

# Microsoft Windows FIPS 140 Validation

**Microsoft Windows 10 (Creators Update)** 

**Microsoft Windows 10 Mobile (Creators Update)** 

Non-Proprietary

# Security Policy Document

Document Information	
Version Number	1.03
Updated On	May 24, 2018

© 2018 Microsoft. All Rights Reserved Page 1 of 21 This Security Policy is non-proprietary and may be reproduced only in its original entirety (without revision).

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.

Complying with all applicable copyright laws is the responsibility of the user. This work is licensed under the Creative Commons Attribution-NoDerivs-NonCommercial License (which allows redistribution of the work). To view a copy of this license, visit <u>http://creativecommons.org/licenses/by-nd-nc/1.0/</u> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

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.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft, Windows, the Windows logo, Windows Server, and BitLocker 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.

#### Version History

Version	Date	Summary of changes
1.0	October 3, 2017	Draft sent to NIST CMVP
1.01	February 12, 2018	Updates from CMVP review
1.02	March 28, 2018	Updates for builds 10.0.15063.728 (1SUB)
1.03	May 24, 2018	Bounded Modules added

#### TABLE OF CONTENTS

<u>SECURI</u>	TY POLICY DOCUMENT1
<u>1</u>	INTRODUCTION
1.1	LIST OF CRYPTOGRAPHIC MODULE BINARY EXECUTABLES
1.2	VALIDATED PLATFORMS
<u>2</u>	CRYPTOGRAPHIC MODULE SPECIFICATION7
2.1	CRYPTOGRAPHIC BOUNDARY7
2.2	FIPS 140-2 Approved Algorithms7
2.3	NON-APPROVED ALGORITHMS
2.4	FIPS 140-2 APPROVED ALGORITHMS FROM BOUNDED MODULES
2.5	CRYPTOGRAPHIC BYPASS
2.6	HARDWARE COMPONENTS OF THE CRYPTOGRAPHIC MODULE
<u>3</u>	CRYPTOGRAPHIC MODULE PORTS AND INTERFACES9
3.1	CONTROL INPUT INTERFACE
3.2	STATUS OUTPUT INTERFACE
3.3	DATA OUTPUT INTERFACE9
3.4	DATA INPUT INTERFACE
<u>4</u>	ROLES, SERVICES AND AUTHENTICATION10
4.1	Roles10
4.2	Services
4.3	AUTHENTICATION
<u>5</u>	FINITE STATE MODEL12
5.1	SPECIFICATION
<u>6</u>	OPERATIONAL ENVIRONMENT14
6.1	SINGLE OPERATOR14
6.2	CRYPTOGRAPHIC ISOLATION

6.3	INTEGRITY CHAIN OF TRUST14
<u>7</u>	CRYPTOGRAPHIC KEY MANAGEMENT16
7.1	CRITICAL SECURITY PARAMETERS16
7.2	ZEROIZATION16
7.2.1	VOLATILE KEYS
7.2.2	Persistent Keys16
7.3	Access Control Policy16
<u>8</u>	SELF-TESTS16
8.1	POWER-ON SELF-TESTS16
8.2	CONDITIONAL SELF-TESTS
<u>9</u>	DESIGN ASSURANCE
<u>10</u>	MITIGATION OF OTHER ATTACKS19
<u>11</u>	SECURITY LEVELS
<u>12</u>	ADDITIONAL DETAILS
<u>13</u>	APPENDIX A – HOW TO VERIFY WINDOWS VERSIONS AND DIGITAL SIGNATURES
13.1	How to Verify Windows Versions21
13.2	How to Verify Windows Digital Signatures21

Page 6 of 21

# **1** Introduction

The Windows Operating System (OS) Loader, which consists of the binaries WINLOAD.EFI and HVLOADER.EFI, is the operating system loader that loads the operating system kernel (ntoskrnl.exe) and other boot stage binary image files. HVLOADER.EFI is loaded when the Hyper-V feature is enabled. Note that the Hyper-V feature, and hence HVLOADER.EFI, is not a requirement for Windows to be operating in FIPS-approved mode. The Windows OS Loader is a part of BitLocker Drive Encryption, which is a data protection feature of the Windows 10 operating system which encrypts data on a storage volume.

