Microsoft Windows

FIPS 140 Validation

Microsoft Windows 10 (November 2019 Update and May 2020 Update)

Non-Proprietary

Security Policy Document

Document Information	
Version Number	1.1
Updated On	October 25, 2022

Version History

Version	Date	Summary of changes
1.0	February 5, 2020	Draft sent to NIST CMVP
1.1	October 25, 2022	Updates in response to NIST comments

Table of Contents

1 Introduction

The TCB Launcher Module, which consists of the binary TCBLAUNCH.EXE, is a software-hybrid module that launches the Hypervisor and Secure Kernel and other binary image files needed to launch those two components. The TCB Launcher Module is separate from the normal Windows OS Loader because it provides a secure method to launch the Hypervisor and Secure Kernel.

1.1 List of Cryptographic Module Binary Executables

The TCB Launcher module contains the following binary. Each binary has a distinct implementation per build for each instruction set (x86, x64, ARM64).

- Tcblaunch.exe

The Windows builds covered by this validation are:

- Windows 10 version 1909 build 10.0.18363
- Windows 10 version 2004 build 10.0.19041

1.2 Validated Platforms

The Windows editions covered by this validation are:

- Microsoft Windows 10 Enterprise Edition (64-bit version)

The TCB Launcher components listed in Section 1.1 were validated using the combination of computers and Windows operating system editions specified in the table below.

- All the computers for Windows 10 and Windows Server listed in the table below are all 64-bit Intel architecture and implement the AES-NI instruction set but not the SHA Extensions.

Table 1 Validated Platforms for Windows 10 and Windows Server version 1909

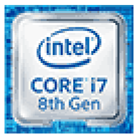
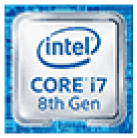
Computer	Windows 10 Enterprise	Processor Image
Dell Latitude 5300 2-in-1 - Intel Core i7-8665U	√	 ark.intel.com

Table 2 Validated Platforms for Windows 10 and Windows Server version 2004

Computer	Windows 10 Enterprise	Processor Image
Panasonic Toughbook FZ 55 - Intel Core i5-8365U	√	 ark.intel.com

2 Cryptographic Module Specification

TCB Launcher 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 configuration prerequisite is required for TCB Launcher to operate in its FIPS-Approved mode of operation:

- Enable Secure Launch[1]

The following configurations and modes of operation will cause TCB Launcher to operate in a non-Approved mode of operation:

- Boot Windows in Debug mode
- Boot Windows with Driver Signing disabled
- Windows enters the ACPI S4 power state

2.1 Cryptographic Boundary

The software-hybrid cryptographic boundary for TCB Launcher consists of disjoint software and hardware components within the same physical boundary of the host platform. The software components are defined as the binary tcblaunch.exe, and the hardware components are the CPUs running on each host platform.

2.2 FIPS 140-2 Approved Algorithms

TCB Launcher implements the following FIPS 140-2 Approved algorithms:[2]

Table 3

Algorithm	Windows 10 version 1909	Windows 10 version 2004

[1] Group policy settings path Console Root\Local Computer Policy\Computer Configuration\Administrative Templates\System\Device Guard and Turn On Virtualization Based Security

[2] This module may not use some of the capabilities described in each CAVP certificate.

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	#C1367	#C1947
FIPS 180-4 SHS SHA-1, SHA-256, SHA-384, and SHA-512	#C 1363	#C1897
NIST SP 800-38D AES-128, AES-192, and AES-256 GCM[3] decryption and GMAC	#C 1363	#C1897
NIST SP 800-90A AES-256 counter mode DRBG	#C 1363	#C1897
NIST SP 800-133 (Section 6.1) Cryptographic Key Generation	Vendor Affirmed	Vendor Affirmed

2.3 Non-Approved Algorithms

TCB Launcher implements the following non-approved algorithms:

- 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 Services.

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, TCB Launcher depends on the following modules and algorithms:

The Windows OS Loader for Windows 10 version 1909 (module certificate #4339) provides

- CAVP certificate #C1367 (Windows 10 and Windows Server) for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificate #C1363 (Windows 10 and Windows Server) for FIPS 180-4 SHS SHA-256

The Windows OS Loader for Windows 10 version 2004 (module certificate #4339) provides

- CAVP certificate #C1947 (Windows 10 and Windows Server) for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificate #C1897 (Windows 10 and Windows Server) for FIPS 180-4 SHS SHA-256

Windows Resume for Windows 10 version 1909 (module certificate #4348) provides

- CAVP certificate #C1367 (Windows 10 and Windows Server) for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificate #C1363 (Windows 10 and Windows Server) for FIPS 180-4 SHS SHA-256

Windows Resume for Windows 10 version 2004 (module certificate #4348) provides

[3] GCM encryption is only used for power-on self-tests.

