Microsoft Corporation

Cryptographic Primitives Library

FIPS 140-3 Non-Proprietary Security Policy Document

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

Microsoft Windows Server 2022
(Standard and Datacenter Editions)

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

COPYRIGHT AND DISCLAIMER

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The example companies, organizations, products, people and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred.

Microsoft, Active Directory, Azure, Visual Basic, Visual Studio, Windows, the Windows logo, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

The Microsoft Windows Cryptographic Primitives Library (the “module”) is a multi-chip standalone software-hybrid cryptographic module that provides cryptographic services to user-mode applications running on the Windows operating system. The module binary, BCRYPTPRIMITIVES.DLL, encapsulates a set of cryptographic algorithms accessible via the Microsoft CNG API, which are exported by BCRYPT.DLL. BCRYPT.DLL is an API wrapper for BCRYPTPRIMITIVES.DLL and can be used by applications for general-purpose FIPS 140-3-compliant cryptography. The relationship between the Cryptographic Primitives Library and its related components is shown in the following diagram.

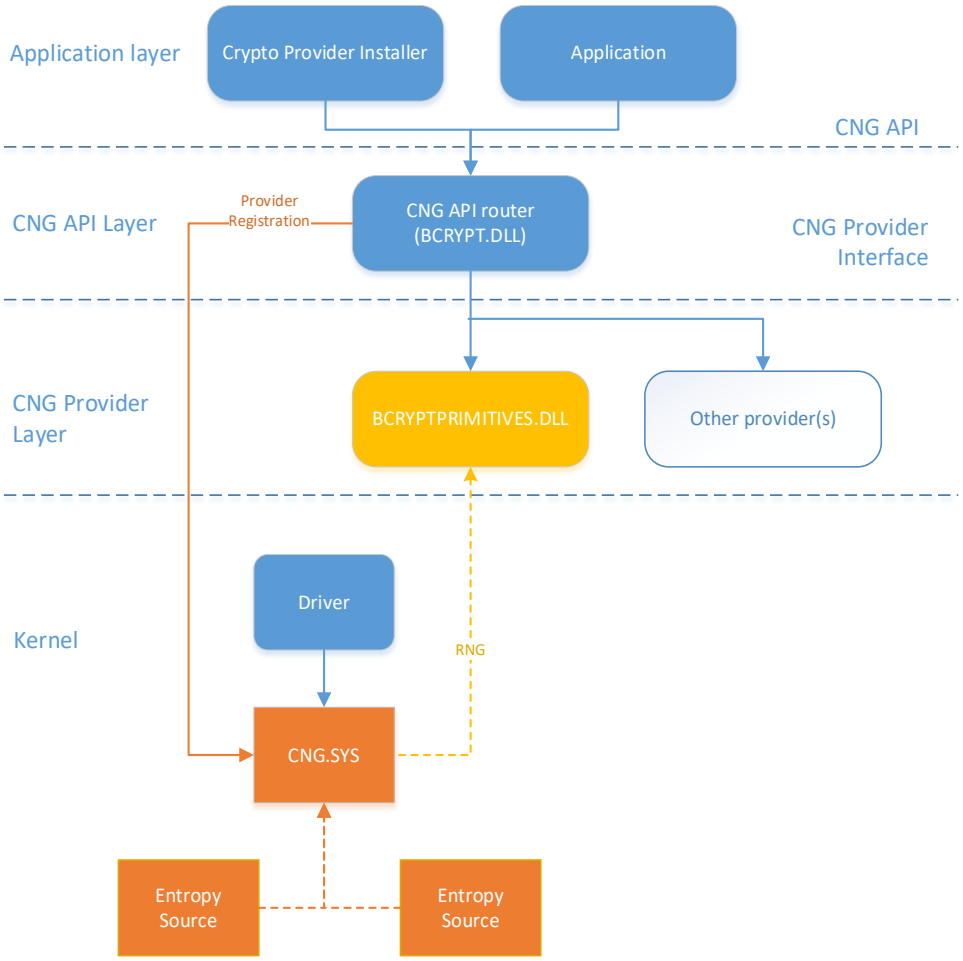


Figure 1: Diagram of the Module and Related Components

This FIPS 140-3 Security Policy contains a specification of the rules under which the module must operate and describes how the module meets the requirements specified in Federal Information Processing Standards Publication 140-3 (FIPS PUB 140-3) and International Standard ISO/IEC 19790:2012 (Information technology – Security techniques – Security requirements for cryptographic modules). This document is intended for the FIPS 140-3 testing lab, the Cryptographic Module Validation Program (CMVP), and administrators and users of the module.

1.2 Security Levels

The overall security rating for the module is level 1. The table below lists the security levels of individual clauses for this validation.

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

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Cryptographic Primitives Library is a cryptographic module that provides cryptographic services to user-mode applications running on Windows through the set of exported functions described in section 3 Cryptographic Module Interfaces. The module includes a set of algorithm providers for the Cryptography Next Generation (CNG) framework in Windows. Each provider represents a single cryptographic algorithm or a set of closely related cryptographic algorithms.

Module Type: Software-hybrid

Module Embodiment: MultiChipStand

Cryptographic Boundary:

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

Software Component	Description
BCRYPTPRIMITIVES.DLL	Binary file that contains the module.

Table 2: Module Software Components

Tested Operational Environment's Physical Perimeter (TOEPP):

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

The following block diagram illustrates the module's components, physical perimeter (TOEPP), and cryptographic boundary. The cryptographic boundary of the module is the module software component, BCRYPTPRIMITIVES.DLL. The BCRYPTPRIMITIVES.DLL binary is loaded from the OS volume in physical storage and executes in the computer memory. The control input, data input / output, and status output of the module exist within the computer memory as well.

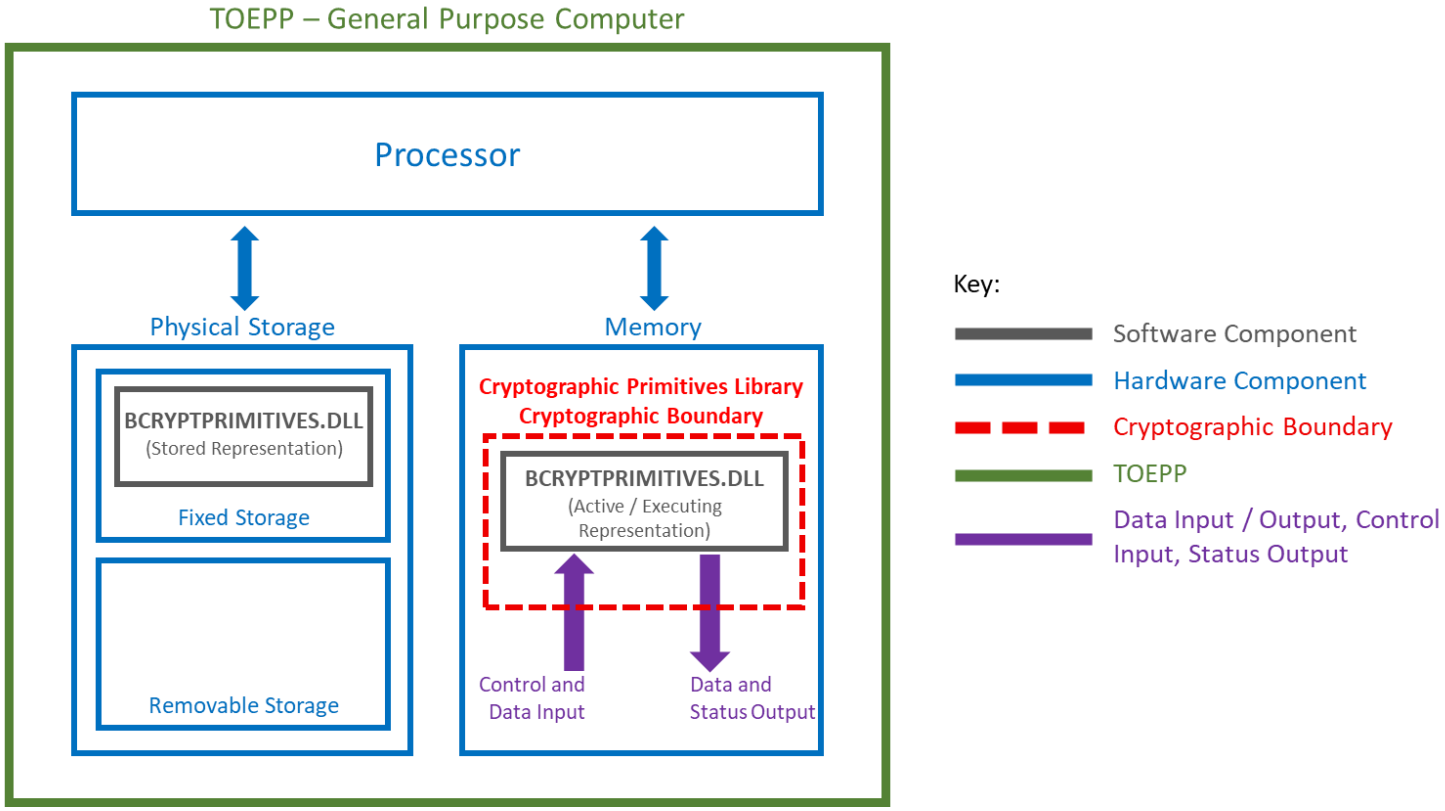
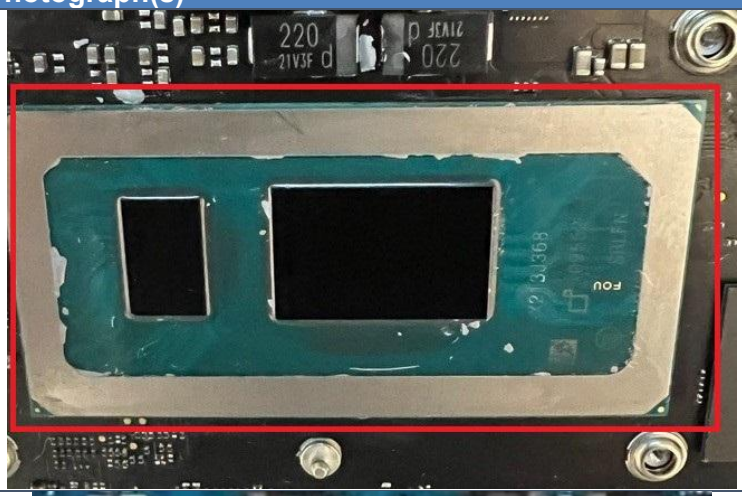
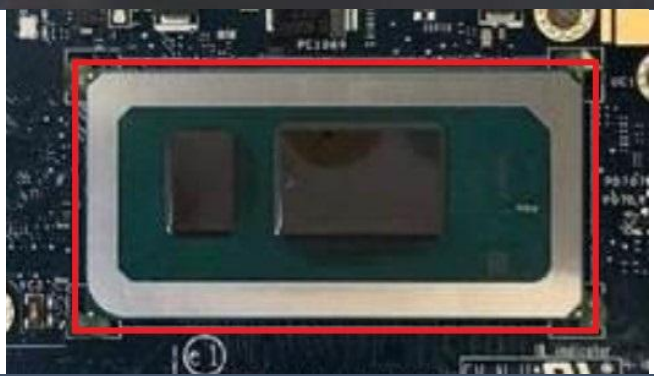


Figure 2: Module Boundary Diagram

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

CPU	Photograph(s)
12th Gen Intel Core i7-1265U (Microsoft Surface Laptop 5)	
11th Gen Intel Core i5-11500H (HP ZBook Power G8)	
11th Gen Intel Core i7-1185G7 (Dell Latitude 7420)	

CPU	Photograph(s)
Intel Xeon Gold 6130 (Dell PowerEdge R640)	

Table 3: CPU Photographs

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Windows Product	Build	Edition(s) in Scope
Windows 11 version 22H2	10.0.22621.30001	Enterprise Edition Home Edition IoT Enterprise Edition Pro Edition Education Edition
Windows Server 2022	10.0.20348.30000 (including the March 2023 updates)	Standard Edition Datacenter Edition

Table 4: Version Information

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
BCRYPTPRIMITIVES.DLL (Windows 11 version 22H2)	Windows 11 version 22H2, build 10.0.22621.30001	N/A	Yes
BCRYPTPRIMITIVES.DLL (Windows Server 2022)	Windows Server 2022, build 10.0.20348.30000	N/A	Yes

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

Tested Module Identification – Hybrid Disjoint Hardware:

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

Table 6: Tested Module Identification – Hybrid Disjoint Hardware

Tested Operational Environments - Software, Firmware, Hybrid:

The operational environment for the module is the Windows operating system running on a supported hardware platform, as listed in the table below. All hardware platforms in the table below are 64-bit Intel architecture. The tested operational environments provide Processor Algorithm Acceleration (PAA) in the form of the Advanced Encryption Standard New Instructions (AES-NI).

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Windows 11 version 22H2, Education Edition	Dell Latitude 7420	11th Gen Intel Core i7-1185G7	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.30001
Windows 11 version 22H2, Enterprise Edition	Microsoft Surface Laptop 5	12th Gen Intel Core i7-1265U	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.30001
Windows 11 version 22H2, Home Edition	Microsoft Surface Laptop 5	12th Gen Intel Core i7-1265U	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.30001
Windows 11 version 22H2, IoT Enterprise Edition	Microsoft Surface Laptop 5	12th Gen Intel Core i7-1265U	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.30001
Windows 11 version 22H2, Pro Edition	HP ZBook Power G8	11th Gen Intel Core i5-11500H	Yes	N/A	Windows 11 version 22H2, build 10.0.22621.30001
Windows Server 2022 Datacenter, including the March 2023 Updates	Dell PowerEdge R640	Intel Xeon Gold 6130	Yes	N/A	Windows Server 2022, build 10.0.20348.30000

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Windows Server 2022 Standard, including the March 2023 Updates	Dell PowerEdge R640	Intel Xeon Gold 6130	Yes	N/A	Windows Server 2022, build 10.0.20348.30000

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

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

Operating System	Hardware Platform
Any Microsoft operating system which is a relabeled version of the operating systems listed in section 2.2.	Any UEFI-based x64 computer

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

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

The following caveat applies when operating the module on any vendor-affirmed operational environment:

No assurance of the minimum strength of generated SSPs (e.g., keys).

2.3 Excluded Components

No components within the cryptographic boundary are claimed as excluded.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Normal operation of the computer, Windows OS, and module.	Approved	Implicit by approved service usage as indicated by output from the BCryptGetProperty function.
Non-Approved Mode	Operation of the module if a non-approved algorithm is called for.	Non-Approved	Implicit by non-approved service usage as indicated by output from the BCryptGetProperty function.

Table 9: Modes List and Description

Mode Change Instructions and Status:

The module operates in its approved mode during normal operation of the module and when approved cryptographic algorithms are called. The mode changes to the non-approved mode implicitly if any non-approved algorithm is called. The calling application is responsible for ensuring that CSPs are not shared between approved and non-approved services and modes of operation.

2.5 Algorithms

Approved Algorithms:

The tables below list the approved algorithms used in the module. The module may not use some of the capabilities described in each CAVP certificate. As the module has separate CAVP certificates for Windows 11 and Windows Server 2022, each approved algorithm is listed twice, with CAVP certificates #A4008 and #A3763 for Windows 11 and CAVP certificates #A4009 and #A3764 for Windows Server 2022. See section 13 Standards References for links to the standards referenced in the tables below.

For its integrity check, the Cryptographic Primitives Library relies on some functionality that is implemented in the Code Integrity cryptographic module (certificate #5406) and the Secure Kernel Code Integrity module (certificate #5407). The Cryptographic Primitives Library also relies on the Kernel Mode Cryptographic Primitives Library (certificate #5408) module's randomness source for the entropy input to its approved AES-CTR-DRBG. All Windows cryptographic modules are installed together as described in section 11 Life-Cycle Assurance. FIPS 140-3 deems the Cryptographic Primitives Library module as binding to the Code Integrity, Secure Kernel Code Integrity, and Kernel Mode Cryptographic Primitives Library modules, which are referred to as Existing Validated Modules (EVMs) in this document. See section 5.1 Integrity Techniques for more information on the dependencies between Windows modules.

