Microsoft Corporation

Boot Manager

FIPS 140-3 Non-Proprietary Security Policy Document

Microsoft Windows 11 version 22H2
(Pro, Enterprise, IoT Enterprise, Education, and Home Editions)

Microsoft Windows Server 2022
(Standard and Datacenter Editions)

Prepared By	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399
Document Version Number	1.0
Updated On	May 4, 2026

COPYRIGHT AND DISCLAIMER

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, Azure, 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

List of Figures

1 General

1.1 Overview

The Windows Boot Manager module (the “module”) is a UEFI application that sets up the boot environment for the Windows operating system to execute. The module implements FIPS 140-3 Approved cryptographic algorithms, and this document is the FIPS 140-3 Security Policy for the module. The Security Policy contains a specification of the rules under which the module must operate and describes how the module meets the requirements specified in Federal Information Processing Standards Publication 140-3 (FIPS PUB 140-3) and International Standard ISO/IEC 19790:2012 (Information technology – Security techniques – Security requirements for cryptographic modules). This document is intended for the FIPS 140-3 testing lab, the Cryptographic Module Validation Program (CMVP), and administrators and users of the module.

1.2 Security Levels

The overall security rating for the module is level 1. The table below lists the security levels of individual clauses for this validation. As a software-hybrid module executing in a modifiable environment, non-invasive security requirements are optional and are out of scope for this validation.

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	1
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	1
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Windows Boot Manager is a multi-chip standalone software-hybrid cryptographic module. Boot Manager is the first Windows OS component to load when the computer powers up. When Secure Boot is enabled, the integrity of Boot Manager is validated before loading by the computer’s UEFI firmware. Along with other startup and initialization tasks, Boot Manager loads and cryptographically validates the integrity of Winload.efi (the Windows OS Loader), the next module in the startup sequence.

The Boot Manager, which includes parts of BitLocker disk encryption, collects authorization factors, known as “protectors”, by reading data or interacting with the user. BitLocker uses these protectors to encrypt entire disk volumes.

Module Type: Software-hybrid

Module Embodiment: MultiChipStand

Module Characteristics:

Cryptographic Boundary:

The software-hybrid cryptographic boundary for Boot Manager consists of disjoint software and hardware components within the same physical perimeter of the host platform. The module's software components are the binaries listed in the following table, and its hardware component is the CPU running on the host platform.

Software Component	Description
BOOTMGR.EFI, BOOTMGFW.EFI, and BOOTX64.EFI	Binary files that contain the software component of the module.

Table 2: Module Software Components

Tested Operational Environment's Physical Perimeter (TOEPP):

The Tested Operational Environment's Physical Perimeter (TOEPP) is the physical perimeter of the computer that contains the module.

The following block diagram illustrates the module's components, physical perimeter (TOEPP), and cryptographic boundary. The cryptographic boundary of the module is the module software component, including the binaries BOOTMGR.EFI, BOOTMGFW.EFI, and BOOTX64.EFI. The binaries are loaded from the OS volume in physical storage and execute in the computer memory. The control input, data input / output, and status output of the module exist within the computer memory as well.

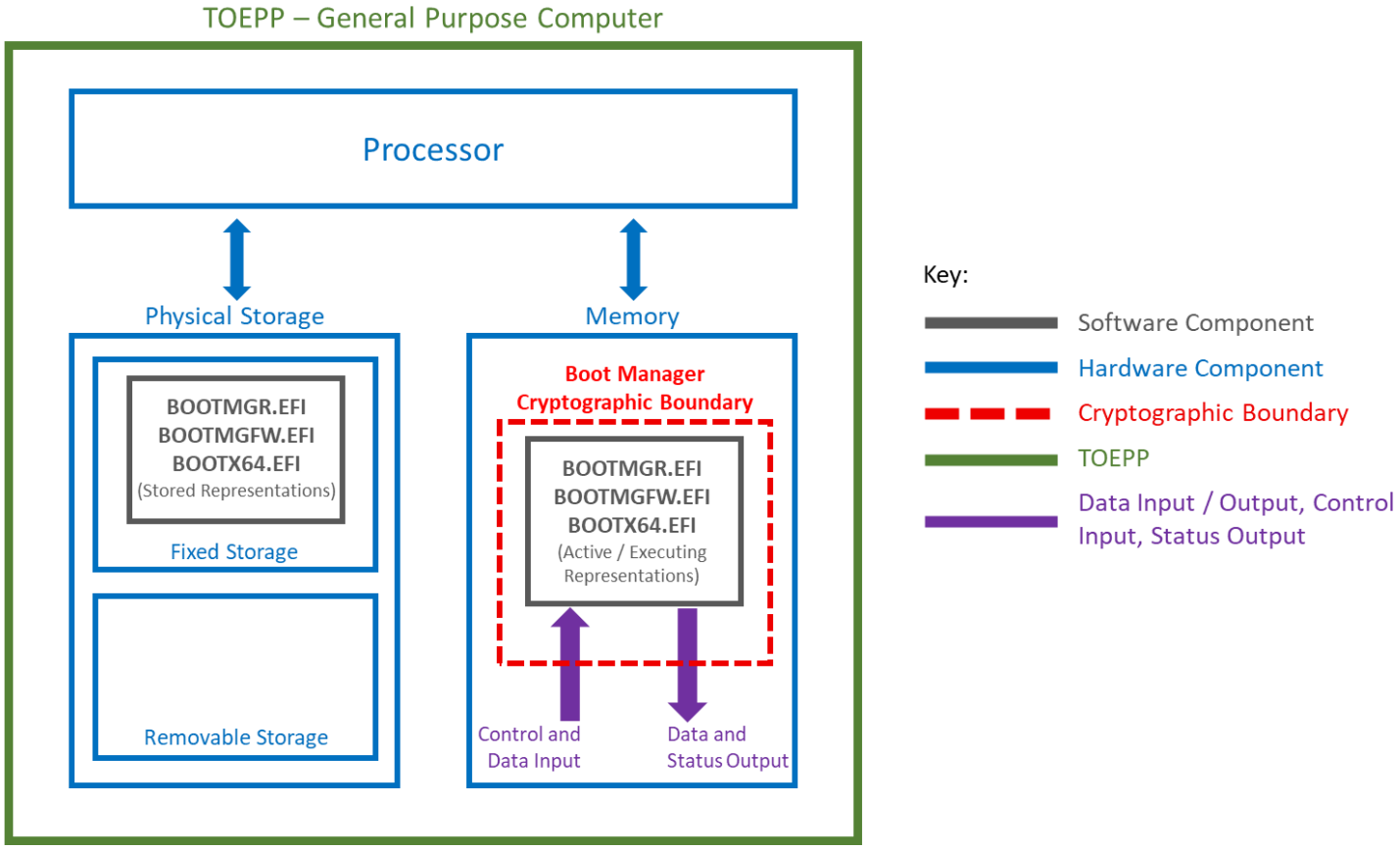
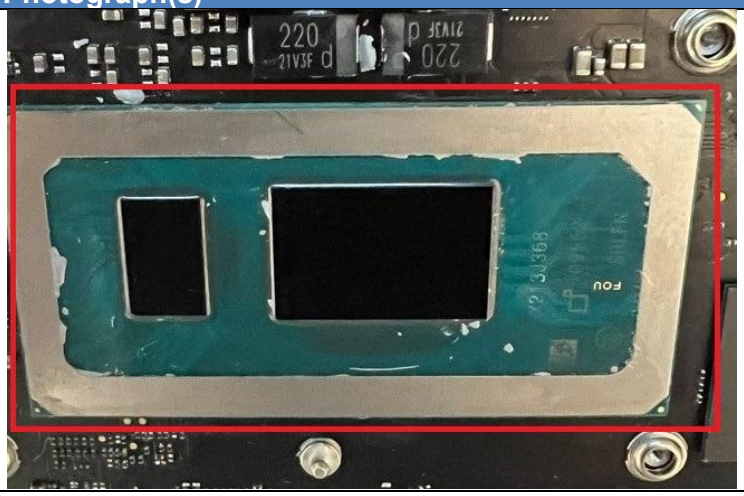


Figure 1: Module Boundary Diagram

The following table includes a photograph of the CPU of each computer listed in Section 2.2 Tested and Vendor Affirmed Module Version and Identification. The processor is highlighted by a red box for clarity. For laptop devices, the processor is shown as integrated into the motherboard. For server devices, the processor is shown both independently and as installed in the computer with its integral heat sink.

CPU	Photograph(s)
12th Gen Intel Core i7-1265U (Microsoft Surface Laptop 5)	

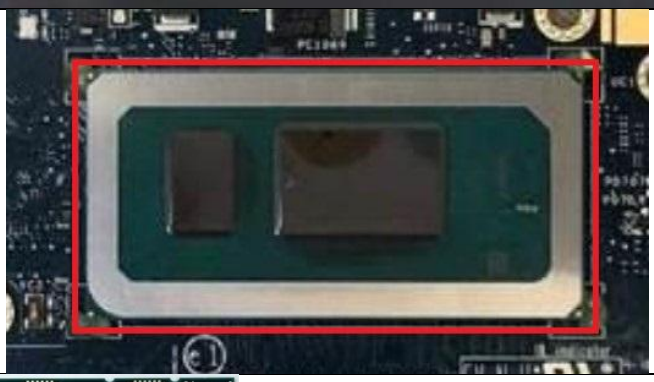
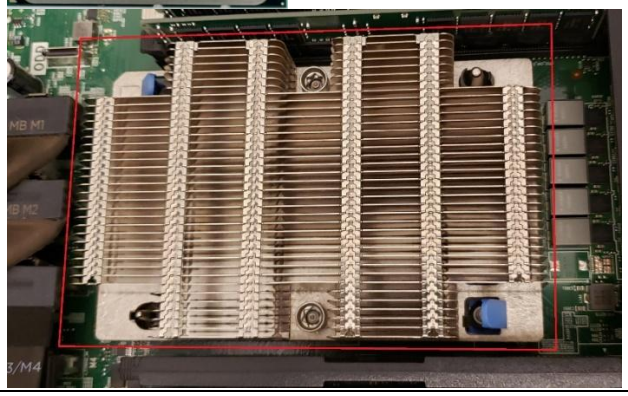
CPU	Photograph(s)
11th Gen Intel Core i5-11500H (HP ZBook Power G8)	
11th Gen Intel Core i7-1185G7 (Dell Latitude 7420)	
Intel Xeon Gold 6130 (Dell PowerEdge R640)	 

Table 3: CPU Photographs

2.2 Tested and Vendor Affirmed Module Version and Identification

This validation includes the following Windows products and versions, each of which can be identified by its build number. The cryptographic module is a distinct implementation in each product build and for each processor architecture. Some Windows products may be installed as different editions; however, the cryptographic module is the same implementation in different editions of the same product.