# 1.1 List of Cryptographic Module Binary Executables

The Windows OS Loader module contains the following binaries and evaluated versions:

- WINLOAD.EFI
- HVLOADER.EFI

The Windows build covered by this validation are:

Version 10.0.15063 Version 10.0.15063.728

#### 1.2 Validated Platforms

The Windows editions covered by this validation are:

- Microsoft Windows 10 Home Edition (32-bit version)
- Microsoft Windows 10 Pro Edition (64-bit version)
- Microsoft Windows 10 Enterprise Edition (64-bit version)
- Microsoft Windows 10 Education Edition (64-bit version)
- Microsoft Windows 10 S Edition (64-bit version)
- Microsoft Windows 10 Mobile
- Microsoft Surface Hub

The Windows OS Loader components listed in Section 1.1 were validated using the combination of computers and Windows operating system editions specified in the following table:

Computer	Windows 10 Home	Windows 10 Pro	Windows 10 Enterprise	Windows 10 Education	Surface Hub	Windows 10 S	Windows 10 Mobile
Microsoft Surface Laptop		V	V			V	
Microsoft Surface Pro		V	V	V			
Microsoft Surface Book			V				
Microsoft Surface Pro 4			V				
Microsoft Surface Pro 3		V					
Microsoft Surface 3			V				
Microsoft Surface 3 with LTE		٧					

© 2018 Microsoft. All Rights Reserved

This Security Policy is non-proprietary and may be reproduced only in its original entirety (without revision).

Page 7 of 21

Microsoft Surface Studio			V		
Microsoft Surface Hub				V	
Windows Server 2016 Hyper-V <sup>1</sup>		V			
Microsoft Lumia 950					V
Microsoft Lumia 950 XL					V
Microsoft Lumia 650					V
Dell Latitude 5285		V			
Dell Inspiron 660s	٧				
Dell Precision Tower 5810MT		V			
Dell PowerEdge R630		V			
HP Elite X3					V
HP Compaq Pro 6305		V			
HP Pro x2 612 G2 Detachable PC with			V		
LTE					
HP Slimline Desktop		V			
Panasonic Toughbook		V			

#### 2 Cryptographic Module Specification

Windows OS Loader is a multi-chip standalone module that operates in FIPS-approved mode during normal operation of the computer and Windows operating system boot sequence.

The following configurations and modes of operation will cause Windows OS Loader to operate in a nonapproved mode of operation:

- Boot Windows in Debug mode
- Boot Windows with Driver Signing disabled

#### 2.1 Cryptographic Boundary

The software cryptographic boundary for Windows OS Loader is defined as the binaries WINLOAD.EFI and HVLOADER.EFI.

#### 2.2 FIPS 140-2 Approved Algorithms

Windows OS Loader implements the following FIPS 140-2 Approved algorithms:<sup>2</sup>

- 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. #2523 for version 10.0.15063)(Cert. #2846 for versions 10.0.15063.728)
- FIPS 180-4 SHS SHA-1, SHA-256, SHA-384, and SHA-512 (Cert. #3790 for version 10.0.15063)(Cert. #4253 for versions 10.0.15063.728)
- FIPS 197 AES CBC 128 and 256, SP 800-38E AES XTS 128 and 256; SP 800-38C AES CCM 256 (Certs. #4624 and #4625 for version 10.0.15063) (Certs. #5300 and #5316 for version 10.0.15063.728)

<sup>&</sup>lt;sup>1</sup> Host OS: Windows Server 2016, hardware platform: Surface Pro 4

<sup>&</sup>lt;sup>2</sup> This module may not use some of the capabilities described in each CAVP certificate.

#### 2.3 Non-Approved Algorithms

Windows OS Loader implements the following non-approved algorithms:

- IEEE 1619-2007 AES-XTS 128 and 256
- A non-deterministic random number generator for entropy that is a not a FIPS Approved algorithm but is allowed by FIPS 140. See **Collecting Initial Entropy for the OS** in <u>Services</u>.