Table 10 below lists the approved algorithms in the Cryptographic Primitives Library module. Table 11 below lists the approved cryptographic algorithms in the Code Integrity, Secure Kernel Code Integrity, and Kernel Mode Cryptographic Primitives Library modules that are used by the Cryptographic Primitives Library.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4008	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4009	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A4008	Key Length - 128, 192, 256	SP 800-38C
AES-CCM	A4009	Key Length - 128, 192, 256	SP 800-38C
AES-CFB128	A4008	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A4009	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A4008	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A4009	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4008	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CMAC	A4009	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A4008	Direction - Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CTR	A4009	Direction - Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4008	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-ECB	A4009	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4008	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GCM	A4009	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A4008	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A4009	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D
AES-KW	A3763	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KW	A3764	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-XTS Testing Revision 2.0	A4008	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
AES-XTS Testing Revision 2.0	A4009	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A4008	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
Counter DRBG	A4009	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
DSA KeyGen (FIPS186-4)	A4008	L - 2048, 3072 N - 224, 256	FIPS 186-4
DSA KeyGen (FIPS186-4)	A4009	L - 2048, 3072 N - 224, 256	FIPS 186-4
DSA PQGGen (FIPS186-4)	A4008	L - 2048, 3072 N - 256 Hash Algorithm - SHA2-256	FIPS 186-4
DSA PQGGen (FIPS186-4)	A4009	L - 2048, 3072 N - 256 Hash Algorithm - SHA2-256	FIPS 186-4
DSA PQGVer (FIPS186-4)	A4008	L - 2048, 3072 N - 256 Hash Algorithm - SHA2-256	FIPS 186-4
DSA PQGVer (FIPS186-4)	A4009	L - 2048, 3072 N - 256 Hash Algorithm - SHA2-256	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyGen (FIPS186-4)	A4008	Curve - P-256, P-384, P-521 Secret Generation Mode - Extra Bits	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A4009	Curve - P-256, P-384, P-521 Secret Generation Mode - Extra Bits	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A4008	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A4009	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4008	Component - No, Yes Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4009	Component - No, Yes Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4008	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4009	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
HMAC-SHA-1	A4008	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA-1	A4009	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4008	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4009	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4008	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4009	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4008	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4009	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC Sp800-56Ar3	A4008	Domain Parameter Generation Methods - P-256, P-384, P-521 Function - Full Validation, Key Pair Generation, Partial Validation Scheme - ephemeralUnified - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 onePassDh - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 staticUnified - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256	SP 800-56A Rev. 3
KAS-ECC Sp800-56Ar3	A4009	Domain Parameter Generation Methods - P-256, P-384, P-521 Function - Full Validation, Key Pair Generation, Partial Validation Scheme - ephemeralUnified - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 onePassDh - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 staticUnified - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256	SP 800-56A Rev. 3
KAS-ECC-SSC Sp800-56Ar3	A4008	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-ECC-SSC Sp800-56Ar3	A4009	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
KAS-FFC Sp800-56Ar3	A4008	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, MODP-2048, MODP-3072, MODP-4096 Function - Full Validation, Key Pair Generation, Partial Validation Scheme - dhEphem - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 dhOneFlow - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 dhStatic - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256	SP 800-56A Rev. 3
KAS-FFC Sp800-56Ar3	A4009	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, MODP-2048, MODP-3072, MODP-4096 Function - Full Validation, Key Pair Generation, Partial Validation Scheme - dhEphem - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 dhOneFlow - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256 dhStatic - KAS Role - Initiator, Responder KDF Methods - oneStepKdf - Key Length - 256	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
KAS-FFC-SSC Sp800-56Ar3	A4008	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, MODP-2048, MODP-3072, MODP-4096 Scheme - dhEphem - KAS Role - initiator, responder dhOneFlow - KAS Role - initiator, responder dhStatic - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A4009	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, MODP-2048, MODP-3072, MODP-4096 Scheme - dhEphem - KAS Role - initiator, responder dhOneFlow - KAS Role - initiator, responder dhStatic - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF SP800-56Cr2	A4008	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8 HMAC Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
KDA HKDF SP800-56Cr2	A4009	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8 HMAC Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
KDF IKEv1 (CVL)	A4008	Authentication Method - Digital Signature, Pre-shared Key, Public Key Encryption Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 224-8192 Increment 8, Diffie-Hellman Shared Secret Length: 256-2048 Increment 8 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512 Preshared Key Length - Preshared Key Length: 64-2048 Increment 8	SP 800-135 Rev. 1
KDF IKEv1 (CVL)	A4009	Authentication Method - Digital Signature, Pre-shared Key, Public Key Encryption Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 224-8192 Increment 8, Diffie-Hellman Shared Secret Length: 256-2048 Increment 8 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512 Preshared Key Length - Preshared Key Length: 64-2048 Increment 8	SP 800-135 Rev. 1
KDF IKEv2 (CVL)	A4008	Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 256-2048 Increment 8 Derived Keying Material Length - Derived Keying Material Length: 192-1792 Increment 8 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Algorithm	CAVP Cert	Properties	Reference
KDF IKEv2 (CVL)	A4009	Diffie-Hellman Shared Secret Length - Diffie-Hellman Shared Secret Length: 256-2048 Increment 8 Derived Keying Material Length - Derived Keying Material Length: 192-1792 Increment 8 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF SP800-108	A3763	KDF Mode - Counter Supported Lengths - Supported Lengths: 160-512 Increment 8	SP 800-108 Rev. 1
KDF SP800-108	A3764	KDF Mode - Counter Supported Lengths - Supported Lengths: 160-512 Increment 8	SP 800-108 Rev. 1
KDF TLS (CVL)	A4008	TLS Version - v1.0/1.1 Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1
KDF TLS (CVL)	A4009	TLS Version - v1.0/1.1 Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1
PBKDF	A4008	Iteration Count - Iteration Count: 10-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
PBKDF	A4009	Iteration Count - Iteration Count: 10-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA Decryption Primitive (CVL)	A4008	Modulus Length - 2048	FIPS 186-4
RSA Decryption Primitive (CVL)	A4009	Modulus Length - 2048	FIPS 186-4
RSA KeyGen (FIPS186-4)	A4008	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4
RSA KeyGen (FIPS186-4)	A4009	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A4008	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigGen (FIPS186-4)	A4009	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA Signature Primitive (CVL)	A4008	Private Key Format - standard	FIPS 186-4
RSA Signature Primitive (CVL)	A4009	Private Key Format - standard	FIPS 186-4
RSA SigVer (FIPS186-4)	A4008	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A4009	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
Safe Primes Key Generation	A4008	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, MODP-2048, MODP-3072, MODP-4096, MODP-6144	SP 800-56A Rev. 3
Safe Primes Key Generation	A4009	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, MODP-2048, MODP-3072, MODP-4096, MODP-6144	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
SHA-1	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA-1	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A4008	Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1
TLS v1.2 KDF RFC7627 (CVL)	A4009	Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1

Table 10: Approved Algorithms -

Existing Validated Module [EVM]

Algorithm	CAVP Cert	Properties	Reference
Counter DRBG	A4008	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
Counter DRBG	A4009	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
RSA SigVer (FIPS186-4)	A4008	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A4009	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
SHA2-256	A4008	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4009	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

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

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
Cryptographic Key Generation (CKG)	Key Type: Symmetric and Asymmetric	N/A	NIST SP 800-133r2 Section 4 Example 1

Table 12: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and Function
MD5 (no security claimed)	Allowed for use with the TLS 1.0/1.1 KDF per FIPS 140-3 IG 2.4.A, example scenario 2a.	Hashing and Message Authentication

Table 13: Non-Approved, Allowed Algorithms with No Security Claimed

Non-Approved, Not Allowed Algorithms:

The following table presents the non-approved, not allowed algorithms. Any use of these non-approved algorithms will cause the module to operate outside of the approved mode.

Name	Use and Function
ANSI X9.42	Key derivation.
ANSI X9.63	Key derivation.
DES	Encryption in ECB, CBC, CFB8, and CFB64 modes.
DSA key and PQG generation	Key generation and PQG generation, except when part of KAS-FFC key generation.
DSA signature generation, verification, and PQG generation	Signature generation; Signature Verification; PQG generation.
ECDSA with non-approved, non-allowed curves	Key agreement with the curves and strengths listed below. Not allowed curves used: brainpoolP224r1 (112 bits); brainpoolP224t1(112bits); brainpoolP256r1(128bits); brainpoolP256t1(128bits); brainpoolP320r1(160bits); brainpoolP320t1(160bits); brainpoolP384r1(192bits); brainpoolP384t1(192bits); brainpoolP512r1(256bits); brainpoolP512t1(256bits); secP224k1(112bits); secP224r1(112bits); secP256k1(128bits); secP256r1(128bits); secP384r1(192bits); secP521r1(256bits); wtls12(112bits); x962P239v1(120bits); x962P239v2(120bits); x962P239v3(120bits); x962P256v1(128bits); brainpoolP192r1(96 bits); brainpoolP192t1(96bits); brainpoolP160r1(80bits); Curve25519(128 bits); ec192wapi(96bits); nistP192(96bits); numsP256t1(128bits); numsP384t1(192bits); numsP512t1(256bits); secP160k1 (80bits); secP160r1(80bits); secP160r2(80bits); secP192k1(96bits); secP192r1(96bits); wtls7(80bits); wtls9(80bits); x962P192v1(96bits); x962P192v2(96bits); x962P192v3(96bits).
HMAC-SHA-1	HMAC generation for message integrity with key sizes less than 112 bits (14 bytes).
KDF TLS (non-compliant without extended master secret)	TLS v1.2 KDF without support for RFC 7627.
Legacy CAPI KDF (proprietary)	Non-approved algorithm for key derivation. Used for backwards compatibility for IT systems prior to the publication of NIST SP 800-108.
MD2	Message digest.
MD4	Message digest.
MD5	Message digest.
NIST SP 800-56Ar2 key establishment	Any key agreement prior to revision 3, including Diffie-Hellman and EC-Diffie Hellman (used by a non-approved service, except for self-tests).
Non-compliant HKDF	Key Derivation.
RC2	Encryption and Decryption.
RC4	Encryption and Decryption.
RSA	Signature generation using 1024 bits; Encryption; Decryption
SHA-1	SHA-1 hash for digital signature generation.
Triple-DES.	Encryption; Decryption; Key and key-pair generation

Table 14: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

The table below lists the module's Security Function Implementations. The Algorithms column presents algorithm and key size information for each CAVP certificate listed in section 2.5 Algorithms. Blank cells are either not applicable or optional according to the CMVP.

Name	Type	Description	Properties	Algorithms
AsymKeyPair1	AsymKeyPair-KeyGen CKG	Asymmetric key generation function used by the Key and Key-Pair Generation service.		DSA KeyGen (FIPS186-4): (A4008, A4009) L, N: 2048, 224 L, N: 2048, 256 L, N: 3072, 256 ECDSA KeyGen (FIPS186-4): (A4008, A4009) Curve: P-256, P-384, P-521 RSA KeyGen (FIPS186-4): (A4008, A4009) Key Generation Mode: B.3.3 Moduli: 2048, 3072, 4096 Key Pair Generation: DRBG, SHA Safe Primes Key Generation: (A4008, A4009) Safe Prime Groups: ffdhe2048, ffdhe3072, MODP-2048, MODP-3072
AsymKeyPair2	AsymKeyPair-KeyVer CKG	Asymmetric key verification function used by the Signing and Verification service.		ECDSA KeyVer (FIPS186-4): (A4008, A4009) Curve: P-256, P-384, P-521

Name	Type	Description	Properties	Algorithms
AsymKeyPair3	AsymKeyPair-DomPar CKG	Asymmetric domain parameter generation and verification used by the Key and Key-Pair Generation service.		DSA PQGGen (FIPS186-4): (A4008, A4009) L, N: 2048, 256 L, N: 3072, 256 Hash Algorithm: SHA2-256 DSA PQGVer (FIPS186-4): (A4008, A4009) L, N: 2048, 256 L, N: 3072, 256 Hash Algorithm: SHA2-256
BC1	BC-UnAuth	Symmetric block cipher function used by the Encryption and Decryption service.		AES-CBC: (A4008, A4009) Key Length: 128, 192, 256 AES-CFB8: (A4008, A4009) Key Length: 128, 192, 256 AES-CFB128: (A4008, A4009) Key Length: 128, 192, 256 AES-CTR: (A4008, A4009) Key Length: 128, 192, 256 AES-ECB: (A4008, A4009) Key Length: 128, 192, 256 AES-XTS Testing Revision 2.0: (A4008, A4009) Key Length: 128, 256
BC2	BC-Auth	Symmetric block cipher function used by the Encryption and Decryption service.		AES-CCM: (A4008, A4009) Key Length: 128, 192, 256 AES-GCM: (A4008, A4009) IV Generation: External IV Generation Mode: 8.2.1 Key Length: 128, 192, 256

Name	Type	Description	Properties	Algorithms
CKG1	CKG	Symmetric key generation function used by the Key and Key-Pair Generation service.		Cryptographic Key Generation (CKG): ()
DigSig-Legacy	DigSig-SigVer	Legacy RSA signature verification used by the Signing and Verification service. SHA secure hash functions support this by verifying the RSA signature.		RSA SigVer (FIPS186-4): (A4008, A4009) Signature Type: PKCS#1v1.5 Modulus: 1024 bits Hash Pair Algorithm: SHA1 with 20-bit salt length
DigSig1	DigSig-SigGen	Digital signature generation functions used by the Signing and Verification service.		ECDSA SigGen (FIPS186-4): (A4008, A4009) Curve, Hash: P-256, SHA2-256 Curve, Hash: P-384, SHA2-384 Curve, Hash: P-521, SHA2-512 RSA SigGen (FIPS186-4): (A4008, A4009) Signature Type: PKCS#1v1.5; PKCSPSS Moduli: 2048, 3072, 4096 bits Hash Pair Algorithms: SHA2-256, SHA2-384, SHA2-512 Salt Length (for PSS): 32, 48, and 64 bits RSA Signature Primitive: (A4008, A4009) Private Key Format: Standard Public Exponent Mode: Fixed

Name	Type	Description	Properties	Algorithms
DigSig2	DigSig-SigVer	Digital signature verification functions used by the Signing and Verification service.		ECDSA SigVer (FIPS186-4): (A4008, A4009) Curve, Hash Algorithm: P-256, SHA2-256 Curve, Hash Algorithm: P-384, SHA2-384 Curve, Hash Algorithm: P-521, SHA2-512 RSA SigVer (FIPS186-4): (A4008, A4009) Signature Type: PKCS#1v1.5; PKCSPSS Moduli: 2048, 3072, 4096 bits Hash Pair Algorithms: SHA2-256, SHA2-384, SHA2-512 Salt Length (for PSS): 32, 48, and 64 bits
DigSig3	DigSig-SigVer	RSA signature verification used for the module pre-operational software integrity test only, executed by the Code Integrity or Secure Kernel Code Integrity module (EVM, IG 1.A Documentation Requirements 5). SHA secure hash functions support this by verifying the RSA signature.		RSA SigVer (FIPS186-4): (A4008, A4009) Signature Type: PKCS#1v1.5; PKCSPSS Modulus: 2048 bits Hash Pair Algorithm: SHA2-256 Salt Length (for PSS): 32 bits
DRBG1	DRBG	Deterministic random bit generator function used by the Random Number Generation service.		Counter DRBG: (A4008, A4009) Mode: AES-256 Entropy Input: 256 bits

Name	Type	Description	Properties	Algorithms
ENT1	ENT-ESV	ESV-validated physical entropy source function used by the Random Number Generation Service.		

