – the key that is used by the encryption algorithm to encrypt the data on the USB flash drive.

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.

Host split authorization factor– an authorization factor that is also a submask; it created by an RNG (optionally on the USB flash drive) that is distributed to and stored on the source and destination hosts.

Key Encryption Key (KEK) – the key that is used to mask the DEK. It is comprised of submasks that are dervied from at least one authorization factor.

Keying material – the KEK, DEK, submasks, 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 to some extent for authorization to the data on the device.

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

Persistent memory – data storage that retains the data when power is turned off.

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.

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 – the extraction or software extraction of the USB flash drive or power down or unintentional loss of power of the host platform.

Software extraction – the process of using the operational environment to remove power from or un-mount the USB flash drive.

Submask – the value derived from an authorization factor that is used to form the KEK.

System files – Files that reside on the USB flash drive that are used in the operation of the device. This includes the software that initially loads into the host environment when the device is inserted into the host. This does not include software that is installed in the operational environment from different media (a driver off of an accompanying CD-ROM, for instance) in preparation for using the device.

TOE (Target of Evaluation) – refers to a product or set of products that fulfill the requirements to encrypt data on a USB flash drive. This includes the USB hardware itself, all firmware implementing the security functions of the device that resides on the USB, and any software used to operate the device (system files).

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 USB flash drive with appropriate IT security commensurate with the value of the user data protected by the USB flash drive.

Unauthorized User – a user who has not been authorized (by possession of the correct authorization factors) to use the TOE.

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.

Appendix F: PP Identification

Tile:	Security Requirements for USB Flash Drives
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
Keywords:	authorization factor, authorization subsystem, DEK, data encryption, encryption subsystem, entropy, KEK, noise source