Microsoft Windows

Common Criteria Evaluation

Microsoft Windows 8

Microsoft Windows Server 2012

Full Disk Encryption Security Target

Document Information	
Version Number	1.0
Updated On	April 3, 2014

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The example companies, organizations, products, people and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred.

Microsoft, Active Directory, Visual Basic, Visual Studio, Windows, the Windows logo, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

TABLE OF CONTENTS

LIST OF TABLES

1 Security Target Introduction

This section presents the following information required for a Common Criteria (CC) evaluation:

- Identifies the Security Target (ST) and the Target of Evaluation (TOE)
- Specifies the security target conventions and conformance claims
- Describes the organization of the security target

1.1 Security Target, TOE, and Common Criteria (CC) Identification

ST Title: Full Disk Encryption Security Target

ST Version: version 1.0; April 3, 2014

TOE Software Identification: The following Windows Operating Systems (OS):

- Microsoft Windows 8 Pro Edition (32-bit and 64-bit versions)
- Microsoft Windows 8 Enterprise Edition (32-bit and 64-bit versions)
- Microsoft Windows Server 2012 Standard Edition
- Microsoft Windows Server 2012 Datacenter Edition

The following security updates and patches must be applied to the above Windows 8 products:

- All critical security updates published as of June 2013.

The following security updates must be applied to the above Windows Server 2012 products:

- All critical security updates published as of June 2013.

TOE Hardware Identification: The following real and virtualized hardware platforms and components are included in the evaluated configuration:

- Microsoft Surface Pro, Intel Core i5, 64-bit
- Dell Optiplex 755, 3.0 GHz Intel Core 2 Duo E8400, 64-bit
- Dell Precision 690, 1.995 GHz Xeon Family 6 Model 15 Stepping 6, 64-bit

TOE Guidance Identification: The following administrator, user, and configuration guides were evaluated as part of the TOE:

- *Microsoft Windows 8, Microsoft Windows Server 2012 Common Criteria Supplemental Admin Guidance for Software Full Disk Encryption*

Evaluation Assurance: As specified in section and specific Assurance Activities defined in the protection profile associated with the security functional requirements from section .

CC Identification: CC for Information Technology (IT) Security Evaluation, Version 3.1, Revision 4, September 2012.

1.2 CC Conformance Claims

This TOE and ST are consistent with the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 3.1, Revision 4, September 2012, extended (Part 2 extended)
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements Version 3.1, Revision 4, September 2012, conformant (Part 3 conformant)
- Software Full Disk Encryption Protection Profile, Version 1.1, March 31, 2014
- Evaluation Assurance Activities specified in section 5.2 and CC Part 3 assurance requirements specified in section 5.1.

1.3 Conventions, Terminology, Acronyms

This section specifies the formatting information used in the security target.

1.3.1 Conventions

The following conventions have been applied in this document:

Security Functional Requirements (SFRs): Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.

- Iteration: allows a component to be used more than once with varying operations.
- Assignment: allows the specification of an identified parameter.
- Selection: allows the specification of one or more elements from a list.
- Refinement: allows the addition of details.

The conventions for the assignment, selection, refinement, and iteration operations are described in Section 5.

Other sections of the security target use a bold font to highlight text of special interest, such as captions.

1.3.2 Terminology

The following terminology is used in the security target:

Term	Definition
Access	Interaction between an entity and an object that results in the flow or modification of data.
Access control	Security service that controls the use of resources[1] and the disclosure and modification of data[2].
Administrator	An authorized user who has been specifically granted the authority to manage some portion or the entire TOE and thus whose actions may affect the TOE Security Policy (TSP). Administrators may possess special privileges that provide capabilities to override portions of the TSP.

[1] Hardware and software
[2] Stored or communicated

Assurance	A measure of confidence that the security features of an IT system are sufficient to enforce the IT system's security policy.
Attack	An intentional act attempting to violate the security policy of an IT system.
Authentication	A security measure that verifies a claimed identity.
Authentication data	The information used to verify a claimed identity.
Authorization	Permission, granted by an entity authorized to do so, to perform functions and access data.
Authorization factor	Data provided by an entity to unlock a data volume protected by BitLocker.
Authorized user	An authenticated user who may, in accordance with the TOE Security Policy, perform an operation.
BitLocker	Windows BitLocker Drive Encryption is a security feature that provides data protection for a computer by encrypting all data stored on the Windows disk volume, especially the operating system volume.
BitLocker To Go	BitLocker To Go is the portion of BitLocker that encrypts removable storage devices.
Compromise	Violation of a security policy.
Confidentiality	A security policy pertaining to disclosure of data.
Critical cryptographic security parameters	Security-related information appearing in plaintext or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.
Cryptographic boundary	An explicitly defined contiguous perimeter that establishes the physical bounds (for hardware) or logical bounds (for software) of a cryptographic module.
Cryptographic key	A parameter used in conjunction with a cryptographic algorithm that produces at least one of the following: • the transformation of plaintext data into ciphertext data • the transformation of ciphertext data into plaintext data • a digital signature computed from data • the verification of a digital signature computed from data • a data authentication code computed from data
Cryptographic module	The set of hardware, software, and/or firmware that implements approved security functions, including cryptographic algorithms and key generation, which is contained within the cryptographic boundary.
Cryptographic module security policy	A precise specification of the security rules under which a cryptographic module must operate.
Defense-in-depth	A security design strategy whereby layers of protection are utilized to establish an adequate security posture for an IT system.
Edition	A distinct variation of a Windows OS version. Examples of editions are Windows Server 2012 [Standard] and Windows Server 2012 Datacenter.
Entity	A subject, object, user or external IT device.
Full Volume Encryption Key (FVEK)	The key used to encrypt/decrypt the sectors on a volume.
General-Purpose Operating System	A general-purpose operating system is designed to meet a variety of goals, including protection between users and applications, fast response time

	for interactive applications, high throughput for server applications, and high overall resource utilization.
Identity	A means of uniquely identifying an authorized user of the TOE.
Intermediate Key (IK)	A randomly generated 256-bit key which is used to protect the Volume Master Key when using public key based protectors.
Key Encrypting Key (KEK)	The key used to protect the FVEK.
Object	An entity under the control of the TOE that contains or receives information and upon which subjects perform operations.
Operating environment	The total environment in which a TOE operates. It includes the physical facility and any physical, procedural, administrative and personnel controls.
Persistent storage	All types of data storage media that maintain data across system boots (e.g., hard disk, removable media).
Resource	A fundamental element in an IT system (e.g., processing time, disk space, and memory) that may be used to create the abstractions of subjects and objects.
Secure State	Condition in which all TOE security policies are enforced.
Security attributes	TOE Security Function (TSF) data associated with subjects, objects and users that is used for the enforcement of the TOE Security Policy (TSP).
Security-enforcing	A term used to indicate that the entity (e.g., module, interface, subsystem) is related to the enforcement of the TOE security policies.
Security-supporting	A term used to indicate that the entity (e.g., module, interface, subsystem) is not security-enforcing; however, the entity's implementation must still preserve the security of the TSF.
Security Target (ST)	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Subject	An active entity within the TOE Scope of Control (TSC) that causes operations to be performed. Subjects can come in two forms: trusted and untrusted. Trusted subjects are exempt from part or all of the TOE security policies. Untrusted subjects are bound by all TOE security policies.
Target of Evaluation (TOE)	An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
Threat	Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.
Unauthorized individual	A type of threat agent in which individuals who have not been granted access to the TOE attempt to gain access to information or functions provided by the TOE.
Unauthorized user	A type of threat agent in which individuals who are registered and have been explicitly granted access to the TOE may attempt to access information or functions that they are not permitted to access.
Version	A Version refers to a release level of the Windows operating system. Windows 7 and Windows 8 are different versions.
Volume Master Key (VMK)	The key used to protect the Full Volume Encryption Key.
Vulnerability	A weakness that can be exploited to violate the TOE security policy.

1.3.3 Acronyms

The acronyms used in this security target are specified in **Appendix A: List of Abbreviations**.

1.4 ST Overview and Organization

The Windows 8 and Windows Server 2012 Full Disk Encryption security target contains the following additional sections:

- TOE Description (Section 2): Provides an overview of the TSF and boundary.
- Security Problem Definition (Section 3): Describes the threats, organizational security policies and assumptions that pertain to the TOE.

1.5 Secure Usage Assumptions

describes the core security assumptions of the environment in which Windows 8 and Windows Server 2012 is intended to be used. It includes information about the physical, personnel, procedural, and connectivity aspects of the environment.

The following specific conditions are assumed to exist in an environment where the TOE is employed in order to conform to the Software Full Disk Encryption Protection Profile:

Table 3-3 Secure Usage Assumptions

Assumption	Description
A.AUTHORIZED_USER	Authorized users will follow all provided user guidance, including keeping passphrases and external tokens secure and stored separately from the disk.
A.ET_AUTH_USE_ONLY	External tokens that contain authorization factors will be used for no other purpose than to store the external token authorization factor.
A.PASSPHRASE_BASED_AUTH_FACTOR	An authorized administrator will be responsible for ensuring that the passphrase authorization factor has sufficient strength and entropy to reflect the sensitivity of the data being protected.
A.PLATFORM_I&A	The TOE will be installed on a platform that supports individual user identification and authentication. This I&A functionality shall remain unaffected by the TOE.
A.PROTECT_INTEGRITY	The user will exercise due diligence in physically protecting the TOE, and ensuring the IT environment will sufficiently protect against logical attacks.
A.SHUTDOWN	An authorized user will not leave the machine in a mode where sensitive information persists in non-volatile storage (e.g., power it down or enter a power managed state, such as a “hibernation mode”).
A.TRAINED_ADMINISTRATORS	Authorized administrators are appropriately trained and follow all administrator guidance.

A.TPM_PIN_ANTI-HAMMER	Authorization factors stored on a TPM device must be protected by a PIN, and the TPM device must implement an anti-hammer capability to prevent brute-force guessing attacks.

- Security Objectives (Section 3.3): Identifies the security objectives that are satisfied by the TOE and the TOE operational environment.
- Security Requirements (Section 5): Presents the security functional and assurance requirements met by the TOE.
- TOE Summary Specification (TSS) (Section 6): Describes the security functions provided by the TOE to satisfy the security requirements and objectives.
- Rationale for the Protection Profile Conformance Claim (Section 7): Presents the rationale concerning compliance of the ST with the Software Full Disk Encryption Protection Profile for the security objectives, requirements, and TOE Summary Specification as to their consistency, completeness and suitability.
- (Section): Presents the rationale for the security objectives, requirements, and TOE Summary Specification as to their consistency, completeness and suitability.

2 TOE Description

The TOE includes the Microsoft Windows 8 operating system, the Microsoft Windows Server 2012 operating system, supporting hardware, and those applications necessary to manage, support and configure the operating system.

The focus of this evaluation is on BitLocker, which is part of the Windows 8 and Server 2012 operating systems. BitLocker encrypts fixed and removable disk volumes, including volumes that contain the operating system and user data. Access to the encrypted volume is protected by one or more protectors, also known as authorization factors, which are specified by the administrator for the computer.

The following sections expand on the cryptographic, data protection, authentication, and management aspects of BitLocker in the context of a protection profile for software full disk encryption.

2.1 Product Types

Windows 8 and Windows Server 2012 are preemptive multitasking, multiprocessor, and multi-user operating systems. In general, operating systems provide users with a convenient interface to manage underlying hardware. They control the allocation and management of computing resources such as processors, memory, and Input / Output (I/O) devices. Windows 8and Windows Server 2012, collectively referred to as Windows, expand these basic operating system capabilities to manage the allocation of higher level IT resources such as security principals (user or machine accounts), files, printing objects, services, window station, desktops, cryptographic keys, network ports traffic, directory objects, and web content. Multi-user operating systems such as Windows keep track of which user is using which resource, grant resource requests, account for resource usage, and mediate conflicting requests from different programs and users.

2.2 Product Description

The TOE includes four product variants of Windows 8 and Windows Server 2012:

- Windows 8 Pro
- Windows 8 Enterprise
- Windows Server 2012 Standard
- Windows Server 2012 Datacenter

Windows 8 is suited for business desktops and notebook computers. It is the workstation product and while it can be used by itself, it is designed to serve as a client within Windows domains. BitLocker is used to protect the user's data.

Built for workloads ranging from the department to the enterprise to the cloud, Windows Server 2012 supports mission-critical solutions for databases, enterprise resource planning software, high-volume real-time transaction processing, server consolidation, public key infrastructure, virtualization, and additional server roles. BitLocker can be used to protect enterprise data used by these applications.

In terms of security and software disk encryption functionality, Windows 8 and Server 2012 share the same security characteristics.

2.3 Security Environment and TOE Boundary

The TOE includes both physical and logical boundaries. Its operational environment is that of a networked environment.