KAS-135KDF-IKE	KAS-135KDF	IKE key derivation functions used by the Key Derivation service.		KDF IKEv1: (A4008) Authentication Methods: Digital Signature, Pre-shared Key, Public Key Encryption Pre-shared Key Length (Pre-shared Key Authentication Method): 64-2048 bits with increments of 8-bit Diffie-Hellman Shared Secret Length: 256-2048 Increment 8 (Digital Signature, Public Key Encryption), 224-8192 Increment 8 (Pre-shared Key) Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 KDF IKEv1: (A4009) Authentication Methods: Digital Signature, Pre-shared Key, Public Key Encryption Pre-shared Key Length (Pre-shared Key Authentication Method): 64-2048 bits with increments of 8-bit Diffie-Hellman Shared Secret Length: 256-2048 bits with Increments of 8-bit (Digital Signature, Public Key Encryption), 224-8192 bits with Increments of 8-bit (Pre-shared Key) Hash Algorithm: SHA2-

Name	Type	Description	Properties	Algorithms
				256, SHA2-384, SHA2-512 KDF IKEv2: (A4008, A4009) Diffie-Hellman Shared Secret Length: 256-2048 bits with increment of 8 bits Derived Keying Material Length: 192-1792 bits with increment of 8 bits Hash Algorithm: SHA2-256, SHA2-384, SHA2-512
KAS-135KDF-TLS	KAS-135KDF	TLS key derivation functions used by the Key Derivation service.		KDF TLS: (A4008, A4009) TLS Version: v1.0/1.1 Hash Algorithm: SHA2-256, SHA2-384 TLS v1.2 KDF RFC7627: (A4008, A4009) Hash Algorithm: SHA2-256, SHA2-384 Key Block Length: 1024 bits MD5: ()
KAS-ECC-SSC1	KAS-SSC	KAS shared secret computation used by the Secret Agreement service.	IG: D.F., scenario 2, path 1	KAS-ECC-SSC Sp800-56Ar3: (A4008, A4009) Domain Parameter Generation Methods: P-256 (hash functions SHA2-256, SHA2-384, SHA2-512), P-384 (hash functions SHA2-384, SHA2-512), and P-521 (hash function SHA2-512) Scheme: ephemeralUnified KAS Roles: initiator, responder

Name	Type	Description	Properties	Algorithms
KAS-ECC1	KAS-SSC/KDF	Key agreement functions used by the Secret Agreement service.	IG: IG D.F., scenario 2, path (2), end-to-end Key confirmation: No Key derivation: KDA (tested as part of the KAS certificate)	KAS-ECC Sp800-56Ar3: (A4008, A4009) Parameter sets: EC, ED, EE Domain Parameter Generation Methods: P-256, P-384, P-521 Auxiliary Function Methods (Curve): SHA2-256 (P-256), SHA2-384 (P-384), SHA2-512 (P-521)
KAS-FFC-SSC1	KAS-SSC	KAS shared secret computation used by the Secret Agreement service.	IG: D.F., scenario 2, path 1	KAS-FFC-SSC Sp800-56Ar3: (A4008, A4009) Domain Parameter Generation Methods: ffdhe2048, MODP-2048, ffdhe3072, MODP-3072, FB, FC Schemes: dhEphem, dhOneFlow, dhStatic KAS Roles: initiator, responder
KAS-FFC1	KAS-SSC/KDF	Key agreement functions used by the Secret Agreement service.	IG: IG D.F., scenario 2, path (2), end-to-end Key confirmation: No Key derivation: KDA (tested as part KAS certificate)	KAS-FFC Sp800-56Ar3: (A4008, A4009) Domain Parameter Generation Methods: FB (p=2048 and 3072, q=224), FC (p=2048, q=256), and safe primes (ffdhe2048, MODP-2048, ffdhe3072, MODP-3072) Auxiliary Function Methods: SHA2-256, SHA2-512

Name	Type	Description	Properties	Algorithms
KBKDF1	KBKDF	Key-based key derivation function used by the Key Derivation service.		KDF SP800-108: (A3763, A3764) KDF Mode: Counter MAC Mode: CMAC-AES128, CMAC-AES192, CMAC-AES256, HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 Supported Lengths: 160-256 bits with increments of 8 bits
KDA1	KAS-56CKDF	HMAC key derivation function used by the Key Derivation service.		KDA HKDF SP800-56Cr2: (A4008, A4009) Derived Key Length: 2048 bits Shared Secret Length: 224-8192 bits with increments of 8 bits HMAC Algorithm: SHA-1, SHA2-256, SHA2-384, SHA2-512
KTS1	BC-AuthDecrypt BC-AuthEncrypt	Key wrapping and unwrapping functions used by the Key Entry and Output service.		AES-KW: (A3763, A3764) Key Length: 128, 192, 256 bits

MAC1	MAC	Message Authentication function used by the Hashing and Message Authentication service		AES-CMAC: (A4008, A4009) Key Length: 128, 192, 256 bits MAC Length: 64-128 bits with increments of 8 bits Message Length: 0-524288 bits with increments of 8 bits AES-GMAC: (A4008, A4009) IV Generation: External Key Length: 128, 192, 256 bits HMAC-SHA-1: (A4008, A4009) MAC Length: 80-160 bits with increments of 8 bits Key Length: 8-2048 bits with increments of 8 bits HMAC-SHA2-256: (A4008, A4009) MAC Length: 80-256 bits with increments of 8 bits Key Length: 8-2048 bits with increments of 8 bits HMAC-SHA2-384: (A4008, A4009) MAC Length: 80-384 bits with increments of 8 bits Key Length: 8-2048 bits with increments of 8 bits HMAC-SHA2-512: (A4008, A4009) MAC Length: 80-512 bits with increments of 8 bits Key Length: 8-2048 bits with increments of 8 bits

Name	Type	Description	Properties	Algorithms
PBKDF1	PBKDF	Password-based key derivation function used by the Key Derivation service.		PBKDF: (A4008, A4009) Iteration Count: 10-10000 bits with increments of 1-bit HMAC Algorithm: SHA-1, SHA2-256, SHA2-384, SHA2-512
RSADP1	AsymKeyPair-Decap	Key Transport component function used by the Encryption and Decryption service.		RSA Decryption Primitive: (A4008, A4009) Modulus Length: 2048 bits
SHS1	SHA	Secure hash function used by the Hashing and Message Authentication service.		SHA-1: (A4008, A4009) Message Length: 0-65536 with increments of 8 SHA2-256: (A4008, A4009) Message Length: 0-65536 with increments of 8 SHA2-384: (A4008, A4009) Message Length: 0-65536 with increments of 8 SHA2-512: (A4008, A4009) Message Length: 0-65536 with increments of 8
SHS2	SHA	Secure hash function used for the module pre-operational software integrity check only, executed by the Code Integrity or Secure Kernel Code Integrity module (EVM, IG 1.A Documentation Requirements 5).		SHA2-256: (A4008, A4009) Message Length: 0-65536 with increments of 8

Table 15: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-GCM

The Crypto Officer can use the module's API to perform AES GCM encryption using internal IV generation. When the operator chooses to have this cryptographic module generate initialization vectors for AES GCM mode, then the call BCryptGenerateSymmetricKey() must set dwFlags to 0x00000020. These IVs are always 96 bits and generated using the approved DRBG internal to the module's boundary in compliance with Scenario 2 of IG C.H.

The module also supports importing of GCM IVs when an initialization vector (IV) is not generated within the module. When imported, the AES-GCM IV must be constructed as described in Implementation Guidance C.H Scenario 1(a) – 1(e). For example, for a module API user implementing TLS 1.2 and/or TLS 1.3 for GCM, IV generation follows RFC 5288 for TLS version 1.2 and RFC 5116 for TLS 1.3. This implementation must be compatible with acceptable AES-GCM cipher suites from SP800-52r2 Section 3.3.1. The IV consists of 12 bytes (96 bits) divided into two fields: the salt (fixed field), which is 4 bytes (32 bits), and the explicit nonce (counter field), which is 8 bytes (64 bits). The counter portion of the IV must be set by the module within its cryptographic boundary.

The module does not implement the TLS protocol. The module's implementation of AES-GCM is used together with an application that runs outside the module's cryptographic boundary. The design of the TLS protocol implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all of its possible values. In compliance with IG C.H section 3, if the module's power is lost and then restored, the key used for the AES GCM encryption / decryption shall be re-distributed.

2.7.2 AES-XTS

The following note applies to the approved AES-XTS algorithms listed in section 2.5 Algorithms.

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

2.7.3 DSA

The following notes apply to the approved DSA algorithms listed in section 2.5 Algorithms according to FIPS 140-3 Implementation Guidance C.K.

1. DSA KeyGen, DSA PQGGen, and DSA PQGVer are only used for KAS-FFC key generation as an approved service.
2. The use of DSA KeyGen, PQGGen, and PQGVer is allowed only for Key Agreement since the module implements FB and FC groups.

2.7.4 RSA

The following notes apply to the approved RSA algorithms listed in section 2.5 Algorithms.

1. RSA signature verification using SHA-1 is used for legacy signature verification only.

2. 1024-bit RSA key is used for legacy signature verification only.
3. Algorithms designated as “legacy” can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 Implementation Guidance C.M.
4. RSA Signature Primitive (CVL) is only used within the context of FIPS 186-5 signature generation.
5. The SP 800-56Br2 RSA Decryption Primitive (CVL) is only used as part of a SP 800-56Brev2 key transport scheme.

2.7.5 FIPS 186-4 and 186-5

The module claims compliance with FIPS 186-5 for the following algorithms, although the CAVP testing for them was completed against FIPS 186-4.

- DSA KeyGen (for generation of KAS-FFC keys, FIPS 186-4)
- DSA SigVer (FIPS186-4)
- ECDSA KeyGen (FIPS186-4)
- ECDSA KeyVer (FIPS186-4)
- ECDSA SigGen (FIPS186-4)
- ECDSA SigVer (FIPS186-4)
- RSA KeyGen (FIPS186-4)
- RSA SigGen (FIPS186-4)
- RSA Signature Primitive (CVL)
- RSA SigVer (FIPS186-4)

Because the FIPS 186-4 RSA CAVP tests are mathematically identical to the FIPS 186-5 RSA CAVP tests, the module can claim a FIPS 186-5 compliance for these tests. The scope of FIPS 186-4 testing complies with FIPS 140-3 Implementation Guidance C.K. Additional Comment 3. Although FIPS 186-4 has been superseded by FIPS 186-5, algorithms implemented under FIPS 186-4 remain approved for use under NIST SP 800-131A Rev. 2.

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

NIST SP 800-132 provides two options to derive the Data Protection Key (DPK) from the Master Key. With this module, it is up to the caller to select which option to generate/protect the DPK. For example, Windows DPAPI uses option 2a. The module provides all the building blocks for the caller to select the desired option.
The module supports the following HMAC hash functions as parameters for PBKDF:

- SHA-1 HMAC
- SHA2-256 HMAC
- SHA2-384 HMAC
- SHA2-512 HMAC

The module supports an iteration count of 10-10,000 (increment 1) and a salt length of 128-4096 bits (increment 8), depending on user application needs.

Keys derived from passwords, as described in SP 800-132, may only be used for storage applications. To run in the approved mode, 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 when the password/passphrase is created and not by this cryptographic module. The probability of guessing a password

is determined by its length and complexity, and an organization should define a policy for these based their threat model, such as the example guidance in NIST SP800-63B, Appendix A.

2.7.7 Key Agreement Schemes (KAS)

The KAS functions listed in section 2.6 Security Function Implementations (SFIs) document the key agreement schemes provided by the module. These include:

- KAS-SSC/KDF SFIs, which offer full-KAS as a service to module callers. The module does not use the KAS-SSC/KDF SFIs to establish module SSPs.
- KAS-SSC SFIs, which offer standalone shared secret computation as a service to module callers. The subsequent use of the generated Z is outside the control of the module.
- KAS-135KDF SFIs, which provide TLS and IKE key derivation functions to module callers.

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

2.7.8 NIST SP 800-38F AES Key Wrapping

As outlined in FIPS 140-3 Implementation Guidance D.G, AES-KW, AES-CCM, and AES-GCM are approved key wrapping algorithms for use as a key transport method, in accordance with NIST SP 800-38F Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping.

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

2.8 RBG and Entropy

The tables below list the RBG certificate, entropy certificates, and entropy source details for the entropy source used by the random number generation service.

The entropy source uses the Intel RDSEED instruction with additional, Microsoft-implemented conditioning functions and DRBGs. The SHA2-512 conditioning components output 512 bits of full entropy per 512-bit output block, and the AES-CTR-DRBGs output 256 bits of full entropy per 256-bit output block.

The entropy source used to seed the CNG AES-CTR-256 DRBG is the CNG Root PRNG and the conditioning chain that initializes and reseeds it. The Root PRNG receives its initial seed from the Windows OS Loader AES-256-CTR DRBG, which itself is seeded by a conditioning chain consisting of RDSEED instruction output processed by a Microsoft-implemented SHA2-512 vetted conditioning component. Subsequent reseeds of the Root PRNG are performed with entropy arriving via the RDSEED instruction output that is then distributed into SHA2-512 entropy pools.

The module's DRBG is part of a SP800-90C compliant RBGC construction, as listed in the RBG certificate table below. The DRBGs in the chain are seeded and instantiated with 256-bits of security strength.

#	Vendor Name	Certificate Number
1	Microsoft Corporation	#G1

Table 16: Random Bit Generator (RBG) Certificates

Cert Number	Vendor Name
E189	Microsoft Corporation
E216	Microsoft Corporation

Table 17: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Windows OS Loader Entropy Source for Windows 11	Physical	1) Microsoft Windows 11 version 22H2 running on a 12th Gen Intel Core i7-1265U with AES-NI; 2) Microsoft Windows 11 version 22H2 running on an 11th Gen Intel i7-1185G7 with AES-NI; 3) Microsoft Windows 11 version 22H2 running on an 11th Gen Intel i5-11500H with AES-NI.	512 bits	512 bits	CBC-MAC with AES-128 (#A2873, #A2668, #A1791); SHA2-512 (#A4008, #A4009)
Windows OS Loader Entropy Source for Windows Server 2022	Physical	1) Microsoft Windows Server 2022 running on an Intel Xeon Gold 6130 with AES-NI.	512 bits	512 bits	CBC-MAC with AES-128 (#A2873, #A2668, #A1791); SHA2-512 (#A4008, #A4009)

Table 18: Entropy Sources

2.9 Key Generation

The module can create and use keys for the following algorithms: AES, RSA, DH, ECDH, DSA, ECDSA, and HMAC, as well as RC2, RC4, DES, and Triple-DES. However, RC2, RC4, DES, and Triple-DES may not be used in the approved mode of operation.

Keys may be generated by calling the BCryptGenerateSymmetricKey() and BCryptGenerateKeyPair() functions. Random data generated by the BCryptGenRandom() function is provided to BCryptGenerateSymmetricKey() function to generate symmetric keys.

AES keys are generated following the techniques given in NIST SP 800-133 r2 (sections 6.1 and 6.2); RSA, Safe Primes, DSA, and ECDSA keys and key pairs are generated following the techniques given in NIST SP 800-133 r2 (sections 5.1 and 5.2); DH and ECDH keys and key-pairs are generated following the techniques given in SP 800-56Arev3 (section 5.8).

Keys generated while operating in the non-approved mode of operation (as described in section 2.4 Modes of Operation) may not be used in the approved mode, and vice versa.

When an application requests the cryptographic module to generate keys for a user, the keys are generated, used, and deleted as requested by applications.

2.10 Key Establishment

In its approved mode of operation, the module supports approved KAS-FFC (Diffie-Hellman) and KAS-ECC (Elliptic Curve Diffie-Hellman) key agreement schemes.

