Non-proprietary Security Policy for

FIPS 140-2 Validation

BitLocker® Windows OS Loader
(winload) in
Microsoft Windows 10
Windows 10 Pro
Windows 10 Enterprise
Windows 10 Enterprise LTSB
Windows 10 Mobile
Windows Server 2016 Standard
Windows Server 2016 Datacenter
Windows Storage Server 2016

DOCUMENT INFORMATION

Version Number	1.3
Updated On	July 26, 2019

CHANGE HISTORY

Date	Version	Updated By	Change
08 DEC 2016	1.0	Tim Myers	First release to validators
28 MAR 2018	1.1	Mike Grimm	Update for build 10.0.14393.1770 (3SUB)
23 MAY 2018	1.2	Iffat Qamar	Updated Bounded Module
26 JULY 2019	1.3	Garrett Burk	Updates in response to comments

TABLE OF CONTENTS

1 Introduction

The BitLocker ® Windows OS Loader, WINLOAD.EXE, is an operating system loader which loads the operating system kernel (ntoskrnl.exe) and other boot stage binary image files. Throughout this document, the BitLocker Windows OS Loader may be called the BitLocker Windows OS Loader (winload) in Microsoft Windows <…>, Windows OS Loader, or Winload for short.

The Operational Environments (OEs) are:

1. Windows 10 Enterprise Anniversary Update (x86) running on a Dell Inspiron 660s - Intel Core i3 without AES-NI or PCLMULQDQ or SSSE 3
2. Windows 10 Enterprise Anniversary Update (x64) running on a Microsoft Surface Pro 3 - Intel Core i7 with AES-NI and PCLMULQDQ and SSSE 3
3. Windows 10 Enterprise Anniversary Update (x64) running on a Microsoft Surface Pro 4 – Intel Core i5 with AES-NI and PCLMULQDQ and SSSE 3
4. Windows 10 Enterprise Anniversary Update (x64) running on a Microsoft Surface Book – Intel Core i7 with AES-NI and PCLMULQDQ and SSSE 3
5. Windows 10 Enterprise Anniversary Update (x64) running on a Dell Precision Tower 5810MT - Intel Xeon with AES-NI and PCLMULQDQ and SSSE 3
6. Windows 10 Enterprise Anniversary Update (x64) running on a HP Compaq Pro 6305 - AMD A4 with AES-NI and PCLMULQDQ and SSSE 3
7. Windows 10 Pro Anniversary Update (x86) running on a Dell Inspiron 660s - Intel Core i3 without AES-NI or PCLMULQDQ or SSSE 3
8. Windows 10 Pro Anniversary Update (x64) running on a Microsoft Surface Pro 3 - Intel Core i7 with AES-NI and PCLMULQDQ and SSSE 3
9. Windows 10 Pro Anniversary Update (x64) running on a Microsoft Surface Pro 4 - Intel Core i5 with AES-NI and PCLMULQDQ and SSSE 3
10. Windows 10 Pro Anniversary Update (x64) running on a Microsoft Surface Book - Intel Core i7 with AES-NI and PCLMULQDQ and SSSE 3
11. Windows 10 Pro Anniversary Update (x64) running on a Dell Precision Tower 5810MT - Intel Xeon with AES-NI and PCLMULQDQ and SSSE 3
12. Windows 10 Anniversary Update (x64) [consumer] running on a Microsoft Surface 3 - Intel Atom x7 with AES-NI and PCLMULQDQ and SSSE 3
13. Windows 10 Anniversary Update (x86) [consumer] running on a Dell Inspiron 660s - Intel Core i3 without AES-NI or PCLMULQDQ or SSSE 3
14. Windows 10 Anniversary Update (x64) [consumer] running on a Dell XPS 8700 - Intel Core i7 with AES-NI and PCLMULQDQ and SSSE 3
15. Windows 10 Enterprise LTSB Anniversary Update (x86) running on a Dell Inspiron 660s - Intel Core i3 without AES-NI or PCLMULQDQ or SSSE 3
16. Windows 10 Enterprise LTSB Anniversary Update (x64) running on a Dell Precision Tower 5810MT - Intel Xeon with AES-NI and PCLMULQDQ and SSSE 3
17. Windows 10 Enterprise LTSB Anniversary Update (x64) running on a Dell XPS 8700 - Intel Core i7 with AES-NI and PCLMULQDQ and SSSE 3
18. Windows Server 2016 Standard Edition running on a HP Compaq Pro 6305 - AMD A4 with AES-NI and PCLMULQDQ and SSSE 3
19. Windows Server 2016 Standard Edition running on a Dell PowerEdge R630 Server - Intel Xeon with AES-NI and PCLMULQDQ and SSSE 3
20. Windows Server 2016 Datacenter Edition running on a Dell PowerEdge R630 Server - Intel Xeon with AES-NI and PCLMULQDQ and SSSE 3