2.3.1 Logical Boundaries

The logical boundary of the TOE includes:

- The **Boot Manager**, which is invoked by the computer's bootstrapping code.
- The **Windows Loader** which loads the operating system into the computer's memory.
- **Windows OS Resume** which reloads an image of the executing operating system from a hibernation file as part of resuming from a hibernated state.
- The **Windows Kernel** which contains device drivers for the Windows NT File System, full volume encryption, the crash dump filter, and the kernel-mode cryptographic library.
- The **Cryptographic Services** module which confirms the signatures of Windows program files.
- **Windows Explorer** which can be used to manage BitLocker and check the integrity of Windows files and updates.
- The **manage-bde** console application to manage BitLocker.
- The **Get-AuthenticodeSignature PowerShell Cmdlet** which can be used to confirm the signatures of Windows program files.
- **PowerShell Cmdlets** to manage BitLocker:
    - **Add-BitLockerKeyProtector**
    - **Backup-BitLockerKeyProtector**
    - **Clear-BitLockerAutoUnlock**
    - **Disable-BitLocker**
    - **Disable-BitLockerAutoUnlock**
    - **Enable-BitLocker**

  - **Enable-BitLockerAutoUnlock**
  - **Get-BitLockerVolume**
  - **Lock-BitLocker**
  - **Remove-BitLockerKeyProtector**
  - **Resume-BitLocker**
  - **Suspend-BitLocker**
  - **Unlock-BitLocker**
- **Local and Group policies** to manage BitLocker
- The **Windows Trusted Installer which installs updates to the Windows operating system.**

2.3.2 Physical Boundaries

Physically, each TOE tablet, workstation, or server uses either an x86 or x64 architecture. The TOE executes on processors from Intel and AMD. Refer to section 1.1 for the specific list of hardware included in the evaluation.

A set of devices may be attached as part of the TOE:

- Display Monitor(s) (required)
- Fixed Disk Drive(s) (including disk drives and solid state drives)
- Removable Disk Drive(s) (including USB storage)
- Network Adaptor (required)
- Keyboard
- Mouse
- Printer
- Audio Adaptor
- CD-ROM Drive
- Smart Card Reader
- Trusted Platform Module (TPM) version 1.2 or 2.0

While this set of devices is larger than is needed to evaluate BitLocker, it is a subset of the devices used as the General Purpose Operating System Protection Profile evaluation. By using the same set of devices for both evaluations, consumers can gain assurance by using both core OS capabilities and BitLocker functionality.

The TOE does not include any network infrastructure components.

2.4 TOE Security Services

This section summarizes the security services provided by the TOE:

- **Cryptographic Protection:** Windows provides FIPS-140-2 validated cryptographic functions that support encryption/decryption, cryptographic signatures, cryptographic hashing, cryptographic key agreement (which is not studied in this evaluation), and random number generation. The TOE additionally provides support for public keys, credential management and certificate

validation functions and provides support for the National Security Agency's Suite B cryptographic algorithms. Windows also provides extensive auditing support of cryptographic operations, the ability to replace cryptographic functions and random number generators with alternative implementations,[3] and a key isolation service designed to limit the potential exposure of secret and private keys. In addition to supporting its own security functions with cryptographic support, the TOE offers access to the cryptographic support functions for user application programs. Public key certificates generated and used by the TOE authenticate users and machines as well as protect user and system data at rest.

    - **Software-based disk encryption:** Windows implements BitLocker to provide encrypted data storage for fixed and removable volumes and protects the disk volume's encryption key with one or more intermediate keys and authorization factors.

- **User Data Protection**: In the context of this evaluation, Windows provides encryption of fixed and removable volumes.
- **Identification & Authentication**: In the context of this evaluation, Windows provides the ability to generate, store, and protect authorization factors which provide access to data on the encrypted fixed and removable volumes.
- **Security Management**: Windows includes several functions to manage local and group security policies. Policy management is controlled through a combination of access control, membership in administrator groups, and privileges.
- **Protection of the TOE's Security Functions**: Windows provides a number of features to ensure the protection of TOE security functions. Windows ensures process isolation security for all processes through private virtual address spaces, execution context, and security context. The Windows data structures defining process address space, execution context, memory protection, and security context are stored in protected kernel-mode memory. Windows 8 and Windows Server 2012 Windows includes self-testing features that ensure the integrity of executable TSF images and Windows cryptographic functions. Finally, Windows provides a trusted update mechanism to update Windows binaries itself.

[3] This option is not included in the Windows Common Criteria evaluation.

3 Security Problem Definition

The security problem definition consists of the threats to security, organizational security policies, and usage assumptions as they relate to Windows 8 and Windows Server 2012. The assumptions, threats, and policies are copied from the Software Full Disk Encryption Protection Profile.

3.1 Threats to Security

Table 3-1 presents known or presumed threats to protected resources that are addressed by Windows 8 and Windows Server 2012 based on conformance to the Software Full Disk Encryption Protection Profile.

Table 3-1 Threats Addressed by Windows 8 and Windows Server 2012

Threat	Description
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 and use these values to gain access to user data.
T.PERSISTENT_INFORMATION	The TOE and/or the Operational Environment can go into a power saving mode so that the data or keying material are left unencrypted in persistent memory.
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.
T.TSF_ COMPROMISE	A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted) to gain access to key material or user data.
T.UNAUTHORIZED_DISK_ACCESS	An unauthorized user that has access to the lost hard disk may gain access to data for which they are not authorized according to the TOE security policy.
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 an authorization factor, resulting in exposure of the KEK, DEK, or user data.

3.2 Organizational Security Policies

An organizational security policy is a set of rules or procedures imposed by an organization upon its operations to protect its sensitive data and IT assets. **Table 3-2** describes organizational security policies that are addressed by Windows 8and Windows Server 2012 which are necessary for conformance to the Software Full Disk Encryption Protection Profile.

Table 3-2 Organizational Security Policies

Security Policy	Description
[No policy]	The Software Full Disk Encryption Protection Profile does not specify any organizational security policies

3.3 Secure Usage Assumptions

describes the core security assumptions of the environment in which Windows 8 and Windows Server 2012 is intended to be used. It includes information about the physical, personnel, procedural, and connectivity aspects of the environment.

The following specific conditions are assumed to exist in an environment where the TOE is employed in order to conform to the Software Full Disk Encryption Protection Profile:

Table 3-3 Secure Usage Assumptions

Assumption	Description
A.AUTHORIZED_USER	Authorized users will follow all provided user guidance, including keeping passphrases and external tokens secure and stored separately from the disk.
A.ET_AUTH_USE_ONLY	External tokens that contain authorization factors will be used for no other purpose than to store the external token authorization factor.
A.PASSPHRASE_BASED_AUTH_FACTOR	An authorized administrator will be responsible for ensuring that the passphrase authorization factor has sufficient strength and entropy to reflect the sensitivity of the data being protected.
A.PLATFORM_I&A	The TOE will be installed on a platform that supports individual user identification and authentication. This I&A functionality shall remain unaffected by the TOE.
A.PROTECT_INTEGRITY	The user will exercise due diligence in physically protecting the TOE, and ensuring the IT environment will sufficiently protect against logical attacks.
A.SHUTDOWN	An authorized user will not leave the machine in a mode where sensitive information persists in non-volatile storage (e.g., power it down or enter a power managed state, such as a “hibernation mode”).
A.TRAINED_ADMINISTRATORS	Authorized administrators are appropriately trained and follow all administrator guidance.
A.TPM_PIN_ANTI-HAMMER	Authorization factors stored on a TPM device must be protected by a PIN, and the TPM device must implement an anti-hammer capability to prevent brute-force guessing attacks.

4 Security Objectives

This section defines the security objectives of Windows 8 and Windows Server 2012 and its supporting environment. Security objectives, categorized as either TOE security objectives or objectives by the supporting environment, reflect the stated intent to counter identified threats, comply with any organizational security policies identified, or address identified assumptions. All of the identified threats, organizational policies, and assumptions are addressed under one of the categories below.

4.1 TOE Security Objectives

Table 4-1 describes the security objectives for Windows 8and Windows Server 2012 which are needed to comply with the Software Full Disk Encryption Protection Profile.

Table 4-1 Security Objectives for the TOE

Security Objective	Source
O.AUTHORIZATION	The TOE must obtain the authorization factor(s) from a user to be able to decrypt the data on the hard disk.
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 data that are stored on a hard drive. (Note that this may exclude the MBR and the bootable partition that it points to.)
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 will zeroize key material as soon as it is no longer needed to decrease the chance that such material could be used to discover a KEK or DEK.
O.MANAGE	The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.
O.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 accessible while the TOE is in operation.
O.POWER_SAVE	The TOE must be configurable so that there exists at least one mechanism that will cause the system to power down after a period of time in the same fashion as the user electing to shutdown the system (O.SHUTDOWN). Any such mechanism (e.g., sleep, hibernate) that does not conform to this

	requirement must be capable of being disabled by the administrator.
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 administrators the capability to update the TOE firmware/software, and verify that updates to the product are received from the intended source.

4.2 Security Objectives for the Operational Environment

Table 4-2 describes the security objectives for the operational environment as specified in the Software Full Disk Encryption Protection Profile.

Table 4-2 Security Objectives for the Operational Environment

Environment Objective	Description
OE.PASSPHRASE_STRENGTH	An authorized administrator will be responsible for ensuring that the passphrase authorization factor conforms to guidance from the Enterprise using the TOE.
OE.PLATFORM_I&A	The Operational Environment will provide individual user identification and authentication mechanisms that operate independently of the authorization factors used by the TOE.
OE.RESTRICTED_FUNCTIONS	Management functions will be limited to an authorized administrator.
OE.SINGLE_USE_ET	External tokens that contain authorization factors will be used for no other purpose than to store the external token authorization factor.
OE.STRONG_ENVIRONMENT_ CRYPTO	The Operating Environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE and Appendix C.
OE.TRAINED_USERS	Authorized users will be properly trained and follow all guidance for securing the TOE and authorization factors.

5 Security Requirements

The section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) for the TOE. The requirements in this section are based on Common Criteria functional requirements described in the Software Full Disk Encryption Protection Profile, version 1.1, March 31, 2014.

Conventions:

Where requirements are drawn from the protection profile, the requirements are copied verbatim, except for some changes to required identifiers to match the iteration convention of this document, from that protection profile and only operations performed in this security target are identified.

The extended requirements, extended component definitions and extended requirement conventions in this security target are drawn from the protection profile; the security target reuses the conventions from the protection profile which include the use of the word "Extended" and the "_EXT" identifier to denote extended functional requirements. The security target assumes that the protection profile correctly defines the extended components and so they are not reproduced in the security target.

Where applicable the following conventions are used to identify operations:

- **Iteration**: Iterated requirements (components and elements) are identified with letter following the base component identifier. For example, iterations of FMT_COP.1 are identified in a manner similar to FCS_COP.1(SIGN) (for the component) and FCS_COP.1(SIGN).1 (for the element).
- **Assignment**: Assignments are identified in brackets and bold (e.g., **[assigned value]**).
- **Selection**: Selections are identified in brackets, bold, and italicized (e.g., ***[selected value]***).
    - Assignments within selections are identified using the previous conventions, except that the assigned value would also be italicized and with additional brackets (e.g., ***[selected value [assigned value]]***).
- **Refinement**: Refinements are identified using bold text (e.g., **added text**) for additions and strike-through text (e.g., ~~deleted text~~) for deletions.

5.1 TOE Security Functional Requirements

This section specifies the SFRs for the TOE.

Table 5-1 TOE Security Functional Requirements

Requirement Class	Requirement Component
Cryptographic Support (FCS)	Cryptographic Key Generation for Disk Encryption Keys(FCS_CKM.1(DEK))
	Cryptographic Key Generation for Key Encryption Key (FCS_CKM.1(KEK))
	Cryptographic Key Generation for Passphrase Conditioning (FCS_CKM.1(PASS))
	Cryptographic Key Generation for External Token Support (FCS_CKM.1(TOKEN))
	Cryptographic Keying Material Destruction (FCS_CKM_EXT.4)
	Cryptographic Operation for Disk Encryption (FCS_COP.1 (SYM))

Requirement Class	Requirement Component
	Cryptographic Operation for Signature Verification (FCS_COP.1 (SIGN))
	Cryptographic Operation for Hashing (FCS_COP.1 (HASH))
	Cryptographic Operation for Key Masking (FCS_COP.1 (MASK))
	Extended: Cryptographic Operation for Random Bit Generation (FCS_RBG_EXT.1)
User Data Protection (FDP)	Extended: Protection of Data on Disk (FDP_DSK_EXT.1)
	Extended: Power Management Function (FDP_PM_EXT.1)
Identification & Authentication (FIA)	Extended: FDE User Authorization (FIA_AUT_EXT.1)
Security Management (FMT)	Specification of Management Functions (FMT_SMF.1)
Protection of the TSF (FPT)	Extended: TSF Testing (FPT_TST_EXT.1)
	Extended: Trusted Update (FPT_TUD_EXT.1)

5.1.1 Cryptographic Support (FCS)

5.1.1.1 Cryptographic Key Generation for Disk Encryption Key (FCS_CKM.1(DEK))

Application Note: FCS_CKM.1(DEK) corresponds to FCS_CKM.1(1) in the Software Full Disk Encryption Protection Profile.

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

Application Note: FCS_CKM.1(DEK) specifies how the Full Volume Encryption Key (FVEK) and Volume Master Key (VMK) are generated.