2.11 Industry Protocols

While the Cryptographic Primitives Library does not contain a TLS implementation, a TLS developer can use the cryptographic primitives implemented in the module to construct a TLS client or server incorporating any of the cipher suites specified in section 3.3.1 of [SP800-52]. The module also provides IKEv1 and IKEv2 key derivation functions for use in the same context by user applications.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

As a software-hybrid module, the module has no physical ports of its own. The physical ports of the module are interpreted as those on the underlying hardware platform and control of them is outside the scope of the module.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	The Data Input Interface consists of the functions listed in section 3.4.1 Export Functions, except for the Control Input Interfaces. Data and options are passed to the interface as input parameters to the export functions. Data Input is kept separate from Control Input by passing Data Input in separate parameters from Control Input.
N/A	Data Output	The Data Output Interface consists of the functions listed in section 3.4.1 Export Functions, except for the Control Input Interfaces. Data is returned to the function's caller via output parameters.
N/A	Control Input	The Control Input Interface are the functions listed in section 3.4.2.1 Algorithm Providers and Properties. Options for control operations are passed as input parameters to these functions.
N/A	Status Output	The Status Output Interface consists of the CNG primitive functions listed in section 3.4.2 CNG Algorithm Primitive Functions. For each function, the status information is returned to the caller as the return value from the function.
N/A	Power	N/A

Table 19: Ports and Interfaces

3.2 Additional Information

3.2.1 Export Functions

The module implements a set of algorithm providers for the CNG framework in Windows. Each provider in the module represents a single cryptographic algorithm or a set of closely related cryptographic algorithms provided by CNG. These algorithm providers are invoked through the CNG algorithm primitive functions, which are sometimes collectively referred to as the BCrypt API or CNG API. These are listed below in section 3.2.2 CNG Algorithm Primitive Functions, and a full list is published in the following topic:

- CNG Algorithm Identifiers, https://learn.microsoft.com/en-us/windows/win32/seccng/cng-algorithm-identifiers

The module exposes its cryptographic services to the operating system through the set of exported functions listed below. These functions are used by the CNG framework to retrieve references to the different algorithm providers, in order to route BCrypt/CNG API calls appropriately to the Cryptographic Primitives Library. These functions return references to implementations of cryptographic functions that correspond directly to functions in the BCrypt/CNG API. For details, please see the CNG documentation for Windows, published in the following topic:

- Cryptography API: Next Generation, https://learn.microsoft.com/en-us/windows/win32/seccng/cng-portal

The following functions are exported by the Cryptographic Primitives Library:

- Functions that return lists of function pointers to the CNG Algorithm Primitive Functions:
  - GetAsymmetricEncryptionInterface
  - GetCipherInterface
  - GetHashInterface
  - GetKeyDerivationInterface
  - GetRngInterface
  - GetSecretAgreementInterface
  - GetSignatureInterface

- Functions that are directly used:
  - ProcessPrng
  - ProcessPrngGuid

3.2.2 CNG Algorithm Primitive Functions

The tables below present the CNG functions used by callers to access the cryptographic services of the module, grouped by service category. These functions are exported by BCRYPT.DLL, and calls are forwarded to the Cryptographic Primitives Library module through the function pointer arrays returned by the GetInterface methods listed above. All the functions are used in the approved mode unless noted otherwise in the description. Furthermore, these are the only approved functions available. The module has additional export functions described in section 3.2.2.9 Non-Security Configuration Interfaces.

3.2.2.1 Algorithm Providers and Properties

Function	Signature	Description
BCryptOpenAlgorithmProvider	NTSTATUS WINAPi BCryptOpenAlgorithmProvider(BCRYPT_ALG_HANDLE *phAlgorithm, LPCWSTR pszAlgId, LPCWSTR pszImplementation, ULONG dwFlags);	The BCryptOpenAlgorithmProvider() function has four parameters: algorithm handle output to the opened algorithm provider, desired algorithm ID input, an optional specific provider name input, and optional flags. This function loads and initializes a CNG provider for a given algorithm and returns a handle to the opened algorithm provider on success. See the Cryptography API: Next Generation documentation at https://docs.microsoft.com/en-us/windows/win32/seccng/cng-portal for CNG providers. Unless the calling function specifies the name of the provider, the default provider is used. The default provider is the first provider listed for a given algorithm. The calling function must pass the BCRYPT_ALG_HANDLE_HMAC_FLAG flag in order to use an HMAC function with a hash algorithm.
BCryptCloseAlgorithmProvider	NTSTATUS WINAPi BCryptCloseAlgorithmProvider(BCRYPT_ALG_HANDLE hAlgorithm, ULONG dwFlags);	This function closes an algorithm provider handle opened by a call to BCryptOpenAlgorithmProvider() function.
BCryptSetProperty	NTSTATUS WINAPi BCryptSetProperty(BCRYPT_HANDLE hObject, LPCWSTR pszProperty, PUCHAR pbInput, ULONG cbInput, ULONG dwFlags);	The BCryptSetProperty() function sets the value of a named property for a CNG object, e.g., a cryptographic key. The CNG object is referenced by a handle, the property name is a NULL terminated string, and the value of the property is a length-specified byte string.
BCryptGetProperty	NTSTATUS WINAPi BCryptGetProperty(BCRYPT_HANDLE hObject, LPCWSTR pszProperty, PUCHAR pbOutput, ULONG cbOutput, ULONG *pcbResult, ULONG dwFlags);	The BCryptGetProperty() function retrieves the value of a named property for a CNG object, e.g., a cryptographic key. The CNG object is referenced by a handle, the property name is a NULL terminated string, and the value of the property is a length-specified byte string.
BCryptFreeBuffer	VOID WINAPi BCryptFreeBuffer(PVOID pvBuffer);	Some of the CNG functions allocate memory on caller's behalf. The BCryptFreeBuffer() function frees memory that was allocated by such a CNG function.

Table 20: CNG Algorithm Primitive Functions for Algorithm Providers and Properties

3.2.2.2 Random Number Generation

Function	Signature	Description
BCryptGenRandom	NTSTATUS WINAPi BCryptGenRandom(BCRYPT_ALG_HANDLE hAlgorithm, PUCHAR pbBuffer, ULONG cbBuffer, ULONG dwFlags);	The BCryptGenRandom() function fills a buffer with random bytes. BCRYPTPRIMITVES.DLL implements the following random number generation algorithm: BCRYPT_RNG_ALGORITHM. This is the AES-256 counter mode based random generator as defined in SP 800-90A r1.

Table 21: CNG Algorithm Primitive Functions for Random Number Generation

3.2.2.3 Key and Key-Pair Generation

Function	Signature	Description
BCryptGenerateSymmetricKey	NTSTATUS WINAPI BCryptGenerateSymmetricKey(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE *phKey, PUCHAR pbKeyObject, ULONG cbKeyObject, PUCHAR pbSecret, ULONG cbSecret, ULONG dwFlags);	The BCryptGenerateSymmetricKey() function generates a symmetric key object from the provided data, which the caller may generate from a DRBG for use with a symmetric encryption algorithm or key derivation algorithm from a supplied *cbSecret* bytes long key value provided in the *pbSecret* memory location. The calling application must specify a handle to the algorithm provider opened with the BCryptOpenAlgorithmProvider() function. The algorithm specified when the provider was opened must support symmetric key encryption or key derivation.
BCryptGenerateKeyPair	NTSTATUS WINAPI BCryptGenerateKeyPair(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE *phKey, ULONG dwLength, ULONG dwFlags);	The BCryptGenerateKeyPair() function creates a public/private key pair object without any cryptographic keys in it. After creating such an empty key pair object using this function, call the BCryptSetProperty() function to set its properties. The key pair can be used only after BCryptFinalizeKeyPair() function is called.
BCryptFinalizeKeyPair	NTSTATUS WINAPI BCryptFinalizeKeyPair(BCRYPT_KEY_HANDLE hKey, ULONG dwFlags);	The BCryptFinalizeKeyPair() function completes a public/private key pair import or generation directly from the output of a DRBG. The key pair cannot be used until this function has been called. After this function has been called, the BCryptSetProperty() function can no longer be used for this key pair.
BCryptDuplicateKey	NTSTATUS WINAPI BCryptDuplicateKey(BCRYPT_KEY_HANDLE hKey, BCRYPT_KEY_HANDLE *phNewKey, PUCHAR pbKeyObject, ULONG cbKeyObject, ULONG dwFlags);	The BCryptDuplicateKey() function creates a duplicate of a symmetric key object.

Function	Signature	Description
BCryptDestroyKey	NTSTATUS WINAPI BCryptDestroyKey(BCRYPT_KEY_HANDLE hKey);	The BCryptDestroyKey() function destroys a key.

Table 22: CNG Algorithm Primitive Functions for Key and Key-Pair Generation

3.2.2.4 Key Entry and Output (Import and Export)

Function	Signature	Description
BCryptImportKey	NTSTATUS WINAPI BCryptImportKey(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE hImportKey, LPCWSTR pszBlobType, BCRYPT_KEY_HANDLE *phKey, PUCHAR pbKeyObject, ULONG cbKeyObject, PUCHAR pbInput, ULONG cbInput, ULONG dwFlags);	The BCryptImportKey() function imports a symmetric key from a key blob.
BCryptImportKeyPair	NTSTATUS WINAPI BCryptImportKeyPair(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE hImportKey, LPCWSTR pszBlobType, BCRYPT_KEY_HANDLE *phKey, PUCHAR pbInput, ULONG cbInput, ULONG dwFlags);	The BCryptImportKeyPair() function is used to import a public/private key pair from a key blob.
BCryptExportKey	NTSTATUS WINAPI BCryptExportKey(BCRYPT_KEY_HANDLE hKey, BCRYPT_KEY_HANDLE hExportKey, LPCWSTR pszBlobType, PUCHAR pbOutput, ULONG cbOutput, ULONG *pcbResult, ULONG dwFlags);	The BCryptExportKey() function exports a key to a memory blob that can be persisted for later use.

Table 23: CNG Algorithm Primitive Functions for Key Entry and Output

3.2.2.5 Encryption and Decryption

Function	Signature	Description
BCryptEncrypt	NTSTATUS WINAPI BCryptEncrypt(BCRYPT_KEY_HANDLE hKey, PUCHAR pbInput, ULONG cbInput, VOID *pPaddingInfo, PUCHAR pbIV, ULONG cbIV, PUCHAR pbOutput, ULONG cbOutput, ULONG *pcbResult, ULONG dwFlags);	The BCryptEncrypt() function encrypts a block of data of given length.
BCryptDecrypt	NTSTATUS WINAPI BCryptDecrypt(BCRYPT_KEY_HANDLE hKey, PUCHAR pbInput, ULONG cbInput, VOID *pPaddingInfo, PUCHAR pbIV, ULONG cbIV, PUCHAR pbOutput, ULONG cbOutput, ULONG *pcbResult, ULONG dwFlags);	The BCryptDecrypt() function decrypts a block of data of given length.

Table 24: CNG Algorithm Primitive Functions for Encryption and Decryption

3.2.2.6 Hashing and Message Authentication

Function	Signature	Description
BCryptCreateHash	NTSTATUS WINAPI BCryptCreateHash(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_HASH_HANDLE *phHash, PUCHAR pbHashObject, ULONG cbHashObject, PUCHAR pbSecret, ULONG cbSecret, ULONG dwFlags);	The BCryptCreateHash() function creates a hash object with an optional key. The optional key is used for HMAC, AES GMAC and AES CMAC.
BCryptHashData	NTSTATUS WINAPI BCryptHashData(BCRYPT_HASH_HANDLE hHash, PUCHAR pbInput, ULONG cbInput, ULONG dwFlags);	The BCryptHashData() function performs a one way hash on a data buffer. Call the BCryptFinishHash() function to finalize the hashing operation to get the hash result.
BCryptDuplicateHash	NTSTATUS WINAPI BCryptDuplicateHash(BCRYPT_HASH_HANDLE hHash, BCRYPT_HASH_HANDLE *phNewHash, PUCHAR pbHashObject, ULONG cbHashObject, ULONG dwFlags);	The BCryptDuplicateHash()function duplicates an existing hash object. The duplicate hash object contains all state and data that was hashed to the point of duplication.
BCryptFinishHash	NTSTATUS WINAPI BCryptFinishHash(BCRYPT_HASH_HANDLE hHash, PUCHAR pbOutput, ULONG cbOutput, ULONG dwFlags);	The BCryptFinishHash() function retrieves the hash value for the data accumulated from prior calls to BCryptHashData() function.

Function	Signature	Description
BCryptDestroyHash	NTSTATUS WINAPI BCryptDestroyHash(BCRYPT_HASH_HANDLE hHash);	The BCryptDestroyHash() function destroys a hash object.
BCryptHash	NTSTATUS WINAPI BCryptHash(BCRYPT_ALG_HANDLE hAlgorithm, PUCHAR pbSecret, ULONG cbSecret, PUCHAR pbInput, ULONG cbInput, PUCHAR pbOutput, ULONG cbOutput);	The function BCryptHash() performs a single hash computation. This is a convenience function that wraps calls to the BCryptCreateHash(), BCryptHashData(), BCryptFinishHash(), and BCryptDestroyHash() functions.
BCryptCreateMultiHash	NTSTATUS WINAPI BCryptCreateMultiHash(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_HASH_HANDLE *phHash, ULONG nHashes, PUCHAR pbHashObject, ULONG cbHashObject, PUCHAR pbSecret, ULONG cbSecret, ULONG dwFlags);	BCryptCreateMultiHash() is a function that creates a new MultiHash object that is used in parallel hashing to improve performance. The MultiHash object is equivalent to an array of normal (reusable) hash objects.
BCryptProcessMultiOperations	NTSTATUS WINAPI BCryptProcessMultiOperations(BCRYPT_HANDLE hObject, BCRYPT_MULTI_OPERATION_TYPE operationType, PVOID pOperations, ULONG cbOperations, ULONG dwFlags);	The BCryptProcessMultiOperations() function is used to perform multiple operations on a single multi-object handle such as a MultiHash object handle. If any of the operations fail, then the function will return an error. Each element of the operations array specifies an operation to be performed on/with the hObject. For hash operations, there are two operation types: • Hash data • Finalize hash These correspond directly to BCryptHashData() and BCryptFinishHash(). Each operation specifies an index of the hash object inside the hObject MultiHash object that this operation applies to. Operations are executed in any order or even in parallel, with the sole restriction that the set of operations that specify the same index are all executed in-order.

Table 25: CNG Algorithm Primitive Functions for Hashing and Message Authentication

3.2.2.7 Signing and Verification

Function	Signature	Description
BCryptSignHash	NTSTATUS WINAPI BCryptSignHash(BCRYPT_KEY_HANDLE hKey, VOID *pPaddingInfo, PUCHAR pbInput, ULONG cbInput, PUCHAR pbOutput, ULONG cbOutput, ULONG *pcbResult, ULONG dwFlags);	The BCryptSignHash() function creates a signature of a hash value. Note: this function accepts SHA-1 hashes, which according to NIST SP 800-131A is *disallowed* for digital signature generation. SHA-1 is currently *legacy-use* for digital signature verification.
BCryptVerifySignature	NTSTATUS WINAPI BCryptVerifySignature(BCRYPT_KEY_HANDLE hKey, VOID *pPaddingInfo, PUCHAR pbHash, ULONG cbHash, PUCHAR pbSignature, ULONG cbSignature, ULONG dwFlags);	The BCryptVerifySignature() function verifies that the specified signature matches the specified hash. Note: this function accepts SHA-1 hashes, which according to NIST SP 800-131A is *disallowed* for digital signature generation. SHA-1 is currently *legacy-use* for digital signature verification.

