Security Policy for FIPS 140-2 Validation

Boot Manager in
Microsoft Windows 8.1 Enterprise
Windows Server 2012 R2
Windows Storage Server 2012 R2
Surface Pro 3
Surface Pro 2
Surface Pro
Surface 2
Surface
Windows RT 8.1
Windows Phone 8.1
Windows Embedded 8.1 Industry Enterprise
StorSimple 8000 Series
Azure StorSimple Virtual Array Windows Server 2012 R2

DOCUMENT INFORMATION

Version Number	2.1
Updated On	April 12, 2017

Change History

Date	Version	Updated By	Change
29 OCT 2013	0.1	Tim Myers	First Draft
7 MAY 2014	0.2	Tim Myers	Second Draft; FIPS 186-4 and FIPS 180-4 updates; processor name updates
1 JUL 2014	0.3	Tim Myers	Third Draft; added Security Levels Table and cryptographic algorithm certificate numbers
11 JUL 2014	1.0	Tim Myers	First Release to Validators; includes all algorithm and module certificate numbers
18 JUL 2014	1.1	Tim Myers	Update platforms; consolidate services
12 DEC 2014	1.2	Tim Myers	Address CMVP comments; update platform names
17 DEC 2014	1.3	Tim Myers	Update IG G.5 platforms; update binary versions; add StorSimple
21 JAN 2015	1.4	Tim Myers	Address CMVP comments
29 JAN 2015	1.5	Tim Myers	Update Design Assurance
20 FEB 2015	1.6	Tim Myers	Add StorSimple to Validated Platforms; correction to algorithms list; update for PBKDF2
9 MAR 2015	1.7	Tim Myers	Add note to section 2.2; prepare for publication
17 MAR 2015	1.8	Tim Myers	Reorder StorSimple 8100 OE description
1 APR 2015	2.0	Tim Myers	Add Microsoft Surface Pro 3 to Validated Platforms
12 APR 2017	2.1	FIPS Team	Add StorSimple Virtual Array to Validated Platforms

TABLE OF CONTENTS

1 Introduction

The Windows Boot Manager is the system boot manager, called by the bootstrapping code that resides in the boot sector. Boot Manager is responsible for capturing the credentials required to unlock the OS volume and for loading and verifying the integrity of the Windows OS Loader, Winload.exe, and Windows OS Resume, winresume.exe. The Boot Manager consists of these binary executables and the Certificate Directory containing the root public key certificate issued by Microsoft.

> Note: credentials only pertain to unlocking the OS volumes. Credentials are not used for authentication to control access to cryptographic module ports and services.

In Microsoft Windows 8.1 Enterprise, Windows Server 2012 R2, Windows Storage Server 2012 R2, Surface Pro 3, Surface Pro 2, Surface Pro, Surface 2, Surface, Windows RT 8.1, Windows Phone 8.1, Windows Embedded 8.1 Industry Enterprise, StorSimple 8000 Series, and Azure StorSimple Virtual Array Windows Server 2012 R2 (herein referred to as Windows 8.1 OEs), the credentials supported for unlocking the OS volumes are:

- A startup key (stored on a USB flash drive; also known as an External key)
- A recovery key (stored on a USB flash drive; also known as an External key)
- An alpha-numeric PIN for Trusted Platform Module (TPM)+PIN or TPM+PIN+ Startup Key (SK) scenarios
- A password
- A key provided over a network by a trusted server

1.1 List of Cryptographic Module Binary Executables

Boot Manager crypto module binaries include:

- BOOTMGR
- bootmgr.exe
- bootmgfw.efi
- bootmgr.efi

The version numbers are:

> Versions 6.3.9600 and 6.3.9600.17031 for Windows 8.1 OEs

1.2 Brief Module Description

Both the older PC/AT BIOS and the newer Unified Extensible Firmware Interface (UEFI) boot methods are supported.

BOOTMGR and bootmgr.exe are the binary executables for booting PC/AT BIOS systems. BOOTMGR logically encapsulates bootmgr.exe.

Bootmgfw.efi and bootmgr.efi are the binary executables for booting Unified Extensible Firmware Interface (UEFI) systems. Bootmgfw.efi logically encapsulates bootmgr.efi.