21. Windows Storage Server 2016 running on a Dell PowerEdge R630 Server - Intel Xeon with AES-NI and PCLMULQDQ and SSSE 3
22. Windows 10 Mobile Anniversary Update running on a Microsoft Lumia 950 - Qualcomm Snapdragon 808 (A57, A53)

herein referred to as Windows 10 OEs.

All the computers for Windows 10 and Windows Server listed above are 64-bit Intel architecture and implement the AES-NI instruction set. The exceptions are:

- Dell Inspiron 660s - Intel Core i3

Windows 10 Mobile runs on the ARM architecture, which does not implement AES-Ni instructions:

- Microsoft Lumia 950 - Qualcomm Snapdragon 808 (A57, A53)

1.1 List of Cryptographic Module Binary Executables

WINLOAD.EXE – Version 10.0.14393.1770 for Windows 10 OEs on systems using conventional BIOS
WINLOAD.EFI – Version 10.0.14393.1770 for Windows 10 OEs on systems using UEFI firmware

Note: both versions of winload exist on all platforms. The firmware determines which is used.

1.2 Version Info

10.0.14393.1770 for Windows 10 OEs

1.3 Brief Module Description

BitLocker Windows OS Loader is the binary executable for loading the Windows operating system.

1.4 Validated Platforms

The BitLocker Windows OS Loader components listed in Section 1.1 were validated using the machine configurations specified in the list of Windows 10 OEs.

1.5 Cryptographic Boundary

The software binary that comprises the cryptographic boundary for Windows OS Loader is Winload.exe or Winload.efi depending on the CPU architecture. The cryptographic boundary is also defined by the enclosure of the computer system, on which Windows OS Loader is to be executed. The physical configuration of Windows OS Loader, as defined in FIPS 140-2, is multi-chip standalone.

2 Security Policy

Windows OS Loader operates under several rules that encapsulate its security policy.

- Windows OS Loader is validated on the platforms listed in Section 1.4.
- Windows OS Loader operates in FIPS mode of operation only when used with the FIPS validated version of Windows 10 OEs Boot Manager (bootmgr) validated to FIPS 140-2 under Cert. #3487, operating in FIPS mode.

- Windows 10 OEs are operating systems supporting a “single user” mode where there is only one interactive user during a logon session.
- Windows OS Loader is only in its Approved mode of operation when Windows is booted normally, meaning Debug mode is disabled and Driver Signing enforcement is enabled.
- The Debug mode status and Driver Signing enforcement status can be viewed by using the bcdedit tool.
- Keys and CSPs defined while operating in the FIPS mode of operation shall not be accessed or shared when operating in a non-Approved mode of operation, and vice versa. The operator of the module must follow the zeroization procedures detailed in this document when switching between modes in order to assure compliance.

The following diagram illustrates the master components of the Windows OS Loader module:

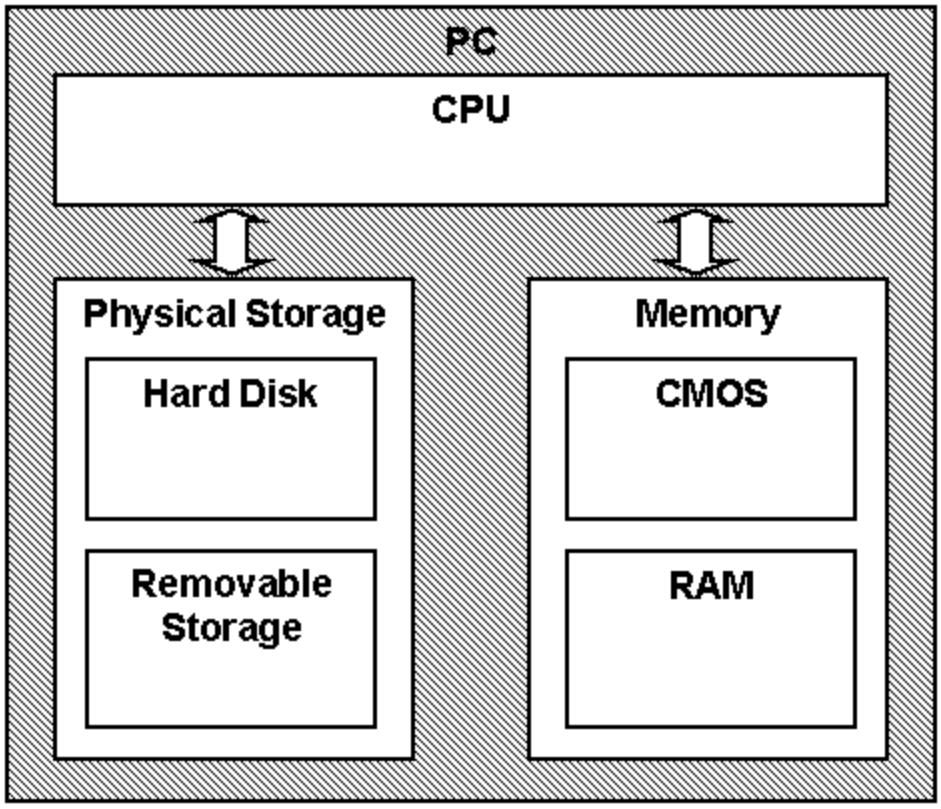

Figure 1 Master Components

The following diagram illustrates Windows OS Loader module interaction with other cryptographic modules:

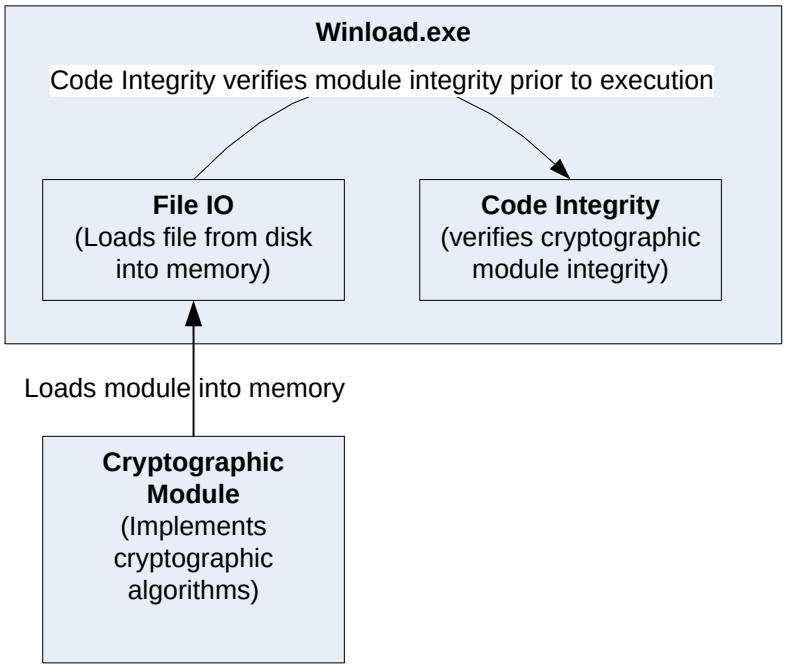

Figure 2 Module Interaction

- Windows OS Loader verifies the integrity of multiple kernel mode crypto modules. This verification relies on RSA 2048-bit signature verification using SHA-256. If the verification fails, the modules are not loaded into memory, and this will prevent Windows from booting. The following binaries are verified in this manner:
  - CI.DLL
  - CNG.SYS
- Windows OS Loader also verifies the integrity of other kernel mode drivers outside of the set of Windows crypto modules. This verification may use other supported RSA modulus sizes (e.g.: 1024 and 3072) and other hash algorithms (e.g.: SHA-1, SHA-384, and SHA-512).