5.1.1.2 Cryptographic Key Generation for Key Encryption Key (FCS_CKM.1(KEK))

Application Note: FCS_CKM.1(KEK) corresponds to FCS_CKM.1(2) in the Software Full Disk Encryption Protection Profile.

FCS_CKM.1(KEK).1 The TSF shall derive KEK cryptographic keys in accordance with a specified cryptographic key derivation algorithm [***None, exclusive OR (XOR)] with the following inputs*** **[**

a submask derived from a passphrase authorization factor conditioned as defined in FCS_CKM.1(Y),[4]

an external token authorization factor that is the same bit-length as the DEK[5]***,***

[

[4] This corresponds to FCS_CKM.1(PASS) in the security target.
[5] This is for the start-up key and recovery key described below.

- **a *TPM-protected authorization factor that is the same bit-length as the DEK*, [**
- ***a PIN that that is conditioned to be the same length as the DEK,***
- ***an enhanced PIN that is conditioned to be the same length as the DEK,***
  ]

that produce submasks that are the same length as the DEK as specified in FCS_CKM.1(1)[6]]]

maintaining the effective strength of each authorization factor and specified cryptographic key sizes [***128 bits, 256 bits***] that meet the following: [No standard].

5.1.1.3 Cryptographic Key Generation for Passphrase Conditioning (FCS_CKM.1(PASS))

Application Note: FCS_CKM.1(PASS) corresponds to FCS_CKM.1(Y) in the Software Full Disk Encryption Protection Profile.

FCS_CKM.1(PASS).1 The TSF shall accept passphrases that are used to generate a submask that contain at least 64 or more characters in the set of [upper case characters, lower case characters, and [**digits 0-9, "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")"**] and shall be conditioned [

- ***using [SHA-256] for 128-bit DEKs;***
- ***using [SHA-256] for 256-bit DEKs;***

];

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

Application Note: The passphrase can be used to unlock data volumes, or the operating system volume on computers that do not have a TPM.

5.1.1.4 Cryptographic Key Generation for External Token Support (FCS_CKM.1(TOKEN))

Application Note: FCS_CKM.1(TOKEN) corresponds to FCS_CKM.1(X) in the Software Full Disk Encryption Protection Profile.

FCS_CKM.1(TOKEN).1 The TSF shall derive an external authorization factor generated by a Random Bit Generator as specified in FCS_RBG_EXT.1 that produces an authorization factor of [***256 bits***] that was seeded with entropy at least equal to the size of the DEK, as specified in FCS_CKM.1(1).[7]

FCS_CKM.1(TOKEN).2 The TSF shall be able to store the authorization factor on [***an external device***].

[6] This corresponds to FCS_CKM.1(KEK) in the security target.
[7] This corresponds to FCS_CKM.1(DEK) in the security target.

5.1.1.5 Cryptographic Key Zeroization (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1 The TSF shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

5.1.1.6 Cryptographic Operation for Disk Encryption (FCS_COP.1(SYM))

Application Note: FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the Software Full Disk Encryption Protection Profile.

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

FCS_COP.1(SYM).2 DEK/IV pairs used by TOE on a platform must be unique.

5.1.1.7 Cryptographic Operation for Signature Verification (FCS_COP.1 (SIGN))

Application Note: FCS_COP.1(SIGN) corresponds to FCS_COP.1(2) in the Software Full Disk Encryption Protection Profile.

FCS_COP.1(SIGN).1 The TSF shall perform cryptographic signature verification for TOE updates in accordance with a [
(2) RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of 2048 bits or greater]

that meets the following:

Case: RSA Digital Signature Algorithm

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

5.1.1.8 Cryptographic Operation for Hashing (FCS_COP.1 (HASH))

Application Note: FCS_COP.1(HASH) corresponds to FCS_COP.1(3) in the Software Full Disk Encryption Protection Profile.

FCS_COP.1(HASH).1 The TSF shall perform cryptographic hashing services in accordance with [***SHA-1, SHA 256***[8]] and message digest sizes [***160, and 256***] bits that meet the following: FIPS Pub 180-3, "Secure Hash Standard".

5.1.1.9 Cryptographic Operation for Key Masking (FCS_COP.1 (MASK))

Application Note: FCS_COP.1(MASK) corresponds to FCS_COP.1(4) in the Software Full Disk Encryption Protection Profile.

FCS_COP.1(MASK).1 The TSF shall perform key masking in accordance with a specified cryptographic algorithm [***AES used in CCM mode; AES Key Wrap***] and the cryptographic key size [***128 bits, 256 bits***] that meet the following: [***"FIPS PUB 197, Advanced***

[8] Windows implements SHA 384 and SHA 512, BitLocker does not use SHA 384 and SHA-512 is used as part of the ECDH smart card authorization factor which is outside the scope of evaluation.

Encryption Standard (AES) for AES, NIST SP 800-38C for CCM mode and NIST SP 800-38F" for AES Key Wrap].

5.1.1.10 Extended: Cryptographic Operation for Random Bit Generation (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1 The TSF shall perform all random bit generation (RBG) services in accordance with [***NIST Special Publication 800-90 using [CTR_DRBG (AES), Dual_EC_DRBG (any)]***] seeded by an entropy source that accumulated entropy from [**a software-based noise source; a hardware-based noise source**].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded with a minimum of [***256 bits***] of entropy at least equal to the greatest bit length of the keys and authorization factors that it will generate.

5.1.2 User Data Protection (FDP)

5.1.2.1 Extended: Protection of Data on Disk (FDP_DSK_EXT.1)

FDP_DSK_EXT.1.1 The TSF shall perform Full Disk Encryption in accordance with FCS_COP.1.1(1).[9]

FDP_DSK_EXT.1.2 The DEK can only exist on an unpowered hard disk if it is masked with a KEK (or intermediate key) derived as specified in FCS_CKM.1(2)[10] and FCS_COP.1(4).[11]

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

FDP_DSK_EXT.1.4 No plaintext keying material shall be written to persistent storage**.**

5.1.2.2 Extended: Power Management Function (FDP_PM_EXT.1)

FDP_PM_EXT.1.1 The TSF shall protect all data stored to the disk drive during the transition to the [**S4, S5, and G3 ACPI power**] state**s** as per FDP_DSK_EXT.1.1.

FDP_PM_EXT.1.2 On the return to a powered-on state from the state indicated in FDP_HIB_EXT.1.1,[12] the TSF shall authorize the user in the manner specified in FIA_AUT_EXT.1.1 before any data is decrypted.

5.1.3 Identification and Authentication (FIA)

5.1.3.1 Extended: FDE User Authorization (FIA_AUT_EXT.1)

FIA_AUT_EXT.1.1 The TSF shall use a mechanism as defined in FCS_CKM.1.1(2),[13] and FCS_COP.1(4)[14] 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 verify that the authorization factors are valid before allowing access to unencrypted data from the device.

---

[9] This corresponds to FCS_CKM.1(SYM) in the security target.
[10] This corresponds to FCS_CKM.1(KEK) in the security target.
[11] This corresponds to FCS_COP.1(MASK) in the security target.
[12] The reference to FDP_HIB_EXT.1.1 is a flaw in the protection profile; it should refer to FDP_PM_EXT.1.1 instead.
[13] This corresponds to FCS_CKM.1(KEK) in the security target.
[14] This corresponds to FCS_COP.1(MASK) in the security target.

FIA_AUT_EXT.1.4 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.

5.1.4 Security Management (FMT)

5.1.4.1 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions:

a) generate the DEK when the disk drive is initialized for encrypted operation or on command by the administrator
b) Wrap the DEK using a KEK formed from submasks derived from user-entered authorization factors; specifically a [***passphrase-based authorization factor, external token authorization factor***] and [***TPM-protected authorization factor, [PIN and enhanced PIN authorization factors]***][15]
c) [***Configure cryptographic functionality, [change passphrase-based authorization factor, generate external authorization factors, disable key recovery functionality, specify a PIN and enhanced PIN, suspend and re-enable BitLocker***.]

5.1.5 Protection of the TSF (FPT)

5.1.5.1 Extended: TSF Self-Test (FPT_TST_EXT.1)

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.

5.1.5.2 Extended: Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1 The TSF shall provide administrators the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The TSF shall provide administrators the ability to initiate updates to TOE software.

FPT_TUD_EXT.1.3 The TSF shall verify firmware/software updates to the TOE using a digital signature mechanism and [***no other functions***] prior to installing those updates.

5.2 TOE Security Assurance Requirements

The security assurance requirements for the TOE are the requirements defined in the Software Full Disk Encryption Protection Profile which in turn is based on assurance requirements that are specified in Part 3 of the Common Criteria. No operations are applied to the assurance components.

In addition, the assurance activities from the Software Full Disk Encryption Protection Profile are used to determine that Windows satisfies the full disk encryption security functional requirements. These assurance activities are described in the protection profile and copied into the security target.

[15] See Table 6-3 TOE BitLocker Authorization Factors for combinations of authorization factors.

5.2.1 CC Part 3 Assurance Requirements

The following table is the collection of CC Part 3 assurance requirements from the Software Full Disk Encryption Protection Profile.

Table 5-2 TOE Security Assurance Requirements

Requirement Class	Requirement Component
Development (ADV)	Basic Functional Specification (ADV_FSP.1)
Guidance Documents (AGD)	Operational User Guidance (AGD_OPE.1)
	Preparative Procedures (AGD_PRE.1)
Testing (ATE)	Independent Testing – Conformance (ATE_IND.1)
Vulnerability Assessment (AVA)	Vulnerability Analysis (AVA_VAN.1)
Life-cycle Support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM Coverage (ALC_CMS.1)

5.2.2 Full Disk Encryption PP Assurance Activities

This section copies the assurance activities from the protection profile in order to ease reading and comparisons between the protection profile and the security target.

5.2.2.1 Cryptographic Support

5.2.2.1.1 Cryptographic Key Generation for Disk Encryption Keys (FCS_CKM.1(DEK))

The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. If the RBG is provided by the Operational Environment, then the evaluator checks to ensure that--for each platform identified in the ST--the TSS describes the interface used by the TOE to invoke this functionality. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to 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), which is also a requirement contained in Appendix C).

5.2.2.1.2 Cryptographic Key Generation for Key Encryption Key (FCS_CKM.1(KEK))

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 passphrase-based factors the examination of the TSS section is performed as part of FCS_CKM.1(Y) assurance activities. For external authorization factors, the TSS shall detail whether the authorization factor must be generated by the TOE (in which case the assurance activities associated with FCS_CKM.1(X) in Appendix C apply); if not, then the TSS section shall specify measures taken by the TOE (or administrative personnel) to ensure the external authorization factor meets the minimum length requirements listed in the ST. Acceptable means include administrator-verification of the length or input validation checks performed at run time by the TOE. Additionally in this case, the evaluator shall verify that the administrative guidance discusses the characteristics of external authorization factors (e.g., how the

authorization factor must be generated; format(s) or standards that the authorization factor must meet, configuration of the TPM device used) that are able to be used by the TOE.

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

The evaluator shall also perform the following tests:

- Test 1 [conditional]: If there is more than one authorization factor, ensure that failure to supply a required authorization factor does not result in access to the encrypted data.
- Test 2 [conditional]: If the TOE supports multiple external tokens of different formats, the evaluators confirm that each format is able to be successfully used in providing authorization information to the TOE.

5.2.2.1.3 Cryptographic Key Generation for Passphrase Conditioning (FCS_CKM.1(PASS))

There are two aspects of this component that require evaluation: the minimum length of a passphrase as specified in the selection is 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 minimum passphrase length

The evaluators shall check the TSS section to determine that it specifies that a capability exists to accept passphrases with the smaller of 64 or the number in the assignment characters. The evaluators shall also check the operational guidance to determine that there are instructions for administrators generating 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: The evaluator shall compose several passphrases that comply with the operational guidance. The evaluators shall ensure at least one of these passphrases is the maximal length as specified in the assignment statement in the requirement. For each composed passphrase, the evaluator shall demonstrate that it is accepted by the TOE and can be used as an authorization factor (that is, the evaluator is able to unlock the disk using this passphrase).
- Test 2 [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. The evaluator ensures that all characters indicated in the requirement can be used in a valid passphrase.

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.

5.2.2.1.4 Cryptographic Key Generation for External Token Support (FCS_CKM.1(TOKEN))

The evaluator reviews the guidance documentation to confirm that the steps necessary for an administrator to generate an external authorization factor are described. The evaluator reviews the TSS portion of the ST to confirm that the external authorization factor generation process is described, including how the generation function uses the RBG, and how the RBG function is seeded. Finally, the evaluator reviews the TSS section (or administrative guidance documentation) to determine how the value generated by the RBG is transferred to the device specified in the selection. It should be noted that the RBG used must be provided by the TOE and meet the requirements specified in FCS_RBG_EXT.1 in order for this requirement to be claimed in the ST.

The following tests must be performed by the evaluator:

- Test 1: Following the administrative guidance, create an external token authorization factor. If possible, confirm the # of bits the authorization factor contains. Load the external authorization factor into its “container”. Ensure that the external authorization factor can then be used to access an encrypted disk.