1.3 Validated Platforms

The Boot Manager components listed in Section 1.1 were validated using the following machine configurations:

1. Microsoft Windows 8.1 Enterprise (x86) running on a Dell PowerEdge SC440 without AES-NI;
2. Microsoft Windows Embedded 8.1 Industry Enterprise (x86) running on a Dell PowerEdge SC440 without AES-NI;
3. Microsoft Windows 8.1 Enterprise (x86) running on a Dell Dimension E521 without AES-NI;
4. Microsoft Windows Embedded 8.1 Industry Enterprise (x86) running on a Dell Dimension E521 without AES-NI;
5. Microsoft Windows 8.1 Enterprise (x86) running on an Intel Core i7 with AES-NI running on an Intel Maho Bay;
6. Microsoft Windows Embedded 8.1 Industry Enterprise (x86) running on an Intel Core i7 with AES-NI running on an Intel Maho Bay;
7. Microsoft Windows 8.1 Enterprise (x86) running on an HP Compaq Pro 6305 with AES-NI;
8. Microsoft Windows Embedded 8.1 Industry Enterprise (x86) running on an HP Compaq Pro 6305 with AES-NI;
9. Microsoft Windows 8.1 Enterprise (x64) running on a Dell PowerEdge SC440 without AES-NI;
10. Microsoft Windows Embedded 8.1 Industry Enterprise (x64) running on a Dell PowerEdge SC440 without AES-NI;
11. Microsoft Windows Server 2012 R2 (x64) running on a Dell PowerEdge SC440 without AES-NI;
12. Microsoft Windows Storage Server 2012 R2 (x64) running on a Dell PowerEdge SC440 without AES-NI;
13. Microsoft Windows 8.1 Enterprise (x64) running on a Dell Dimension E521 without AES-NI;
14. Microsoft Windows Embedded 8.1 Industry Enterprise (x64) running on a Dell Dimension E521 without AES-NI;
15. Microsoft Windows Server 2012 R2 (x64) running on a Dell Dimension E521 without AES-NI;
16. Microsoft Windows Storage Server 2012 R2 (x64) running on a Dell Dimension E521 without AES-NI;
17. Microsoft Windows 8.1 Enterprise (x64) running on an Intel Core i7 with AES-NI running on an Intel Maho Bay;
18. Microsoft Windows Embedded 8.1 Industry Enterprise (x64) running on an Intel Core i7 with AES-NI running on an Intel Maho Bay;
19. Microsoft Windows Server 2012 R2 (x64) running on an Intel Core i7 with AES-NI running on an Intel Maho Bay;
20. Microsoft Windows Storage Server 2012 R2 (x64) running on an Intel Core i7 with AES-NI running on an Intel Maho Bay;
21. Microsoft Windows 8.1 Pro (x64) running on an Intel x64 Processor with AES-NI running on a Microsoft Surface Pro;
22. Microsoft Windows 8.1 Pro (x64) running on an Intel i5 with AES-NI running on a Microsoft Surface Pro 2;
23. Microsoft Windows 8.1 Enterprise (x64) running on an HP Compaq Pro 6305 with AES-NI;
24. Microsoft Windows Embedded 8.1 Industry Enterprise (x64) running on an HP Compaq Pro 6305 with AES-NI;
25. Microsoft Windows Server 2012 R2 (x64) running on an HP Compaq Pro 6305 with AES-NI;
26. Microsoft Windows Storage Server 2012 R2 (x64) running on an HP Compaq Pro 6305 with AES-NI;

27. Microsoft Windows RT 8.1 (ARMv7 Thumb-2) running on an NVIDIA Tegra 3 Tablet;
28. Microsoft Windows RT 8.1 (ARMv7 Thumb-2) running on a Microsoft Surface RT;
29. Microsoft Windows RT 8.1 (ARMv7 Thumb-2) running on a Microsoft Surface 2;
30. Microsoft Windows RT 8.1 (ARMv7 Thumb-2) running on a Qualcomm Tablet;
31. Microsoft Windows Phone 8.1 (ARMv7 Thumb-2) running on a Qualcomm Snapdragon S4 running on a Windows Phone 8.1;
32. Microsoft Windows Phone 8.1 (ARMv7 Thumb-2) running on a Qualcomm Snapdragon 400 running on a Windows Phone 8.1;
33. Microsoft Windows Phone 8.1 (ARMv7 Thumb-2) running on a Qualcomm Snapdragon 800 running on a Windows Phone 8.1;
34. Microsoft Windows 8.1 Enterprise (x64) running on a Dell Inspiron 660s without AES-NI and with PCLMULQDQ and SSSE 3;
35. Microsoft Windows Embedded 8.1 Industry Enterprise (x64) running on a Dell Inspiron 660s without AES-NI and with PCLMULQDQ and SSSE 3;
36. Microsoft Windows Server 2012 R2 (x64) running on a Dell Inspiron 660s without AES-NI and with PCLMULQDQ and SSSE 3;
37. Microsoft Windows Storage Server 2012 R2 (x64) running on a Dell Inspiron 660s without AES-NI and with PCLMULQDQ and SSSE 3;
38. Microsoft Windows 8.1 Enterprise (x64) running on an Intel Core i5 with AES-NI and with PCLMULQDQ and SSSE 3 running on a Microsoft Surface Pro 2;
39. Microsoft Windows Embedded 8.1 Industry Enterprise (x64) running on an Intel Core i7 with AES-NI and with PCLMULQDQ and SSSE 3 running on an Intel Maho Bay;
40. Microsoft Windows Server 2012 R2 (x64) running on an Intel Core i7 with AES-NI and with PCLMULQDQ and SSSE 3 running on an Intel Maho Bay;
41. Microsoft Windows Storage Server 2012 R2 (x64) running on an Intel Core i7 with AES-NI and with PCLMULQDQ and SSSE 3 running on an Intel Maho Bay;
42. Microsoft Windows 8.1 Enterprise (x64) running on an HP Compaq Pro 6305 with AES-NI and with PCLMULQDQ and SSSE 3;
43. Microsoft Windows Embedded 8.1 Industry Enterprise (x64) running on an HP Compaq Pro 6305 with AES-NI and with PCLMULQDQ and SSSE 3;
44. Microsoft Windows Server 2012 R2 (x64) running on an HP Compaq Pro 6305 with AES-NI and with PCLMULQDQ and SSSE 3;
45. Microsoft Windows Storage Server 2012 R2 (x64) running on an HP Compaq Pro 6305 with AES-NI and with PCLMULQDQ and SSSE 3;
46. Windows Server 2012 R2 (x64) running on a Microsoft StorSimple 8100 with an Intel Xeon E5-2648L without AES-NI;
47. Windows Server 2012 R2 (x64) running on a Microsoft StorSimple 8100 with an Intel Xeon E5-2648L with AES-NI;
48. Microsoft Windows 8.1 Pro (x64) running on an Intel Core i7 with AES-NI and with PCLMULQDQ and SSSE 3 running on a Microsoft Surface Pro 3;
49. Azure StorSimple Virtual Array Windows Server 2012 R2 on Hyper-V 6.3 on Windows Server 2012 R2 (x64) running on a Dell Precision Tower 5810 with PAA;
50. Azure StorSimple Virtual Array Windows Server 2012 R2 on VMware Workstation 12.5 on Windows Server 2012 R2 (x64) running on a Dell XPS 8700 with PAA