Windows Product	Build	Edition(s) in Scope
Windows 11 version 22H2	10.0.22621.1	Enterprise Edition Home Edition IoT Enterprise Edition Pro Edition Education Edition
Windows Server 2022	10.0.20348.1668 (including the March 2023 updates)	Standard Edition Datacenter Edition

Table 4: Version Information

Tested Module Identification – Hardware:

N/A for this module.

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
bootmgr.efi, bootmgfw.efi, and bootx64.efi (Windows 11 version 22H2)	Windows 11 version 22H2, build 10.0.22621.1	N/A	Yes
bootmgr.efi, bootmgfw.efi, and bootx64.efi (Windows Server 2022)	Windows Server 2022, build 10.0.20348.1668	N/A	Yes

Table 5: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Module Identification – Hybrid Disjoint Hardware:

Model and/or Part Number	Hardware Version	Firmware Version	Processors	Features
Dell Latitude 7420	11th Gen Intel Core i7-1185G7	N/A	11th Gen Intel Core i7-1185G7	N/A
Dell PowerEdge R640	Intel Xeon Gold 6130	N/A	Intel Xeon Gold 6130	N/A
HP ZBook Power G8	11th Gen Intel Core i5-11500H	N/A	11th Gen Intel Core i5-11500H	N/A
Microsoft Surface Laptop 5	12th Gen Intel Core i7-1265U	N/A	12th Gen Intel Core i7-1265U	N/A

Table 6: Tested Module Identification – Hybrid Disjoint Hardware

Tested Operational Environments - Software, Firmware, Hybrid:

The operational environment for the module is the Windows operating system running on a supported hardware platform, as listed in the table below. All hardware platforms in the table below are 64-bit Intel

architecture. The tested operational environments provide Processor Algorithm Acceleration (PAA) in the form of the Advanced Encryption Standard New Instructions (AES-NI).

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Windows 11 version 22H2, Education Edition	Dell Latitude 7420	11th Gen Intel Core i7-1185G7	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.1
Windows 11 version 22H2, Enterprise Edition	Microsoft Surface Laptop 5	12th Gen Intel Core i7-1265U	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.1
Windows 11 version 22H2, Home Edition	Microsoft Surface Laptop 5	12th Gen Intel Core i7-1265U	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.1
Windows 11 version 22H2, IoT Enterprise Edition	Microsoft Surface Laptop 5	12th Gen Intel Core i7-1265U	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.1
Windows 11 version 22H2, Pro Edition	HP ZBook Power G8	11th Gen Intel Core i5-11500H	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.1
Windows Server 2022 Datacenter, including the March 2023 Updates	Dell PowerEdge R640	Intel Xeon Gold 6130	Yes	N/A	Windows Server 2022, build 10.0.20348.1668
Windows Server 2022 Standard, including the March 2023 Updates	Dell PowerEdge R640	Intel Xeon Gold 6130	Yes	N/A	Windows Server 2022, build 10.0.20348.1668

Table 7: Tested Operational Environments - Software, Firmware, Hybrid

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

The following table presents the vendor-affirmed operational environment(s).

Operating System	Hardware Platform
Any Microsoft operating system which is a relabeled version of the operating systems listed in section 2.2.1.	Any UEFI-based x64 computer

Table 8: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

No components within the cryptographic boundary are excluded.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Normal operation of the computer, Windows OS, and module. This is the only mode claimed by the module.	Approved	N/A

Table 9: Modes List and Description

2.5 Algorithms

Approved Algorithms:

The table below lists the approved algorithms used in the module. The module may not use some of the capabilities described in each CAVP certificate. As the module has separate CAVP certificates for Windows 11 and Windows Server 2022, each approved algorithm is listed twice, with CAVP certificates #A4008, #A3748, and #A3767 for Windows 11 and CAVP certificates #A4009, #A3749, and #A3768 for Windows Server 2022. See Section 13 Standards References for links to the standards referenced in the table below.

The Windows Boot Manager cryptographic module uses a cryptographic key that is generated by the Windows Kernel Mode Cryptographic Primitives Library cryptographic module (CNG). FIPS 140-3 deems the Windows Boot Manager module as binding to the Windows Kernel Mode Cryptographic Primitives Library module (certificate #5408), which is referred to as an Existing Validated Module (EVM) in this document. All Windows cryptographic modules are installed together as described in section 11 Life-Cycle Assurance. See Section 5.1 Integrity Techniques for more information on the dependencies between Windows modules.

Table 10 below lists the approved algorithms in Windows Boot Manager. Table 11 below lists the approved cryptographic algorithms in the Windows Kernel Mode Cryptographic Primitives Library for key generation.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4008	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38A
AES-CBC	A4009	Direction - Decrypt Key Length - 128, 256	SP 800-38A
AES-CCM	A3748	Key Length - 256	SP 800-38C
AES-CCM	A3749	Key Length - 256	SP 800-38C
AES-XTS Testing Revision 2.0	A4008	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
AES-XTS Testing Revision 2.0	A4009	Direction - Decrypt Key Length - 128, 256	SP 800-38E
HMAC-SHA2-256	A4008	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4009	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
PBKDF	A4008	Iteration Count - Iteration Count: 10-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
PBKDF	A4009	Iteration Count - Iteration Count: 10-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA SigVer (FIPS186-4)	A3767	Signature Type - PKCS 1.5 Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A3768	Signature Type - PKCS 1.5 Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
SHA-1	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA-1	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-256	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Table 10: Approved Algorithms -

Existing Validated Module [EVM]

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4008	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38A
AES-CBC	A4009	Direction - Decrypt Key Length - 128, 256	SP 800-38A
Counter DRBG	A4008	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
Counter DRBG	A4009	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1

Table 11: Approved Algorithms - Existing Validated Module [EVM]

Vendor-Affirmed Algorithms:

The following table presents the vendor-affirmed algorithms. See Section 2.9.1 BitLocker Authorization Factors for more information.

Name	Properties	Implementation	Reference
CKG	Key type: Symmetric	N/A	NIST SP 800-133 Rev. 2 Section 6.3 method 2

Table 12: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

N/A for this module.

2.6 Security Function Implementations

The table below lists the module's Security Function Implementations. The Algorithms column presents algorithm and key size information for each CAVP certificate listed in Section 2.5 Algorithms. Blank cells are either not applicable or optional according to the CMVP. As the module has separate CAVP certificates for the Windows 11 and Windows Server 2022, each approved algorithm is listed twice in the Algorithms column.

Name	Type	Description	Properties	Algorithms
BC1	KTS-Wrap	Symmetric key unwrap using AES-CCM SP 800-38C and SP 800-38F. KTS (key unwrapping) per IG D.G. Used by the Unlocking the OS Volume service to decrypt multiple keys used in the boot process.	Standard: SP 800-38F IG D.G: Approved Method 2 from IG D.G Caveat: Key establishment methodology provides 256 bits of security strength	AES-CCM: (A3748, A3749) Key size: 256 bits AES-CBC: (A4008, A4009) Key size: 256 bits
BC2	BC-UnAuth	Symmetric block cipher decryption used by the Decrypting the OS Volume service to decrypt the system volume.		AES-CBC: (A4008, A4009) Key size: 128 bits Key size: 256 bits AES-XTS Testing Revision 2.0: (A4008, A4009) Key size: 128 bits Key size: 256 bits
CKG1	CKG	Key generation used by the Unlocking the OS Volume service to generate the Intermediate Key, Derived Key, and Network Key.		CKG: () Key type: Symmetric Counter DRBG: (A4008, A4009) [EVM] DRBG output: Bounded [EVM] CNG.SYS AES-CBC: (A4008, A4009) Bounded [EVM] CNG.SYS: 256 bits
DS-Legacy	DigSig-SigVer	RSA signature verification used by the module services for integrity verification.		RSA SigVer (FIPS186-4): (A3767, A3768) RSA Modulus: 1024 bits (For legacy signature verification only) SHA-1: (A4008, A4009) SHA Size: 160 bits (For legacy signature verification only)