#### 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 <u>Integrity Chain of Trust</u> section, the Windows OS Loader depends on the following algorithms implemented in the Boot Manager (module certificate #3089):

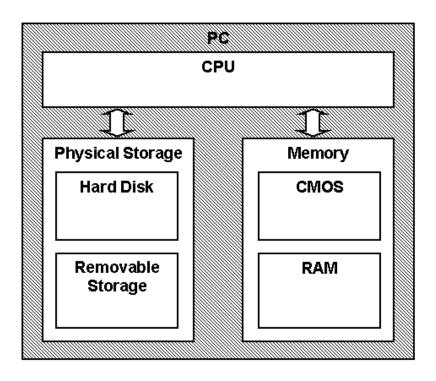
- CAVP certificates #2523 for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificates #2846 for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificates #3790 for FIPS 180-4 SHS SHA-256
- CAVP certificates #4253 for FIPS 180-4 SHS SHA-256

#### 2.5 Cryptographic Bypass

Cryptographic bypass is not supported by Windows OS Loader.

#### 2.6 Hardware Components of the Cryptographic Module

The physical boundary of the module is the physical boundary of the computer that contains the module. The following diagram illustrates the hardware components used by the Windows OS Loader module:



# 3 Cryptographic Module Ports and Interfaces

# 3.1 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. OslMain (WINLOAD) This function receives and parses the Boot Application parameters, which are passed to the module when execution is passed from Boot Manager.
- 2. HvlMain (HVLOADER) This function receives and parses the Boot Application parameters, which are passed to the module when execution is passed from Boot Manager.
- 3. BlBdInitialize Reads the system status to determine if a boot debugger is attached.
- 4. BllnitializeLibrary Performs the parsing Boot Application parameters.
- 5. BIXmiRead Reads the operator selection from the Windows OS Loader user interface.

# 3.2 Status Output Interface

The Status Output Interface is the BIXmiWrite function that is responsible for displaying any integrity verification errors to the display. The Status Output Interface is also defined as the BILogData responsible for writing the name of the corrupt driver to the bootlog.

# 3.3 Data Output Interface

The Data Output Interface is represented by the OslArchTransferToKernel (ARM only) function and the AhCreateLoadOptionsString function. OslArchTransferToKernel is responsible for transferring the

execution from Windows OS Loader to the initial execution point of the Windows 10 kernel. Data exits the module in the form of the initial instruction address of the Windows 10 kernel.

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

#### 3.4 Data Input Interface

The Data Input Interface is represented by the BIFileReadEx function and the BIDeviceRead function. BIFileReadEx 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. BIDeviceRead is responsible for reading data directly from devices.

# 4 Roles, Services and Authentication

#### 4.1 Roles

In Windows 10, authentication and assignment of roles happens after the OS boots. Since Windows OS Loader functions only during the period between power-on and OS initialization, the module's functions are fully automatic and not configurable. FIPS 140 validations define formal "User" and "Cryptographic Officer" roles. Both roles can use any Windows OS Loader service.

#### 4.2 Services

Windows OS Loader services are described below. It does not export any cryptographic functions.

- 1. Loading the OS The main service is to load the Windows 10 operating system kernel (ntoskrnl.exe), the Windows hypervisor (hvix64.exe for Intel processors and hvax64.exe for AMD processors), and other boot stage binary image files, including Code Integrity (ci.dll), Secure Kernel Code Integrity (skci.dll), and Kernel Mode Cryptographic Primitives Library (cng.sys) after it validates their integrity using the FIPS 140-2 Approved cryptographic algorithm implementations described below. After the verified kernel and boot stage binary image files are loaded, Windows OS Loader passes the execution control to the operating system kernel and it terminates its own execution. If the integrity of any module is not verified, Windows OS Loader does not transfer the execution to the kernel and displays the boot failure page.
- 2. **Show Status** If Windows OS Loader fails any of the checks specified in the <u>Finite State Model</u> then it reports a boot failure status on the computer monitor.
- 3. **Perform Self-Tests** The module provides a power-up self-tests service that is automatically executed when the module is loaded into memory. See <u>Self-Tests</u>.
- 4. Zeroizing Cryptographic Materal see Cryptographic Key Management.
- 5. **Collecting Initial Entropy** for the OS Windows OS Loader also implements an entropy source for the operating system. Entropy is gathered from the following sources, when they are available on the computer:

- 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 is not 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 OEMO ACPI table in the machine firmware.
- f. If the CPU supports instructions for the RDSEED (used when available) or RDRAND (used if RDSEED is not available), the output from RDSEED or RDRAND.
- g. 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.

This 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 FIPSapproved DRBGs.

Service	Algorithms	CSPs	Invocation
Loading the OS	FIPS 186-4 RSA PKCS#1 (v1.5) verify with public key FIPS 180-4 SHS: SHA-256 hash SHA-384 hash SHA-512 hash AES CBC 128 and 256 bits AES XTS 128 and 256 bits <sup>3</sup> AES CCM 256 bits	Full Volume Encryption Key (FVEK) (to load the BitLocker encrypted data containing the OS) RSA public key (to verify the integrity of Windows OS Loader)	This service is fully automatic.
Show Status	None	None	This service is fully automatic.
Perform Self- Tests	FIPS 186-4 RSA PKCS#1 (v1.5) verify with public key KAT and signature verification KAT	None	This service is fully automatic.

The following table maps the services to their corresponding algorithms and critical security parameters (CSPs) as described in Cryptographic Key Management.

<sup>&</sup>lt;sup>3</sup> The length of the data unit does not exceed 2<sup>20</sup> AES blocks for storage applications such as BitLocker..

Service	Algorithms	CSPs	Invocation
	FIPS 180-4 SHS:		
	SHA-1 KAT		
	SHA-256 KAT		
	SHA-512 KAT		
	AES CBC KAT		
	AES CCM KAT		
	AES XTS KAT		
Zeroizing	None	Full Volume Encryption Key	See Zeroization.
Cryptographic		(FVEK)	
Material			
Collecting	SHA-512	None	This service is fully
Initial Entropy			automatic.

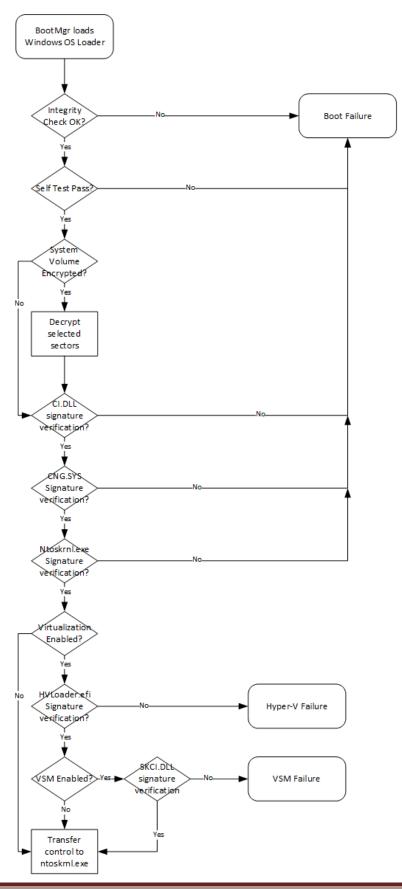
#### 4.3 Authentication

The Windows OS Loader does not implement any authentication services.

# 5 Finite State Model

#### 5.1 Specification

The following diagram shows the finite state model for Windows OS Loader:



# 6 Operational Environment

The operational environment for Windows OS Loader is the Windows 10 operating system running on a supported hardware platform.