Boot Manager maintains FIPS 140-2 validation compliance (according to FIPS 140-2 PUB Implementation Guidance G.5) on the following platforms:

x86 Microsoft Windows 8.1
x86 Microsoft Windows 8.1 Pro

x64 Microsoft Windows 8.1
x64 Microsoft Windows Server 2012 R2 Datacenter

x64-AES-NI Microsoft Windows 8.1
x64-AES-NI Microsoft Windows Server 2012 R2 Datacenter

1.4 Cryptographic Boundary

The software cryptographic boundary for Boot Manager is defined as the binaries BOOTMGR, bootmgr.exe, bootmgfw.efi, and bootmgr.efi. The physical configuration of Boot Manager, as defined in FIPS-140-2, is multi-chip standalone.

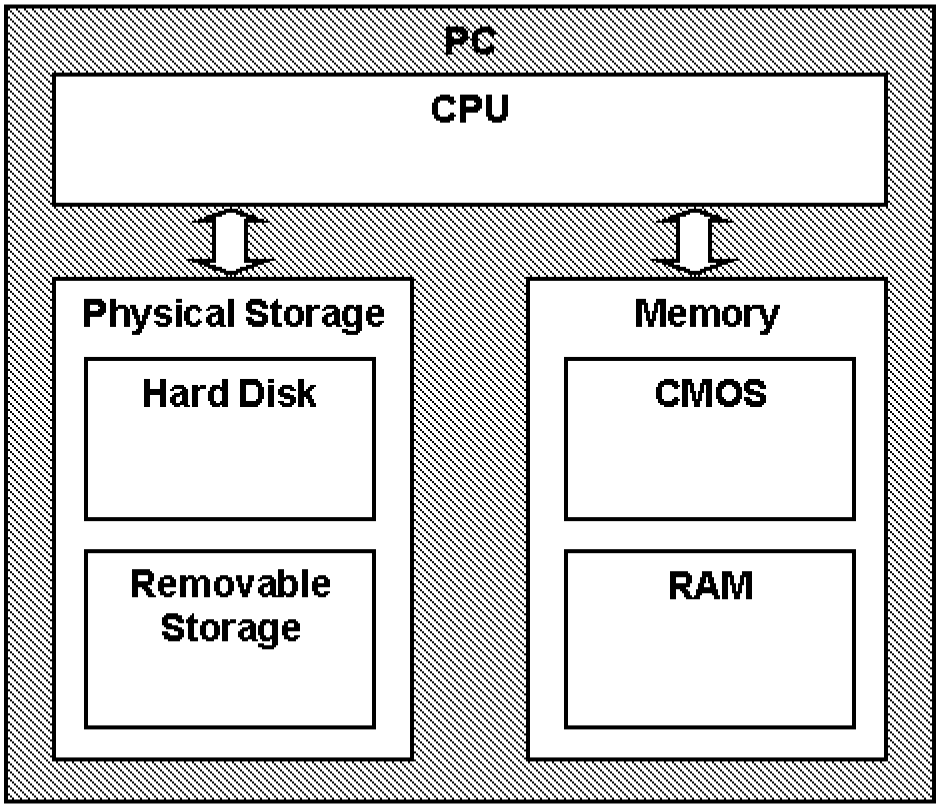
2 Security Policy

Boot Manager operates under several rules that encapsulate its security policy.