Name	Type	Description	Properties	Algorithms
DS1	DigSig-SigVer	RSA signature verification used by the Secure Boot service to verify the integrity of the Secure Boot policy. SHA hash functions support this by verifying the RSA signature.		RSA SigVer (FIPS186-4): (A3767) RSA moduli: 2048, 3072, and 4096 bits RSA SigVer (FIPS186-4): (A3768) RSA moduli: 2048, 3072, and 4096 bits SHA2-256: (A4008, A4009) SHA Size: 256 bits SHA2-384: (A4008, A4009) SHA Size: 384 bits SHA2-512: (A4008, A4009) SHA Size: 512 bits
DS2	DigSig-SigVer	RSA signature verification used by the Secure Boot and Loading and Verifying OS Loader service to verify the integrity of the Windows OS Loader. SHA hash functions support this by verifying the RSA signature.		RSA SigVer (FIPS186-4): (A3767) RSA moduli: 2048, 3072, and 4096 bits RSA SigVer (FIPS186-4): (A3768) RSA moduli: 2048, 3072, and 4096 bits SHA2-256: (A4008, A4009) SHA Size: 256 bits SHA2-384: (A4008, A4009) SHA Size: 384 bits SHA2-512: (A4008, A4009) SHA Size: 512 bits
DS3	DigSig-SigVer	RSA signature verification used by the Unlocking the OS service to verify the integrity of the MUI resource file in some scenarios. SHA hash functions support this by verifying the RSA signature.		RSA SigVer (FIPS186-4): (A3767) RSA Moduli: 2048, 3072, and 4096 bits RSA SigVer (FIPS186-4): (A3768) RSA Moduli: 2048, 3072, and 4096 bits SHA2-256: (A4008, A4009) SHA Size: 256 bits SHA2-384: (A4008, A4009) SHA Size: 384 bits

Name	Type	Description	Properties	Algorithms
				SHA2-512: (A4008, A4009) SHA Size: 512 bits
PBKDF1	PBKDF	Password based key derivation function used by the Unlocking the OS Volume service to generate a key from user-input data, which is then used for TPM scenarios.		PBKDF: (A4008, A4009) MAC used: HMAC-SHA2-256 HMAC-SHA2-256: (A4008) SHA Size: 256 bits HMAC-SHA2-256: (A4009) SHA Size: 256 bits SHA2-256: (A4008, A4009) SHA Size: 256 bits

Table 13: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 FIPS 186-4 and 186-5

The module claims compliance with FIPS 186-5 for RSA signature verification, although the CAVP testing for RSA was completed against FIPS 186-4. Because the FIPS 186-4 RSA CAVP tests are mathematically identical to the FIPS 186-5 RSA CAVP tests, the module can claim a FIPS 186-5 compliance for these tests. The scope of FIPS 186-4 testing complies with IG C.K. Additional Comment 3. Although FIPS 186-4 has been superseded by FIPS 186-5, algorithms implemented under FIPS 186-4 remain approved for use under NIST SP 800-131A Rev. 2.

2.7.2 Password Based Key Derivation Function Usage

When BitLocker is configured to use a password or PIN protector to protect the system volume, a Password Based Key Derivation Function (PBKDF) is used to derive a suitable key for storage applications such as BitLocker. The PBKDF implemented in this module is NIST SP 800-132 compliant, including the following characteristics:

- 128-bit Salt
- Iteration count is 220

Note: The password length is enforced by the caller of the PBKDF interfaces at the time the password/passphrase is created and not by this cryptographic module. Boot Manager is not involved in the creation of any password. At a minimum, Microsoft recommends that passwords be at least 14 characters long, contain secret or random information, are significantly different from any previous passwords, and contain a mix of uppercase letters, lowercase letters, numerals, symbols, numbers, and characters from the extended ANSI character set. It is the responsibility of the operator / caller to define and enforce an appropriate password policy. Windows provides a variety of security policies to administrators to enforce password length and complexity.

NIST SP 800-132 Section 5.4, Using the Derived Master Key to Protect Data, describes two options for protecting data using a Master Key (MK). Boot Manager uses Option 2 in which the MK produced by the PBKDF is used to decrypt a Data Protection Key (DPK). In Windows, the DPK corresponds to the Volume Master Key (VMK), which is used to decrypt the Full Volume Encryption Key (FVEK) which is used for data encryption/decryption of the storage volume. The VMK and FVEK are described later in this document.

2.7.3 Legacy RSA Signature Verification

1. RSA signature verification using SHA-1 is used for legacy signature verification only.
2. 1024-bit RSA key is used for legacy signature verification only.
3. Algorithms designated as “Legacy” can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M.

2.7.4 AES-XTS

AES-XTS is approved only for storage applications such as BitLocker. The length of data encrypted does not exceed 2^{20} AES blocks. Key 1 and Key 2 are generated independently, as required by IG C.I., and the two keys are explicitly checked to ensure that they are not equal before use.

2.8 RBG and Entropy

N/A for this module.

N/A for this module.

2.9 Key Generation

Boot Manager implements NIST SP 800-133 r2 symmetric key generation. This is used by the Unlocking the OS Volume service to generate the Intermediate Key and Network Key. The symmetric key generation combines multiple keys and other data in an Exclusive-Or (XOR) operation. This conforms with the requirements for vendor affirmation as per IG D.H. and the use of random bit strings for key generation as per SP 800-133 Rev. 2 section 4. See section 2.9.1 BitLocker Authorization Factors below for more information.

Boot Manager also implements NIST SP 800-132 password based key derivation (PBKDF) using HMAC-SHA2-256. This is used by the Unlocking the OS Volume service to generate a key from user-input data, which is then used for TPM scenarios.

2.9.1 BitLocker Authorization Factors

The authorization factors supplied for BitLocker in the table below describe the “knowledge factor” (something you know) or “possession factor” (something you have) used in authorization. This data is provided as part of the claim for SP 800-133 CKG.

Authorization Factor	Justification
A password entered by the user	• The VMK meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • The password meets the requirements in NIST SP 800-132, Recommendation for Password-Based Key Derivation. • The key derived from the password is used to decrypt the Volume Master Key (VMK).
An External Key which may be used during boot if it is stored on a USB drive, or during recovery if it is stored on a USB drive or typed in manually	• The VMK meets the requirements in NIST SP 800-133 Rev 2, Recommendation for Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • The External Key meets the requirements in NIST SP 800-133 Rev 2, Recommendation for Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • The external key is used to decrypt the Volume Master Key (VMK).
A key stored in the TPM	• The VMK meets the requirements in NIST SP 800-133 Rev 2, Recommendation for Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • The unsealed intermediate key is used to decrypt the Volume Master Key (VMK).

Authorization Factor	Justification
A TPM key combined with a user-provided PIN	• The VMK meets the requirements in NIST SP 800-133 Rev 2, Recommendation for Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • The user-provided PIN meets the requirements in NIST SP 800-132, Recommendation for Password-Based Key Derivation. • Combining the TPM-protected Intermediate Key and User-provided PIN meets the requirements in NIST SP 800-133 Rev 2 Cryptographic Key Generation: specifically, Section 6.3, Symmetric Keys Produced by Combining (Multiple) Keys and Other Data.
A TPM key combined with the External Key	• The VMK meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • The External Key meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • Combining the TPM-protected Intermediate Key and External Key meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.3, Symmetric Keys Produced by Combining (Multiple) Keys and Other Data.
A TPM key combined with a PIN and the External Key	• The VMK meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.1 The “Direct Generation” of Symmetric Keys. • The External Key meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.1 “The “Direct Generation” of Symmetric Keys. • The user-provided PIN meets the requirements in NIST SP 800-132, Recommendation for Password-Based Key Derivation. • Combining the TPM-protected Intermediate Key, External Key, and User-provided PIN meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation; specifically, Section 6.3, Symmetric Keys Produced by Combining (Multiple) Keys and Other Data.
A TPM key combined with a Network Key	• The VMK meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6 Generation of Keys for Symmetric-Key Algorithms. • The Network Key meets the requirements NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.1 “The “Direct Generation” of Symmetric Keys”. • Combining the TPM-protected Intermediate Key, and Network Key meets the requirements in NIST SP 800-133 Rev 2, Cryptographic Key Generation: specifically, Section 6.3, Symmetric Keys Produced by Combining (Multiple) Keys and Other Data.
A key stored on disk and only used when BitLocker is disabled but the drive is encrypted	• Meets the requirements in NIST SP 800-133 Rev 2; specifically, Section 6.1 “The “Direct Generation” of Symmetric Keys.”

Table 14: BitLocker Authorization Factors

2.10 Key Establishment

N/A as the module has no key establishment functions.

2.11 Industry Protocols

N/A as the module claims none.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

As a software-hybrid module, the module has no physical ports of its own. The physical ports of the module are interpreted as those on the underlying hardware platform and control of them is outside the scope of the module. The module logical interfaces are described in the table below. Boot Manager does not export any cryptographic functions that can be called or externally invoked.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	The Data Input Interface includes the BlFileReadEx function. BlFileReadEx is responsible for reading the binary data of unverified components from the computer hard drive. Additionally, the computer's USB port also forms a part of the Data Input Interface. This interface is used to enter the BitLocker Startup key or Recovery key. The keyboard, USB, and recovery key may also serve as a Data Input Interface for password-based protection factors.
N/A	Data Output	The Data Output Interface includes two different kinds of functions: initialization and transfer. The initialization function ImgpInitializeBootApplicationParameters output are the input parameters for the boot application which Boot Manager launches. This function is called before transferring execution to the boot application. The following function is a transfer function: Archpx64TransferTo64BitApplicationAsm. These functions are responsible for transferring the execution from Boot Manager to the initial execution point of the Windows OS Loader. Data exits the module in the form of the initial instruction address of winload.efi.
N/A	Control Input	The Boot Manager Control Input Interface is the set of internal functions responsible for reading control input. These input signals are read from various system locations and are not directly provided by the operator. Examples of the internal function calls include: BlBdDebuggerEnabled - Reads the system flag to determine if the boot debugger is enabled; BlXmiRead - Reads the operator selection from the Boot Selection menu; BlGetBootOptionBoolean - Reads control input from a protected area of the Boot Configuration Data registry. The computer's keyboard can also be used as control input when it is necessary for an operator to provide a response to a prompt for input or in response to an error indicator.
N/A	Status Output	The Status Output Interface is the BlStatusPrint function that is responsible for displaying any integrity verification errors to the display. The Status Output Interface defined as BsdpWriteAtLogOffset is responsible for writing the name of the corrupt driver to the boot log.
N/A	Power	N/A

Table 15: Ports and Interfaces

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

In Windows, authentication and assignment of user roles happens after the OS boots. When BitLocker is used to encrypt the system volume, the user booting the system may interact with BitLocker by providing an authorization factor (CSP) as input to the module. Otherwise, since Boot Manager executes between power-on and the start of OS initialization, its functions are fully automatic and not configurable.