Table 26: CNG Algorithm Primitive Functions for Signing and Verification

3.2.2.8 Secret Agreement and Key Derivation

Function	Signature	Description
BCryptSecretAgreement	NTSTATUS WINAPI BCryptSecretAgreement(BCRYPT_KEY_HANDLE hPrivKey, BCRYPT_KEY_HANDLE hPubKey, BCRYPT_SECRET_HANDLE *phAgreedSecret, ULONG dwFlags);	The BCryptSecretAgreement() function creates a secret agreement value from a private and a public key is used with SP 800-56Arev3 compliant Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) algorithms.
BCryptDeriveKey	NTSTATUS WINAPI BCryptDeriveKey(BCRYPT_SECRET_HANDLE hSharedSecret, LPCWSTR pwszKDF, BCryptBufferDesc *pParameterList, PUCHAR pbDerivedKey, ULONG cbDerivedKey, ULONG *pcbResult, ULONG dwFlags);	The BCryptDeriveKey() function derives a key from a secret agreement value.
BCryptDestroySecret	NTSTATUS WINAPI BCryptDestroySecret(BCRYPT_SECRET_HANDLE hSecret);	The BCryptDestroySecret() function destroys a secret agreement handle that was created by using the BCryptSecretAgreement() function.

Function	Signature	Description
BCryptKeyDerivation	NTSTATUS WINAPI BCryptKeyDerivation(BCRYPT_KEY_HANDLE hKey, BCryptBufferDesc *pParameterList, (cbDerivedKey, *pcbResult) PUCHAR pbDerivedKey, ULONG cbDerivedKey, ULONG *pcbResult, ULONG dwFlags);	The BCryptKeyDerivation() function executes a Key Derivation Function (KDF) on a key generated with BCryptGenerateSymmetricKey() function. It differs from the BCryptDeriveKey() function in that it does not require a secret agreement step to create a shared secret.
BCryptDeriveKeyPBKDF2	NTSTATUS WINAPI BCryptDeriveKeyPBKDF2(BCRYPT_ALG_HANDLE hPrf, PUCHAR pbPassword, ULONG cbPassword, PUCHAR pbSalt, ULONG cbSalt, ULONGLONG cIterations, PUCHAR pbDerivedKey, ULONG cbDerivedKey, ULONG dwFlags);	The BCryptDeriveKeyPBKDF2() function derives a key from a hash value by using the password based key derivation function as defined by NIST SP 800-132 PBKDF and IETF RFC 2898 (specified as PBKDF2).

Table 27: CNG Algorithm Primitive Functions for Secret Agreement and Key Derivation

3.2.2.9 Non-Security Configuration Interfaces

The following non-cryptographic functions are used to configure cryptographic providers on the system. See also the BCrypt API documentation at https://learn.microsoft.com/en-us/windows/win32/api/bcrypt/ for more information on these functions.

Function Name	Description
BCryptEnumAlgorithms	Enumerates the algorithms for a given set of operations.
BCryptEnumProviders	Returns a list of CNG providers for a given algorithm.
BCryptRegisterConfigChangeNotify	Deprecated interface.
BCryptResolveProviders	Resolves queries against the set of providers currently registered on the local system and the configuration information specified in the machine and domain configuration tables, returning an ordered list of references to one or more providers matching the specified criteria.
BCryptAddContextFunctionProvider	Adds a cryptographic function provider to the list of providers that are supported by an existing CNG context.
BCryptRegisterProvider	Registers a CNG provider.
BCryptUnregisterProvider	Unregisters a CNG provider.
BCryptUnregisterConfigChangeNotify	Removes a CNG configuration change event handler.
BCryptGetFipsAlgorithmMode	Determines whether the module is operating in FIPS mode. Some applications use the value returned by this API to alter their own behavior, such as blocking the use of some SSL versions.
BCryptQueryProviderRegistration	Retrieves information about a CNG provider.
BCryptEnumRegisteredProviders	Retrieves information about the registered providers.
BCryptCreateContext	Creates a new CNG configuration context.
BCryptDeleteContext	Deletes an existing CNG configuration context.
BCryptEnumContexts	Obtains the identifiers of the contexts in the specified configuration table.
BCryptConfigureContext	Sets the configuration information for an existing CNG context.
BCryptQueryContextConfiguration	Retrieves the current configuration for the specified CNG context.

Function Name	Description
BCryptAddContextFunction	Adds a cryptographic function to the list of functions that are supported by an existing CNG context.
BCryptRemoveContextFunction	Removes a cryptographic function from the list of functions that are supported by an existing CNG context.
BCryptEnumContextFunctions	Obtains the cryptographic functions for a context in the specified configuration table.
BCryptConfigureContextFunction	Sets the configuration information for the cryptographic function of an existing CNG context.
BCryptQueryContextFunctionConfiguration	Obtains the cryptographic function configuration information for an existing CNG context.
BCryptEnumContextFunctionProviders	Obtains the providers for the cryptographic functions for a context in the specified configuration table.
BCryptSetContextFunctionProperty	Sets the value of a named property or a cryptographic function in an existing CNG context.
BCryptQueryContextFunctionProperty	Obtains the value of a named property for a cryptographic function in an existing CNG context.
BCryptSetAuditingInterface	Sets the auditing interface.

Table 28: Non-Security Relevant Configuration Interfaces

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

4.2 Roles

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

Name	Type	Operator Type	Authentication Methods
Cryptographic Officer (CO)	Role	CO	None

Table 29: Roles

4.3 Approved Services

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

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Algorithm Providers and Properties	Loads and initializes algorithm providers, gets / sets algorithm object properties, and frees buffer taken by algorithm operations.	"Algorithm Providers and Properties" output from the BCryptGetProperty function for the service indicator function.	Invoked when appropriate input parameters are provided to the functions: BCryptOpenAlgorithmProvider BCryptCloseAlgorithmProvider BCryptSetProperty BCryptGetProperty BCryptFreeBuffer	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	None	Cryptographic Officer (CO)
Encryption and Decryption	Encrypts and decrypts a block of data.	The "Encryption and Decryption" output from the BCryptGetProperty function is the service indicator function for the approved methods.	Invoked when appropriate input parameters are provided to the functions: BcryptEncrypt BcryptDecrypt	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	BC1 BC2 RSADP1	Cryptographic Officer (CO) - AES-GCM IV: G,W,E - Asymmetric RSA Private Keys: W,E - Asymmetric RSA Public Keys: W,E - Symmetric AES Keys: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Hashing and Message Authentication	Generates hashes and authenticates messages.	The "Hashing and Message Authentication" output from the BCryptGetProperty function is the service indicator function for the approved methods.	Invoked when appropriate input parameters are provided to the functions: BcryptCreateHash BcryptHashData BcryptDuplicateHash BcryptFinishHash BcryptDestroyHash BcryptHash BcryptCreateMultiHash BcryptProcessMultiOperations	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	MAC1 SHS1	Cryptographic Officer (CO) - HMAC Keys: W,E - Symmetric AES Keys: W,E
Key and Key-Pair Generation	Generates symmetric keys and public / private key pairs.	"Key and Key-Pair Generation" output from the BCryptGetProperty function for the service indicator function for the approved key and key pair generation methods.	Invoked when appropriate input parameters are provided to the functions: BcryptGenerateSymmetricKey BcryptGenerateKeyPair BcryptFinalizeKeyPair BcryptDuplicateKey BcryptDestroyKey	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	AsymKeyPair 1 AsymKeyPair 2 AsymKeyPair 3 CKG1	Cryptographic Officer (CO) - AES-CTR DRBG Entropy Input: E - AES-CTR DRBG key: E - AES-CTR DRBG Seed: E - AES-CTR DRBG V: E - Asymmetric ECDSA Private Keys: G - Asymmetric ECDSA Public Keys: G - Asymmetric RSA Private Keys: G - Asymmetric RSA Public Keys: G - DH Private Values: G - DH Public Values: G - ECDH Private Values: G - ECDH Public Values: G - HMAC Keys: G - Symmetric AES Keys: G

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Derivation	Provides key derivation.	The "Key Derivation" output from the BCryptGetProperty function for the service indicator function.	Invoked when appropriate input parameters are provided to the functions: BcryptDeriveKey BcryptKeyDerivation BCryptDeriveKeyPBKDF2	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	KAS-135KDF-IKE KAS-135KDF-TLS KBKDF1 PBKDF1 KDA1	Cryptographic Officer (CO) - Derived Key: G,R - Key Derivation Key: R,W - PBKDF Password: E - Z - Shared Secret Input for IKE KDFs: R,W - Z - Shared Secret Input for KBKDF: R,W - Z - Shared Secret Input for KDA-HKDF: R,W
Key Entry and Output	Imports and exports symmetric keys and public / private key pairs.	The "Key Entry and Output" output from the BCryptGetProperty function is the service indicator function for approved key entry and output methods.	Invoked when appropriate input parameters are provided to the functions: BcryptImportKey BcryptImportKeyPair BcryptExportKey	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	KTS1	Cryptographic Officer (CO) - Asymmetric ECDSA Private Keys: R,W - Asymmetric ECDSA Public Keys: R,W - Asymmetric RSA Private Keys: R,W - Asymmetric RSA Public Keys: R,W - HMAC Keys: R,W - Symmetric AES Keys: R,W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform Cryptographic Algorithm Self-Tests	The module provides a power-up cryptographic algorithm self-test service that automatically executes when the module is loaded into memory.	Self-test success is indicated by module and algorithm availability; failure is indicated by an error.	Conditional self-tests are automatically executed when services are called. See section 10 Self-Tests for details.	Success is implicit in module availability. If any self-test fails, the module returns an error code. See section 10 Self-Tests for details.	AsymKeyPair 1 AsymKeyPair 2 AsymKeyPair 3 BC1 BC2 CKG1 DigSig-Legacy DigSig1 DigSig2 DRBG1 ENT1 KAS-ECC1 KAS-FFC1 KAS-ECC-SSC1 KAS-FFC-SSC1 KAS-135KDF-IKE KAS-135KDF-TLS KBKDF1 KDA1 RSADP1 MAC1 PBKDF1 SHS1	Cryptographic Officer (CO)

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform Pre-Operational Software Integrity Test [EVM]	The pre-operational self-test is executed by the [EVM] Secure Kernel Code Integrity module (IG 1.A Documentation Requirements 5).	Integrity test success is indicated by the Cryptographic Primitives Library module being loaded into memory.	This service is fully automatic and executed before the module is loaded into memory.	The Kernel Mode Cryptographic Primitives Library module is loaded into memory.	DigSig3 SHS2	Cryptographic Officer (CO)

Perform Zeroization	Zeroizes cryptographic material.	SSPs are zeroized. See Section 9.3 SSP Zeroization Methods for more information.	Invoked when appropriate input parameters are provided to the functions: BcryptDestroyKey BcryptDestroySecret Executed automatically as part of module shutdown.	Keys are zeroized when the key objects are deleted, which is required prior to unloading the module from memory.	None	Cryptographic Officer (CO) - AES-CTR DRBG Entropy Input: Z - AES-CTR DRBG key: Z - AES-CTR DRBG Seed: Z - AES-CTR DRBG V: Z - AES-GCM IV: Z - Asymmetric ECDSA Private Keys: Z - Asymmetric ECDSA Public Keys: Z - Asymmetric RSA Private Keys: Z - Asymmetric RSA Public Keys: Z - Derived Key: Z - DH Private Values: Z - DH Public Values: Z - ECDH Private Values: Z - ECDH Public Values: Z - Embedded X.509 Certificate for BCRYPTPRIMITIVES.DLL (This is not an SSP): Z - Hash Value for BCRYPTPRIMITIVES.DLL (This is not an SSP): Z - HMAC Keys: Z - Key Derivation Key: Z - PBKDF Password: Z - Symmetric AES Keys: Z - TLS Pre-Master Secret: Z - Z - Shared Secret Input for IKE KDFs: Z - Z - Shared Secret Input

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						for KBKDF: Z - Z - Shared Secret Input for KDA-HKDF: Z - Z - Shared Secret Output for KAS-ECC: Z - Z - Shared Secret Output for KAS-FFC: Z
Random Number Generation	Fills a buffer with random bytes using the AES-256 CTR mode DRBG	"Random Number Generation" output from the BCryptGetProperty function for the service indicator function.	Invoked when appropriate input parameters are provided to the function: BCryptGenRandom	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	DRBG1	Cryptographic Officer (CO) - AES-CTR DRBG Entropy Input: W,E - AES-CTR DRBG key: W,E - AES-CTR DRBG Seed: W,E - AES-CTR DRBG V: W,E
Secret Agreement	Provides key agreement.	The "Secret Agreement" output from the BCryptGetProperty function for the service indicator function for approved methods.	Invoked when appropriate input parameters are provided to the functions: BcryptSecretAgreement BcryptDestroySecret	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	KAS-ECC1 KAS-FFC1 KAS-ECC-SSC1 KAS-FFC-SSC1	Cryptographic Officer (CO) - Derived Key: G,R - DH Private Values: E - DH Public Values: E - ECDH Private Values: E - ECDH Public Values: E - Z - Shared Secret Output for KAS-ECC: R,W - Z - Shared Secret Output for KAS-FFC: R,W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show Status	Provides the module status response.	Provided via NTSTATUS by all functions.	This service is fully automatic and occurs when any function is called.	Status is output across the module's Status Output logical interface, to the computer monitor or to log files.	None	Cryptographic Officer (CO)
Show Version	Provides the module version number.	Version information is provided in the Portable Executable (PE) header of each module binary. The PE header contains Windows-specific fields such as the major and minor version, which equate to the module version. See the public documentation for the PE Format for more information.	Module binary file.	Version information is embedded in the PE header of the binary.	None	Cryptographic Officer (CO)

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Signing and Verification	Generates and verifies digital signatures.	The "Signing and Verification" output from the BCryptGetProperty function for the service indicator function for the approved signing and verification functions.	Invoked when appropriate input parameters are provided to the functions: BcryptSignHash BcryptVerifySignature	The cryptographic operation is executed, with output per the interfaces defined in section 3 Cryptographic Module Interfaces.	DigSig-Legacy DigSig1 DigSig2	Cryptographic Officer (CO) - Asymmetric RSA Private Keys: E - Asymmetric RSA Public Keys: E

Table 30: Approved Services

4.4 Non-Approved Services

The following table identifies the non-approved security services of the module. For additional details on the algorithms accessed by the non-approved services, see the non-approved algorithm details in section 2.5 Algorithms.

Name	Description	Algorithms	Role
Non-Approved Encryption and Decryption	Encryption and decryption using the non-approved algorithms listed to the right.	DES RC2 RC4 RSA Triple-DES.	CO
Non-Approved Hashing and Message Authentication	Hashing and message authentication using the non-approved algorithms listed to the right.	HMAC-SHA-1 MD2 MD4 MD5	CO
Non-Approved Key and Key-Pair Generation	Key and key-pair generation using the non-approved algorithms listed to the right.	DSA key and PQG generation ECDSA with non-approved, non-allowed curves	CO

Name	Description	Algorithms	Role
Non-Approved Key Entry and Output	Key entry and output using one of the non-approved algorithms listed to the right.	ECDSA with non-approved, non-allowed curves	CO
Non-Approved Secret Agreement and Key Derivation	Key derivation using one of the non-approved algorithms listed to the right. For HKDF, the service derives a key from a hash value when the exported function BCryptDeriveKeyCapi is called.	ANSI X9.42 ANSI X9.63 DSA key and PQG generation DSA signature and PQG generation ECDSA with non-approved, non-allowed curves KDF TLS (non-compliant without extended master secret) Legacy CAPI KDF (proprietary) NIST SP 800-56Ar2 key establishment Non-compliant HKDF	CO
Non-Approved Signing and Verification	Signing and verification using the non-approved algorithms listed to the right.	ECDSA with non-approved, non-allowed curves RSA SHA-1 DSA signature and PQG generation	CO

Table 31: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

5 Software/Firmware Security

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

Windows dynamic link libraries, which include BCRYPTPRIMITIVES.DLL, are loaded into a user-mode process to expose the services offered by that DLL. The operating system environment enforces process isolation including access to memory (where keys and intermediate key data are stored) and access to the CPU.