5.2.2.1.5 Cryptographic Key Zeroization (FCS_CKM_EXT.4)

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key; when they are zeroized (for example, immediately after use, on system shutdown, etc.); and the type of zeroization procedure that is performed (overwrite with zeros, overwrite three times with random pattern, etc.). If different types of

memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the zeroization procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are zeroized by overwriting once with zeros, while secret keys stored on the internal hard drive are zeroized by overwriting three times with a random pattern that is changed before each write").

5.2.2.1.6 Cryptographic Operation for Disk Encryption (FCS_COP.1 (SYM))

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.

The evaluator shall ensure that the TSS describes-for each supported platform listed in the ST, the process by which the disk is encrypted. If multiple disks are supported, then this description shall cover this cases as well. This description shall cover how (or if) the disk is divided into encryptable portions so that it is clear when an IV needs to be used, and the number of IVs that are potentially present in the system. The description shall also cover IV generation so that the evaluator can determine that all DEK/IV pairs used on the system are unique.

CBC Mode

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

The evaluators shall run a set of known answer tests for each key size and mode 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 these modes 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

The evaluators shall perform a multi-block message test for each key length and mode 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).

The evaluators shall perform a Monte Carlo test for each supported mode. 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.x of [AESAVS] and is dependent on the mode being tested.

5.2.2.1.7 Cryptographic Operation for Signature Verification (FCS_COP.1 (SIGN))

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.

Hash functions and/or random number generation required by these algorithms must be specified in the ST; therefore the assurance activities associated with those functions is contained in the associated Cryptographic Hashing and Random Bit Generation sections. 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.

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 hard drive 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).

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

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

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

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.

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.

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 psuedorandomly-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.

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.

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.

ECDSA

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

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.

5.2.2.1.8 Cryptographic Operation for Hashing (FCS_COP.1 (HASH))

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.

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.

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.

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

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 psuedorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. 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

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 psuedorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

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

5.2.2.1.9 Cryptographic Operation for Key Masking (FCS_COP.1 (MASK))

If the DEK masking method is using "XOR", the evaluators shall verify that the use of XOR is stated in the TSS.

If AES in ECB mode is used, then the following assurance activities will be performed.

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.).

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

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.

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

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).

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].

If AES in CCM mode is used, then the following assurance activities will be performed.

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.).

The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

128 bit and 256 bit keys

Two payload lengths. One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).

Two or three associated data lengths. One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 216 bytes, an associated data length of 216 bytes shall be tested.

Nonce lengths. All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.

Tag lengths. All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.

To test the generation-encryption functionality of AES-CCM, the evaluator shall perform the following four tests:

Test 1. For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

Test 2. For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

Test 3. For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.

Test 4. For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

To determine correctness in each of the above tests, the evaluator shall compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator shall supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS

result with the decrypted payload. The evaluator shall supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

If AES Key Wrap or AES Key Wrap with padding is used, then the following assurance activities shall be performed.

The evaluation team shall check the TSS to ensure it vigorously asserts that the Key Wrap specification is met.

5.2.2.1.10 Extended: Cryptographic Operation for Random Bit Generation (FCS_RBG_EXT.1)

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Annex G, Entropy Documentation and Assessment The evaluator shall perform the following tests, depending on the standard to which the RBG conforms.

Implementations Conforming to NIST Special Publication 800-90

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.

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).

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

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

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

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

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

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

5.2.2.2 User Data Protection

5.2.2.2.1 Extended: Protection of Data on Disk (FDP_DSK_EXT.1)

The evaluator shall consult the TSS section 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 disk and the point at which the encryption function is applied. Since the implementation of the encryption/decryption functionality is implemented entirely in software on the host operating system, then the TSS must make the case that all methods of accessing the disk will pass through these functions. This must be true for all hard drives contained on the platform on which the TOE runs.

In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on power-up relating to the loading of the MBR and the portions of the TOE that perform the authorization function. The TSS should also cover the initialization of the TOE and the activities that are performed to ensure that all disks are entirely encrypted when the TOE is first established; areas of the disk that are not encrypted (e.g., portions associated with the MBR) shall also be described.

The evaluator shall ensure that the description also covers how the cryptographic functions in the FCS requirements are being used to perform the encryption functions, including how the DEK is unwrapped and stored in the TOE. For the cryptographic functions that are provided by the Operational Environment, the evaluator shall check the TSS to ensure it describes--for each platform identified in the ST--the interface(s) used by the TOE to invoke this functionality. The evaluator shall ensure that the TSS describes, for each power-down scenario (see assurance activities for FCS_CKM_EXT.4) how the TOE ensures the DEK is wrapped with the KEK.

The evaluator shall ensure that the TSS describes how other functions available in the system (e.g., regeneration of the DEK) ensure that no unencrypted data or key material are present on the disk.

If the TOE supports multiple disk encryption, the evaluator shall examine the administration guidance to ensure the initialization procedure addresses the need for all disks on the platform to be encrypted.

The evaluator reviews the TSS to determine that it makes a case that key material is not written unencrypted to the hard disk. Since in normal use all writes to the disks will be encrypted, one approach is to make an argument concerning the exceptional cases when data are written to the unencrypted portions of the disk, then detailing how key material is prevented from being written in these areas.

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

The evaluators shall perform the following test activities:

- Test 1: Ensure that following the initialization activities results in all disks 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 or sensitive TSF data can be written to these areas. The examination of the disks can be done in several ways. Physically removing the drive and then inserting it into another computer is one method. Alternatively, booting the system that contains the encrypted hard drive from an external device and then directly accessing the encrypted drive is another example of an acceptable method for examination.
- Test 2: Ensure that the data (including data that may be stored in page files in the OS) are encrypted when written to disk. The extent that this is tested is consistent with the previous test; that is, it is acceptable to power on the system “normally”, cause data to be written to the disk, and then use the methods mentioned in the previous test to ensure those data do not appear unencrypted on the device(s).

5.2.2.2.2 Extended: Power Management Function (FDP_PM_EXT.1)

The evaluator shall examine the TSS to ensure that it describes the state(s) that are supported by this capability. For each state, the evaluator ensures that the TSS contains a description of how the state is entered, and the actions of the TSF on entering the state. The TSF shall also describe how the state is exited, and how the requirements are met during this transition to an operational state.

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

The following tests must be performed by the evaluator for each supported State:

- Test 1: Following the administrative guidance, configure the Operational Environment and the TOE so that the lower power state of the platform is protected. Invoke the lower power state. On resumption of normal power, observe that an incorrect entry of the authorization factor(s) does not result in access to the system, and that correct entry of the authorization factor(s) does result in access to the system. The intent here is that the user should be first prompted for the authorization factor such that the only activity performed by the system is for the TOE to verify

that the authorization factors are valid before allowing access to unencrypted data from the device.

5.2.2.3 Identification and Authentication

5.2.2.3.1 Extended: FDE User Authorization (FIA_AUT_EXT.1)

The evaluator shall check the TSS section to determine it describes how the TOE is initialized; that is, the sequence of events including power on; MBR access; and load of the code that will perform the authorization activities. If the operational guidance describes different start-up modes (e.g., pressing certain function keys during the boot process), the evaluator shall ensure the TSS describes how these modes do not allow access the data on the hard disk prior to entering the authorization factors.

The evaluator shall check that the TSS describes how the authorization factors are validated prior to allowing the user to access the data on a drive. 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).

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

The evaluator shall perform the following tests:

- Test 1: Ensure that the authorization factors are prompted for prior to allowing any access to unencrypted data on the hard drive devices.
- Test 2: 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.
- Test 3 [conditional]: If any bypass or alternate boot modes are provided, test to ensure that the modes are consistent with the requirements (that is, the appropriate authorization factors have to be entered prior to access to unencrypted data).
- Test 4: Ensure that after access to the device is obtained by correctly entering the authorization factors, the underlying platform still requires identification and authentication distinct from the authorization factors already entered.

5.2.2.4 Security Management

5.2.2.4.1 Specification of Management Functions (FMT_SMF.1)

The assurance activities for this component will be driven by the selections made by the ST author. This section describes assurance activities for the selections in the above component (with the exception of the “other management functions” assignment); 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.

Required Activities

While it is a requirement of products conformant to this PP to restrict (with potentially significant help from the Operational Environment) the above-mentioned functions to an administrator (a privileged group that is a subset of the set of users of the TOE), the requirement to do so and associated assurance activities are levied elsewhere in the ST, depending on the particular TOE implementation. As detailed in section 1.1.4 of this PP, there are a range of scenarios that could be implemented, and the assurance activities associated with each—while similar—differ in scope and implementation.

General

The evaluator reviews the TSS and AGD guidance and shall determine that it identifies all of the non-TOE products needed to perform administration. For instance, if the management capability is provided via Java Applets or through a web browser interface, the details of what needs to be in place are provided in the TSS.

Generate DEK

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 or re-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. The TSS shall also describe the processing (if any) that occurs for existing data when a new DEK is generated and installed. The evaluator shall also perform the following tests:

- Test 1: On a "clean" installation, the administrator is able to generate the DEK.
- Test 2: With the disk already encrypted, generate a new DEK for the disk(s) and verify that the new DEK is different from the previous DEK.
- Test 3: Using information in the TSS and AGD_OPR/AGD_PRE guidance, ensure any claims made relating to functionality that is visible to users of the TOE as a function of the DEK regeneration process are verified (e.g., files encrypted with the previous DEK are still visible after a new DEK is generated).

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

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 wrapping of the DEK (or the wrapping of a new DEK) with the KEK produced from the authorization factors. The authorization factors can be per-disk or per-user. The evaluator shall review the AGD guidance to determine, for each supported authorization factor, 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 various authorization factors are 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 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 tests:

- Test 4: For each acceptable combination 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 authorization factors allowed by the implementation. Authorization factors should be tested on a variety supported Operational Environments (as claimed in the TOE), especially if there are different authorization factors supported in different environments (for example, environments possessing a TPM vs. those that don’t).

Conditional Activities

Item c 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 Nation Scheme overseeing the evaluation needs to be consulted to determine if compliance to the PP can be claimed.

If passphrase-based authorization factors are used by the TOE, then the "change passphrase-based authorization factor" item is selected and the following assurance activities shall be performed. The evaluator shall examine the operational guidance to ensure that it describes how the passphrase-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 hard disk device 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 5 [conditional]: The evaluator shall establish a passphrase authorization factor for the hard disk 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 passphrase on the device Once the current authorization factor is changed, they shall ensure that they are still able to access the 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.

It may be the case that the hard disk device 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 6: [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.

Two of the activities in section d concern the generation of authorization factors: passphrases and the bit-string to be put on an external token. 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 are generated. For this requirement, 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. The tests associated with these mechanisms are specified as part of that particular mechanisms' assurance activities.

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.

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, as well as an additional documentation associated with the TOE in terms of the cryptographic functionality. Armed with this information, the evaluator shall review the AGD documentation to determine that there are instructions for manipulating all of the claimed mechanisms. The evaluator shall also perform the following test:

- Test 7 [conditional]: For each configurable cryptographic mode supported, the evaluator follows the AGD instructions to determine that TOE functionality operates as expected (that is, the hard disk(s) are encrypted/decrypted appropriately). While it is not required that the details be verified at this assurance level (that is, if the TOE allows AES 128-bit or 256-bit keys, the evaluator is not required to run the process in a debugger that allows them to determine the length of the key), it is required that some comparative analysis be performed. For example, if different AES modes are supported, choosing a different mode with the same DEK should result in different ciphertext being present on the hard disk(s).

If the TOE supports key recovery, this must be stated in the TSS. The TSS shall also describe how to disable this functionality. This includes a description of how the recovery material is provided to the

recovery holder. The intent is that this description can be used by the evaluators in testing to determine whether the key recovery 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 8 [conditional] If the TOE provides key recovery capability whose effects are visible at the TOE interface, then the evaluator shall devise a test that ensures that the key recovery capability has been or can be disabled following the guidance provided by the vendor.

5.2.2.5 TSF Protection

5.2.2.5.1 Extended: TSF Self-Test (FPT_TST_EXT.1)

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

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

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

5.2.2.5.2 Extended: Trusted Update (FPT_TUD_EXT.1)

Updates to the TSF are signed by an authorized source, and may have an associated hash. 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 installed in the Operational Environment. The evaluator ensures this information is contained in the TSS and that any instructions dealing with the installation of the update credentials is detailed in the operational guidance. 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 optionally calculation of the hash); and the actions that take place for successful (signature was verified and optionally hash was verified) and unsuccessful (signature could not be verified, optional hash was incorrect) cases. If the digital signature/hashing is performed by the Operational Environment, then the evaluator shall check the TSS to ensure it describes--for each platform identified in the ST--the interface(s) used by the TOE to invoke this cryptographic functionality.

The location of the software that is performing the processing must also be described in the TSS and verified by the evaluators. The evaluators shall perform the following tests (if an optional hash is supported by the TOE, then the evaluator performs tests 2 and 3 for different combinations of valid and invalid digital signatures and hashes, as well as for digital signature alone):

- Test 1: The evaluator performs the version verification activity to determine the current version of the 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 described 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.