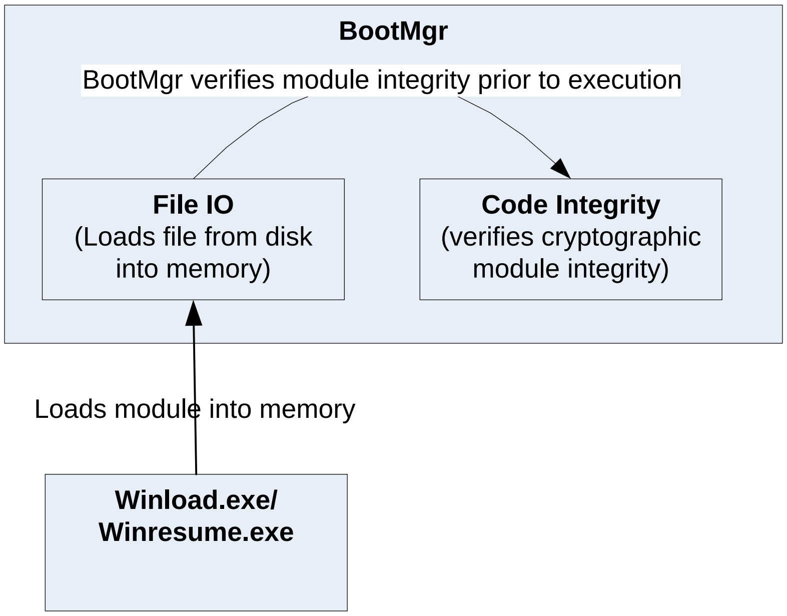
- Boot Manager is validated on the platforms listed in Section 1.3.
- Windows 8.1 OEs are operating systems supporting a “single user” mode where there is only one interactive user during a logon session.
- Boot Manager is only in its Approved mode of operation when Windows is booted normally, meaning 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.

The following diagram illustrates the master components of the Boot Manager module:

Boot Manager



The following diagram illustrates Boot Manager module interaction with the cryptographic module:


BootMgr
BootMgr verifies module integrity prior to execution
File IO (Loads file from disk into memory)
Code Integrity (verifies cryptographic module integrity)
Loads module into memory
Winload.exe/ Winresume.exe

- Boot Manager loads the Windows 8.1 OEs operating system loader (Winload.exe) or winresume.exe, after it determines component's integrity using its cryptographic algorithm implementations using the FIPS 140-2 approved algorithms mentioned below. After the verified binary image file is loaded, Boot Manager passes the execution control to it and no longer executes until the next reboot. The Crypto officer and User have access to the services Boot Manager supports.
- If the integrity of components being loaded is not verified, Boot Manager does not transfer the execution from itself.
- Boot Manager has a service for the encryption and decryption functionality used with BitLocker® Drive Encryption[1] operations related to bootstrapping the Windows 8.1 OEs operating system.
- The module provides a power-up self-tests service that is automatically executed when the module is loaded into memory, as well as, a show status service, that is automatically executed by the module to provide the status response of the module either via output to the general purpose computer (GPC) monitor or to log files.
- Boot Manager implements a self-integrity check using an RSA digital signature during its initialization process. Boot Manager will not complete its initialization if the signature is invalid.

2.1 FIPS 140-2 Approved Algorithms

- Boot Manager implements the following FIPS-140-2 Approved algorithms. Note that not all algorithms/modes verified through the CAVP certificates listed are implemented by the Boot Manager module.
  - 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. #1494)
  - FIPS 180-4 SHS SHA-1, SHA-256, SHA-384, and SHA-512 (Certs. #2373 and #2396)
  - FIPS 197 AES CBC 128 and 256; SP 800-38C AES CCM 256 (Cert. #2832)
  - FIPS PUB 198-1 HMAC-SHA-1[2] and HMAC-SHA-256 (Cert. #1773)
  - SP 800-132 PBKDF supporting HMAC-SHA-256 (vendor affirmed)

2.2 Non-Approved Algorithms

- Boot Manager implements the following non-approved algorithms:
  - MD5[3] – used for legacy certificate chain authentication
  - An algorithm for VMK key derivation
  - An algorithm for a password-based key derivation function (PBKDF)

Note: the non-approved algorithms for VMK key derivation and PBKDF cannot be used when the module is configured in the approved mode of operation because the keys derived from them cannot be used in the approved mode of operation.