2.1 FIPS 140-2 Approved Algorithms

Windows OS Loader implements the following FIPS 140-2 Approved algorithms:

- FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 1024, 2048 and 3072 moduli; supporting SHA-1, SHA-256 , SHA-384 and SHA-512 (Cert. # 2833)
- FIPS 180-4 SHS SHA-1, SHA-256, SHA-384 and SHA-512 (Cert. # 4250)
- FIPS 197 AES CBC 128 and 256, SP 800-38E AES XTS[1] 128 and 256; SP 800-38C AES CCM 256 (Cert. # 5297 and Cert. # 5295)

[1] For XTS-AES the length of the data unit does not exceed 2^20 blocks. XTS-AES mode is only used by the module for the cryptographic protection of data on storage devices.

Note: not all the algorithms / modes verified through the CAVP certificates listed are implemented by this module.

2.2 Non-Approved Algorithms

Windows OS Loader also has a legacy implementation of MD5[2] for backwards compatibility with the verification of the certificate chain of old certificates that might have been used by certificate authorities (CAs) to sign certificates on kernel mode drivers outside of Windows. This legacy implementation of MD5 is not used for checking the integrity of this cryptographic module nor any other Windows cryptographic module. Windows OS Loader has an NDRNG that is a non-Approved, but allowed algorithm. The IEEE 1619-2007 XTS-AES algorithm is also implemented in the Windows OS Loader. This XTS-AES algorithm is not allowed for usage in the FIPS Approved mode of operation.

2.3 Cryptographic Bypass

Cryptographic bypass is not supported by Windows OS Loader.

2.4 FIPS 140-2 Approved Algorithms from Bounded Modules

A bounded module is a FIPS 140 module which provides cryptographic functionality that is relied on by a downstream module. As described in the Integrity Chain of Trust section, Windows OS Loader depends on the following modules and algorithms:

Implemented in Boot Manager (module certificate #3487):

- CAVP certificate #2833 for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificate # 4250 for FIPS 180-4 SHS supporting SHA-256 (Cert. #4250)

2.5 Machine Configurations

Windows OS Loader was tested using the machine configurations listed in Section 1.4 - Validated Platforms

3 Operational Environment

The operational environment for Windows OS Loader is the Windows 10 OEs running on the software and hardware configurations listed in Section 1.4 - Validated Platforms.

Windows OS Loader services are only available before the startup of the operating system. This is done inside the Trusted Computing Base (TCB).

4 Integrity Chain of Trust

4.1 Conventional BIOS and UEFI without Secure Boot Enabled

Boot Manager is the start of the chain of trust. It cryptographically checks its own integrity during its startup. It then cryptographically checks the integrity of the Windows OS Loader before starting it. The

[2] MD5 is not allowed for usage in FIPS mode.

Windows OS Loader then checks the integrity of the Code Integrity crypto module, the operating system kernel, and other boot stage binary images. An RSA signature with a 2048-bit key and SHA-256 message digest are used.

4.2 UEFI with Secure Boot Enabled

On UEFI systems with Secure Boot enabled, Boot Manager is still the OS binary from which the integrity of all other OS binaries is rooted, and it does cryptographically check its own integrity. However, Boot Manager's integrity is also checked and verified by the UEFI firmware, which is the root of trust on Secure Boot enabled systems. An RSA signature with a 2048-bit key and SHA-256 message digest are used.

5 Ports and Interfaces

5.1 Control Input Interface

The Windows OS Loader Control Input Interface is the set of internal functions responsible for intercepting control input. These functions are:

1. BlBdInitialize – Reads the system status to determine if a boot debugger is attached.
2. OslMain – This function receives and parses the Boot Application parameters, which are passed to the module when execution is passed from Boot Manager.
3. BlInitializeLibrary – Performs the parsing Boot Application parameters.
4. BlXmiRead – Reads the operator selection from the Winload user interface.

5.2 Status Output Interface

The Status Output Interface is the BlXmiWrite function that is responsible for displaying the integrity verification errors to the screen. The Status Output Interface is also defined as the BlLogData responsible for writing the name of the corrupt driver to the bootlog.