4.2 Roles

The module has a single role, Cryptographic Officer (CO). All services are accessible by this role.

Name	Type	Operator Type	Authentication Methods
Cryptographic Officer	Role	CO	None

Table 16: Roles

4.3 Approved Services

The module provides approved services only. The indicator for each service is the successful completion of the service. The table below provides additional indicator details to enable the operator to verify each service's successful completion.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Booting the OS	Boots the next boot application in the overall boot sequence for the Windows OS (Windows OS Loader).	The Windows boot process continues. See 5.1 Integrity Techniques for more information.	This service is fully automatic.	Loads the next boot application, the Windows OS Loader component.	None	Cryptographic Officer
Decrypting the OS Volume	Decrypts the OS volume.	The OS volume is decrypted, and the Windows boot process continues.	This service is executed automatically when BitLocker is enabled.	The cryptographic operation is completed, and status is returned.	BC2	Cryptographic Officer - Full Volume Encryption Key (FVEK): R,W,E
Loading and Verifying OS Loader	Loads and verifies the integrity of the Windows OS Loader (winload.efi).	OS Loader (winload.efi) executes.	This service is fully automatic.	The cryptographic operation is completed and status is returned.	DS2	Cryptographic Officer - Microsoft Root Certificate Authority Public Key (This is not an SSP): E - SHA Hashes (This is not an SSP): E
Perform Self-Tests	The module provides a power-up self-test service that is automatically executed when the module is loaded into memory.	Self-test success is indicated by module and algorithm availability; failure is indicated by an error.	This service is fully automatic, executed when the module is loaded into memory.	The availability of the module is the service output. See section 10 Self-Tests for more information.	BC1 BC2 CKG1 DS1 DS2 DS3 PBKDF1 DS-Legacy	Cryptographic Officer
Perform Zeroization	Zeroizes cryptographic material.	Volatile keys are zeroized. See 9.3 SSP Zeroization Methods for more information.	This service is fully automatic.	Keys are zeroized and the	BC1 PBKDF1 CKG1	Cryptographic Officer - Derived Key

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
				module unloads from memory.		(DK): Z - External Key (ExK): Z - Full Volume Encryption Key (FVEK): Z - Intermediate Key (IK): Z - Network Intermediate Key (NIK): Z - Network Key (NK): Z - Password: Z - PIN: Z
Secure Boot	Reads the Secure Boot policy when UEFI boot is used.	UEFI boot validates the integrity of Boot Manager. See 5.1 Integrity Techniques for more information.	This service is fully automatic after Secure Boot has been enabled.	The cryptographic operation is completed, and status is returned.	DS-Legacy DS1 DS2	Cryptographic Officer - Microsoft Root Certificate Authority Public Key (This is not an SSP): E - SHA Hashes (This is not an SSP): E
Show Status	Provides the module status response.	Provided via NTSTATUS by all functions of approved services, as noted above.	This service is fully automatic and occurs when any function is called.	Status is output across the module's Status Output logical interface, to the computer monitor or to log files.	None	Unauthenticated
Show Version	Provides the module version number.	Version information is provided in the Portable Executable (PE) header of each module binary. The PE header contains Windows-specific fields such as the major and minor version (10.0.22621.1 for Windows 11 and	Module binary file.	Version information is embedded in the PE header of the binary.	None	Unauthenticated

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		10.0.20348.1668 for Windows Server). See the PE Format public documentation published on https://learn.microsoft.com/ for more information.				
Unlocking the OS Volume	Decrypts the FVEK. Reads the secure boot policy when UEFI boot is used. In some BitLocker scenarios Boot Manager must display a prompt to the user. In these scenarios, this service also validates and then loads the Multilingual User Interface (MUI) resource file.	FVEK is decrypted and BitLocker drive unlock proceeds. In scenarios where Boot Manager gathers one or more BitLocker authorization factors, a prompt for each authorization factor is displayed.	See the finite state model diagrams in the section Error States for the user actions during "System Volume Unlock" stage. This service is executed automatically when BitLocker is enabled.	The cryptographic operation is completed, and status is returned.	BC1 CKG1 DS3 PBKDF1	Cryptographic Officer - Clear Key (CK): W,E - Derived Key (DK): G,E - External Key (ExK): W,E - Full Volume Encryption Key (FVEK): W,E - Intermediate Key (IK): G,W,E - Microsoft Root Certificate Authority Public Key (This is not an SSP): E - Network Intermediate Key (NIK): W,E - Network Key (NK): G,E - Password: W,E - PIN: W,E - Session Key (SK): W,E - SHA Hashes (This is not an SSP): E - Volume Master Key (VMK): W,E

Table 17: Approved Services

4.4 Non-Approved Services

N/A for this module.

4.5 External Software/Firmware Loaded

The module loads no external software or firmware. While it is running, Boot Manager is the only process running on the computer.

5 Software/Firmware Security

The secure installation, generation, and startup procedures of this module are part of the secure installation, configuration, and startup procedures of the Windows operating systems named in Section 2.2 Tested and Vendor Affirmed Module Version and Identification.

5.1 Integrity Techniques

Windows uses several mechanisms to provide integrity verification depending on the stage in the boot sequence, the hardware, and the configuration.

- The integrity of Boot Manager in its executable form (bootmgr.efi, bootmgfw.efi, and bootx64.efi) is verified by UEFI when Secure Boot is enabled and by the Boot Manager itself. To perform the module integrity test on demand, the user may restart the computer.
- Boot Manager verifies Windows OS Loader before transferring control to that component.
- Windows binaries include a SHA2-256, SHA2-384, or SHA2-512 hash of the binary signed with the 2048-bit Microsoft RSA code-signing key (i.e., the key associated with the Microsoft code-signing certificate). The integrity check uses the public key component of the Microsoft code signing certificate to verify the signed hash of the binary.

The figure below shows the integrity chain of trust for the Windows modules in scope for this validation.

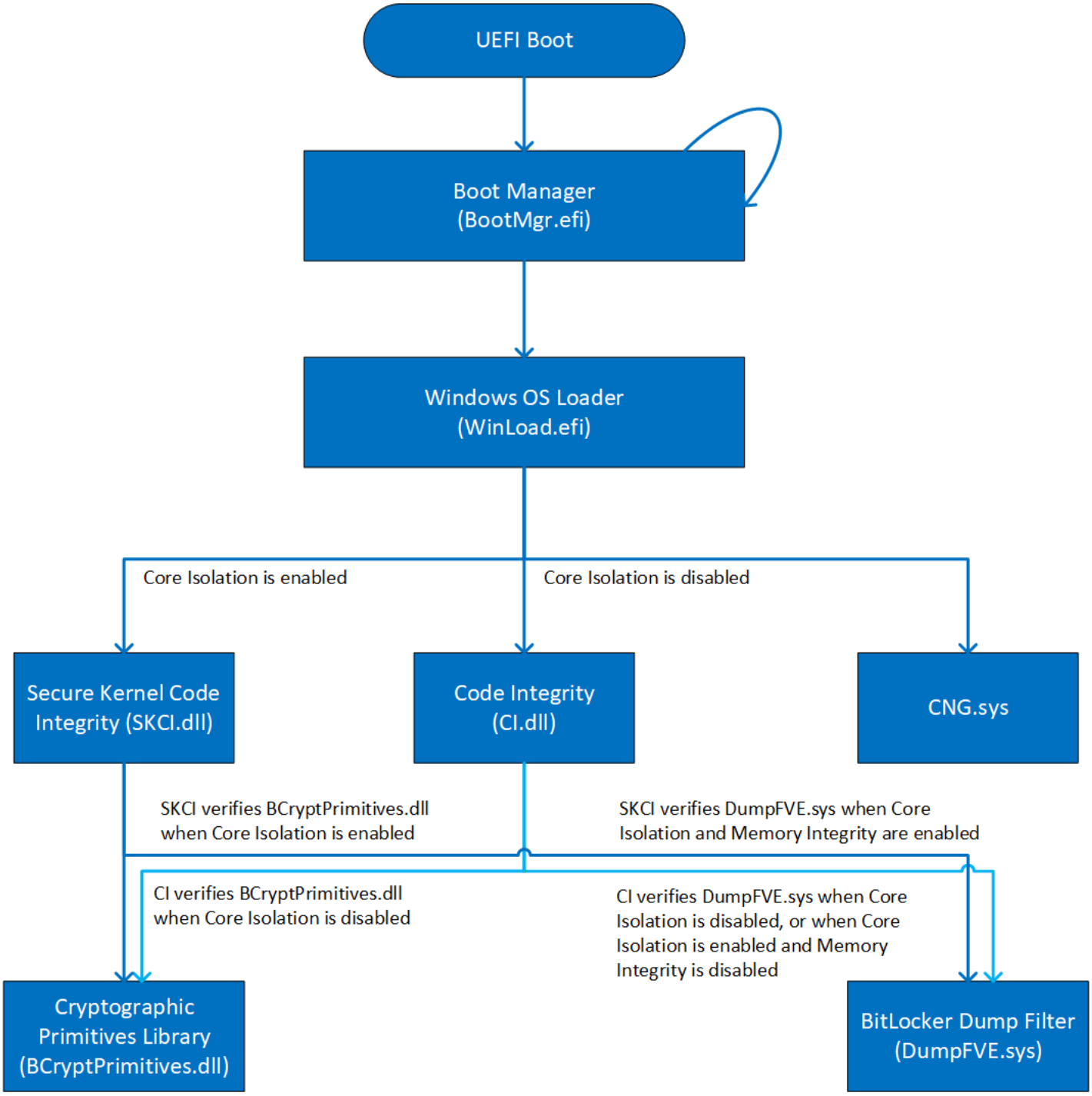


Figure 2: Integrity chain of trust

5.2 Initiate on Demand

To initiate the integrity test on demand, the operator may restart the computer.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The modifiable operational environment for Boot Manager is the Windows operating system running on a supported hardware platform, as listed in Section 2.2 Tested and Vendor Affirmed Module Version and Identification.