2.3 Cryptographic Bypass

Cryptographic bypass is not supported by Boot Manager.

[1] BitLocker is a registered trademark for the full volume encryption functionality in Windows. BitLocker is not a separate binary executable.
[2] For HMAC, only key sizes that are >= 112 bits in length are used by the module in FIPS mode.
[3] MD5 is not allowed for usage in FIPS mode.

2.4 Machine Configurations

Boot Manager was tested using the machine configurations listed in Section 1.3 - Validated Platforms.

2.5 NIST SP 800-132 Password Based Key Derivation Function (PBKDF) Usage

Keys derived from passwords, as shown in SP 800-132, may only be used in storage applications. To run in a FIPS Approved manner, strong passwords must be used and they may only be used for storage applications. The password/passphrase length is enforced by the caller of the PBKDF interfaces at the time the password/passphrase is created and not by this cryptographic module. (This module is not involved in the creation of any password/passphrase.)

The GPC'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.

3 Operational Environment

The operational environment for Boot Manager is the Windows 8.1 OEs running on the software and hardware configurations listed in Section 1.3 - Validated Platforms.

4 Integrity Chain of Trust

Boot Manager is the very 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 (Winload.exe) or Windows OS Resume (Winresume.exe) before starting it.

5 Ports and Interfaces

5.1 Control Input Interface

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 GPC'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.

5.2 Status Output Interface

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

5.3 Data Output Interface

The Data Output Interface includes two different kinds of functions: initialization and transfer.

The initialization function ImgpInitializeBootApplicationParameters is providing output in the form of input parameters for the boot application. This function is called before transferring execution to the boot application.

The following functions are transfer functions: Archx86TransferTo32BitApplicationAsm, Archx86TransferTo64BitApplicationAsm, and Archpx64TransferTo64BitApplicationAsm. These functions are responsible for transferring the execution from Boot Manager to the initial execution point of the Windows OS Loader or Windows OS Resume. Data exits the module in the form of the initial instruction address of Winload.exe or Winresume.exe.

5.4 Data Input Interface

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 GPC's USB port also forms a part of the Data Input interface. This interface is used to enter the Startup key or Recovery Key used by the BitLocker® Drive Encryption in Windows 8.1 OEs. The GPC's keyboard can also serve as a Data Input Interface when the method to protect the Volume Master Key (VMK) value relies on an operator supplied PIN.

6 Specification of Roles

Boot Manager supports both User and Cryptographic Officer roles (as defined in FIPS-140-2). Both roles have access to all services implemented in Boot Manager. Therefore, roles are assumed implicitly by booting the Windows 8.1 OEs operating system.

Services available to the Cryptographic Officer role:

- Configure BitLocker into FIPS mode
- Unlock the operating system volume
- Boot the Windows operating system

Services available to the User role:

- Unlock the operating system volume
- Boot the Windows operating system

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

Boot Manager services are:

1. Configuration of BitLocker into FIPS mode
2. The encryption and decryption functionality used with BitLocker Drive Encryption for file I/O that supports the bootstrapping of the Windows 8.1 OEs operating system
3. Unlocking the operating system volume
4. Loading and verifying the integrity of the Windows 8.1 OEs operating system loader (winload.exe or winresume.exe)
5. Booting the Windows operating system
6. Zeroization (see Section 8 Cryptographic Key Management)
7. Legacy certificate chain authentication (non-FIPS Approved service)

The User and Cryptographic Officer roles have the same use of the encryption, decryption, unlocking, OS loading, and booting services. Boot Manager does not export any cryptographic functions that can be called or externally invoked.

Boot Manager has a service of unlocking the operating system volume with methods that are Approved and non-Approved.

Always Approved

- Clear Key
- External Key (TPM-less startup key scenario)
- External Key (i.e. USB startup key or USB recovery key)
- TPM
- Recovery password

Configurable to be Approved

- TPM + USB (with system configured to use Approved SP 800-132 PBKDF)

Non-Approved

- TPM + Network unlock
- TPM + PIN
- TPM + PIN + USB

The Non-Approved methods of the service of unlocking the operating system volume are equivalent to storing the VMK in plaintext.

Boot Manager also has a Non-Approved service of legacy certificate chain authentication using the MD5 algorithm. See Section 2.2 non-Approved Algorithms.