6 TOE Summary Specification (TSS)

This chapter describes the Windows 8and Windows Server2012 security functions in the context of BitLocker. The Windows 8 and Windows Server 2012 Security Functions (SFs) satisfy the security functional requirements of the Software Full Disk Encryption Protection Profile.

The TOE performs the following security functions:

- Cryptographic Protection
- User Data Protection
- Identification and Authentication
- Security Management
- TSF Protection

The following sections present each TOE Security Functions (TSFs) and a mapping of security functions to Security Functional Requirements (SFRs).

6.1 Cryptographic Support

6.1.1 TSS Description

Cryptography API: Next Generation (CNG) API is designed to be extensible at many levels and agnostic to cryptographic algorithm suites and is a part of Windows 8 and Server 2012. An important feature of CNG is its native implementation of the Suite B algorithms. CNG includes support for Suite B that extends to all required algorithms: AES (128, 256 key sizes), the SHA family (SHA-256and SHA-384) of hashing algorithms, elliptic curve Diffie Hellman (ECDH), and elliptical curve DSA (ECDSA) over the NIST-standard prime curves P-256 and P-384 along with other FIPS Approved algorithms.

Deterministic random bit generation (DRBG) is implemented in accordance with NIST Special Publication 800-90. Windows generates random bits by taking the output of a cascade of two SP800-90 AES-256 counter mode based DRBGs in kernel-mode and four cascaded SP800-90 AES-256 DRBGs in user-mode; all are seeded from the Windows entropy pool. The entropy pool is populated using the following values:

- An initial entropy value from a seed file provided to the Windows OS Loader at boot time (512 bits of entropy).[16, 17]
- A calculated value based on the high-resolution CPU cycle counter which fires after every 1024 interrupts (a continuous source providing 16384 bits of entropy).
- Random values gathered periodically from the Trusted Platform Module (TPM), if one is available on the system (320 bits of entropy on boot, 384 bits thereafter).

[16] The Windows OS Loader is the process of being validated as a FIPS 140-2 Level 1 module.
[17] The Windows OS Loader implements a SP 800-90 AES-CTR-DRBG and passes along 384 bits of entropy to the kernel for CNG to be use during initialization. This DBRG uses the same algorithms to obtain entropy from the CPU cycle counter, TPM, and RDRAND as described above.

- Random values gathered periodically by calling the RDRAND CPU instruction, if supported by the CPU (256 bits of entropy).

The main source of entropy in the system is the CPU cycle counter which tracks hardware interrupts. This is a sufficient health test; if the computer were not accumulating hardware and software interrupts it would not be running and therefore there would be no need for random bit generation. In the same manner, a failure of the TPM chip or processor would be a critical error that halts the computer. In addition, if the user chooses to operate Windows in the FIPS validated mode, it will run FIPS 140 AES-256 Counter Mode DBRG Known Answer Tests (instantiate, generate) and Dual-EC DRBG Known Answer Tests (instantiate, generate) on start-up. Windows always runs the SP 800-90-mandated self-tests for AES-CTR-DRBG during a reseed and runs the Dual-EC reseed self-test if the user chooses to operate Windows in the FIPS validated mode.

Each entropy source is independent of the other sources and does not depend on time. The CPU cycle counter inputs vary by environmental conditions such as data received on a network interface card, key presses on a keyboard, mouse movement and clicks, and touch input.

The TSF defends against tampering of the random number generation (RNG)/ pseudorandom number generation (PRNG) sources by encapsulating its use in Kernel Security Device Driver. The interface for the Windows random number generator is BCryptGenRandom. By default, the CNG provider for random number generation is the AES_CTR_DRBG, however CNG can be configured to use the Dual EC DRBG.

The encryption and decryption operations are performed by independent modules, known as Cryptographic Service Providers (CSPs). The Windows CSPs, specifically the Kernel Cryptographic Primitives Library, are FIPS 140-2 Level 1 compliant. The Kernel Cryptographic Primitives Library is the CSP which is used by the NTFS device stack.

In addition to encryption and decryption services, the TSF provides other cryptographic operations such as hashing and digital signatures. The compliance with these cryptographic standards has been demonstrated as follows:

Table 6-1 Cryptographic Standards and Evaluation Methods

Cryptographic Operation	Standard	Evaluation Method
Encryption/Decryption	FIPS 197 AES For ECB, CBC, CFB8, CCM, and GCM modes	NIST CAVP #2197, #2216
Digital signature	FIPS 186-3 rDSA	NIST CAVP #1134, #1133
Digital signature	FIPS 186-3 ECDSA	NIST CAVP #341
Hashing	FIPS 180-3	NIST CAVP #1903
Random number generation	NIST SP 800-90	NIST CAVP #259 for Dual_EC_DRBG NIST CAVP #258 for CTR_DRBG

The TSF overwrites each intermediate storage area for plaintext key/critical cryptographic security parameter (i.e., any storage, such as memory buffers, that is included in the path of such data). This overwriting is performed by overwriting the memory area with zeros using a forced inline function to minimize execution time. Because the memory word is explicitly set to '0', it is not necessary to do a read-back test. Windows uses only one type of system memory.

The following table describes the secret keys, private keys, and critical security parameters used by BitLocker within Windows:

Table 6-2 BitLocker Key Types and Critical Security Parameters

Security Relevant Data Item	Description
Symmetric encryption/decryption keys	Keys used for AES encryption/decryption for the storage volume and key protectors.
Asymmetric RSA Public Keys	Keys used for the verification of RSA digital signatures.
Asymmetric RSA Private Keys	Keys used for the calculation of RSA digital signatures.
AES-CTR DRBG Seed	A secret value maintained internal to the module that provides the seed material for AES-CTR DRBG output
AES-CTR DRBG Entropy Input	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output
AES-CTR DRBG V	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output
AES-CTR DRBG key	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output
DUAL EC DRBG Seed	A secret value maintained internal to the module that provides the seed material for DUAL EC DRBG output
DUAL EC DRBG Entropy Input	A secret value maintained internal to the module that provides the entropy material for DUAL EC DRBG output
DUAL EC DRBG V	A secret value maintained internal to the module that provides the entropy material for DUAL EC DRBG output
DUAL EC DRBG key	A secret value maintained internal to the module that provides the entropy material for DUAL EC DRBG output
Disk Encrypting Key (DEK)	A symmetric key used to encrypt a fixed or removable storage volume, also known as the FVEK.
Key Encrypting Key (KEK)	A symmetric key used to protect the DEK, also known as the VMK.
Full Volume Encryption Key (FVEK)	An AES key used to encrypt data on disk sectors.
Volume Master Key (VMK)	An AES key used to decrypt the FVEK.
Intermediate Key (IK)	An AES key value that is stored encrypted and forms the basis of a key which decrypts the VMK, by combining with another 256-bit AES key via key derivation or XOR.
External Key (ExK)	An AES key stored outside the cryptographic boundary (for example a USB device). This key is entered into the module via the USB port and is the only method used to decrypt the VMK that results in the VMK being considered encrypted, as other methods rely upon non-

	approved key derivation methods. Logically, the external key represents either a startup key or a recovery key.
Clear Key	An AES key that is used to decrypt the FVEK after the administrator has decided to disable BitLocker. This key is never generated when BitLocker operates within the evaluated configuration.

These keys and critical security parameters are zeroized, using the process described above, when they are no longer needed.

Windows will use the certificate embedded in the Microsoft Update Standalone Package (.msu) file to verify the digital signature of the update package. Windows 8 and Server 2012 contain two certificates: a legacy 2048 bit RSA certificate with a SHA-1 digest and a 2048 bit RSA certificate with a SHA-256 digest. The Code Integrity service within Windows validates the signature and checks that the certificate has not been revoked. Like any other data, an update package can be received from either a network or by removable storage.

6.1.2 SFR Mapping

- **FCS_CKM_EXT.4**: Windows overwrites critical cryptographic parameters immediately after that data is no longer needed.
- **FCS_COP.1(SYM)**: See **Table 6-1 Cryptographic Standards and Evaluation Methods**.
- **FCS_COP.1(SIGN)**: See **Table 6-1 Cryptographic Standards and Evaluation Methods**.
- **FCS_COP.1(HASH)**: See **Table 6-1 Cryptographic Standards and Evaluation Methods**.
- **FCS_RBG_EXT.1**: See **Table 6-1 Cryptographic Standards and Evaluation Methods**.

6.2 TPM Background Information

Computers that incorporate a TPM have the ability to create cryptographic keys and encrypt them so that they can be decrypted only by the TPM. This process, often called "wrapping" or "binding" a key, can help protect the key from disclosure. Each TPM has a master "wrapping" key, called the Storage Root Key (SRK), which is stored within the TPM itself. The private portion of a key created in a TPM is never exposed to any other component, software, process, or user.

Computers that incorporate a TPM can also create a key that has not only been wrapped, but also tied to certain platform measurements. This type of key can only be unwrapped when those platform measurements have the same values that they had when the key was created. This process is called "sealing" the key to the TPM. Decrypting it is called "unsealing." The TPM can also seal and unseal data generated outside of the TPM, i.e., generated by the operating system. [18]

6.3 BitLocker

BitLocker operates in two environments. In the **preboot environment**, which is the execution context after the firmware has started a boot manager, BitLocker will perform platform integrity tests, check authorization access requests to the encrypted drive, and then unlock the encrypted drive if the

[18] This section was copied from http://technet.microsoft.com/en-us/library/cc766159(v=WS.10).aspx.

integrity and authorization checks pass. In the normal OS runtime environment, BitLocker will unlock data drives based on the authorization factors for authorized users, and manage BitLocker capabilities.

6.3.1 Storage Volumes

BitLocker can encrypt several types of storage volumes:

- An **operating system drive** which contains an OS that the Boot Manager will boot into.
- **Fixed data drives** which contain user data.
- **Removable data drives** which contain user data.

6.3.2 Authorization Factors

In order to unlock, or authorize access, to a drive, BitLocker will check one or more authorization factors. The set of authorization factors which can be used to unlock the drive will depend on the type of storage volume since some authorization factors are not available in the preboot environment.

For the Windows 8 family of operating systems the set of authorization factors are:

Table 6-3 TOE BitLocker Authorization Factors

Protector	Input into the TOE by	Unlocks	Part of evaluated configuration?
TPM	TPM	Operating System Drive	No
TPM + PIN	TPM + keyboard	Operating System Drive	No
TPM + Start-up Key	TPM + USB	Operating System Drive	Yes
TPM + PIN + Start-up Key	TPM + keyboard + USB	Operating System Drive	Yes
TPM + Enhanced PIN	TPM + keyboard	Operating System Drive	No
TPM + Enhanced PIN + USB [Start-up Key]	TPM + keyboard + USB	Operating System Drive	Yes
External Key[19]	USB	Operating System Drive and Data Volume	Yes
Recovery Password	Keyboard	Operating System Drive and Data Volume	No
Clear Key[20]	See below	Operating System Drive	No
Passphrase	Keyboard	Data Volumes (and Windows To Go) and Operating System Drive	Yes
Public Key (RSA and ECDH)[21]	Smart card	Data Volume	No

[19] This can be startup key or a recovery key.
[20] BitLocker generates the clear key when the administrator chooses to suspend BitLocker.

To unlock the operating system drive, the user must supply all of the configured protectors; to unlock data volumes, the user can supply any configured protector.

While BitLocker can operate without a TPM chip, BitLocker's functionality is limited to disk encryption and will not include platform integrity verification because BitLocker measurements and authorization factors cannot be sealed to the TPM.

The start-up key can be any storage device that is recognized by the BIOS or UEFI firmware. Typically, it will be a USB drive.

The PIN is a 4 – 20 digit number that is typed either using the numeric keys on the keyboard or the F1 – F10 keys with F10 representing 0. The enhanced PIN includes uppercase and lowercase English letters, symbols on an EN-US keyboard, numbers, and spaces. Once an administrator chooses to enable enhanced PINs, all new PINs will be treated as enhanced PINs. Keyboards with non-English characters may not be able to use enhanced PINs.

The two protectors for data volumes are the passphrase and public key protectors. The passphrase protector is a string in the user's selected locale, and the public key protector is the public key portion of a RSA or ECDH key pair that is typically stored on a smart card.

The recovery password is a 48 digit password that a user can provide to the BitLocker preboot environment in case they have lost the start-up key. An additional recovery method is the external key which is used by Data Recovery Agents. BitLocker prompts the user to save it as a file or print the recovery password.

The clear key is generated by BitLocker when the administrator decides to suspend BitLocker; this results in the Windows booting from the encrypted disk without a user-supplied authorization factor. Users who need to follow the procedures in the *Supplemental Admin Guidance for Software Full Disk Encryption* should not suspend BitLocker because the computer will no longer be operating in the evaluated configuration.

When Windows detects a change in the UEFI or BIOS configuration or needs to modify one of the files that are used in the early boot measurements described below, it will place BitLocker into "recovery" mode for system maintenance. In recovery mode BitLocker will generate the clear key, which is used to protect the VMK. When Windows exits recovery mode, BitLocker will delete the clear key and generate a new VMK.