5.3 Data Output Interface

The Data Output Interface is represented by the OslArchTransferToKernel function and the AhCreateLoadOptionsString function. OslArchTransferToKernel is responsible for transferring the execution from Winload to the initial execution point of the Windows 10 OEs kernel. Data exits the module in the form of the initial instruction address of the Windows 10 OEs kernel.

Data exits the module from the AhCreateLoadOptionsString function in the form of boot application parameters passed to the Windows 10 OEs kernel.

5.4 Data Input Interface

The Data Input Interface is represented by the BlFileReadEx function and the BlDeviceRead function. BlFileReadEx is responsible for reading the binary data of unverified components from the computer hard drive. In addition the BitLocker Full Volume Encryption Key (FVEK) can also be entered into the module over the module's data input interface. BlDeviceRead is responsible for reading data directly from devices.

6 Specification of Roles

Windows OS Loader supports both User and Cryptographic Officer roles (as defined in FIPS 140-2). Both roles have access to all services implemented in Windows OS Loader. The module does not implement any authentication services. Therefore, roles are assumed implicitly by booting the Windows 10 OEs operating systems.

6.1 Maintenance Roles

Maintenance roles are not supported.

6.2 Multiple Concurrent Interactive Operators

There is only one interactive operator in Single User Mode. When run in this configuration, multiple concurrent interactive operators are not supported.

7 Services

Windows OS Loader services are described below. It does not export any cryptographic functions. The only service triggered by the User/Cryptographic Officer is zeroization. Everything else is started by the Boot Manager. The only service for which there is any output to the User/Cryptographic Officer is the Show Status service.

1. **Load the OS** - The main service is to load the Windows 10 OEs operating system kernel (ntoskrnl.exe) and other boot stage binary image files, including Code Integrity cryptographic module (ci.dll), after it validates their integrity using its cryptographic algorithm implementations using the FIPS 140-2 approved algorithms mentioned below. After the verified kernel and boot stage binary image files, including Code Integrity, are loaded, Windows OS Loader passes the execution control to the kernel and it terminates its own execution. In addition to this service, Windows OS Loader also provides status and self-test services. The Crypto officer and User have access to all services WINLOAD supports. If the integrity of the kernel or Code Integrity is not verified, Windows OS Loader does not transfer the execution to the kernel.
2. **Show Status** – The module provides a show status service that is automatically executed by the module to provide the status response of the module either via output to the GPC monitor or to log files.
3. **Self-Tests** - The module provides a power-up self-tests service that is automatically executed when the module is loaded into memory.
4. **Signature Checks** - Verifies the integrity of system policies and implements validation of signature checks for the allowed list of legacy policies
5. **Zeroization** (see Section 9 Cryptographic Key Management)
6. **Entropy** - Windows OS Loader also implements an entropy source which is used by subsequently-loaded Windows components. This source gathers entropy from the following sources:
    a. The contents of the registry value HKLM\System\RNG\Seed, which is written by the Kernel Mode Cryptographic Primitives Library (cng.sys) during its normal operation.

b. The contents of the registry value HKLM\System\RNG\ExternalEntropy, which can be populated by system administrators. This value is overwritten after reading, to ensure that it does not get reused.
c. If a Trusted Platform Module (TPM) is available, the output of a TPM_GetRandom call to the TPM.
d. The current system time.
e. The contents of the OEM0 ACPI table in the machine firmware.
f. If the CPU supports the RDRAND CPU instruction, the output of such an operation.
g. If booted from UEFI firmware which supports the UEFI entropy protocol, the output of the UEFI random number generator.
h. The CPU timings.

These inputs are then combined using SHA-512, and the entropy source is conditioned using a non-Approved RNG. From this NDRNG, a block of output bytes is passed to the Windows kernel at boot time. This block of output bytes is used by CNG.SYS as one of its entropy sources.

The Entropy service is considered a "Non-Approved, but Allowed" service. It is only "Allowed" in the context that it is being used by another FIPS 140-2 Approved module, i.e., the Kernel Mode Cryptographic Primitives Library (cng.sys), in order to provide entropy to one of the FIPS-approved DRBGs.