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

Service	Algorithms	CSPs	Invocation
Configuration of BitLocker into FIPS mode	None	None	See section 2.
The encryption and decryption functionality used with BitLocker Drive Encryption for file I/O that supports the bootstrapping of the Windows 8.1 OEs operating system	AES CBC (128 and 256 bit)	FVEK	This service is fully automatic. The User / Cryptographic Officer do not take any actions to start this service.
Unlocking the operating system volume	HMAC-SHA-256 SP 800-132 PBKDF AES in CCM mode (128 and 256 bit) VMK key derivation (KDF, non-FIPS Approved algorithm) Legacy password-based key derivation (PBKDF, non-FIPS Approved algorithm)	PIN, Password, DK, ExK, CC, IK, SK, NK, VMK	See Flow Logic diagram for the actions that the User / Crypto Officer must take to reach the "UNLOCK & BOOT" stage. This service is executed automatically once this stage has been reached.
Loading and verifying the integrity of the Windows 8.1 OEs operating system loader (winload.exe or winresume.exe)	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.
Booting the Windows operating system	None	None	The User / Crypto Officer must power up the device to start this service.
Zeroization	None	All CSPs	See section 8.2.
Legacy certificate chain authentication (non-FIPS Approved service)	MD5 (non-FIPS Approved algorithm)	None	This service is fully automatic. The User / Cryptographic Officer do 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 described in Section 5 Ports and Interfaces.

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 Cryptographic Key Management

Note: authentication in the FIPS 140-2 standard pertains exclusively to controlling access to cryptographic module ports and services. That particular use of the word “authentication” is different than how it is used in a broader information security sense. The keys described here are credentials used to unlock Windows OS volumes.

8.1 Critical Security Parameters

Boot Manager does not store any secret or private cryptographic keys across power-cycles. However, it does use certain AES keys in support of the BitLocker feature in FIPS mode. Boot Manager uses the following critical security parameters (CSPs), which include cryptographic keys, in FIPS mode:

Critical Security Parameters	CSP / Key Description
External Key (ExK) or Clear Key (CC)	256-bit AES key stored outside the cryptographic boundary (for example a USB device). This key enters the module via the USB port. It is the only method used to decrypt the VMK for which the VMK could be considered as having been encrypted to start with because other methods rely upon non-FIPS Approved key derivation methods. Logically, the external key represents either a startup key or a recovery key. Key used for AES decryption of the VMK.
Intermediate Key (IK)	256-bit AES key value that is stored encrypted and forms the basis of another AES key, such as the NK or VMK, by combining with another 256-bit AES key via key derivation or XOR. When a non-Approved KDF or XOR is used to derive this key, the VMK is considered to be stored in plaintext.
Session Key (SK)	256-bit AES key value that is stored in plaintext on disk and used to decrypt an IK transported over a trusted network to Boot Manager during Network Unlock authentication[4]. Boot Manager does the actual decryption of the IK using AES-CCM.

[4] Network Unlock authentication concerns BitLocker functionality. It is not referring to FIPS 140-2 standard authentication used to control access to the ports and services of the cryptographic module.

Network Key (NK)	256-bit 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 delivered over a trusted network. A KDF is not used. Because an XOR is used to derive this key, the VMK is considered to be stored in plaintext.
Volume Master Key (VMK)	256-bit AES key used for AES-CCM decryption of the FVEK
Full Volume Encryption Key (FVEK)	128 or 256-bit AES key used for AES encryption/decryption of data on disk sectors This key is stored persistently. It is encrypted by the VMK using AES-CCM.
Derived Key (DK)	256-bit AES key value used for AES decryption of the VMK. The value is not stored long-term and is derived using a method defined by the system configuration. Derived Keys are used in Password and Recovery Password authentication.
Microsoft Root Certificate Authority (CA) Public Key	Key used for RSA PKCS#1 (v1.5) verification of digital signatures
PIN	An alpha-numeric PIN for Trusted Platform Module (TPM) + PIN or TPM + PIN + USB scenarios
Password	A password

The VMK is always stored in encrypted form for volumes encrypted in FIPS mode. Based on system configuration, the method used to perform the encryption may use an Approved password-based key derivation method (SP 800-132 PBKDF). (Volumes created by systems *not* in FIPS mode may have performed the VMK encryption with an older, non-Approved key derivation method, which cannot be used while operating in FIPS mode.) The VMK may alternately or additionally be stored encrypted with the ExK/CC (identified above). The VMK value can be zeroized by following the guidance for zeroization of persistent keys in section 8.2.2 below.

Note that the FVEK is stored in encrypted form across power cycles, and thus is not subject to the zeroization requirements, but can be zeroized in the same manner as the VMK. The ExK/CC, is stored only in memory and is zeroized by following the guidance for zeroization of ephemeral keys in section 8.2.1 below.

Boot Manager also uses the Microsoft root CA public key certificate stored on the computer hard disk to verify digital signatures using its implementation of RSA PKCS#1 (v1.5) verify. This public key is available to both roles. Zeroization is performed by following the guidance for zeroization of persistent keys in section 8.2.2 below.