During the operating system boot process there is no logged-on user, so the single operator requirement is met.

7 Physical Security

7.1 Mechanisms and Actions Required

Boot Manager is a multi-chip standalone software-hybrid module whose host platforms meet the Level 1 physical security requirements. The host platform consists of production-grade physical security components that include standard passivation techniques and is entirely contained within a metal or hard plastic production-grade enclosure that may include doors or removable covers.

Tables identifying the voltage and temperature boundaries that trigger zeroization or shutdown are not included in this Security Policy as they are N/A for a Level 1 validation of a hybrid module.

8 Non-Invasive Security

N/A for this module.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
Hard Disk	Persistent SSPs are stored on the operating system volume, USB device, or other non-volatile storage outside the cryptographic boundary (see: Hard Disk and Removable Storage in the block diagram).	Static
RAM	Volatile SSPs are temporarily stored in the computer's memory (see: RAM in the block diagram).	Dynamic

Table 18: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input from encrypted non-volatile storage	Hard disk with operating system volume.	RAM	Encrypted	Manual	Electronic	BC1

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input from non-encrypted non-volatile storage	Hard Disk	RAM	Plaintext	Manual	Electronic	
Resides in RAM as plaintext	RAM	Output to BitLocker runtime (fvevol.sys) in the clear via in-memory / in-RAM sharing.	Plaintext	Manual	Electronic	
User Input	Input by the Cryptographic Officer	RAM	Plaintext	Manual	Direct	

Table 19: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Procedural zeroization	Operators may choose to overwrite persistent CSPs by reformatting and overwriting the storage media for the operating system.	Reformatting and overwriting the OS volume at least once ensures adequate zeroization. Operators may choose to overwrite multiple times at their discretion.	Operators may use the Format command together with the /P parameter to format and overwrite every sector on the volume with zeros. Operators may specify the number of overwrite passes using the /P parameter. See the public documentation for the Format command for more information.
Zeroization after use	For all volatile SSPs, the module overwrites the temporary storage area for the SSP immediately after use.	Zeroization 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.	None (programmatically executed by the module after CSPs are used).

Table 20: SSP Zeroization Methods

9.4 SSPs

The tables below list the keys and SSPs used by the module. Per the CMVP, public keys and file hashes used for the module integrity check are not considered SSPs and, as such, have no input method assigned and are categorized as neither PSP nor CSP. Blank cells are either not applicable or optional according to the CMVP.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Clear Key (CK)	An AES key that is used to decrypt the VMK after BitLocker enters Suspend mode.	Size: 256-bit - Strength: 256-bit	Symmetric Key (AES) - CSP	Externally generated by BitLocker runtime components when BitLocker enters Suspend mode.		BC1
Derived Key (DK)	An AES key used to decrypt the VMK. The value does not persist and is derived using a method defined by the system configuration. Derived Keys are used in Password TPM combination mechanisms.	Size: 256-bit - Strength: 256-bit	Symmetric Key (AES) - CSP	CKG1 PBKDF1		BC1 CKG1
External Key (ExK)	An AES key stored outside the cryptographic boundary, for example, on a USB device. The external key represents either a startup key or a recovery key and is used for AES decryption of the VMK.	Size: 256-bit - Strength: 256 bits	Symmetric Key (AES) - CSP	Externally generated by BitLocker runtime components outside of the cryptographic boundary.		BC1
Full Volume Encryption Key (FVEK)	An AES key used for encryption / decryption of data on disk sectors. This key is stored encrypted on the system volume. It is encrypted and decrypted by the VMK using AES-CCM.	Size: 128-bit or 256-bit (administrator configurable) - Strength: The strength is the key size	Symmetric Key (AES) - CSP	Externally generated by BitLocker runtime components outside of the cryptographic boundary.	BC1	BC1
Intermediate Key (IK)	An AES key that is protected by (sealed to) the platform TPM and forms the basis of another AES key, such as the NK or VMK, by combining with another 256-bit AES key using an Exclusive-Or (XOR) operation. This key is generated by the DRBG output of the bounded [EVM] for the following BitLocker configurations (Table 14): A key stored in the TPM, a TPM key combined with a user-provided PIN, a TPM key combined with the External Key, a TPM key combined with a PIN	Size: 256-bit - Strength: 256-bit	Symmetric Key (AES) - CSP	CKG1		BC1 CKG1

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	and the External Key, a TPM key combined with a Network Key.					
Microsoft Root Certificate Authority Public Key (This is not an SSP)	Public key used for RSA PKCS #1 (v1.5) verification of digital signatures.	Size: 2048-bit - Strength: 112 bits	Asymmetric Public Key (RSA) - Neither	Generated external to the module by the Microsoft root CA.		DS1 DS2 DS3
Network Intermediate Key (NIK)	An AES key used for the Network Unlock Authenticator . Boot Manager does the decryption of the NIK using AES-CCM with the Session Key.	Size: 256-bit - Strength: 256 bits	Symmetric Key (AES) - CSP	Externally generated by BitLocker runtime components outside of the cryptographic boundary.		BC1
Network Key (NK)	An AES key used for AES decryption of the VMK in Network Unlock authentication. Composed by XOR of an IK protected by the TPM and another IK imported over a trusted network (the NIK).	Size: 256-bit - Strength: 256 bits	Symmetric Key (AES) - CSP	CKG1		BC1
Password	A password entered by the user. Used for NIST SP 800-132 password-based key derivation; the resulting CSP is generated following NIST SP 800-132 recommendations for PBKDF.	Size: Administrator defined - Strength: Guidelines in Section 2.7.2 of the Security Policy advise on password strength.	User-generated text string - CSP	User-generated		PBKDF1
PIN	An alphanumeric PIN for Trusted Platform Module (TPM) + PIN or TPM + PIN + USB scenarios. Used for NIST SP 800-132 password-based key derivation; the resulting CSP is generated following NIST SP 800-132 recommendations for PBKDF.	Size: Administrator defined - Strength: Guidelines in Section 2.7.2 of the Security Policy advise on password strength.	User-generated text string - CSP	User-generated		PBKDF1
Session Key (SK)	An AES key stored in plaintext on disk and used to decrypt the NIK using AES-CCM.	Size: 256-bit - Strength: 256-bit	Symmetric Key (AES) - CSP	Externally generated by BitLocker runtime components outside of the cryptographic boundary.		BC1
SHA Hashes (This is not an SSP)	File hashes used to verify the integrity of binaries. This is not an SSP.	Size: 160, 256, 384, or 512 bits - Strength: 160, 256, 384, or 512 bits	File Hash - Neither	Generated external to the module when the binary files are written to disk.		DS1 DS2 DS3

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Volume Master Key (VMK)	An AES key used for AES-CCM decryption of the FVEK.	Size: 256-bit - Strength: 256-bit	Symmetric Key (AES) - CSP	Externally generated and encrypted by BitLocker runtime components outside of the cryptographic boundary.		BC1

Table 21: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Clear Key (CK)	Input from non-encrypted non-volatile storage	RAM:Plaintext	Until the module is unloaded.	Procedural zeroization	Volume Master Key (VMK):Decrypts in BitLocker Suspend scenarios.
Derived Key (DK)		RAM:Plaintext	Temporary during service execution.	Zeroization after use	Volume Master Key (VMK):Decrypts
External Key (ExK)	Input from non-encrypted non-volatile storage	RAM:Plaintext	Temporary during service execution.	Zeroization after use	Volume Master Key (VMK):Decrypts
Full Volume Encryption Key (FVEK)	Input from encrypted non-volatile storage Resides in RAM as plaintext	RAM:Encrypted RAM:Plaintext	Until module is unloaded.	Procedural zeroization	Volume Master Key (VMK): Decrypted using the VMK.
Intermediate Key (IK)	Input from encrypted non-volatile storage	RAM:Encrypted	Temporary during service execution.	Zeroization after use	Volume Master Key (VMK):Forms the basis using XOR Network Key (NK):Forms the basis using XOR
Microsoft Root Certificate Authority Public Key (This is not an SSP)		Hard Disk:Plaintext		Procedural zeroization	
Network Intermediate Key (NIK)	Input from encrypted non-volatile storage	RAM:Encrypted	Temporary during service execution.	Zeroization after use	Session Key (SK): Decrypted using AES-CCM with the SK.
Network Key (NK)		RAM:Plaintext	Temporary during service execution.	Zeroization after use	Volume Master Key (VMK): Used to decrypt the VMK in Network Unlock authentication scenarios.
Password	User Input	RAM:Plaintext	Temporary during service execution.	Zeroization after use	
PIN	User Input	RAM:Plaintext	Temporary during service execution.	Zeroization after use	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Session Key (SK)	Input from non-encrypted non-volatile storage	RAM:Plaintext	Temporary during service execution.	Zeroization after use	Network Intermediate Key (NIK):Decrypts
SHA Hashes (This is not an SSP)		RAM:Plaintext		Procedural zeroization	
Volume Master Key (VMK)	Input from encrypted non-volatile storage	RAM:Encrypted RAM:Plaintext	Until module is unloaded.	Procedural zeroization	Full Volume Encryption Key (FVEK):Decrypts

Table 22: SSP Table 2

For details about the keys and network protocol used for Network Unlock authentication, see the Network Key Protector Unlock Protocol Specification [MS-NKPU].