5.1 Integrity Techniques

Windows uses several mechanisms to provide integrity verification depending on the stage in the boot sequence, hardware, and configuration. The algorithms used for integrity verification are included in section 2.5 Algorithms, Table 11.

The integrity of the Cryptographic Primitives Library is checked before it is loaded into process memory by the Code Integrity module or the Secure Kernel Code Integrity module (EVMs, IG 1.A Documentation Requirements 5). See the table below for more information on the Code Integrity and Secure Kernel Code Integrity modules. To perform the module integrity test on demand, the user may restart the computer.

Windows binaries include a SHA2-256 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.

EVM Name	CMVP Certificate	Version Details
Code Integrity (Windows 11 v 22H2 and Windows Server 2022)	#5406	Windows 11 version 22H2, build 10.0.22621.1 Windows Server 2022, build 10.0.20348.1668
Secure Kernel Code Integrity (Windows 11 v 22H2 and Windows Server 2022)	#5407	Windows 11 version 22H2, build 10.0.22621.1 Windows Server 2022, build 10.0.20348.1668

Table 32: EVM Details

The figure below shows the Integrity Chain of trust for the Windows builds and modules in scope for this validation.

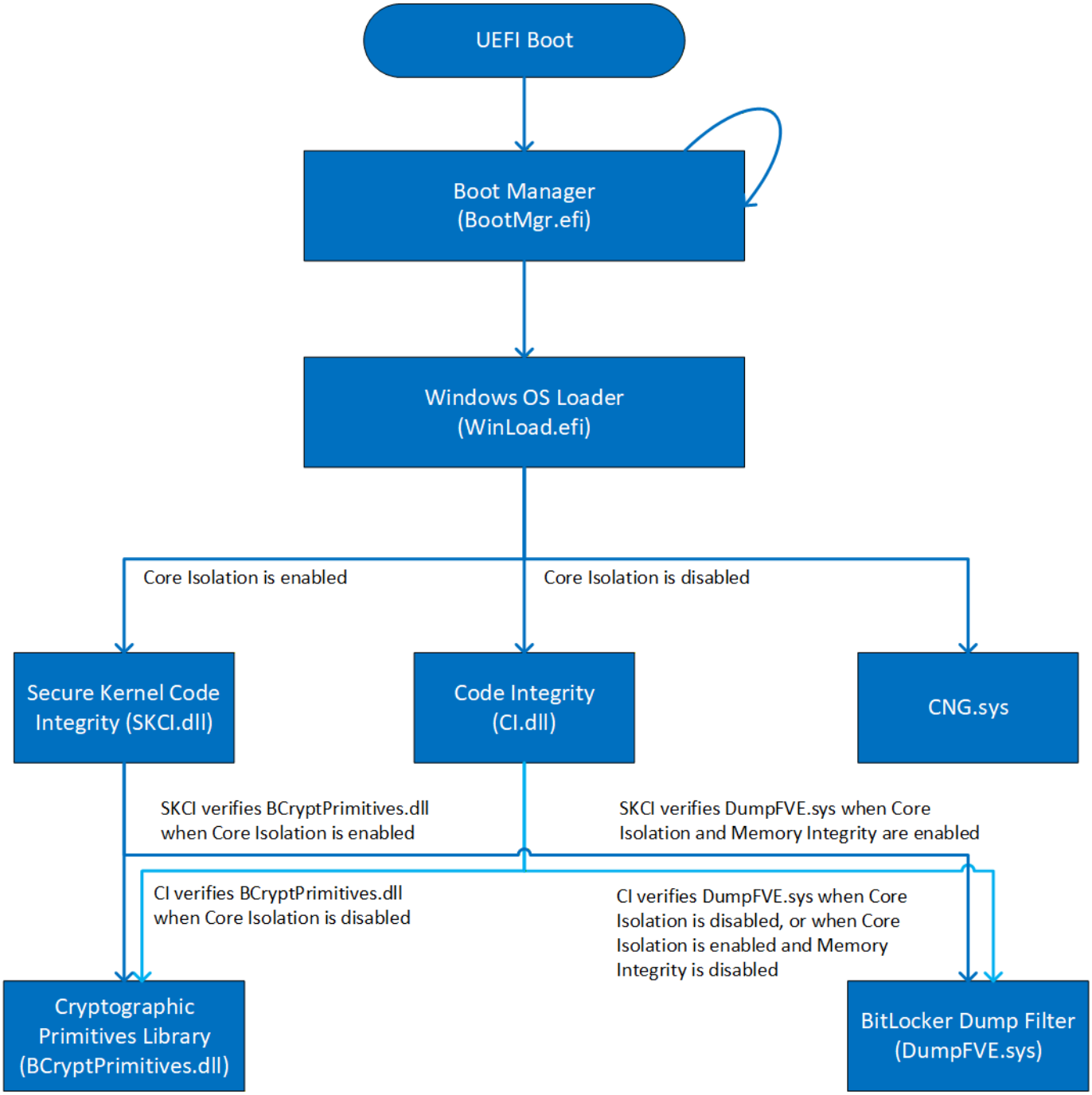


Figure 3: Integrity Chain of Trust

5.2 Initiate on Demand

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

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The modifiable operational environment for the module is the Windows operating system running on a supported hardware platform, as listed in section 2.2 Tested and Vendor Affirmed Module Version and Identification. The Cryptographic Primitives Library is loaded into process memory for a single application.

7 Physical Security

7.1 Mechanisms and Actions Required

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

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

8 Non-Invasive Security

N/A for this module.

9 Sensitive Security Parameters Management

9.1 Storage Areas

The module stores SSPs in the following manner. The module does not directly persist cryptographic keys. The operator may choose to export a cryptographic key, but management of the secure archival of that key is the responsibility of the user.

Storage Area Name	Description	Persistence Type
Hard Disk	Pre-loaded keys for integrity verification are stored on the operating system volume (see: Hard Disk in the block diagram).	Static
RAM	Volatile SSPs used by the module as part of service execution are temporarily stored in the computer's memory (see: RAM in the block diagram).	Dynamic

Table 33: Storage Areas

9.2 SSP Input-Output Methods

Whenever a user application dynamically links with the Cryptographic Primitives Library, the DLL is instantiated and no keys exist within it. The application is responsible for inputting keys into the Cryptographic Primitives Library or using the Cryptographic Primitives Library's functions to generate keys.

Keys may be output from and input into the module via the BCryptExportKey(), BCryptImportKey(), and BCryptImportKeyPair() functions. Symmetric key input and output may also be done by exchanging keys using the recipient's asymmetric public key via the BCryptSecretAgreement() and BCryptDeriveKey() functions. For more information on these functions and how they are accessed, see section 3.2 Additional Information.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input from non-encrypted volatile storage	Outside cryptographic boundary	RAM	Plaintext	Manual	Electronic	
Output to non-encrypted volatile storage	RAM	Outside cryptographic boundary	Plaintext	Manual	Electronic	

Table 34: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Perform zeroization service	Dynamic keys may be zeroized using one of the zeroization functions described in 3.2 Additional Information. All keys are destroyed and their memory location zeroized (replaced with zeroes) when the operator calls BCryptDestroyKey() or BCryptDestroySecret() on that key handle.	The module does not persist SSPs. The SSPs are not retrievable or reusable after being overwritten with zeroes.	The operator calls zeroization functions on an SSP.
Procedural zeroization	Operators may zeroize SSPs in volatile storage by powering off the host General Purpose Computer (GPC).	The module does not persist SSPs. The SSPs are not retrievable or reusable after being overwritten with zeroes.	Operators may power off or reboot the host GPC to zeroize SSPs stored in volatile RAM.

Table 35: SSP Zeroization Methods

9.4 SSPs

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

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES-CTR DRBG Entropy Input	Entropy material for AES_CTR DRBG.	Size: 256-bit - Strength: 256 bits	Entropy Material - CSP	ESV		DRBG1
AES-CTR DRBG key	Entropy material for AES_CTR DRBG.	Size: 256-bit - Strength: 256 bits	Entropy Material - CSP	DRBG1		DRBG1
AES-CTR DRBG Seed	Seed material for AES_CTR DRBG output.	Size: 384-bit - Strength: 384 bits	Entropy Material - CSP	DRBG1		DRBG1
AES-CTR DRBG V	Entropy material for AES_CTR DRBG.	Size: 128-bit - Strength: 128 bits	Entropy Material - CSP	DRBG1		DRBG1
AES-GCM IV	Initialization vector for AES-GCM	Size: 96 bits - Strength: 96 bits	Initialization Vector - CSP	BC2 May be imported to the module depending on user application needs.		BC2
Asymmetric ECDSA Private Keys	Used for digital signature generation.	Size: P-256, P-384, or P-521 - Strength: 128, 192, or 256 bits	Asymmetric Private Key - CSP	AsymKeyPair1 May be imported to the module depending on user application needs.		DigSig-Legacy DigSig1 DigSig2 AsymKeyPair1 AsymKeyPair2

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Asymmetric ECDSA Public Keys	Used for digital signature verification.	Size: P-256, P-384, or P-521 - Strength: 128, 192, or 256 bits	Asymmetric Public Key - PSP	AsymKeyPair1 May be imported to the module depending on user application needs.		DigSig-Legacy DigSig1 DigSig2 AsymKeyPair1 AsymKeyPair2
Asymmetric RSA Private Keys	Used for digital signature generation, key transport.	Size: 2048, 3072, or 4096 bits - Strength: 112, 128, or 150 bits	Asymmetric Private Key - CSP	AsymKeyPair1 May be imported to the module depending on user application needs.		AsymKeyPair1 DigSig1 DigSig2 DigSig3 RSADP1
Asymmetric RSA Public Keys	Used for digital signature verification, key transport.	Size: 2048, 3072, or 4096 bits - Strength: 112, 128, or 150 bits	Asymmetric Public Key - PSP	AsymKeyPair1 May be imported to the module depending on user application needs.		AsymKeyPair1 DigSig-Legacy DigSig1 DigSig2 RSADP1
Derived Key	Symmetric key output of KDFs	Size: 128 or 256 bits, depending on protocol derivation function. - Strength: 128 or 256 bits	Symmetric Key - CSP	KAS-135KDF-IKE KAS-135KDF-TLS KBKDF1 KDA1 PBKDF1		KAS-135KDF-IKE KAS-135KDF-TLS KBKDF1 KDA1 PBKDF1
DH Private Values	Used for DH key establishment.	Size: 2048 or 4096 bits - Strength: 112 or 152 bits	Private Key Pair - CSP	AsymKeyPair1 May be imported to the module depending on user application needs.		KAS-FFC1 KAS-FFC-SSC1

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DH Public Values	Used for DH key establishment.	Size: 2048 or 4096 bits - Strength: 112 or 152 bits	Public Key Pair - PSP	AsymKeyPair1 May be imported to the module depending on user application needs.		KAS-FFC1 KAS-FFC-SSC1
ECDH Private Values	Used for ECDH key establishment.	Size: P-256, P-384, or P-521 - Strength: 128, 192, or 256 bits	Private Key Pair - CSP	AsymKeyPair1 May be imported to the module depending on user application needs.		KAS-ECC1 KAS-ECC-SSC1
ECDH Public Values	Used for ECDH key establishment.	Size: P-256, P-384, or P-521 - Strength: 128, 192, or 256 bits	Public Key Pair - PSP	AsymKeyPair1 May be imported to the module depending on user application needs.		KAS-ECC1 KAS-ECC-SSC1
Embedded X.509 Certificate for BCRYPTPRIMITIVES.DLL (This is not an SSP)	Public key used for RSA PKCS #1 (v1.5) integrity verification of BCRYPTPRIMITIVES.DLL.	Size: 2048-bit - Strength: 112 bits	Asymmetric Public Key (RSA) - Neither	Generated external to the module by the Microsoft Windows build process.		DigSig3
Hash Value for BCRYPTPRIMITIVES.DLL (This is not an SSP)	File hash used to verify the integrity of BCRYPTPRIMITIVES.DLL.	Size: 256 bits - Strength: 128 bits	File Hash - Neither	Generated external to the module by the Microsoft Windows build process.		SHS2

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
HMAC Keys	Used for, hashing and message authentication and key derivation.	Size: 160 (SHA1), 256 (SHA2-256), 384 (SHA2-384), or 512 (SHA2-512) bits - Strength: 160 (SHA-1), 256 (SHA2-256), 384 (SHA2-384), and 512 (SHA2-512) bits	Symmetric Key (HMAC) - CSP	CKG1 May be imported to the module depending on user application needs.		MAC1
Key Derivation Key	Internal key for two-step KDFs.	Size: 256 bits - Strength: 256 bits	Secret Key - CSP	KDA1		KAS-135KDF-IKE KAS-135KDF-TLS KBKDF1
PBKDF Password	PBKDF Password	Size: 8 to 1024 bits - Strength: N/A	Password - CSP			PBKDF1
Symmetric AES Keys	Symmetric keys used for AES encryption / decryption.	Size: 128, 192, or 256 bits - Strength: 128, 192, or 256 bits	Symmetric Key (AES) - CSP	CKG1 May be imported to the module depending on user application needs.		BC1 BC2 CKG1 MAC1 KAS-135KDF-IKE KAS-135KDF-TLS
TLS Pre-Master Secret	Used for key derivation for TLS.	Size: 384-bit - Strength: 384 bits	Secret Key - CSP	Generated externally and imported into the module.		KAS-135KDF-TLS

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Z - Shared Secret Input for IKE KDFs	Shared secret input for IKEv1 and IKEv2 KDFs	Size: 256-2048 bits (KDF-IKEv1 and KDF-IKEv2) - Strength: Between 112 to 256 bits	Secret Key - CSP	Shared secret input is generated externally and input into the module		KAS-135KDF-IKE
Z - Shared Secret Input for KBKDF	Shared secret input for KBKDF	Size: 160-256 bits - Strength 160-256 bits	Secret Key - CSP	Shared secret input is generated externally and input into the module		KBKDF1
Z - Shared Secret Input for KDA-HKDF	Shared secret input for KDA-HKDF	Size: 224-8192 bits - Strength: Between 112 to 256 bits	Secret Key - CSP	Shared secret input is generated externally and input into the module		KDA1
Z - Shared Secret Output for KAS-ECC	Shared secret calculation output for KAS-ECC SP 800-56Ar3 key agreement	Size: P-256, P-384, or P-521 - Strength: 128, 192, or 256 bits	Secret Key - CSP	KAS-ECC-SSC1 KAS-ECC1		KAS-ECC1
Z - Shared Secret Output for KAS-FFC	Shared secret calculation output for KAS-FFC SP 800-56Ar3 key agreement	Size: 2048, 3072, or 4096 bits - Strength: 112, 128, or 150 bits	Secret Key - CSP	KAS-FFC-SSC1 KAS-FFC1		KAS-FFC1

Table 36: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES-CTR DRBG Entropy Input	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	AES-CTR DRBG Seed:Used With AES-CTR DRBG V:Used With AES-CTR DRBG key:Used With
AES-CTR DRBG key		RAM:Plaintext	Temporary during service execution.	Perform zeroization service	AES-CTR DRBG Entropy Input:Used With AES-CTR DRBG Seed:Used With AES-CTR DRBG V:Used With
AES-CTR DRBG Seed	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	AES-CTR DRBG Entropy Input:Used With AES-CTR DRBG V:Used With AES-CTR DRBG key:Used With
AES-CTR DRBG V		RAM:Plaintext	Temporary during service execution.	Perform zeroization service	AES-CTR DRBG Entropy Input:Used With AES-CTR DRBG Seed:Used With AES-CTR DRBG key:Used With
AES-GCM IV	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Asymmetric ECDSA Private Keys	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	Asymmetric ECDSA Public Keys:Paired With
Asymmetric ECDSA Public Keys	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	Asymmetric ECDSA Private Keys:Paired With
Asymmetric RSA Private Keys	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	Asymmetric RSA Public Keys:Paired With
Asymmetric RSA Public Keys	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	Asymmetric RSA Private Keys:Paired With
Derived Key	Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
DH Private Values	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	DH Public Values:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DH Public Values	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	DH Private Values:Paired With
ECDH Private Values	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	ECDH Public Values:Paired With
ECDH Public Values	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	ECDH Private Values:Paired With
Embedded X.509 Certificate for BCRYPTPRIMITIVES.DLL (This is not an SSP)		Hard Disk:Plaintext		Procedural zeroization	
Hash Value for BCRYPTPRIMITIVES.DLL (This is not an SSP)		Hard Disk:Plaintext		Procedural zeroization	
HMAC Keys	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
Key Derivation Key		RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
PBKDF Password	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Symmetric AES Keys	Input from non-encrypted volatile storage Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
TLS Pre-Master Secret		RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
Z - Shared Secret Input for IKE KDFs	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
Z - Shared Secret Input for KBKDF	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
Z - Shared Secret Input for KDA-HKDF	Input from non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
Z - Shared Secret Output for KAS-ECC	Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	
Z - Shared Secret Output for KAS-FFC	Output to non-encrypted volatile storage	RAM:Plaintext	Temporary during service execution.	Perform zeroization service	

Table 37: SSP Table 2

9.5 Transitions

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

- The SHA-1 algorithm will become disallowed for applying cryptographic protection starting January 1, 2031, however, its use for legacy digital signature verification will continue to be allowed.
- RSA with a security strength of less than 128 bits will become deprecated starting January 1, 2031, however, its use for legacy digital signature verification will continue to be allowed.
- FIPS 186-4 has been superseded by FIPS 186-5. This transition began on July 25, 2023, and concluded on February 3, 2024. Although testing for this module was completed against FIPS 186-4, it claims compliance with FIPS 186-5 for RSA signature verification – see Section 2.7.5 FIPS 186-4 and 186-5 for more information.