Boot Manager will seek appropriate keys to decrypt the encrypted (i.e. “BitLocker® protected”) volume at boot time and waking up from hibernation, in the following sequence:

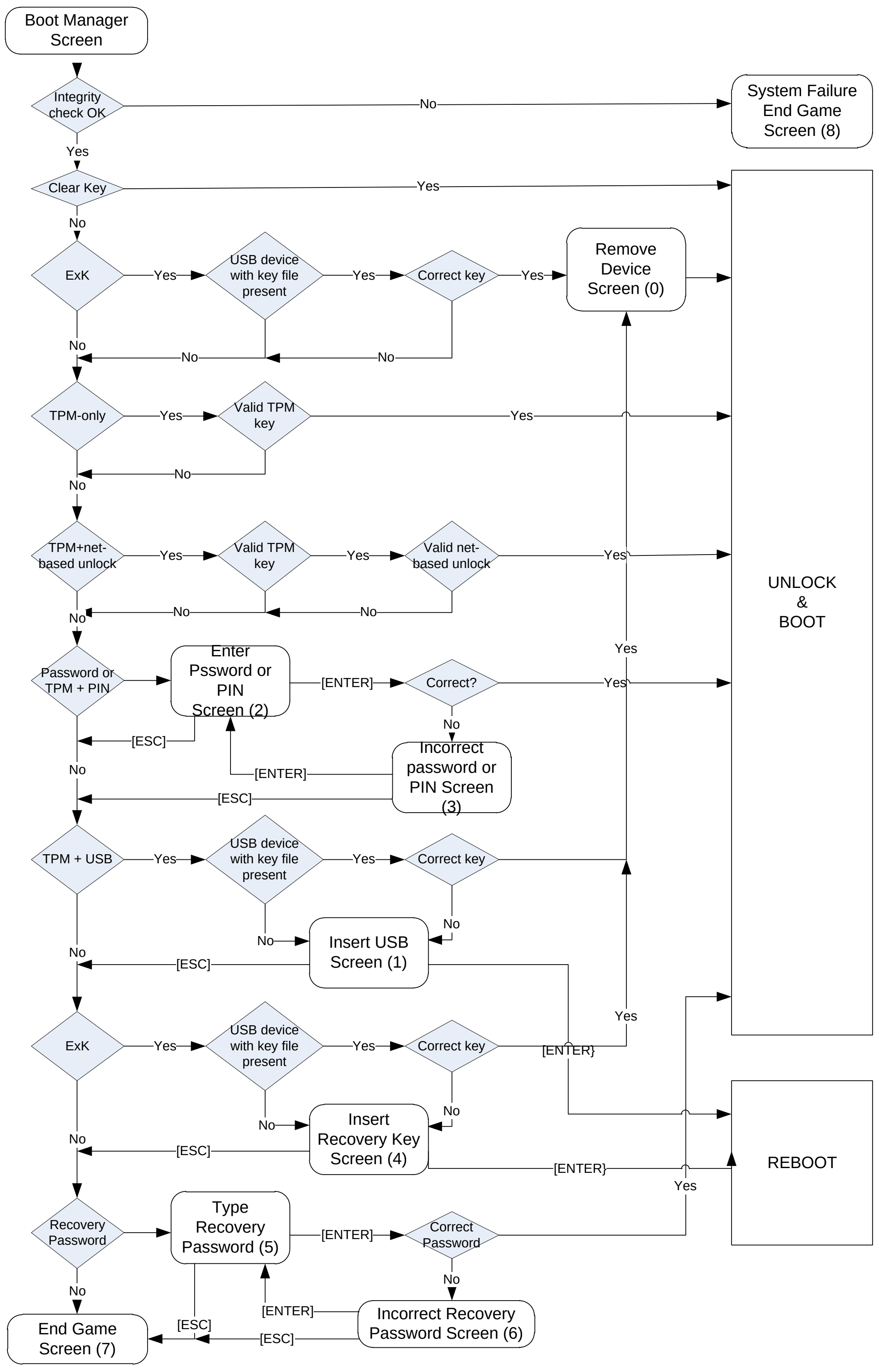
1. Clear Key
2. No-TPM required and no user input required

    a. ExK (TPM-less startup key scenario)

3. TPM system and no user input required
    a. TPM
    b. TPM + Network unlock
    c. TPM + USB
4. UI Required (TPM system or no-TPM system)
    a. TPM + PIN
    b. TPM + PIN + USB
    c. ExK (i.e. USB startup key or USB recovery key)
    d. Recovery password

The TPM + Network unlock, TPM + PIN, and TPM + PIN + USB scenarios use a method of key derivation that cannot be used while operating in FIPS mode. If these methods of unlocking are used, the VMK is considered to be stored in plaintext.

The following diagram illustrates the flow logic in the system.