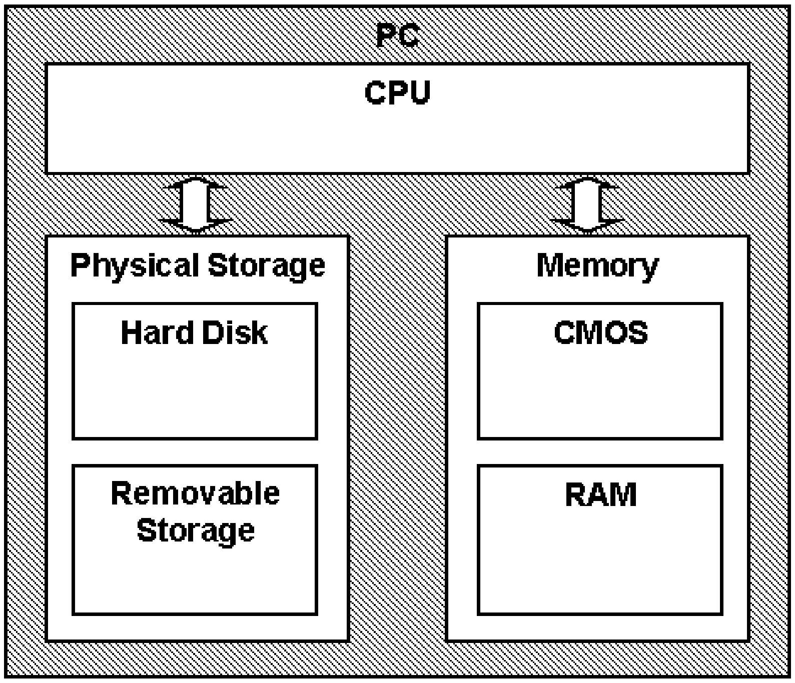
- CAVP certificate #C1947 (Windows 10 and Windows Server) for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificate #C1897 (Windows 10 and Windows Server) for FIPS 180-4 SHS SHA-256

2.5 Cryptographic Bypass

Cryptographic bypass is not supported by TCB Launcher.

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 TCB Launcher module:



3 Ports and Interfaces

3.1 Control Input Interface

The TCB Launcher Module Control Input Interface is the set of internal functions responsible for intercepting control input. These functions are:

- TcbpMain (TCBLAUNCH) – This function receives control and performs the functions of the TCB Launcher Module.

3.2 Status Output Interface

The Status Output Interface is the BlLogDiagWrite function that is responsible for writing diagnostic logging data.

3.3 Data Output Interface

Data output is returned as an output parameter when the TCB Launcher module terminates

3.4 Data Input Interface

The data input into TCB Launcher is which boot application, e.g., Windows OS Loader and Windows Resume, TCB Launcher will transfer control back to when TCB Launcher terminates.

4 Roles, Services and Authentication

4.1 Roles

In Windows 10, authentication and assignment of roles happens after the OS boots. Since TCB Launcher Module does not interact with the user through any service, 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 TCB Launcher Module service.

4.2 Services

TCB Launcher Module services are described below. It does not export any cryptographic functions.

1. **Launching the Hypervisor** - The service launches the Hypervisor, verifying signatures on the binaries that it loads. The service launches the Hypervisor in two different scenarios:

    1) Cold boot – In this scenario the system is being booted from scratch.
    2) Resuming from hibernation – In this scenario the system was hibernated and the system is be restored from the hibernated state.

    In the resume scenario the service decrypts the secure memory of the Hypervisor and then calls the dispatcher to remap the memory and resume the Hypervisor.

2. **Launching the Secure Kernel –** The service launches the Secure Kernel, verifying signatures on the binaries that it loads. The service launches the Secure Kernel in two different scenarios:

    1) Cold boot – In this scenario the system is being booted from scratch.
    2) Resuming from hibernation – In this scenario the system was hibernated and the system is be restored from the hibernated state.

    In the resume scenario the service decrypts the secure memory of the Secure Kernel and then calls the dispatcher to remap the memory and resume the Secure Kernel.
3. **Show Status** – The module does not provide an explicit status interface. Operational status is indicated by successfully initializing Hyper-V and the Secure Kernel.
4. **Perform Self-Tests** - The module provides a power-up self-tests service that is automatically executed when the module is loaded into memory. See Self-Tests.
5. **Zeroizing Cryptographic Materal -** see Cryptographic Key Management.