10 Self-Tests

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

10.1 Pre-Operational Self-Tests

As described in the sections above, the Code Integrity module (EVM) or Secure Kernel Code Integrity module (EVM) checks the integrity of BCRYPTPRIMITIVES.DLL before it is loaded. The algorithms used for the pre-operational integrity check pass their own algorithm self-tests before integrity verification is performed. See section 5.1 Integrity Techniques for details on the EVMs and how the module's integrity is checked.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
RSA SigVer (FIPS186-4) (A4008)	RSA 2048-bit key with SHA2-256 (Windows 11 version 22H2)	Software Integrity	SW/FW Integrity	The Perform Cryptographic Algorithm Self-Tests service runs.	[EVM] Signature verification.
RSA SigVer (FIPS186-4) (A4009)	RSA 2048-bit key with SHA2-256 (Windows Server 2022).	Software Integrity	SW/FW Integrity	The Perform Cryptographic Algorithm Self-Tests service runs.	[EVM] Signature verification.

Table 38: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

The module performs the conditional cryptographic algorithm self-tests (CASTs) for all approved algorithms each time the module is loaded into a process via the default DLL entry point, DllMain, after the pre-operational software integrity tests described above have completed. If any self-test fails, the module does not load and

the Cryptographic Primitives Library DllMain returns an error code. The caller may attempt to reload the module. The following conditional self-tests are included.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
[EVM] Counter DRBG (A4008)	256-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	[EVM] DRBG Known Answer Test (KAT) with instantiate, generate, and reseed tests	Run at every module initialization after the module integrity is verified.
[EVM] Counter DRBG (A4009)	256-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	[EVM] DRBG Known Answer Test (KAT) with instantiate, generate, and reseed tests	Run at every module initialization after the module integrity is verified.
[EVM] RSA SigVer (A4008)	RSA PKCS#1v1.5 with 2048-bit key and SHA2-256 (Windows 11 22H2)	KAT	CAST	The Perform Cryptographic Algorithm Self-Tests service runs.	[EVM] Signature Verification	The self-tests are run prior to pre-operational integrity test during module initialization.
[EVM] RSA SigVer (A4009)	RSA PKCS#1v1.5 with 2048-bit key and SHA2-256 (Windows Server 2022)	KAT	CAST	The Perform Cryptographic Algorithm Self-Tests service runs.	[EVM] Signature Verification	The self-tests are run prior to pre-operational integrity test during module initialization.
[EVM] SHA2-256 (A4008)	256 bits (Windows 11 version 22H2)	KAT	CAST	The Perform Cryptographic Algorithm Self-Tests service runs.	[EVM] Secure hash	The self-tests are run prior to pre-operational integrity test during module initialization.
[EVM] SHA2-256 (A4009)	256 bits (Windows Server 2022)	KAT	CAST	The Perform Cryptographic Algorithm Self-Tests service runs.	[EVM] Secure hash	The self-tests are run prior to pre-operational integrity test during module initialization.
AES-CBC (A4008) - Decrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A4008) - Encrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-CBC (A4009) - Decrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-CBC (A4009) - Encrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-CCM (A4008) - Decrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-CCM (A4008) - Encrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-CCM (A4009) - Decrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-CCM (A4009) - Encrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-CMAC (A4008)	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	AES-CMAC known answer test.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CMAC (A4009)	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	AES-CMAC known answer test.	Run when module is loaded via the default entry point.
AES-ECB (A4008) - Decrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-ECB (A4008) - Encrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-ECB (A4009) - Decrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-ECB (A4009) - Encrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-GCM (A4008) - Decrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-GCM (A4008) - Encrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-GCM (A4009) - Decrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A4009) - Encrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-XTS Testing Revision 2.0 (A4008)	128-bit AES key (Windows 11 22H2)	Fault Detection	CAST	Crypto functions execute and status is returned via interface.	Key equivalence test in compliance with FIPS 140-3 IG C.I.	When BCRYPT_ENABLE_INCOMPATIBLE_FIPS_CHECKS flag (required by policy) is used with BCryptGenerateSymmetricKey.
AES-XTS Testing Revision 2.0 (A4008) - Decrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-XTS Testing Revision 2.0 (A4008) - Encrypt	128-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.
AES-XTS Testing Revision 2.0 (A4009)	128-bit AES key (Windows Server 2022)	Fault Detection	CAST	Crypto functions execute and status is returned via interface.	Key equivalence test in compliance with FIPS 140-3 IG C.I.	When BCRYPT_ENABLE_INCOMPATIBLE_FIPS_CHECKS flag (required by policy) is used with BCryptGenerateSymmetricKey.
AES-XTS Testing Revision 2.0 (A4009) - Decrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Decrypt KAT.	Run when module is loaded via the default entry point.
AES-XTS Testing Revision 2.0 (A4009) - Encrypt	128-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Encrypt KAT.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Counter DRBG (A4008)	256-bit AES key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	DRBG Known Answer Test (KAT) with instantiate, generate, and reseed tests	Run at every module initialization after the module integrity is verified.
Counter DRBG (A4009)	256-bit AES key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	DRBG Known Answer Test (KAT) with instantiate, generate, and reseed tests	Run at every module initialization after the module integrity is verified.
DSA KeyGen (FIPS186-4) (A4008)	2048-bit (Windows 11 22H2)	PCT	PCT	Crypto functions execute and status is returned via interface.	KeyGen pairwise consistency test.	Run at each invocation of DSA KeyGen.
DSA KeyGen (FIPS186-4) (A4009)	2048-bit (Windows Server 2022)	PCT	PCT	Crypto functions execute and status is returned via interface.	KeyGen pairwise consistency test.	Run at each invocation of DSA KeyGen.
ECDSA KeyGen (FIPS186-4) (A4008)	P-256 Curve (Windows 11 22H2)	PCT	PCT	Crypto functions execute and status is returned via interface.	KeyGen pairwise consistency test.	Run at each invocation of ECDSA KeyGen (on key generation and key import).
ECDSA KeyGen (FIPS186-4) (A4009)	P-256 Curve (Windows Server 2022)	PCT	PCT	Crypto functions execute and status is returned via interface.	KeyGen pairwise consistency test.	Run at each invocation of ECDSA KeyGen (on key generation and key import).
ECDSA KeyVer (FIPS186-4) (A4008)	P-256 Curve (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KeyVer known answer test.	Run before the first ECDSA key verification.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA KeyVer (FIPS186-4) (A4009)	P-256 Curve (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KeyVer known answer test.	Run before the first ECDSA key verification.
ECDSA SigGen (FIPS186-4) (A4008)	P-256 curve (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Sign known answer test.	Run before the first ECDSA signature generation
ECDSA SigGen (FIPS186-4) (A4009)	P-256 curve (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Sign known answer test.	Run before the first ECDSA signature generation
ECDSA SigVer (FIPS 186-4) (A4008)	P-256 curve (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Verify known answer test.	Run before the first ECDSA signature verification
ECDSA SigVer (FIPS 186-4) (A4009)	P-256 curve (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Verify known answer test.	Run before the first ECDSA signature verification
HMAC-SHA-1 (A4008)	112-bit HMAC key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Message authentication.	Run when module is loaded via the default entry point.
HMAC-SHA-1 (A4009)	112-bit HMAC key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Message authentication.	Run when module is loaded via the default entry point.
HMAC-SHA2-256 (A4008)	256-bit HMAC key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Message authentication.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A4009)	256-bit HMAC key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Message authentication.	Run when module is loaded via the default entry point.
HMAC-SHA2-512 (A4008)	512-bit HMAC key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Message authentication.	Run when module is loaded via the default entry point.
HMAC-SHA2-512 (A4009)	512-bit HMAC key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Message authentication.	Run when module is loaded via the default entry point.
Intel-based Entropy Source (E189)	256-bit entropy input (Windows Server 2022)	Fault detection	CAST	The entropy source is instantiated and a status is returned via the interface.	Start-up and continuous noise source health tests and start-up logic integrity self-test.	Run at start-up and continuously.
Intel-based Entropy Source (E216)	256-bit entropy input (Windows 11 22H2)	Fault detection	CAST	The entropy source is instantiated and a status is returned via the interface.	Start-up and continuous noise source health tests and start-up logic integrity self-test.	Run at start-up and continuously.
KAS-ECC SP800-56Ar3 (A4008)	P-256 curve (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secret agreement known answer test.	Run when module is loaded via the default entry point.
KAS-ECC SP800-56Ar3 (A4008) - PCT	P-256 curve (Windows 11 22H2)	PCT	PCT	Crypto functions execute and status is returned via interface.	Perform ECDH assurances (including pairwise consistency tests) according to NIST SP 800-56Arev3.	Run at each invocation of KAS-ECC.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-ECC SP800-56Ar3 (A4009)	P-256 curve (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secret agreement known answer test.	Run when module is loaded via the default entry point.
KAS-ECC SP800-56Ar3 (A4009) - PCT	P-256 curve (Windows Server 2022)	PCT	PCT	Crypto functions execute and status is returned via interface.	Perform ECDH assurances (including pairwise consistency tests) according to NIST SP 800-56Arev3.	Run at each invocation of KAS-ECC.
KAS-FFC SP800-56Ar3 (A4008)	2048-bit key (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secret agreement known answer test.	Run when module is loaded via the default entry point.
KAS-FFC SP800-56Ar3 (A4008) - PCT	2048-bit key (Windows 11 22H2)	PCT	PCT	Crypto functions execute and status is returned via interface.	Perform DH assurances (including pairwise consistency tests) according to NIST SP 800-56Arev3.	Run when module is loaded via the default entry point.
KAS-FFC SP800-56Ar3 (A4009)	2048-bit key (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secret agreement known answer test.	Run when module is loaded via the default entry point.
KAS-FFC SP800-56Ar3 (A4009) - PCT	2048-bit key (Windows Server 2022)	PCT	PCT	Crypto functions execute and status is returned via interface.	Perform DH assurances (including pairwise consistency tests) according to NIST SP 800-56Arev3.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDA HKDF SP800-56Cr2 (A4008)	Derived key material (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDA HKDF SP800-56Cr2 (A4009)	Derived key material (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDF SP 800-108 (A3763)	Derived key material (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDF SP 800-108 (A3764)	Derived key material (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDF TLS (A4008)	Derived key material (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDF TLS (A4009)	Derived key material (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDF-IKEv1 (A4008)	Derived key material (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDF-IKEv1 (A4009)	Derived key material (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDF-IKEv2 (A4008)	Derived key material (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
KDF-IKEv2 (A4009)	Derived key material (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
NIST SP 800-132 PBKDF (A4008)	Derived key material (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
NIST SP 800-132 PBKDF (A4009)	Derived key material (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
RSA KeyGen (FIPS 186-4) (A4008)	N/A (Windows 11 22H2)	PCT	PCT	Crypto functions execute and status is returned via interface.	KeyGen pairwise consistency test.	Run at each invocation of RSA KeyGen (on key generation and key import).
RSA KeyGen (FIPS 186-4) (A4009)	N/A (Windows Server 2022)	PCT	PCT	Crypto functions execute and status is returned via interface.	KeyGen pairwise consistency test.	Run at each invocation of RSA KeyGen (on key generation and key import).
RSA SigGen (FIPS 186-4) (A4008)	RSA_SHA256_PKCS1 signature (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Sign known answer test.	Run before first RSA signature generation.
RSA SigGen (FIPS 186-4) (A4009)	RSA_SHA256_PKCS1 signature (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Sign known answer test.	Run before first RSA signature generation.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA SigVer (FIPS 186-4) (A4008)	RSA_SHA256_PKCS1 signature (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Verify known answer test.	Run before first RSA signature verification.
RSA SigVer (FIPS 186-4) (A4009)	RSA_SHA256_PKCS1 signature (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Verify known answer test.	Run before first RSA signature verification.
SHA-1 (A4008)	SHA-1 (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.
SHA-1 (A4009)	SHA-1 (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.
SHA2-256 (A4008)	SHA2-256 (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.
SHA2-256 (A4009)	SHA2-256 (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.
SHA2-384 (A4008)	SHA2-384 (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.
SHA2-384 (A4009)	SHA2-384 (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA2-512 (A4008)	SHA2-512 (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.
SHA2-512 (A4009)	SHA2-512 (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	Secure hash.	Run when module is loaded via the default entry point.
TLS v1.2 KDF RFC7627 (A4008)	Derived key material (Windows 11 22H2)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.
TLS v1.2 KDF RFC7627 (A4009)	Derived key material (Windows Server 2022)	KAT	CAST	Crypto functions execute and status is returned via interface.	KAT with derived key material.	Run when module is loaded via the default entry point.

Table 39: Conditional Self-Tests

10.3 Periodic Self-Test Information

The tables below present the periodic self-test information for the module. The set of self-tests presented in tables 25 and 26 below is identical to the set of self-tests presented in tables 23 and 24 above. The Code Integrity module (EVM) or Secure Kernel code integrity module (EVM) checks the integrity of BCRYPTPRIMITIVES.DLL before it is loaded. See section 5.1 Integrity Techniques for details on the EVMs and how the module's integrity is checked.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-4) (A4008)	Software Integrity	SW/FW Integrity	Pre-operational integrity test is run at every module initialization before the CASTs (Windows 11 version 22H2).	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-4) (A4009)	Software Integrity	SW/FW Integrity	Pre-operational integrity test is run at every module initialization before the CASTs (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).