The user specifies their human-readable PIN, enhanced PIN, and passphrase protectors; BitLocker generates the values for the other authorization factors, however the administrator may optionally specify a recovery password. BitLocker conditions these into authorization factors, which is described below.

In the preboot environment, BitLocker will search for authorization factors in this order:

- Clear Key

- External Key (USB)
- TPM
- TPM + [Enhanced] PIN
- TPM + Start-up Key or External Key
- TPM + PIN + Start-up Key
- Recovery Password / External Key (USB)

The second check for an external key is to determine if BitLocker should switch to recovery mode.

6.3.3 BitLocker Keys

BitLocker uses several keys to protect storage volumes, to provide a large set of authorization providers, and to provide recovery options in the case of a lost or forgotten password. The following table describes the different keys, key lengths, and operational properties.

Table 6-4 TOE BitLocker Keys and Security Attributes

Key or Input for Derived Key	Length	Algorithm in which is used	Visible to user	Storage Place (default)
FVEK	128, 256	AES	No	On disk (metadata)
VMK	256	AES	No	On disk (metadata)
SRK	2048	RSA	No	TPM
Start-up Key	256	AES	Yes	External device
Recovery Key	256	AES	Yes	External device
(Enhanced) PIN	4 to 20 digits	SHA256	Yes	-
Passphrase	8 or more characters	SHA256	Yes	-

The Full Volume Encryption Key (FVEK) encrypts or decrypts sectors on a storage volume; the Software Full Disk Encryption Protection Profile describes this as the Disk Encrypting Key (DEK). Copies of the FVEK are stored in multiple places on the encrypted disk, and it is protected by the Volume Master Key (VMK).

Windows generates the Full Volume Encryption Key (FVEK) by requesting the Windows FIPS 140 Approved random bit generator for 128 or 256 bits using the BCryptGenRandom function.[22] Note that the Software Full Disk Encryption Protection Profile limits the allowable size to be either 128 or 256 bits.

The OS always obtains the Full Volume Encryption key by AES CCM decryption using the 256 bit Volume Master Key, which is also generated by the Windows RBG. The decrypted FVEK is stored in kernel memory.

[22] Windows does not distinguish between this and “normal” operations.

When BitLocker uses more than one authorization factor, the factors need to be combined prior to generating a VMK. The Software Full Disk Encryption Protection Profile refers to the resulting keys as "intermediate keys" (IK), this security target refers to the result of the combined authorization factors to generate an intermediate key as a composed key. The process to derive a composed key will vary based on which authorization factors are specified for a storage volume. The next section describes the steps to derive the composed key(s) and VMK from the authorization factors.

6.3.4 VMK Protection

This section describes how BitLocker protects and accesses the VMK through the various authorization factors and is arranged in the order that BitLocker will search for authorization factors (i.e., keys). These authorization factors are then combined to form an intermediate key that decrypts the VMK. This section also mentions cases where the length of authorization factors or intermediate keys is truncated or stretched.

6.3.4.1 Clear Key

The clear key is a 256 bit value, which is stored unencrypted on the disk, that AES encrypts/decrypts the VMK. The clear key is used when BitLocker goes into suspended mode, which is not part of the evaluated configuration. When BitLocker leaves suspended mode it will generate a new VMK.

6.3.4.2 Start-up Key Only (USB Token)

The OS reads the 256 bit start-up key from the USB token, and decrypts the AES key wrapped blob to obtain the VMK using the start-up key without any additional key derivation. The VMK is then used to AES CCM decrypt the FVEK.

6.3.4.3 Recovery or External Key

When a recovery or external key is used, the VMK is decrypted using the same method as for a start-up key.

6.3.4.4 TPM Only

BitLocker provides the encrypted VMK blob[23] to the TPM, which unseals the VMK using the SRK only when the current Platform Configuration Registers match the original values used when the blob was encrypted. The VMK then decrypts the FVEK.[24]

6.3.4.5 TPM and Start-up Key (USB Token)

The OS uses the Windows random bit generator to generate both a 256 bit intermediate key (IK1) which is sealed/unsealed using the SRK in the TPM, and a 256 bit start-up key which is stored on the USB token. These two values are XOR'ed to produce a 256 bit composed key, CK_{VMK}, which decrypts/encrypts the VMK using AES Counter with CBC-MAC (CCM) mode.[25]

[23] The blob includes the encrypted VMK and metadata such as the initialization vector.
[24] While the protection profile does not consider TPM-only to be a valid option, this explanation is a foundation for the other TPM options.
[25] CK_{VMK} is a Key Encrypting Key using the terminology from the protection profile.

6.3.4.6 TPM and PIN

BitLocker obtains two 128 bit random numbers from the Windows random bit generator, these are known as $Salt_1$ and $Salt_2$. The user enters a 4 – 20 character numerical or enhanced PIN that is passed through a key stretching algorithm, which will hash the PIN with $Salt_1$ 2^{20} times to produce a 256-bit value. Call this result AuthData. The same PIN data is passed through the key stretching algorithm again, this time using $Salt_2$ to produce a second 256-bit value. Call this result Remix. AuthData serves as a composed key ($CK_{TPM+PIN}$) and is sealed to the TPM as described above to protect the VMK. During boot, the composed key $CK_{TPM+PIN}$ is then XOR'ed with Remix to produce another composed key, CK_{VMK}, which decrypts/encrypts the VMK.[26]

6.3.4.7 TPM and PIN and Start-up Key

BitLocker obtains two 128 bit random numbers from the Windows random bit generator, these are known as $Salt_1$ and $Salt_2$. The user enters a 4 – 20 character numerical or enhanced PIN that is passed through a key stretching algorithm, which will hash the PIN with $Salt_1$ 2^{20} times. Call this 256-bit result AuthData. The same PIN data is passed through the key stretching algorithm again, this time using $Salt_2$. Call this 256-bit result Remix. AuthData serves as a composed key ($CK_{TPM+PIN}$) and is sealed to the TPM as described in the previous section. The composed key $CK_{TPM+PIN}$ is then XOR'ed with Remix to produce another composed key, $CK2_{TPM+PIN}$ which is, XOR'ed with the start-up key to produce the composed key CK_{VMK}, which decrypts/encrypts the VMK using AES Counter with CBC-MAC (CCM) mode.[27]

6.3.4.8 Passphrase

The passphrase is a character string that was selected by the authorized administrator during BitLocker initialization to protect fixed data drives and removable drives. The passphrase, represented as an array of characters, is directly hashed with a SHA-256 function to produce a 256 bit initial hash value which is then hashed again with a SHA-256 function to produce a 256 bit intermediate value. The resulting value is concatenated with the results of the previous round of computation, a salt, and a counter which is then passed into a SHA-256 function. After 2^{20} rounds through this hash function, the final result is the 256 bit composed key, CK_{VMK}, which decrypts/encrypts the VMK for the drive using AES Counter with CBC-MAC (CCM) mode.

6.3.4.9 Public Key

BitLocker supports using RSA and ECDH X.509 certificates as authorization factors for the data drives, however the process to decrypt the VMK is different for each type of certificate. These certificates are stored on a smart card, i.e. as an external token, and are presented to BitLocker using CNG.

Note that the Public Key protector was not included in the evaluation due to lack of assurance activities for public key authorization factors.

[26] CK_{VMK} is a Key Encrypting Key using the terminology from the protection profile.
[27] CK_{VMK} is a Key Encrypting Key using the terminology from the protection profile.

6.3.4.10 Recovery Password

If BitLocker is not able to access the VMK using any of the authorization mechanisms, the user will be prompted with a recovery screen that asks the user to provide a recovery password that will unlock the operating system drive.

If the user cannot unlock the drive using the recovery password, they will be prompted to insert the USB flash drive that holds the recovery key, and then restart the computer.

Note that the Recovery Password was not included in the evaluation.

6.3.5 PINs, Recovery Passwords, and Passphrases

BitLocker has separate user authorization factors for PINs, enhanced PINs, recovery passwords, and passphrases.

PINs are digits which are typed using either the keys for 0 – 9 on the keyboard or F1 – F10, with F10 representing 0. The enhanced PIN includes number, uppercase and lowercase English letters, symbol keys on an EN-US keyboard, numbers, and spaces.

The passphrase is the is the part of BitLocker covered by the FCS_CKM.1(PASS) functional requirement.

The supplemental CC administrative guidance specifies that the passphrases will contain at least sixty-four characters. The allowed set of characters includes numbers 0 – 9, upper and lower English letters and the following symbols: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”)”.

When a user decides to change their passphrase, BitLocker generates a new VMK using the same process described above in section 6.3.4.8 Passphrase.

The recovery password is not part of the evaluated configuration and the administrator should not generate a recovery password while managing BitLocker.

6.3.6 Start-up Key

BitLocker uses the Windows RBG to generate the random bit string that serves as the start-up key. BitLocker then stores the start-up key on an external store provided by the authorized administrator, typically this will be a USB drive.

6.3.7 Recovery Keys

Functionally, a recovery key is the same as a start-up key. When BitLocker is initialized for the operating system drive, the authorized administrator has the option to create a recovery key for the drive. If a user has forgotten their PIN or lost their start-up key, they can use the recovery key to unlock the operating system drive and then boot Windows.

After the authorized administrator has decided to generate a recovery key for the operating system volume, the only way to prevent the recovery key from being used is to turn BitLocker off and then turn it on again.

6.3.8 SFR Mapping

- **FCS_CKM.1(DEK)**: Windows uses a FIPS Approved and FIPS Validated random bit generator to generate a 128 bit or 256 bit BitLocker Full Volume Encryption Key (FVEK).
- **FCS_CKM.1(KEK)**: Windows derives the BitLocker Volume Master Key (VMK) based the administrator-specified combination of a PIN, passphrase, external token, or TPM-protected authorization factor.
- **FCS_CKM.1(PASS)**: The administrative guidance provides instructions to select passphrases that have the appropriate length and number of words to serve as an authorization factor.
- **FCS_CKM.1(TOKEN)**: Windows uses a FIPS Approved random bit generator to generate a 256 bit External Key (ExK), and stores the External Key on either a USB device or a TPM.
- **FCS_COP.1(SYM)**: Windows uses a FIPS Approved and FIPS Validated AES implementation to encrypt and decrypt disk sectors.
- **FCS_COP.1(SIGN)**: Windows uses a FIPS Approved and FIPS Validated RSA implementation to verify digital signatures for Microsoft Update Standalone Package and device drivers.
- **FCS_COP.1(HASH)**: Windows uses a FIPS Approved and FIPS Validated SHA-1 and SHA-2 implementations as part of digital signature verification.
- **FCS_COP.1(MASK)**: Windows protects the Full Volume Encryption Key using either AES CCM or AES Key Wrap, depending on the authorization factors used to protect the storage volume.

6.4 User Data Protection

6.4.1 TSS Description

This description applies to all fixed and removable storage volumes which the administrator deems to protect using BitLocker. A storage volume is a resource managed by the operating system's kernel, requests for access to the volume by users, or their subject processes, is regulated by the kernel. When the volume is recognized by Windows Plug-and-Play, a set of layered device drivers will read from or write to the volume. The NT File System driver, ntfs.sys, is the driver in the lowest level of the stack that performs the read/write operations. Above the NT File System driver is the BitLocker File Encryption drivers, fvevol.sys, which encrypts (or decrypts) data prior to the NT File System driver writing the data blocks to disk. During normal operations, all reads and writes to the disk pass through the NTFS I/O stack, BitLocker cannot be bypassed. In cases when the operating system halts, either due to a planned transition to a hibernated state or during a system crash, the BitLocker Drive Encryption Crash Dump Filter ensures that the data written to the hibernation file or crash dump file is encrypted in the same manner as the BitLocker File Encryption driver. BitLocker will use a different initialization vector, which is generated by the Windows RBG, for each encrypted volume.

After the computer has been switched on and power-on self-tests have completed, the firmware invokes the boot manager for the operating system. The Windows Boot Manager is responsible for (1) capturing and verifying the authorization factors required to unlock the OS volume and (2) loading and verifying the integrity of the Windows OS Loader, winload.exe or winload.efi, and Windows OS Resume, winresume.exe. Depending on the underlying hardware, the Boot Manager executable will be one of the following: BOOTMGR, bootmgr.exe, bootmgfw.efi, bootmgr.efi. After verifying the integrity of OS Loader

or OS Resume, the Boot Manager hands control to those programs which will either complete initialization of the OS or reload the OS into memory. Since at this point the kernel and the file system device driver stack have not been loaded, the Boot Manager, OS Loader, and OS Resume, all read from the encrypted volume. An administrator can choose to encrypt the entire disk (including both used and free space) or only the used space.

On disk, the DEK is always protected by the KEK. In the case of hibernation or a system crash, the in-memory copy of the DEK is encrypted by the BitLocker Drive Encryption Crashdump Filter prior to the hibernation file or crash dump file being written to disk.

Apart from the master boot record and the Boot Manager, the storage volume is encrypted.

All cryptographic operations are implemented within Windows using FIPS Approved algorithms as described above in section 6.3 BitLocker.