6. **System Integrity During Hibernation (Random Number Generation)** to generate keys for system integrity during OS hibernation.

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

Table 4

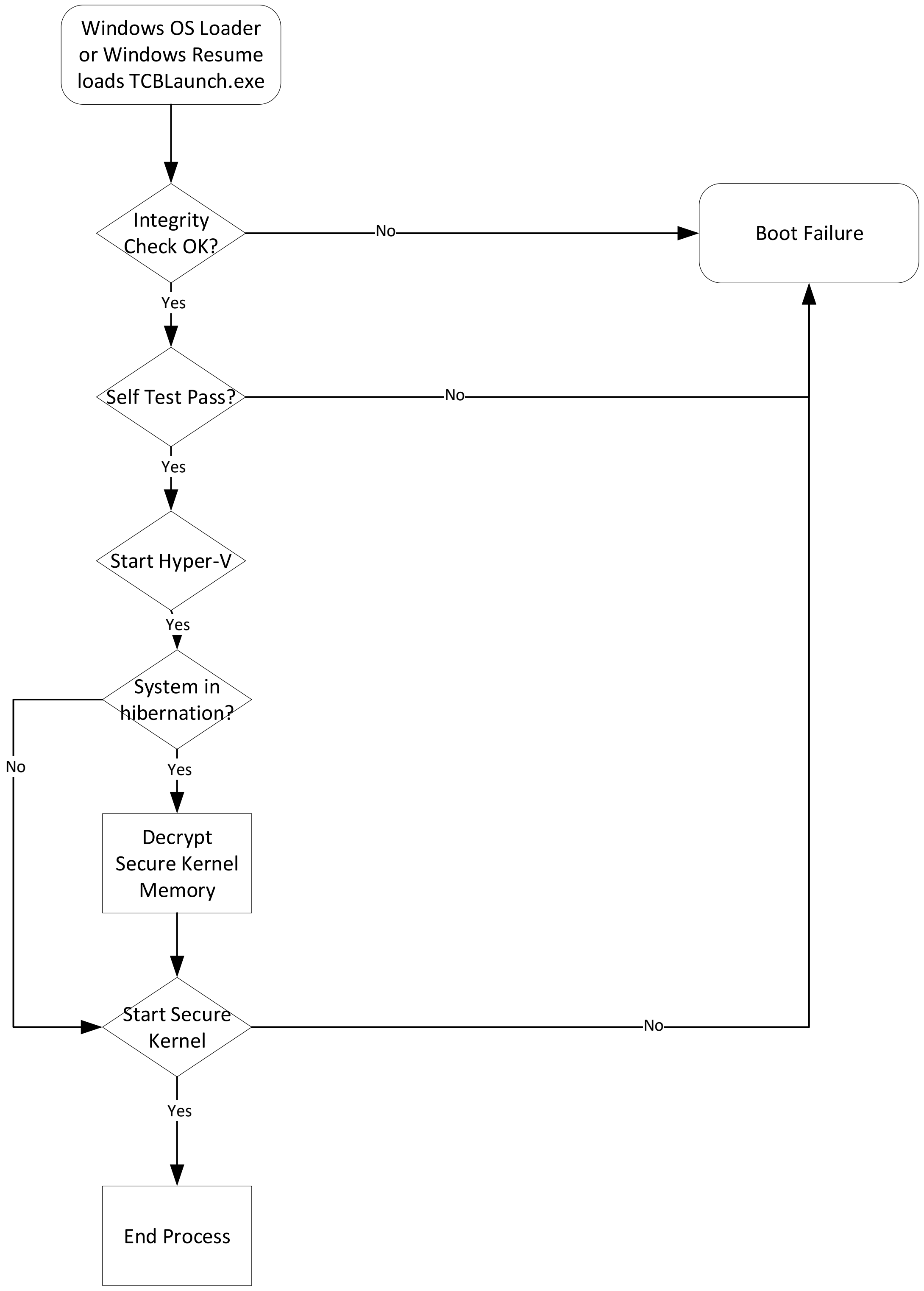
Service	Algorithms	CSPs	Invocation
Loading the Hypervisor	FIPS 186-4 RSA PKCS#1 (v1.5) verify with public key FIPS 180-4 SHS: SHA-256 hash SHA-384 hash SHA-512 hash AES GCM 256 bits	Secure Memory Encryption Key (to decrypt the memory of the Hypervisor in the hibernation scenario) RSA public key to verify the integrity of components mentioned in Integrity Chain of Trust	This service is fully automatic.
Loading the Secure Kernel	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 GCM 256 bits	Secure Memory Encryption Key (to decrypt the memory of the Secure Kernel in the hibernation scenario) RSA public key to verify the integrity of components mentioned in Integrity Chain of Trust	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 FIPS 180-4 SHS: SHA-1 KAT SHA-256 KAT SHA-512 KAT AES GCM KAT	None	This service is fully automatic.
Zeroizing Cryptographic Material	None	All	See Zeroization.