7. **Legacy Certificate Chain Authentication** (non-FIPS Approved service; see Section 2.2 Non-Approved Algorithms)

The following table maps the services to their corresponding algorithms and critical security parameters (CSPs).

Table 1

Service	Algorithms	CSPs	Invocation
Load the OS	FIPS 186-4 RSA PKCS#1 (v1.5) verify with public key FIPS 180-4 SHS: SHA-256 hash SHA-512 hash AES CBC 128 and 256 bits AES XTS 128 and 256 bits AES CCM 256 bits IEEE 1619-2007 XTS-AES (non-FIPS Approved algorithm)	Asymmetric Public keys (to verify digital signatures of OS components) Full Volume Encryption Key (FVEK) (to load the BitLocker encrypted data containing the OS)	This service is fully automatic. The User / Cryptographic Officer does not take any actions to start this service.
Show Status	None	None	This service is fully automatic. The User / Cryptographic Officer does not take any

			actions to start this service.
Self-Tests	FIPS 186-4 RSA PKCS#1 (v1.5) verify with public key KAT and signature verification KAT FIPS 180-4 SHS: SHA-1 KAT SHA-256 KAT SHA-512 KAT AES CBC KAT AES CCM KAT AES XTS KAT	None	This service is fully automatic. The User / Cryptographic Officer does not take any actions to start this service.
Signature Checks	RSA PKCS#1 (v1.5) verify with public key SHA-1 hash SHA-256 hash SHA-384 hash SHA-512 hash	Microsoft Root Certificate Authority (CA) Public Key	This service is fully automatic. The User / Cryptographic Officer do not take any actions to start this service.
Zeroization	None	All CSPs	See section 9.
Entropy	SHA-512	None	This service is fully automatic. The User / Cryptographic Officer does not take any actions to start this service.
Legacy certificate chain authentication (non-FIPS Approved service)	MD5 (non-FIPS Approved algorithm)	Asymmetric Public keys	This service is fully automatic. The User / Cryptographic Officer does not take any actions to start this service.

7.1 Show Status Services

The User and Cryptographic Officer roles have the same Show Status functionality, which is, for each function, the status information is returned to the caller as the return value from the function.

7.2 Self-Test Services

The User and Cryptographic Officer roles have the same Self-Test functionality, which is described in Section 10 Self-Tests.

7.3 Service Inputs / Outputs

The User and Cryptographic Officer roles have service inputs and outputs as specified in Section 5 Ports and Interfaces.

8 Authentication

The Windows OS Loader does not implement any authentication services. The User and Cryptographic Officer roles are assumed implicitly by booting the Windows operating system.

9 Cryptographic Key Management

Windows OS Loader does not store any secret or private cryptographic keys across power-cycles. However, it does use an AES key in support of the BitLocker feature:

- Full Volume Encryption Key (FVEK) - 128 or 256-bit AES key that is used to decrypt data on disk sectors of the hard drive.

Procedural zeroization of this ephemeral key (RAM only) for this software cryptographic module consists of rebooting the operating system.

The key enters the module via machine memory in plaintext; a pointer to this memory is provided to Winload by Boot Manager after it verifies the integrity of Winload.

Windows OS Loader also uses the Microsoft root CA public key certificate stored on the computer hard disk in plaintext to verify digital signatures using its implementation of RSA PKCS#1 (v1.5) verify. This public key is available to both roles. Procedural zeroization of persistent keys for this software cryptographic module consists of uninstallation of the cryptographic module and reformatting and overwriting, at least once, the hard drive or other permanent storage media upon which the operating system was installed.

9.1 Access Control Policy

All the keys (mentioned above) are accessed only by the Windows OS Loader service that loads the operating system kernel (ntoskrnl.exe) and other boot stage binary image files, including Code Integrity. This service only has execute access to the keys mentioned above. For this reason, an access control policy table is not included in this document.