6.4.1.1 Power Management and BitLocker

A computer running a Windows operating system provides several power management states ranging from the switched-off machine to a computer with all accessory devices powered up. For an ACPI-based computer, these states are: [28]

Table 6-5 Windows Power States

Power State	ACPI State	Description
Working	S0	The system is fully usable. Devices that are not in use can save power by entering a lower power state.
Sleep	S1, S2, S3	The system appears to be off. Power consumption is reduced to one of several levels, depending on how the system is to be used. The lower the level of power consumption, the more time it takes the system to return to the working state.
Hibernation	S4	The system appears to be off. Power consumption is reduced to the lowest level. The system saves the contents of memory to a hibernation file, preserving the state of the operating system, applications, and open documents.
Soft Off	S5	The system appears to be off. Some components remain powered so the computer can wake from input from the keyboard, LAN, or a USB device. The working context can be restored if it is stored on nonvolatile media.
Mechanical Off	G3	The system is completely off and consumes no power. The system returns to the working state only after a full reboot.

In the context of BitLocker, the relevant power states are mechanical off (G3), soft off (S5) working (S0), hibernation (S4), because Connected Standby operates in the working (S0) state and the sleep states

[28] From System Power States, http://msdn.microsoft.com/en-us/library/windows/desktop/aa373229(v=vs.85).aspx

(S1, S2, and S3) are not entered by Windows. The following diagram illustrates the power transitions between states:

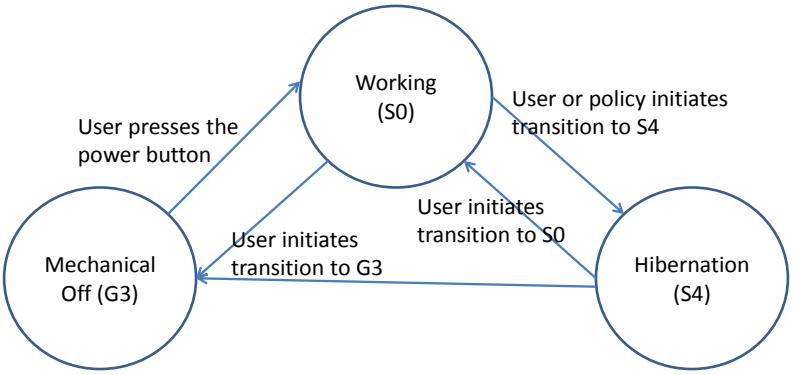


Figure 1 BitLocker and Windows Power States

The previous section described BitLocker operations for startup and both planned and unplanned shutdowns. When Windows enters the hibernation state, the snapshot of the system's memory and related metadata are saved in a hibernation file. By default the hibernation file is in the root directory of the operating system drive, and when that drive is protected by BitLocker, the hibernation file will be encrypted. The typical action that a user will take to exit the hibernation state is to either open a laptop lid or press the power button, depending on the policy for that computer. Before attempting to unlock the operating system drive, BitLocker will prompt the user for any authorization factors. If all authorization factors are correct, BitLocker will load the hibernation file into memory and restore the state of the processor(s). The option to hibernate is not enabled by default for Windows 8 and Server 2012. When transitioning from S4, S5, or the G3 power states, BitLocker will prompt the user to provide authorization factors to unlock the operating system drive.

6.4.2 SFR Mapping

The **User Data Protection** function satisfies the following SFR:

- **FDP_DSK_EXT.1**: Using BitLocker, Windows encrypts all data on fixed and removable storage volumes using FIPS Approved cryptographic algorithms without exposing the disk encrypting key.

- **FDP_PM_EXT.1**: When a computer is turned on, or resumes from hibernation, Windows will prompt the user for authorization factor(s) before booting the operating system or resuming from hibernation.

6.5 Identification and Authentication

6.5.1 TSS Description

Section 6.4, User Data Protection, describes Windows initialization and authorization to the bootable disk volume by the Boot Manager. The Boot Manager verifies that the authorization factors are correct during boot; during runtime, the Windows kernel will verify the user-provided authorization factors for other fixed drives and all removable drives. This verification is the result of computing the intermediate composed key (as described in **VMK Protection**), which is then used to decrypt the encrypted VMK blob. If the metadata fields in the blob match the expected values then Windows will attempt to decrypt the FVEK using the VMK. The DEK is not exposed because it is stored on disk in an encrypted form; the KEK is computed only when all of the authorization factors have been successfully verified. During boot, the user has the ability to provide only a PIN or a startup key (in addition to the system-provided TPM value), the DEK and KEK are never exposed.

6.5.2 SFR Mapping

The **Identification & Authentication** function satisfies the following SFRs:

- **FIA_AUT_EXT.1**: Windows ensures that a user provides one or more authorization factors prior to allowing the operating system to start, thus preventing access to user on the device.

6.6 Security Management

6.6.1 TSS Description

Windows contains all the functionality needed to manage BitLocker, there are no external dependencies.

Windows provides several tools to manage BitLocker, including:

- **Windows Explorer** which can be used to manage BitLocker and check the integrity of Windows files and updates.
- The **manage-bde** console application to manage BitLocker.
- The **Get-AuthenticodeSignature PowerShell Cmdlet** which can be used to confirm the signatures of Windows program files.
- **PowerShell Cmdlets** to manage BitLocker:
    - **Add-BitLockerKeyProtector**
    - **Backup-BitLockerKeyProtector**
    - **Clear-BitLockerAutoUnlock**
    - **Disable-BitLocker**
    - **Disable-BitLockerAutoUnlock**
    - **Enable-BitLocker**

  - **Enable-BitLockerAutoUnlock**
  - **Get-BitLockerVolume**
  - **Lock-BitLocker**
  - **Remove-BitLockerKeyProtector**
  - **Resume-BitLocker**
  - **Suspend-BitLocker**
  - **Unlock-BitLocker**
- **Local and Group policies** to manage BitLocker

The group policies for BitLocker are located under \Computer Configuration\Administrative Templates\Windows Components\BitLocker Drive Encryption. This security target only mentions policies that directly affect the security functional requirements.

The operating system drive policy "Choose how BitLocker-protected operating system drives can be recovered" specifies a collection of recovery options which can be enabled or disabled that includes recovery passwords, recovery keys, whether to archive recovery information for the computer in the Active Directory, and data recovery agents. In the evaluated configuration, the administrator should not choose the "Allow data recovery agent" option, although by default, BitLocker will not archive recovery information to Active Directory (AD) or use a data recovery agent. The local recovery key and password are presented directly to the user as a file, the recovery information in AD is available to any administrator who can read the attributes in AD.

The FVEK is generated implicitly when the administrator initializes BitLocker for a storage volume. When BitLocker is initialized for a volume that already contains data, that data is encrypted as part of background processing. A user can check on the progress of the encryption using either Windows Explorer or the `manage-bde -status` command.

Authorization factors are managed on a per-volume basis. Management functions to initialize, modify, or remove an authorization factor are typically the same for the operating system drive and data drives but there are a few exceptions:

- The `changekey` option in `Manage-bde` applies only to the operating system drive.
- The `autounlock` option in `Manage-bde` should not be set because that will bypass user authorization to unlock a volume.

An administrator can use the `Enable-BitLocker` PowerShell cmdlet to specify either AES-128 or AES-256 encryption for the storage volume, however those are the only cryptographic settings which can be managed.

If the administrator desires to use a different FVEK, they must turn off and then turn on BitLocker for the disk volume; however a different VMK will be generated whenever an authorization factor is changed.

Note that the administrator must follow the instructions in the *Supplemental Admin Guidance for Software Full Disk Encryption* to encrypt all of the locally attached storage volumes for the computer.

6.6.2 SFR Mapping

- **FMT_SMF.1**: Windows provides the authorized administrator with the capability to administer the security functions described in the security target. The mappings to specific functions are described in each applicable section of the TOE Summary Specification.

6.7 TSF Protection

6.7.1 TSS Description

6.7.1.1 Windows Platform Integrity Tests

On computers with a TPM before BitLocker will unlock the operating system drive, it will verify the integrity of the early boot components in order to prevent tampering and to ensure that the drive is in the same computer as when BitLocker was initialized.

BitLocker checks that the file integrity of early boot components has not been compromised and ensures that the files have not been modified, which mitigates the risk of rootkits and viruses, and that the data elements that contribute to creating the composite keys, which will ultimately unlock the operating system drive, have not been compromised.

BitLocker collects these file measurements and seals them to the TPM as described in section 6.2, TPM Background Information. In addition, when a PIN is used as an authorization factor, BitLocker will also seal the derived PIN value to the TPM.

When BitLocker starts in the preboot environment, it will unseal these values from the TPM and if those values do not match the calculated values, BitLocker will lock the system (which prevents booting) and display a warning on the computer monitor.

6.7.1.2 Windows Cryptographic Self-Tests

The Windows self-tests are a collection of tests which verify that the Windows is operating correctly. The self-tests are enabled when the administrator sets the “System Cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing” policy; Windows will always run the self-tests described in this section.

The kernel-mode startup self-tests are:[29]

- AES-128 encrypt/decrypt EBC Known Answer Test
- AES-128 encrypt/decrypt CBC Known Answer Test
- AES-128 CMAC Known Answer Test
- AES-128 encrypt/decrypt CCM Known Answer Test
- AES-128 encrypt/decrypt GCM Known Answer Test
- RSA Known Answer Test
- ECDSA sign/verify test on P256 curve
- ECDH secret agreement Known Answer Test on P256 curve

[29] When the System Cryptography policy is set, Windows will always perform these self-tests however the evaluated configuration does not use the ECDH, HMAC, and SP800-56A algorithms.

- HMAC-SHA-1 Known Answer Test
- HMAC-SHA-256 and HMAC-SHA-512 Known Answer Tests
- SP800-56A concatenation KDF Known Answer Tests (same as Diffie-Hellman KAT)
- SP800-90 AES-256 counter mode DRBG Known Answer Tests (instantiate, generate and reseed)
- SP800-90 Dual-EC DRBG Known Answer Tests (instantiate, generate and reseed)

The Windows kernel-mode cryptographic module, the Kernel Mode Cryptographic Primitives Library, also performs pair-wise consistency checks upon each invocation of RSA, ECDH, and ECDSA key-pair generation and import as defined in FIPS 140-2. SP 800-56A conditional self-tests are also performed. A continuous RNG test (CRNGT) is used for the random number generators of this cryptographic module. All approved and non-approved RNGs have a CRNGT. The SP 800-90 DRBGs have health tests. A pair-wise consistency test is done for Diffie-Hellman.

The Kernel Mode Cryptographic Primitives Library is loaded into the kernel's memory early during the boot process. If there is a failure in any startup self-test, the Kernel Mode Cryptographic Primitives Library DriverEntry function will fail to return the STATUS_SUCCESS status to its caller. The only way to recover from the failure of a startup self-test is to attempt to invoke DriverEntry again, which will rerun the self-tests, and will only succeed if the self-tests passes.

By thoroughly exercising the cryptography used for Software Disk Encryption, Windows will prevent situations where user data is not stored in an encrypted state.

All operations on the TSF ultimately involve the use of cryptography, and so the FIPS 140 health tests are sufficient to determine that BitLocker is operating correctly.

6.7.1.3 Windows Code Integrity

A Windows operating system verifies the integrity of Windows program code using the Code Integrity capability in Windows. Kernel-mode code signing (KMCS) prevents kernel-mode device drivers, such as the BitLocker Drive Encryption Drivers (fvevol.sys), from loading unless they are published and digitally signed by developers who have been vetted by one of a handful of trusted certificate authorities (CAs). KMCS, using public-key cryptography technologies, requires that kernel-mode code include a digital signature generated by one of the trusted certificate authorities. When a kernel device driver tries to load, Windows decrypts the hash included with the driver using the public key stored in the certificate, then verifies that the hash matches the one computed with the code. The authenticity of the certificate is checked in the same way, but using the certificate authority's public key, which is trusted by Windows. The root public key of the certificate chain that used to verify the signature must match one of the Microsoft's root public keys indicating that Microsoft is the publisher of the Windows image files. These Microsoft's root public keys are hardcoded in the Windows boot loader.

Starting with the Windows 8 wave of operating systems, code integrity additionally supports larger digital certificates and hash sizes.

The cryptography used by Code Integrity is validated as part of the Windows FIPS 140 validation.