9.5 Transitions

The following transition timelines apply to the approved algorithms named in the bullets below:

- The SHA-1 algorithm will become disallowed for applying cryptographic protection starting January 1, 2031, however, its use for legacy digital signature verification will continue to be allowed.
- RSA with a security strength of less than 128 bits will become deprecated starting January 1, 2031, however, its use for legacy digital signature verification will continue to be allowed.
- FIPS 186-4 has been superseded by FIPS 186-5. This transition began on July 25, 2023, and concluded on February 3, 2024. Although testing for this module was completed against FIPS 186-4, it claims compliance with FIPS 186-5 for RSA signature verification – see Section 2.7.1 FIPS 186-4 and 186-5 for more information.

10 Self-Tests

The module performs self-tests to ensure integrity and correct functionality. The module will not perform cryptographic functions while in its self-test or error states. If the self-test fails, the module enters the error state. If the status is not STATUS_SUCCESS, then that is the indicator a self-test failed. If the self-test passes, the module is loaded and cryptographic functions are available for use. As the module has separate CAVP certificates for Windows 11 and Windows Server 2022, the self-test tables below list two identical rows for each algorithm self-test, one for Windows 11 and one for Windows Server 2022.

10.1 Pre-Operational Self-Tests

The algorithms used for the pre-operational self-tests pass their own algorithm self-tests before integrity verification is performed.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
RSA SigVer (FIPS186-4) (A3767)	RSA PKCS#1v1.5 2048-bit key with SHA2-256	Software Integrity	SW/FW Integrity	Module available and Unlocking the OS Service runs.	Signature verification
RSA SigVer (FIPS186-4) (A3768)	RSA PKCS#1v1.5 2048-bit key with SHA2-256.	Software Integrity	SW/FW Integrity	Module available and Unlocking the OS Service runs.	Signature verification

Table 23: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

The Boot Manager module performs the conditional cryptographic algorithm self-tests automatically when the module is loaded into memory, after the pre-operational software integrity tests described above have completed.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
[EVM] AES-CBC (A4009) - Encrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
						module integrity is verified.
[EVM] AES-CBC (A4008) - Decrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
[EVM] AES-CBC (A4008) - Encrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the module integrity is verified.
[EVM] AES-CBC (A4009) - Decrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
[EVM] Counter DRBG (A4008)	256-bit AES key	KAT	CAST	Unlocking the OS Service runs.	[EVM] DRBG Known Answer Test (KAT) with instantiate, generate, and reseed tests.	Run at every module initialization after the module integrity is verified.
[EVM] Counter DRBG (A4009)	256-bit AES key	KAT	CAST	Unlocking the OS Service runs.	[EVM] DRBG Known Answer Test (KAT) with instantiate, generate, and reseed tests.	Run at every module initialization after the module integrity is verified.
[EVM] SHA2-256 (A4008)	256 bit hash (Windows 11 version 22H2)	KAT	CAST	The module's integrity verification is performed.	[EVM] Secure Hash	Runs before module's integrity verification is performed.
[EVM] SHA2-256 (A4009)	256 bit hash (Windows Server 2022)	KAT	CAST	The module's integrity verification is performed.	[EVM] Secure Hash	Runs before module's integrity verification is performed.
AES-CBC (A4008) - Decrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-CBC (A4008) - Encrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the module integrity is verified.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A4009) - Decrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-CBC (A4009) - Encrypt	AES-CBC with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-CCM (A3748) - Decrypt	AES-CCM with 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-CCM (A3748) - Encrypt	AES-CCM with 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-CCM (A3749) - Decrypt	AES-CCM with 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-CCM (A3749) - Encrypt	AES-CCM with 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-XTS Testing Revision 2.0 (A4008) - Decrypt	AES-XTS with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-XTS Testing Revision 2.0 (A4008) - Encrypt	AES-XTS with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-XTS Testing Revision 2.0 (A4008) - Key Equivalence	AES-XTS with 128 or 256 bits key	XTS-AES Key_1 Key_2 check in compliance with FIPS 140-3 IG C.I.	CAST	Unlocking the OS Service runs.	Key equivalence test	Run at every module initialization after the module integrity is verified.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-XTS Testing Revision 2.0 (A4009) - Decrypt	AES-XTS with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Decrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-XTS Testing Revision 2.0 (A4009) - Encrypt	AES-XTS with 128 or 256 bits key	KAT	CAST	Unlocking the OS Service runs.	Encrypt KAT.	Run at every module initialization after the module integrity is verified.
AES-XTS Testing Revision 2.0 (A4009) - Key Equivalence	AES-XTS with 128 or 256 bits key	XTS-AES Key_1 Key_2 check in compliance with FIPS 140-3 IG C.I.	CAST	Unlocking the OS Service runs.	Key equivalence test	Run at every module initialization after the module integrity is verified.
PBKDF (A4008)	Derived key material using a password that meets the requirements described in section 2.7.2, HMAC-SHA2-256, 128-bit Salt, 256-bit Volume Master Key, and iteration count of 220.	KAT	CAST	Unlocking the OS Service runs.	KAT with derived key material.	Run at every module initialization after the module integrity is verified.
PBKDF (A4009)	Derived key material using a password that meets the requirements described in section 2.7.2, HMAC-SHA2-256, 128-bit Salt, 256-bit Volume Master Key, and iteration count of 220.	KAT	CAST	Unlocking the OS Service runs.	KAT with derived key material.	Run at every module initialization after the module integrity is verified.
RSA SigVer (FIPS186-4) (A3767)	RSA PKCS#1v1.5 2048-bit key with SHA2-256.	KAT	CAST	Unlocking the OS Service runs.	Signature verification.	Run at every module initialization after the module integrity is verified.
RSA SigVer (FIPS186-4) (A3768)	RSA PKCS#1v1.5 2048-bit key with SHA2-256.	KAT	CAST	Unlocking the OS Service runs.	Signature verification.	Run at every module initialization after the module integrity is verified.
SHA-1 (A4008)	160 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
						module integrity is verified.
SHA-1 (A4009)	160 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the module integrity is verified.
SHA2-256 (A4008)	256 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the module integrity is verified.
SHA2-256 (A4009)	256 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the module integrity is verified.
SHA2-384 (A4008)	384 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the module integrity is verified.
SHA2-384 (A4009)	384 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the module integrity is verified.
SHA2-512 (A4008)	512 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the module integrity is verified.
SHA2-512 (A4009)	512 bits	KAT	CAST	Unlocking the OS Service runs.	Secure hash.	Run at every module initialization after the module integrity is verified.

Table 24: Conditional Self-Tests

10.3 Periodic Self-Test Information

The tables below present the periodic self-test information for the module. The set of self-tests presented in tables 25 and 26 below is identical to the set of self-tests presented in tables 23 and 24 above.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-4) (A3767)	Software Integrity	SW/FW Integrity	Pre-operational integrity test is run at every module initialization after the CASTs.	Manual on-demand (operator initiated)
RSA SigVer (FIPS186-4) (A3768)	Software Integrity	SW/FW Integrity	Pre-operational integrity test is run at every module initialization after the CASTs.	Manual on-demand (operator initiated)

Table 25: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
[EVM] AES-CBC (A4009) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
[EVM] AES-CBC (A4008) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
[EVM] AES-CBC (A4008) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
[EVM] AES-CBC (A4009) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
[EVM] Counter DRBG (A4008)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
[EVM] Counter DRBG (A4009)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
[EVM] SHA2-256 (A4008)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified (Windows 11 version 22H2).	Manual on-demand (operator initiated by rebooting the computer)
[EVM] SHA2-256 (A4009)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified	Manual on-demand (operator initiated by rebooting the computer)

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
			(Windows Server 2022).	
AES-CBC (A4008) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-CBC (A4008) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-CBC (A4009) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-CBC (A4009) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-CCM (A3748) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-CCM (A3748) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-CCM (A3749) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-CCM (A3749) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-XTS Testing Revision 2.0 (A4008) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-XTS Testing Revision 2.0 (A4008) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-XTS Testing Revision 2.0 (A4008) - Key Equivalence	XTS-AES Key_1 Key_2 check in compliance with FIPS 140-3 IG C.I.	CAST	Conditional CASTs are run at every module initialization	Manual on-demand (operator initiated).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
			before the module integrity is verified.	
AES-XTS Testing Revision 2.0 (A4009) - Decrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-XTS Testing Revision 2.0 (A4009) - Encrypt	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
AES-XTS Testing Revision 2.0 (A4009) - Key Equivalence	XTS-AES Key_1 Key_2 check in compliance with FIPS 140-3 IG C.I.	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
PBKDF (A4008)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
PBKDF (A4009)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
RSA SigVer (FIPS186-4) (A3767)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
RSA SigVer (FIPS186-4) (A3768)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
SHA-1 (A4008)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
SHA-1 (A4009)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
SHA2-256 (A4008)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
SHA2-256 (A4009)	KAT	CAST	Conditional CASTs are run at every module initialization	Manual on-demand (operator initiated).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
			before the module integrity is verified.	
SHA2-384 (A4008)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
SHA2-384 (A4009)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
SHA2-512 (A4008)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).
SHA2-512 (A4009)	KAT	CAST	Conditional CASTs are run at every module initialization before the module integrity is verified.	Manual on-demand (operator initiated).

Table 26: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Boot Fail	Boot failure	Error state occurs if the Boot Manager integrity test fails, if any of the cryptographic algorithm self-tests fail, or if the boot policy file integrity test fails.	Restart the computer.	Boot failure blue screen error message.