Boot Manager Screen
Integrity check OK
No
System Failure End Game Screen (8)
Yes
Clear Key
Yes
No
ExK
Yes
USB device with key file present
Yes
Correct key
Yes
Remove Device Screen (0)
No
No
No
TPM-only
Yes
Valid TPM key
Yes
No
No
TPM+net-based unlock
Yes
Valid TPM key
Yes
Valid net-based unlock
Yes
No
No
No
UNLOCK & BOOT
Yes
Password or TPM + PIN
Enter Pssword or PIN Screen (2)
[ENTER]
Correct?
Yes
No
[ESC]
No
Incorrect password or PIN Screen (3)
[ENTER]
[ESC]
TPM + USB
Yes
USB device with key file present
Yes
Correct key
No
No
Insert USB Screen (1)
No
[ESC]
Yes
ExK
Yes
USB device with key file present
Yes
Correct key
[ENTER]
No
No
Insert Recovery Key Screen (4)
No
[ESC]
[ENTER]
REBOOT
Yes
Recovery Password
Type Recovery Password (5)
[ENTER]
Correct Password
No
No
[ENTER]
[ESC]
Incorrect Recovery Password Screen (6)
[ESC]
End Game Screen (7)

Flow Logic

Details about the keys and network protocol used for Network Unlock authentication are in the Network Key Protector Unlock Protocol Specification [MS-NKPU], which is available at:

https://msdn.microsoft.com/en-us/library/hh537327.aspx

8.2 Zeroization Procedures

8.2.1 Ephemeral Keys

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

8.2.2 Persistent Keys

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.

8.3 Access Control Policy

The Boot Manager crypto module does not allow access to the cryptographic keys contained within it. For this reason, an access control table is not included in this document. Boot Manager receives keys from outside and then manages them appropriately once received. Boot Manager prevents access to its keys by zeroizing them.

8.4 Access Control Policy

The Boot Manager crypto module does not allow access to the cryptographic keys contained within it. For this reason, an access control table is not included in this document. Boot Manager receives keys from outside and then manages them appropriately once received. Boot Manager prevents access to its keys by zeroizing them.

9 Authentication

Boot Manager does not implement any authentication services as defined by the FIPS 140-2 standard, which is concerned exclusively with controlling access to cryptographic module ports and services. The User and Cryptographic Officer roles are assumed implicitly by booting the Windows operating system.

10 Self-Tests

10.1 Power-On Self-Tests

Boot Manager performs the following power-on (startup) self-tests.

- SHA-1 Known Answer Test
- SHA-256 Known Answer Test

- SHA-512 Known Answer Test
- RSA PKCS#1 (v1.5) signature verification Known Answer Test
- Software Integrity Test – RSA PKCS#1 (v1.5) verify with public key (RSA 2048 w/ SHA-256)
- AES-CBC - Encrypt/Decrypt KATs
- AES-CCM - Encrypt/Decrypt KATs
- HMAC-SHA-1 Known Answer Test
- HMAC-SHA-256 Known Answer Test

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

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 8.1 OEs. The various methods of delivery and installation for each product are listed in the following table.

Product	Delivery and Installation Method
Windows 8.1	• DVD • Pre-installed on the computer by OEM • Download that updates Windows 8
Windows Server 2012 R2	• DVD • Pre-installed on the computer by OEM • Download that updates Windows Server 2012
Windows Storage Server 2012 R2	• Pre-installed by the OEM (Third party)
Surface Pro 3, Surface Pro 2, Surface Pro, Surface 2, Surface	• Pre-installed by the OEM (Microsoft)
Windows RT 8.1	• Pre-installed on the device by OEM • Download that updates Windows RT
Windows Phone 8.1	• Pre-installed on the device by OEM or mobile network operator • Download that updates Windows 8
Windows Embedded 8.1 Industry Enterprise	• Pre-installed by the OEM (Third party)
StorSimple 8000 Series	• Pre-installed by the OEM (Third party)
Azure StorSimple Virtual Array Windows Server 2012 R2	• Pre-installed by Azure

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

The installed version of Windows 8.1 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:

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:

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:
http://windows.microsoft.com
For more information about FIPS 140 evaluations of Microsoft products, please see:
http://technet.microsoft.com/en-us/library/cc750357.aspx

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

15.1 How to Verify Windows Versions

The installed version of Windows 8.1 OEs must be verified to match the version that was validated using one of the following methods:

1. The ver command
    a. From Start, open the Search charm.
    b. In the search field type "cmd" and press the Enter key.
    c. The command window will open with a "C:\>" prompt.
    d. At the prompt, type "ver" and press the Enter key.
    e. You should see the answer "`Microsoft Windows [Version 6.3.9600]`".
2. The systeminfo command
    a. From Start, open the Search charm.
    b. In the search field type "cmd" and press the Enter key.
    c. The command window will open with a "C:\>" prompt.
    d. At the prompt, type "systeminfo" and press the Enter key.
    e. Wait for the information to be loaded by the tool.
    f. Near the top of the output, you should see:

```
OS Name:                   Microsoft Windows 8.1 Enterprise
OS Version:                6.3.9600 N/A Build 9600
OS Manufacturer:           Microsoft Corporation
```

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: x.x.xxxx.xxxxx.
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.