#### 6.1 Single Operator

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

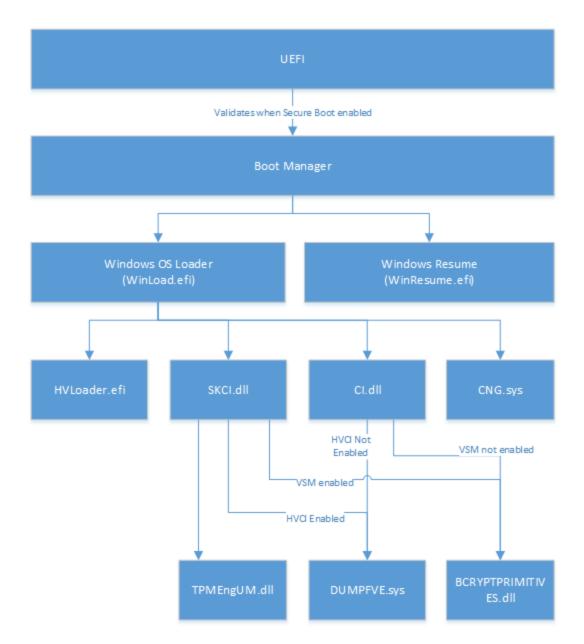
#### 6.2 Cryptographic Isolation

While it is running, Windows OS Loader is the only process running on the computer.

#### 6.3 Integrity Chain of Trust

Windows uses several mechanisms to provide integrity verification depending on the stage in the boot sequence and also on the hardware and configuration. The following diagram describes the Integrity Chain of trust for each supported configuration:

#### Boot Sequence & Chain of Trust



Boot Manager checks the integrity of the WINLOAD.EFI component of the Windows OS Loader before it is loaded.

Windows binaries include a SHA-256 hash of the binary signed with the 2048 bit Microsoft RSA codesigning 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 Windows OS Loader component WINLOAD.EFI verifies the integrity of multiple kernel mode cryptographic modules using the same process described above. The following binaries are verified:

- CNG.SYS
- CI.DLL
- SKCI.DLL (When VSM is enabled)
- HVLOADER.EFI (When Hyper-V is installed)

Windows OS Loader also verifies the integrity of other early boot kernel mode drivers that are not cryptographic modules.

# 7 Cryptographic Key Management

#### 7.1 Critical Security Parameters

When the System Volume is encrypted with Bitlocker, the WINLOAD.EFI component of the Windows OS Loader uses this critical security parameter (CSP):

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

The FVEK is provided to the WINLOAD.EFI component of the Windows OS Loader by Boot Manager.

Windows OS Loader also uses as a CSP the public key component of the Microsoft code signing certificate as described in <u>Integrity Chain of Trust</u>.

#### 7.2 Zeroization

#### 7.2.1 Volatile Keys

The FVEK is zeroized when the module component WINLOAD.EFI is unloaded from memory after control is transferred to ntokrnl.exe.

#### 7.2.2 Persistent Keys

Windows OS Loader does not have any persistent keys.

# 7.3 Access Control Policy

The Windows OS Loader does not allow access to the cryptographic keys contained within it, so an access control table is not included in this document. Windows OS Loader receives keys from outside and then manages them appropriately once received. Windows OS Loader prevents access to its keys by zeroizing them after use.

# 8 Self-Tests

#### 8.1 Power-On Self-Tests

The WINLOAD.EFI component of the 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.

The HVLOADER.EFI component of the 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.

#### 8.2 Conditional Self-Tests

Windows OS Loader performs the following conditional self-test:

• Non-Approved RNG CRNGT (entropy pool)

#### 9 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 operating system.

The Windows 10 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 10 version downloaded from Windows Update.

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

The installed version of Windows 10 must be checked to match the version that was validated. See <u>Appendix A</u> for details on how to do this.

For Windows Updates, the client only accepts binaries signed with Microsoft certificates. The Windows Update client only accepts content whose signed 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 <u>Appendix A</u> for details on how to do this.

# **10 Mitigation of Other Attacks**

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

Algorithm	Protected Against	Mitigation
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 running on a processor that implements AES-NI

# **11 Security Levels**

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

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

# **12 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://technet.microsoft.com/en-us/library/cc750357.aspx

# **13** Appendix A – How to Verify Windows Versions and Digital Signatures

#### **13.1 How to Verify Windows Versions**

The installed version of Windows 10 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.

#### **13.2 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.