Table 27: Error States

10.5 Operator Initiation of Self-Tests

To perform the module self-tests on demand, including running the approved services Perform Pre-operational Software Integrity Test [EVM] and Perform Cryptographic Algorithm Self-Tests, the user may reboot the computer.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The Windows operating system must be pre-installed on a computer by an OEM, installed by the end-user, by an organization's IT administrator, or updated from a previous Windows version downloaded from Windows Update. An inspection of authenticity can be made by following the guidance at this Microsoft web site: www.microsoft.com/en-us/howtotell/default.aspx.

For Windows Updates, the client only accepts binaries signed by Microsoft certificates. The Windows Update client only accepts content whose SHA2 hash matches the SHA2 hash specified in the metadata. All metadata communication is done over a Transport Layer Security (TLS) port. Using SSL ensures that the client is communicating with the real server and so prevents a spoof server from sending the client harmful requests. The version and digital signature of new cryptographic module releases must be verified to match the version that was validated. See Section 11.3.1 General Guidance - 11.2.2 Verifying the Cryptographic Module Version and its Signature for details on how to do this.

Module initialization occurs automatically as part of the Windows boot process. The following diagrams visualize different aspects of Windows and module initialization, beginning with the finite state model. Every state of the module can transition to the power-off state through power-cycle/rebooting the host machine.

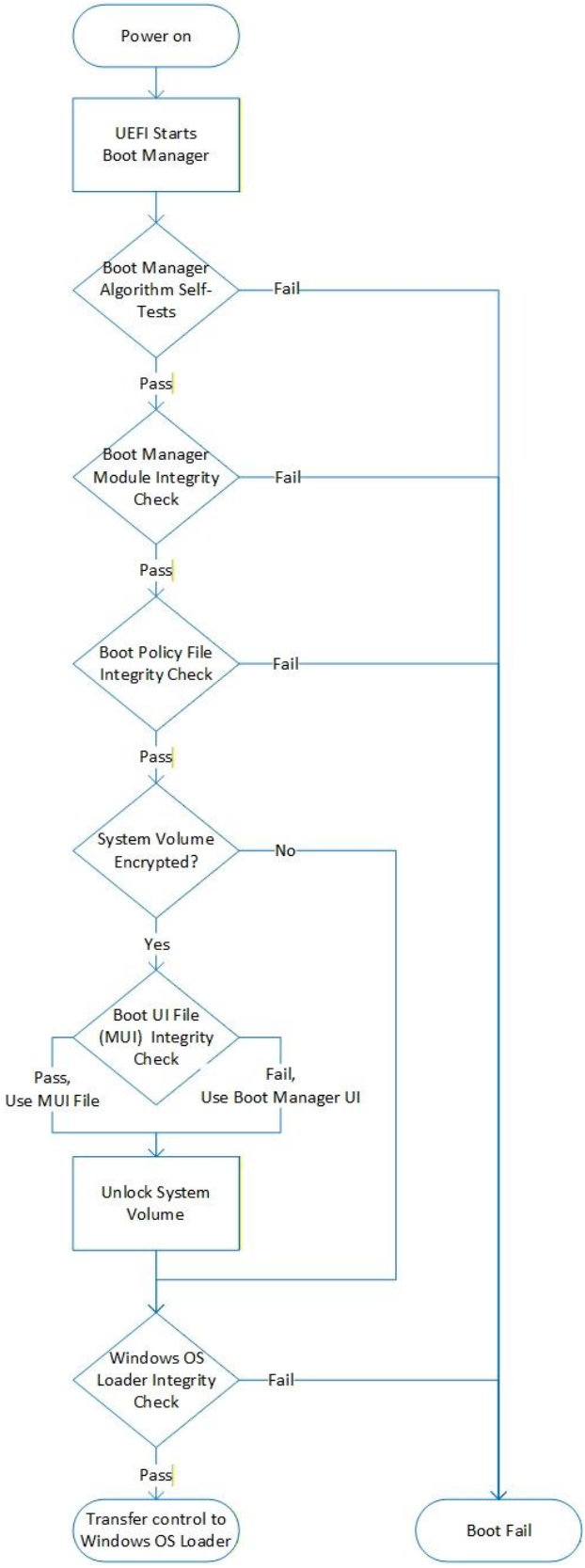


Figure 3: Finite State Model

The following state diagram shows states and user inputs for the Unlock System Volume sequence.

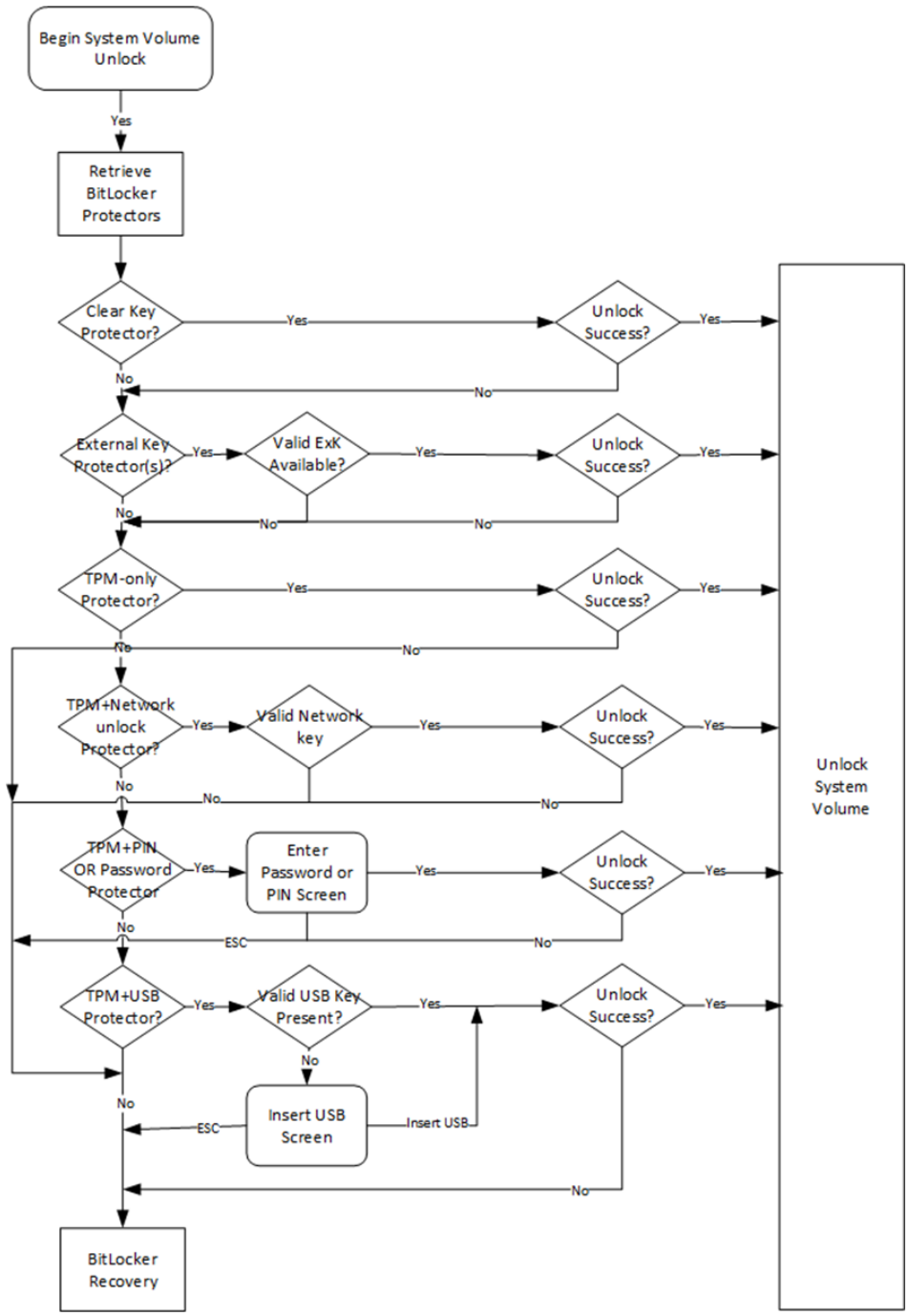


Figure 4: States and User Inputs for the Optional Unlock System Volume Sequence

The following state diagram shows the BitLocker Recovery sequence.