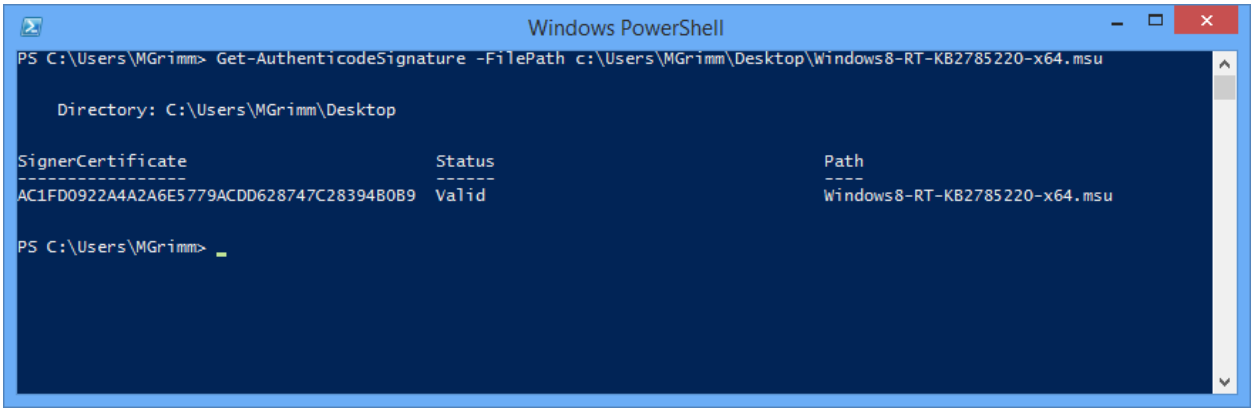
6.7.1.4 Windows Updates

Updates to Windows are delivered as Microsoft Update Standalone Package files (.msu files) and are signed by Microsoft with two digital signatures, a SHA1 signature for legacy applications and a SHA256 signature for modern applications. The RSA SHA256 digital signature is signed by *Microsoft Corporation*, with a certification path through a Microsoft Code Signing certificate and ultimately the Microsoft Root Certification Authority. These certificates are checked by the Windows Trusted Installer prior to installing the update.

The Windows operating system will check that the certificate is valid and has not been revoked using a standard PKI CRL. Once the Trusted Installer determines that the package is valid, it will update Windows; otherwise the installation will abort and there will be an error message in the event log.

Updates to Windows are delivered through the Windows Update capability, which is enabled by default, or the user can go to http://www.microsoft.com/security/default.aspx to search and obtain security updates on their own volition.

A user can then check that the signature is valid either by viewing the digital signature details of the file from Windows Explorer or by using the Get-`AuthenticodeSignature` PowerShell Cmdlet. The following is an example of using Powershell:



If the Get-`AuthenticodeSignature` PowerShell Cmdlet or Windows Explorer could not verify the signature, the status will be marked as invalid. This verification check uses the same functionality described above.

6.7.2 SFR Mapping

The **TSF Protection** function satisfies the following SFRs:

- **FPT_TST_EXT.1**: Windows runs a series of self-tests that confirm that essential cryptographic operations are performed correctly and halts if the self-tests fail. Those cryptographic functions are then used to check integrity of TOE executables.
- **FPT_TUD_EXT.1**: Windows has an update mechanism to deliver updated binaries and a means for a user to confirm that the digital signatures, which ensure the integrity of the update, are

valid. Windows provides a mechanism to check the current version of the product and binary files that comprise the product.

7 Rationale for the Protection Profile Conformance Claim

This security target is in strict compliance with Software Full Disk Encryption Protection Profile, version 1.1, March 31, 2014 (“SW FD PP”).

For all of the content incorporated from the protection profile, the corresponding rationale in that protection profile remains applicable to demonstrate the correspondence between the TOE security functional requirements and TOE security objectives.

The requirements in the Software Full Disk Encryption Protection Profile are assumed to represent a complete set of requirements that serve to address any interdependencies. Given that all of the functional requirements in the SW FD PP have been copied into this security target, the dependency analysis for those requirements is not reproduced here.

8 Rationale for Modifications to the Security Requirements

This section provides a rationale that describes how the Security Target reproduced the security functional requirements and security assurance requirements from the protection profile.

8.1 Functional Requirements

This Security Target includes security functional requirements (SFRs) that can be mapped to SFRs found in the protection profile along with SFRs that describe additional features and capabilities. The mapping from protection profile SFRs to security target SFRs along with rationale for operations is presented in Table 8-1 Rationale for Operations. SFR operations left incomplete in the protection profile have been completed in this security and are identified within each SFR in section 5.1 TOE Security Functional Requirements.

Table 8-1 Rationale for Operations

Protection Profile Requirement	Security Target Requirement	CC Operation and Rationale
FCS_CKM.1(1)	FCS_CKM.1(DEK)	A selection which is allowed by the PP.
FCS_CKM.1(2)	FCS_CKM.1(KEK)	Assignment and multiple selections which are allowed by the PP.
FCS_CKM.1(Y)	FCS_CKM.1(PASS)	Multiple assignments and selections which are allowed by the PP.
FCS_CKM.1(X)	FCS_CKM.1(TOKEN)	Two selections which are allowed by the PP.
FCS_CKM_EXT.4	FCS_CKM_EXT.4	Copied from the PP with no operations.
FCS_COP.1(1)	FCS_COP.1(SYM)	Three selections which are allowed by the PP.
FCS_COP.1(2)	FCS_COP.1(SIGN)	Three selections which are allowed by the PP.
FCS_COP.1(3)	FCS_COP.1(HASH)	Two selections which are allowed by the PP.
FCS_COP.1(4)	FCS_COP.1(MASK)	Multiple selections which are allowed by the PP.
FCS_RBG_EXT.1	FCS_RBG_EXT.1	Three selections which are allowed by the PP.
FDP_DSK_EXT.1	FDP_DSK_EXT.1	Copied from the PP with no operations.
FDP_PM_EXT.1	FDP_PM_EXT.1	Assignment which is allowed by the PP. Refinement to provide context for TSF operations.
FIA_AUT_EXT.1	FIA_AUT_EXT.1	Refinement to remove a reference to a SFR that is not in the protection profile.

Protection Profile Requirement	Security Target Requirement	CC Operation and Rationale
FMT_SMF.1	FMT_SMF.1	Assignment and multiple selections which are allowed by the PP.
FPT_TST_EXT.1	FPT_TST_EXT.1	Copied from the PP with no operations.
FPT_TUD_EXT.1	FPT_TUD_EXT.1	Copied from the PP with no operations.

8.2 Assurance Requirements

The statement of security assurance requirements (SARs) found in section 5.2 TOE Security Assurance Requirements, is in strict conformance with the assurance requirements in the Software Full Disk Encryption Protection Profile.

8.3 Rationale for the TOE Summary Specification

This section, in conjunction with section 6, the TOE Summary Specification (TSS), provides evidence that the security functions are suitable to meet the TOE security requirements.

Each subsection in section 6, TOE Security Functions (TSFs), describes a Security Function (SF) of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding SF. The set of security functions work together to satisfy all of the functional requirements. Furthermore, all the security functions are necessary in order for the TSF to provide the required security functionality.

The set of security functions work together to provide all of the security requirements as indicated in **Table 8-2**. The security functions described in the TOE Summary Specification and listed in the tables below are all necessary for the required security functionality in the TSF.

Table 8-2 Requirement to Security Function Correspondence

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection
FCS_CKM.1(DEK)		X				
FCS_CKM.1(KEK)		X				
FCS_CKM.1(PASS)		X				
FCS_CKM.1(TOKEN)		X				
FCS_CKM_EXT.4		X				

FCS_COP.1(SYM)		X				
FCS_COP.1(SIGN)		X				
FCS_COP.1(HASH)		X				
FCS_COP.1(MASK)		X				
FCS_RBG_EXT.1		X				
FDP_DSK_EXT.1			X			
FDP_PM_EXT.1			X			
FIA_AUT_EXT.1				X		
FMT_SMF.1					X	
FPT_TST_EXT.1						X
FPT_TUD_EXT.1						X

9 Appendix A: List of Abbreviations

Abbreviation	Meaning
3DES	Triple DES
ACE	Access Control Entry
ACL	Access Control List
ACP	Access Control Policy
AD	Active Directory
ADAM	Active Directory Application Mode
AES	Advanced Encryption Standard
AGD	Administrator Guidance Document
AH	Authentication Header
ALPC	Advanced Local Process Communication
ANSI	American National Standards Institute
API	Application Programming Interface
APIC	Advanced Programmable Interrupt Controller
BDE	BitLocker Drive Encryption
BTG	BitLocker To Go
CA	Certificate Authority
CBAC	Claims Basic Access Control, see DYN
CBC	Cipher Block Chaining
CC	Common Criteria
CD-ROM	Compact Disk Read Only Memory
CIFS	Common Internet File System
CIMCPP	Certificate Issuing and Management Components For Basic Robustness Environments Protection Profile, Version 1.0, April 27, 2009
CM	Configuration Management; Control Management
COM	Component Object Model
CNG	Cryptography API: Next Generation
CP	Content Provider

CPU	Central Processing Unit
CRL	Certificate Revocation List
CryptoAPI	Cryptographic API
CSP	Critical Security Parameter
CSP	Cryptographic Service Provider
DAC	Discretionary Access Control
DACL	Discretionary Access Control List
DC	Domain Controller
DEK	Disk Encryption Key
DEP	Data Execution Prevention
DES	Data Encryption Standard
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DFS	Distributed File System
DMA	Direct Memory Access
DNS	Domain Name System
DS	Directory Service
DSA	Digital Signature Algorithm
DYN	Dynamic Access Control
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
EFS	Encrypting File System
ESP	Encapsulating Security Protocol
FEK	File Encryption Key
FDE	Full Drive Encryption
FIPS	Federal Information Processing Standard
FRS	File Replication Service
FSMO	Flexible Single Master Operation
FTP	File Transfer Protocol
FVE	Full Volume Encryption
FVEK	Full Volume Encryption Key
GB	Gigabyte
GC	Global Catalog
GHz	Gigahertz
GPC	Group Policy Container
GPO	Group Policy Object
GPOSPP	US Government Protection Profile for General-Purpose Operating System in a Networked Environment
GPT	Group Policy Template
GPT	GUID Partition Table
GUI	Graphical User Interface
GUID	Globally Unique Identifiers
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP

I/O	Input / Output
I&A	Identification and Authentication
IA	Information Assurance
ICF	Internet Connection Firewall
ICMP	Internet Control Message Protocol
ICS	Internet Connection Sharing
ID	Identification
IDE	Integrated Drive Electronics
IETF	Internet Engineering Task Force
IFS	Installable File System
IIS	Internet Information Services
IK	Intermediate Key
IKE	Internet Key Exchange
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
IPC	Inter-process Communication
IPI	Inter-process Interrupt
IPSec	IP Security
ISAPI	Internet Server API
IT	Information Technology
IV	Initialization Vector
KDC	Key Distribution Center
KEK	Key Encrypting Key
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LPC	Local Procedure Call
LSA	Local Security Authority
LSASS	LSA Subsystem Service
LUA	Least-privilege User Account
MAC	Message Authentication Code
MB	Megabyte
MMC	Microsoft Management Console
MSR	Model Specific Register
NAC	(Cisco) Network Admission Control
NAP	Network Access Protection
NAT	Network Address Translation
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NLB	Network Load Balancing
NMI	Non-maskable Interrupt
NTFS	New Technology File System
NTLM	New Technology LAN Manager
OS	Operating System

PAE	Physical Address Extension
PC/SC	Personal Computer/Smart Card
PIN	Personal Identification Number
PKCS	Public Key Certificate Standard
PKI	Public Key Infrastructure
PP	Protection Profile
RADIUS	Remote Authentication Dial In Service
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAS	Remote Access Service
RBG	Random Bit Generator
RC4	Rivest's Cipher 4
RID	Relative Identifier
RNG	Random Number Generator
RPC	Remote Procedure Call
RSA	Rivest, Shamir and Adleman
RSASSA	RSA Signature Scheme with Appendix
SA	Security Association
SACL	System Access Control List
SAM	Security Assurance Measure
SAML	Security Assertion Markup Language
SAR	Security Assurance Requirement
SAS	Secure Attention Sequence
SD	Security Descriptor
SHA	Secure Hash Algorithm
SID	Security Identifier
SIP	Session Initiation Protocol
SIPI	Startup IPI
SF	Security Functions
SFP	Security Functional Policy
SFR	Security Functional Requirement
SMB	Server Message Block
SMI	System Management Interrupt
SMTP	Simple Mail Transport Protocol
SP	Service Pack
SPI	Security Parameters Index
SPI	Stateful Packet Inspection
SRM	Security Reference Monitor
SSL	Secure Sockets Layer
SSP	Security Support Providers
SSPI	Security Support Provider Interface
ST	Security Target
SYSVOL	System Volume
TCP	Transmission Control Protocol

TDI	Transport Driver Interface
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSC	TOE Scope of Control
TSF	TOE Security Functions
TSS	TOE Summary Specification
UART	Universal Asynchronous Receiver / Transmitter
UI	User Interface
UID	User Identifier
UNC	Universal Naming Convention
US	United States
UPN	User Principal Name
URL	Uniform Resource Locator
USB	Universal Serial Bus
USN	Update Sequence Number
v5	Version 5
VDS	Virtual Disk Service
VMK	Volume Master Key
VPN	Virtual Private Network
VSS	Volume Shadow Copy Service
WAN	Wide Area Network
WCF	Windows Communications Framework
WebDAV	Web Document Authoring and Versioning
WebSSO	Web Single Sign On
WDM	Windows Driver Model
WIF	Windows Identity Framework
WMI	Windows Management Instrumentation
WSC	Windows Security Center
WU	Windows Update
WSDL	Web Service Description Language
WWW	World-Wide Web
X64	A 64-bit instruction set architecture
X86	A 32-bit instruction set architecture