Service	Algorithms	CSPs	Invocation
System Integrity During Hibernation (Random Number Generation)	AES-256 CTR DRBG NDRNG (allowed, used to provide entropy to DRBG)	AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key	This service is fully automatic.

5 Finite State Model

5.1 Specification

The following diagram shows the finite state model for TCB Launcher:

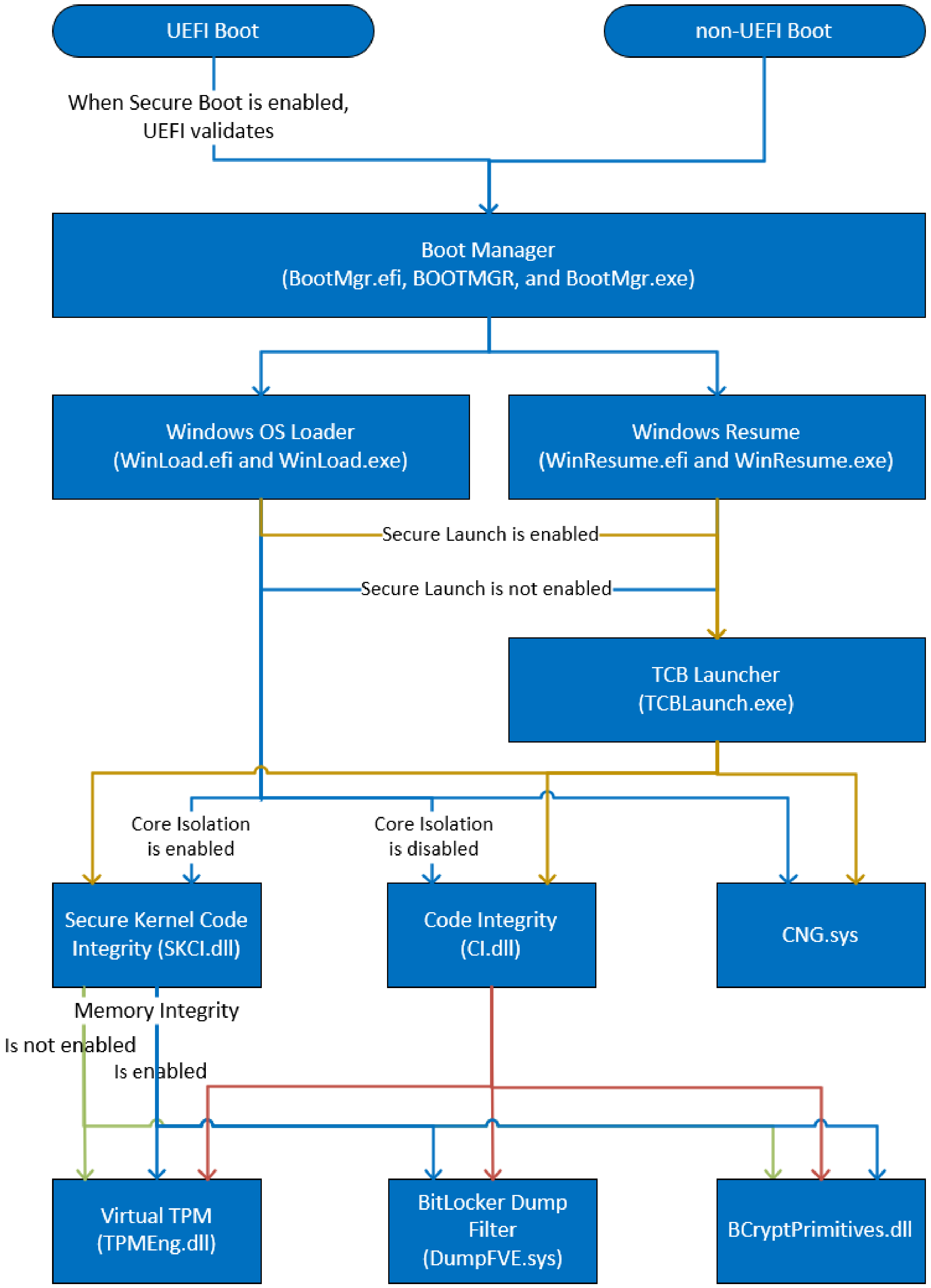


6 Operational Environment

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

6.1 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:


UEFI Boot
non-UEFI Boot
When Secure Boot is enabled, UEFI validates
Boot Manager (BootMgr.efi, BOOTMGR, and BootMgr.exe)
Windows OS Loader (WinLoad.efi and WinLoad.exe)
Windows Resume (WinResume.efi and WinResume.exe)
Secure Launch is enabled
Secure Launch is not enabled
TCB Launcher (TCBLaunch.exe)
Core Isolation is enabled
Core Isolation is disabled
Secure Kernel Code Integrity (SKCI.dll)
Code Integrity (CI.dll)
CNG.sys
Memory Integrity
Is not enabled
Is enabled
Virtual TPM (TPMEng.dll)
BitLocker Dump Filter (DumpFVE.sys)
BCryptPrimitives.dll

Windows OS Loader and Windows Resume check the integrity of the TCBLAUNCH.EXE component of the TCB Launcher Module before it is loaded.

Windows binaries include a SHA-256 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 TCB Launcher Module verifies the integrity of the cryptographic modules Secure Kernel Code Integrity and CNG.sys. It also verifies the integrity of Hypervisor and Secure Kernel binaries that are not cryptographic modules.

7 Cryptographic Key Management

7.1 Critical Security Parameters

When resuming from a hibernation scenario the TCBLAUNCH.EXE component of the TCB Launcher Module uses these critical security parameter (CSP):

- Secure Memory Encryption Key - 256-bit AES key generated by the DRBG that is used to decrypt memory of the Hypervisor and Secure Kernel that has been saved to the disk.
- AES-CTR DRBG Entropy Input – A secret value that is at least 256 bits and maintained internal to the module that provides the entropy material for AES-CTR DRBG output
- AES-CTR DRBG Seed – A 384 bit secret value maintained internal to the module that provides the seed material for AES-CTR DRBG output
- AES-CTR DRBG V – A 128 bit secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output
- AES-CTR DRBG Key – A 256 bit secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output

The Secure Memory Encryption Key is unsealed by the TPM and used by the TCBLAUNCH.EXE component of the TCB Launcher Module to decrypt memory of the Hypervisor and Secure Kernel.

TCB Launcher Module also uses as a CSP the public key component of the Microsoft code signing certificate as described in Integrity Chain of Trust.

7.2 Zeroization

7.2.1 Volatile Keys

The Secure Memory Encryption Key and DRBG CSPs are zeroized after use when the module component TCBLAUNCH.EXE is unloaded from memory.

7.2.2 Persistent Keys

TCB Launcher will use the TPM to seal keys generated by the RBG; if the computer does not have a TPM, the TCB Launcher will not persist any keys.

To zeroize a key that has been sealed by the TPM the operator should reformat the computer's storage volume overwrite preexisting data.

7.3 Access Control Policy

The TCB Launcher Module does not allow access to the cryptographic keys contained within it, so an access control table is not necessary. TCB Launcher Module receives keys from outside and then manages them appropriately once received. TCB Launcher Module prevents access to its keys by zeroizing them after use.

8 Self-Tests

The TCBLAUNCH.EXE component of the TCB Launcher Module 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-256 decrypt GCM Known Answer Tests
- SP 800-90A AES-256 counter mode DRBG Known Answer Tests (instantiate, generate and reseed)

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.

8.1 Conditional Self-Tests

TCB Launcher performs the following conditional self-tests:

- A Continuous Random Number Generator Test (CRNGT) is performed for SP 800-90A AES-256 CTR DRBG.
- A CRNGT is performed for the non-approved NDRNG per FIPS 140-2 IG 9.8.

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: https://www.microsoft.com/en-us/howtotell/default.aspx

The installed version of Windows 10 must be checked 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 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 Appendix A 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:

Table 5

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 used with AES NI

11 Security Levels

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

Table 6

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

TCB Launcher is a multi-chip standalone software-hybrid module whose host platforms meet the level 1 physical security requirements. The module consists of production-grade 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.

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

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.