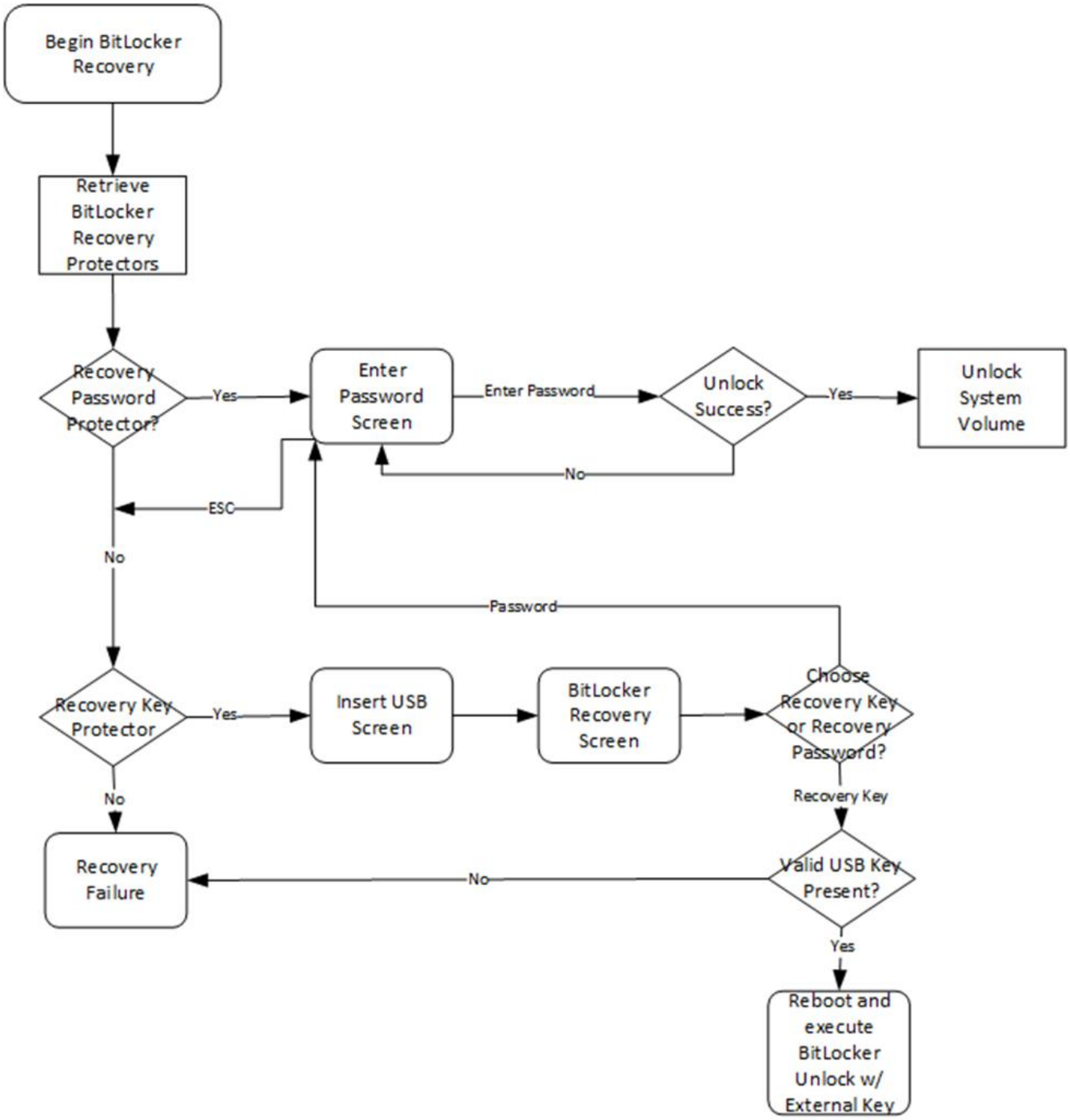


Figure 5: BitLocker Recovery Sequence

11.2 Administrator Guidance

The installed version of Windows must be checked to match the version that was validated. See 11.2.1 Verifying the Installed Windows Version below for details on how to do this. To sanitize the module, the operator should reformat the hard drive or wipe the device as part of unenrollment for Azure Active Directory.

11.2.1 Verifying the Installed Windows Version

The following methods may be used to check the installed version of Windows against the version number listed in Section 2.2 Tested and Vendor Affirmed Module Version and Identification.

Using the Windows command prompt or Windows PowerShell (local or remote):

- Open a command prompt or PowerShell window.
- At the prompt, type **systeminfo** and press the Enter key.
- Near the top of the output, information like the following is displayed. The OS Version field lists the installed Windows version. Compare this version number against the version number listed in Section 2.2 Tested and Vendor Affirmed Module Version and Identification.

```
OS Name:                   Microsoft Windows 11 Enterprise
OS Version:                10.0.xxxxx N/A Build xxxxx
OS Manufacturer:           Microsoft Corporation
```

For Windows installations without a user interface, e.g., Windows Server with the Core Installation option, a server management solution may also be used to validate the installed Windows version. For example, the Overview page of Windows Admin Center Server Manager lists the version number under the Operating System category. For more information, see Manage Servers with Windows Admin Center.

11.2.2 Verifying the Cryptographic Module Version and its Signature

To confirm the version number or digital signature of the module, locate the module binary or binaries named in Section 2.1 Description in their default installation location. The list below identifies the default install locations for the Windows cryptographic module binaries for a system where Windows has been installed on the C: drive.

- bootmgfw.efi - C:\Windows\Boot\EFI\
- bootmgr.efi - C:\Windows\Boot\EFI\
- bootx64.efi - C:\Windows\Boot\EFI\

To validate the module version number, use Windows Explorer or PowerShell (local or remote).

- Using Windows Explorer:
  - Open Windows Explorer and navigate to the folder where the binary is installed, referencing the list above for the correct location.
  - Find the file in the folder and **right click** on the file's icon.
  - Select **Properties** from the context menu.
  - Select the **Details** tab.
  - Compare the version number in the **File version** field against the version identified in .

- Using PowerShell:
  - Open a **PowerShell** window.
  - Use the **Get-ItemProperty cmdlet** together with the path of the cryptographic module binary identified above, formatting the output as a list. For example, if the binary path is `C:\Windows\Boot\EFI\bootmgfw.efi`, the PowerShell command is:

    ```
    Get-ItemProperty -Path "C:\Windows\Boot\EFI\bootmgfw.efi" | Format-List
    ```

  - The cmdlet will return output that summarizes file property details, including the version number. If the version number listed in the **VersionInfo / ProductVersion** field matches one of the version numbers identified in Section 2.2 Tested and Vendor Affirmed Module Version and Identification, then the module version has been verified.
  - Full documentation for the Get-ItemProperty cmdlet may be found at: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-itemproperty.

To validate the Windows digital signature for the module binary, use Windows Explorer or PowerShell (local or remote).

- Using Windows Explorer:
  - Open Windows Explorer and navigate to the folder where the binary is installed, referencing the list at the beginning of this section for the correct location.
  - Find the file in the folder and **right click** on the file's icon.
  - Select **Properties** from the context menu.
  - Select the **Digital Signatures** tab.
  - In the **Signature** list, select the **Microsoft Windows** signer.
  - Click the **Details** button.
  - Under the Digital Signature Information, you should see: "`This digital signature is OK.`" If that condition is true then the digital signature has been verified.

- Using PowerShell:
  - Open a **PowerShell** window.
  - Use the **Get-AuthenticodeSignature cmdlet** together with the path the path of the cryptographic module binary identified at the beginning of this section. For example, if the binary path is `C:\Windows\Boot\EFI\bootmgfw.efi`, the PowerShell command is:

    ```
    Get-AuthenticodeSignature -FilePath "C:\Windows\Boot\EFI\bootmgfw.efi"
    ```

- The cmdlet will return output that summarizes signature details. If the signature is valid, the **Status** field will show "`Valid`" and the **StatusMessage** field will show "`Signature verified.`"
- Full documentation for the Get-AuthenticodeSignature cmdlet may be found at: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-authenticodesignature.

11.2.3 Boot Manager and BitLocker Guidance

BitLocker, which includes parts of the Boot Manager, collects authorization factors, known as “protectors”, by reading data or interacting with the user. BitLocker uses these protectors to encrypt entire disk volumes. In Windows, the following key types may be derived from the authorization factors and used to unlock BitLocker encrypted OS volumes:

Authorization Factor	Additional Guidance
A password entered by the user.	The user should choose a strong password.
An external key which may be used during boot if it is stored on a USB drive, or during recovery if it is stored on a USB drive or typed in manually.	None.
A key stored in the TPM	Consider using a TPM that contains a validated cryptographic module.
A TPM key combined with a user-provided PIN	Consider using a TPM that contains a validated cryptographic module.
A TPM key combined with the external key	Consider using a TPM that contains a validated cryptographic module.
A TPM key combined with a PIN and the external key	Consider using a TPM that contains a validated cryptographic module.
A TPM key combined with a network key	Consider using a TPM that contains a validated cryptographic module.
A key stored on disk and only used when BitLocker is disabled but the drive is encrypted	None.

Table 28: BitLocker Authorization Factors

For more information on the boot sequence and how Boot Manager collects and uses these protectors, see the diagrams in section 11.1 Installation, Initialization, and Startup Procedures. See section 2.9.1 BitLocker Authorization Factors for the rationale of why each protector is used in the approved mode of operation.

11.3 Non-Administrator Guidance

The module implements a single role only, Cryptographic Officer. See the Administrator Guidance above.

11.4 Design and Rules

The module is a multi-chip standalone software-hybrid module that provides cryptographic services within the Operational Environments listed in Section 2.2 Tested and Vendor Affirmed Module Version and Identification, Table 7. The other sections of this Security Policy provide additional details on the design of the module and its rules of operation.

12 Mitigation of Other Attacks

12.1 Attack List

The following table lists the mitigations of other attacks for this cryptographic module:

Algorithm	Protected Against	Mitigation
SHA1	Timing Analysis Attack	Constant time implementation

Algorithm	Protected Against	Mitigation
	Cache Attack	Memory access pattern is independent of any confidential data
SHA2	Timing Analysis Attack	Constant time implementation
	Cache Attack	Memory access pattern is independent of any confidential data
AES	Timing Analysis Attack	Constant time implementation
	Cache Attack	Memory access pattern is independent of any confidential data
		Protected against cache attacks only when running on a processor that implements AES-NI

Table 29: Mitigation of Other Attacks

13 Standards References

- FIPS 140-3, Security Requirements for Cryptographic Modules, https://csrc.nist.gov/publications/detail/fips/140/3/final
- FIPS 180-4, Secure Hash Standard (SHS), https://csrc.nist.gov/publications/detail/fips/180/4/final
- FIPS 186-4, Digital Signature Standard (DSS), https://csrc.nist.gov/publications/detail/fips/186/4/final
- FIPS 186-5, Digital Signature Standard (DSS), https://csrc.nist.gov/pubs/fips/186-5/final
- FIPS 197, Advanced Encryption Standard (AES), https://csrc.nist.gov/publications/detail/fips/197/final
- NIST SP 800-132, Recommendation for Password-Based Key Derivation: Part 1: Storage Applications, https://csrc.nist.gov/publications/detail/sp/800-132/final
- NIST SP 800-133 Rev. 2, Recommendation for Cryptographic Key Generation, https://csrc.nist.gov/publications/detail/sp/800-133/rev-2/final