10 Self-Tests

10.1 Power-On Self-Tests

Windows OS Loader performs the following power-on (startup) self-tests:

- RSA PKCS#1 (v1.5) verify with public key Known Answer Test
  - RSA signature verification Known Answer Test with 1024-bit key and SHA-1 message digest
  - RSA signature verification Known Answer Test with 2048-bit key and SHA-256 message digest
- SHS (SHA-1) Known Answer Test
- SHS (SHA-256) Known Answer Test
- SHS (SHA-512) Known Answer Test

- AES-CBC Encrypt/Decrypt Known Answer Tests
- AES-CCM Encrypt/Decrypt Known Answer Tests
- XTS-AES Encrypt/Decrypt Known Answer Tests

If the self-test fails, the module will not load and status will be returned. If the status is not STATUS_SUCCESS, then that is the indicator a self-test failed.

10.2 Conditional Self-Tests

Windows OS Loader performs the following conditional self-test:

- Non-Approved RNG CRNGT (entropy pool)

11 Design Assurance

The secure installation, generation, and startup procedures of this cryptographic module are part of the overall operating system secure installation, configuration, and startup procedures for the Windows 10 OEs. The various methods of delivery and installation for each product are listed in the following table.

Table 2

Product	Delivery and Installation Method
Windows 10, Windows 10 Pro, Windows 10 Enterprise, Windows Enterprise LTSB, Windows Server 2016 Standard, Windows Server 2016 Datacenter	• Pre-installed on the computer by OEM • Download that updates to Windows 10 • Enterprise IT deployment
Surface Book, Surface Pro 4, Surface Pro 3, Surface 3, Lumia 950, Windows Storage Server 2016	• Pre-installed by the OEM (Microsoft)

After the operating system has been installed, it must be configured by enabling the "System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing" policy setting followed by restarting the system. This procedure is all the crypto officer and user behavior necessary for the secure operation of this cryptographic module.

An inspection of authenticity of the physical medium can be made by following the guidance at this Microsoft web site: https://www.microsoft.com/en-us/howtotell/default.aspx

The installed version of Windows 10 OEs must be verified to match the version that was validated. See Appendix A for details on how to do this.

For Windows Updates, the client only accepts binaries signed by Microsoft certificates. The Windows Update client only accepts content whose SHA-2 hash matches the SHA-2 hash specified in the metadata. All metadata communication is done over a Secure Sockets Layer (SSL) 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 Appendix A for details on how to do this.

12 Mitigation of Other Attacks

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

Table 3

Algorithm	Protected Against	Mitigation	Comments
SHA1	Timing Analysis Attack	Constant Time Implementation	
	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 used with AES NI

13 Security Levels

The security level for each FIPS 140-2 security requirement is given in the following table.

Table 4

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	NA
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	2
Mitigation of Other Attacks	1

14 Additional Details

For the latest information on Microsoft Windows, check out the Microsoft web site at:

https://www.microsoft.com/en-us/windows

For more information about FIPS 140 validations of Microsoft products, please see:

https://docs.microsoft.com/en-us/windows/security/threat-protection/fips-140-validation

15 Appendix A – How to Verify Windows Versions and Digital Signatures

15.1 How to Verify Windows Versions

The installed version of Windows 10 OEs must be verified to match the version that was validated using the following method:

1. In the Search box type "cmd" and open the Command Prompt desktop app.
2. The command window will open.
3. At the prompt, enter "ver".
4. The version information will be displayed in a format like this:
   `Microsoft Windows [Version 10.0.xxxxx]`

If the version number reported by the utility matches the expected output, then the installed version has been validated to be correct.

15.2 How to Verify Windows Digital Signatures

After performing a Windows Update that includes changes to a cryptographic module, the digital signature and file version of the binary executable file must be verified. This is done like so:

1. Open a new window in Windows Explorer.
2. Type "C:\Windows\" in the file path field at the top of the window.
3. Type the cryptographic module binary executable file name (for example, "CNG.SYS") in the search field at the top right of the window, then press the Enter key.
4. The file will appear in the window.
5. Right click on the file's icon.
6. Select Properties from the menu and the Properties window opens.
7. Select the Details tab.
8. Note the File version Property and its value, which has a number in this format: xx.x.xxxxx.xxxx.
9. If the file version number matches one of the version numbers that appear at the start of this security policy document, then the version number has been verified.
10. Select the Digital Signatures tab.
11. In the Signature list, select the Microsoft Windows signer.
12. Click the Details button.
13. 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.