Table 40: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
[EVM] Counter DRBG (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer)
[EVM] Counter DRBG (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer)
[EVM] RSA SigVer (A4008)	KAT	CAST	Conditional CAST is run at every module initialization prior to module's integrity check (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer)
[EVM] RSA SigVer (A4009)	KAT	CAST	Conditional CAST is run at every module initialization prior to module's integrity check (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer)
[EVM] SHA2-256 (A4008)	KAT	CAST	Conditional CAST is run at every module initialization prior to module's integrity check (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer)
[EVM] SHA2-256 (A4009)	KAT	CAST	Conditional CAST is run at every module initialization prior to module's integrity check (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer)

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A4008) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-CBC (A4008) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-CBC (A4009) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-CBC (A4009) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-CCM (A4008) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-CCM (A4008) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-CCM (A4009) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-CCM (A4009) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CMAC (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-CMAC (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-ECB (A4008) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-ECB (A4008) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-ECB (A4009) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-ECB (A4009) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-GCM (A4008) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-GCM (A4008) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM (A4009) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-GCM (A4009) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-XTS Testing Revision 2.0 (A4008)	Fault Detection	CAST	When BCRYPT_ENABLE_ INCOMPATIBLE_FIPS_ CHECKS flag (required by policy) is used with BCryptGenerateSymmetricKey (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-XTS Testing Revision 2.0 (A4008) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-XTS Testing Revision 2.0 (A4008) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
AES-XTS Testing Revision 2.0 (A4009)	Fault Detection	CAST	When BCRYPT_ENABLE_ INCOMPATIBLE_FIPS_ CHECKS flag (required by policy) is used with BCryptGenerateSymmetricKey (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
AES-XTS Testing Revision 2.0 (A4009) - Decrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
AES-XTS Testing Revision 2.0 (A4009) - Encrypt	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
Counter DRBG (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
Counter DRBG (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
DSA KeyGen (FIPS186-4) (A4008)	PCT	PCT	Run at each invocation of DSA KeyGen (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
DSA KeyGen (FIPS186-4) (A4009)	PCT	PCT	Run at each invocation of DSA KeyGen (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
ECDSA KeyGen (FIPS186-4) (A4008)	PCT	PCT	Run at each invocation of ECDSA KeyGen (on key generation and key import; Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
ECDSA KeyGen (FIPS186-4) (A4009)	PCT	PCT	Run at each invocation of ECDSA KeyGen (on key generation and key import; Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
ECDSA KeyVer (FIPS186-4) (A4008)	KAT	CAST	Run before the first ECDSA key verification. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
ECDSA KeyVer (FIPS186-4) (A4009)	KAT	CAST	Run before the first ECDSA key verification.(Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigGen (FIPS186-4) (A4008)	KAT	CAST	Run before the first ECDSA signature generation (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
ECDSA SigGen (FIPS186-4) (A4009)	KAT	CAST	Run before the first ECDSA signature generation (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
ECDSA SigVer (FIPS 186-4) (A4008)	KAT	CAST	Run before the first ECDSA signature verification (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
ECDSA SigVer (FIPS 186-4) (A4009)	KAT	CAST	Run before the first ECDSA signature verification (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
HMAC-SHA-1 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
HMAC-SHA-1 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
HMAC-SHA2-256 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
HMAC-SHA2-256 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-512 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
HMAC-SHA2-512 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
Intel-based Entropy Source (E189)	Fault detection	CAST	Continuously checks the health of entropy output (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
Intel-based Entropy Source (E216)	Fault detection	CAST	Continuously checks the health of entropy output (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KAS-ECC SP800-56Ar3 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KAS-ECC SP800-56Ar3 (A4008) - PCT	PCT	PCT	Run at each invocation of KAS-ECC (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KAS-ECC SP800-56Ar3 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
KAS-ECC SP800-56Ar3 (A4009) - PCT	PCT	PCT	Run at each invocation of KAS-ECC (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-FFC SP800-56Ar3 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KAS-FFC SP800-56Ar3 (A4008) - PCT	PCT	PCT	Run at each invocation of KAS-ECC (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KAS-FFC SP800-56Ar3 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
KAS-FFC SP800-56Ar3 (A4009) - PCT	PCT	PCT	Run at each invocation of KAS-ECC (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
KDA HKDF SP800-56Cr2 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
KDA HKDF SP800-56Cr2 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
KDF SP 800-108 (A3763)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KDF SP 800-108 (A3764)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KDF TLS (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KDF TLS (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
KDF-IKEv1 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KDF-IKEv1 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
KDF-IKEv2 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows 11 22H2).	Manual on-demand (operator initiated by rebooting the computer).
KDF-IKEv2 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified (Windows Server 2022).	Manual on-demand (operator initiated by rebooting the computer).
NIST SP 800-132 PBKDF (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
NIST SP 800-132 PBKDF (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA KeyGen (FIPS 186-4) (A4008)	PCT	PCT	Run at each invocation of RSA KeyGen (on key generation and key import). (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
RSA KeyGen (FIPS 186-4) (A4009)	PCT	PCT	Run at each invocation of RSA KeyGen (on key generation and key import). (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
RSA SigGen (FIPS 186-4) (A4008)	KAT	CAST	Run before first RSA signature generation. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
RSA SigGen (FIPS 186-4) (A4009)	KAT	CAST	Run before first RSA signature generation. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
RSA SigVer (FIPS 186-4) (A4008)	KAT	CAST	Run before first RSA signature verification. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
RSA SigVer (FIPS 186-4) (A4009)	KAT	CAST	Run before first RSA signature verification. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
SHA-1 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
SHA-1 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA2-256 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
SHA2-256 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
SHA2-384 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
SHA2-384 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
SHA2-512 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
SHA2-512 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).
TLS v1.2 KDF RFC7627 (A4008)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows 11 22H2)	Manual on-demand (operator initiated by rebooting the computer).
TLS v1.2 KDF RFC7627 (A4009)	KAT	CAST	Run when module is loaded via the default entry point, after the module integrity is verified. (Windows Server 2022)	Manual on-demand (operator initiated by rebooting the computer).

Table 41: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Boot Failure	Boot failure.	Occurs if the module integrity test fails or if one of the conditional self-tests fails.	Restart the computer.	Boot failure blue screen error message.

Table 42: Error States

10.5 Operator Initiation of Self-Tests

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

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

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

For Windows Updates, the client only accepts binaries signed by Microsoft certificates. The Windows Update client only accepts content whose SHA2 hash matches the SHA2 hash specified in the metadata. All metadata communication is done over a Transport Layer Security (TLS) port. Using TLS ensures that the client is communicating with the real server and so prevents a malicious TLS server from communicating to the TLS client. The version and digital signature of any new cryptographic module must be verified to match the version that was validated. See section 11.2 Administrator Guidance for details on how to do this.

Module initialization occurs automatically as part of the Windows boot process. The finite state model diagram below visualizes the initialization process along with other module states. Every state of the module can transition to the power-off state through power-cycle/rebooting the host machine.

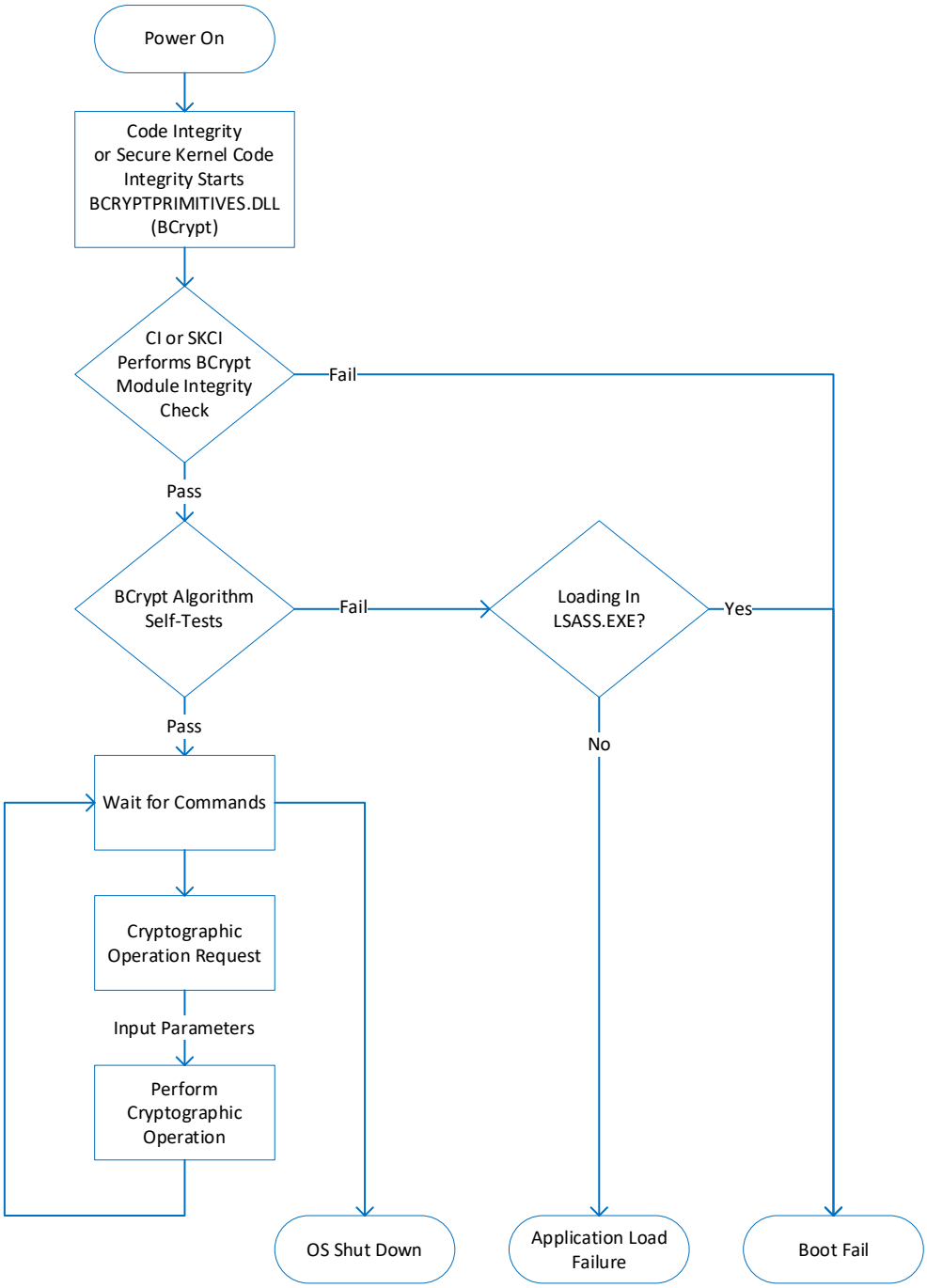


Figure 4: Finite State Model

11.2 Administrator Guidance

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

11.2.1 Verifying the Installed Windows Version

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

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

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

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

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

11.2.2 Verifying the Cryptographic Module Version and its Signature

To confirm the version number or digital signature of the module, locate the module binary or binaries named in 2.2 Tested and Vendor Affirmed Module Version and Identification in their default installation location. The list below identifies the default install locations for the Windows cryptographic module binaries for a system where Windows has been installed on the C: drive.

- BCRYPTPRIMITIVES.DLL - C:\Windows\System32\ and C:\Windows\SysWOW64\

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

- Using Windows Explorer:
  - Open Windows Explorer and navigate to the folder where the binary is installed, referencing the list above for the correct location.
  - Find the file in the folder and **right click** on the file's icon.
  - Select **Properties** from the context menu.
  - Select the **Details** tab.
  - Compare the version number in the **File version** field against the version identified in 2.2 Tested and Vendor Affirmed Module Version and Identification.

- Using PowerShell:
  - Open a **PowerShell** window.
  - Use the **Get-ItemProperty cmdlet** together with the path of the cryptographic module binary identified above, formatting the output as a list. For example, if the binary path is `C:\Windows\System32\bcryptprimitives.dll`, the PowerShell command is:

```
Get-ItemProperty -Path
"C:\Windows\System32\bcryptprimitives.dll" | Format-List
```

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

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

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

- Using PowerShell:
  - Open a **PowerShell** window.
  - Use the **Get-AuthenticodeSignature cmdlet** together with the path the path of the cryptographic module binary identified at the beginning of this section. For example, if the binary path is `C:\Windows\System32\bcryptprimitives.dll`, the PowerShell command is:

```
Get-AuthenticodeSignature -FilePath
"C:\Windows\System32\bcryptprimitives.dll"
```

  - The cmdlet will return output that summarizes signature details. If the signature is valid, the **Status** field will show "`Valid`" and the **StatusMessage** field will show "`Signature verified.`"

- Full documentation for the Get-AuthenticodeSignature cmdlet may be found at: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-authenticodesignature.

11.3 Non-Administrator Guidance

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

11.4 Design and Rules

The module is a multi-chip standalone software-hybrid module that operates in its approved mode during normal operation of the computer and Windows operating system and provides cryptographic services within the Operational Environments listed in section 2.2 Tested and Vendor Affirmed Module Version and Identification. The other sections of this Security Policy provide additional details on the design of the module and its rules of operation.

12 Mitigation of Other Attacks

12.1 Attack List

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

Algorithm	Protected Against	Mitigation	Constraints / Guidance
SHA1	Timing Analysis Attack	Constant time implementation.	None
	Cache Attack	Memory access pattern is independent of any confidential data.	None
SHA2	Timing Analysis Attack	Constant time implementation.	None
	Cache Attack	Memory access pattern is independent of any confidential data.	None
AES	Timing Analysis Attack	Constant time implementation.	None
	Cache Attack	Memory access pattern is independent of any confidential data.	None
		Module prevents observation of memory for AES rounds.	Effective only for computers that use the AES-NI instruction set.

Table 43: Mitigation of Other Attacks

13 Standards References

- FIPS 140-3, Security Requirements for Cryptographic Modules, https://csrc.nist.gov/publications/detail/fips/140/3/final
- FIPS 180-4, Secure Hash Standard (SHS), https://csrc.nist.gov/publications/detail/fips/180/4/final

- FIPS 186-4, Digital Signature Standard (DSS), https://csrc.nist.gov/publications/detail/fips/186/4/final
- FIPS 186-5, Digital Signature Standard (DSS), https://csrc.nist.gov/pubs/fips/186-5/final
- FIPS 197, Advanced Encryption Standard (AES), https://csrc.nist.gov/publications/detail/fips/197/final
- FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC), https://csrc.nist.gov/publications/detail/fips/198/1/final
- FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, https://csrc.nist.gov/publications/detail/fips/202/final
- NIST SP 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques, https://csrc.nist.gov/publications/detail/sp/800-38a/final
- NIST SP 800-38B, Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication, https://csrc.nist.gov/publications/detail/sp/800-38b/final
- NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality, https://csrc.nist.gov/publications/detail/sp/800-38c/final
- NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, https://csrc.nist.gov/publications/detail/sp/800-38d/final
- NIST SP 800-38E, Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices, https://csrc.nist.gov/publications/detail/sp/800-38e/final
- NIST SP 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, https://csrc.nist.gov/publications/detail/sp/800-38f/final
- NIST SP 800-56A Rev. 3, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final
- NIST SP 800-56B Rev. 2, Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography, https://csrc.nist.gov/publications/detail/sp/800-56b/rev-2/final
- NIST SP 800-90A Rev. 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final
- NIST SP 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation, https://csrc.nist.gov/publications/detail/sp/800-90b/final
- NIST SP 800-108 Rev. 1, Recommendation for Key Derivation Using Pseudorandom Functions, https://csrc.nist.gov/publications/detail/sp/800-108/rev-1/final
- NIST SP 800-131A Rev. 2, Transitioning the Use of Cryptographic Algorithms and Key Lengths, https://csrc.nist.gov/publications/detail/sp/800-131a/rev-2/final
- NIST SP 800-132, Recommendation for Password-Based Key Derivation: Part 1: Storage Applications, https://csrc.nist.gov/publications/detail/sp/800-132/final
- NIST SP 800-133 Rev. 2, Recommendation for Cryptographic Key Generation, https://csrc.nist.gov/publications/detail/sp/800-133/rev-2/final
- NIST SP 800-135 Rev. 1, Recommendation for Existing Application-Specific Key Derivation Functions, https://csrc.nist.gov/publications/detail/sp/800-135/rev-1/final