



Microsoft Windows

FIPS 140 Validation

Microsoft Windows Server 2019

Microsoft Azure Stack Edge

Microsoft Azure Stack Hub

Microsoft Azure Stack Edge Rugged

Non-Proprietary

Security Policy Document

Document Information	
Version Number	1.3
Updated On	January 23, 2024

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

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

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

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

© 2024 Microsoft Corporation. All rights reserved.

Microsoft, Windows, the Windows logo, Windows Server, and BitLocker are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

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

Version History

Version	Date	Summary of changes
1.0	November 4, 2020	Draft sent to NIST CMVP
1.1	November 3, 2022	Updates in response to NIST feedback
1.2	December 12, 2023	Updates to bounded module certificates
1.3	January 23, 2024	Updates in response to NIST feedback

TABLE OF CONTENTS

<u>SECURITY POLICY DOCUMENT</u>	1
<u>VERSION HISTORY</u>	3
<u>1 INTRODUCTION</u>	7
1.1 LIST OF CRYPTOGRAPHIC MODULE BINARY EXECUTABLES.....	7
1.2 VALIDATED PLATFORMS.....	7
<u>2 CRYPTOGRAPHIC MODULE SPECIFICATION</u>	8
2.1 CRYPTOGRAPHIC BOUNDARY	9
2.2 FIPS 140-2 APPROVED ALGORITHMS	9
2.3 NON-APPROVED ALGORITHMS.....	10
2.4 FIPS 140-2 APPROVED AND ALLOWED ALGORITHMS FROM BOUNDED MODULES.....	11
2.5 CRYPTOGRAPHIC BYPASS	11
2.6 HARDWARE COMPONENTS OF THE CRYPTOGRAPHIC MODULE.....	11
<u>3 CRYPTOGRAPHIC MODULE PORTS AND INTERFACES</u>	13
3.1 VTPM EXPORT FUNCTIONS	13
3.1.1 VTPMCOLDINITWITHPERSISTENTSTATE	13
3.1.2 VTPMWARMINIT	13
3.1.3 VTPMSHUTDOWN	13
3.1.4 VTPMEXECUTECOMMAND.....	13
3.1.5 VTPMSETCANCELFLAG.....	17
3.1.6 VTPMGETRUNTIMESTATE	17
3.2 CONTROL INPUT INTERFACE	17
3.3 STATUS OUTPUT INTERFACE	17
3.4 DATA INPUT INTERFACE	17
3.5 DATA OUTPUT INTERFACE.....	17
3.6 NON-SECURITY RELEVANT INTERFACES.....	18
<u>4 ROLES, SERVICES AND AUTHENTICATION</u>	18
4.1 ROLES	18
4.2 SERVICES.....	18
4.2.1 MAPPING OF SERVICES, ALGORITHMS, AND CRITICAL SECURITY PARAMETERS.....	18

4.3	AUTHENTICATION.....	22
5	<u>FINITE STATE MODEL</u>	<u>22</u>
5.1	STARTUP.....	23
5.2	COMMAND PROCESSING.....	24
6	<u>OPERATIONAL ENVIRONMENT</u>	<u>25</u>
6.1	CRYPTOGRAPHIC ISOLATION	25
6.2	INTEGRITY CHAIN OF TRUST.....	25
7	<u>CRYPTOGRAPHIC KEY MANAGEMENT.....</u>	<u>27</u>
7.1	ACCESS CONTROL POLICY	28
7.2	KEY MATERIAL	31
7.3	KEY GENERATION.....	31
7.4	KEY AGREEMENT AND DERIVATION	31
7.5	KEY ENTRY AND OUTPUT	32
7.6	KEY STORAGE.....	32
7.7	KEY ARCHIVAL.....	32
7.8	KEY ZEROIZATION.....	32
8	<u>SELF-TESTS.....</u>	<u>32</u>
8.1	POWER-ON SELF-TESTS.....	32
8.2	CONDITIONAL SELF-TESTS	33
9	<u>DESIGN ASSURANCE.....</u>	<u>33</u>
10	<u>MITIGATION OF OTHER ATTACKS</u>	<u>34</u>
11	<u>SECURITY LEVELS</u>	<u>35</u>
12	<u>ADDITIONAL DETAILS</u>	<u>35</u>
13	<u>APPENDIX A – HOW TO VERIFY WINDOWS VERSIONS AND DIGITAL SIGNATURES</u>	<u>36</u>
13.1	HOW TO VERIFY WINDOWS VERSIONS	36

13.2 HOW TO VERIFY WINDOWS DIGITAL SIGNATURES.....36

1 Introduction

The Virtual Trusted Platform Module (Virtual TPM or VTPM) is a dynamically linked library, TPMEngUM.dll, that provides TPM 2.0 cryptographic services to virtual machines that are running in guest partitions on the host Windows operating system. For the purpose of this validation, the Virtual Trusted Platform Module is classified as a Software-Hybrid cryptographic module because the validated platforms all implement the AES-NI instruction set.

The only requirements for VTPM services are the requirements needed to configure and run Hypervisor as described [here](#).

1.1 List of Cryptographic Module Binary Executables

The Virtual Trusted Platform Module (VTPM) contains the following binaries:

- TPMEngUM.dll

The Windows builds covered by this validation are:

- Windows Server 2019 build 10.0.17763.10021 and 10.0.17763.10127

1.2 Validated Platforms


The editions covered by this validation are:

- Windows Server 2019 Datacenter Core

The Virtual Trusted Platform Module components listed in Section 1.1 were validated using the combination of computers and Windows operating system editions specified in the table below.

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

Table 1 Validated Platforms

Computer	Windows Server 2019 Datacenter Core	Processor Image
Microsoft Azure Stack Edge - Dell XR2 - Intel Xeon Silver 4114	√	 wikichip.org

Microsoft Azure Stack Hub - Dell PowerEdge R640 - Intel Xeon Gold 6230	<p style="text-align: center;">√</p>	 <p style="text-align: center;">wikichip.org</p>
Microsoft Azure Stack Hub - Dell PowerEdge R840 - Intel Xeon Platinum 8260	<p style="text-align: center;">√</p>	 <p style="text-align: center;">wikichip.org</p>
Microsoft Azure Stack Edge Rugged - Rugged Mobile Appliance – Intel Xeon D-1559	<p style="text-align: center;">√</p>	 <p style="text-align: center;">wikichip.org</p>

2 Cryptographic Module Specification

Virtual Trusted Platform Module is a multi-chip standalone module that operates in FIPS-approved mode during normal operation of the computer and Windows operating system. See [Self-Tests](#) for a description of how self-tests are implemented.

The following configurations and modes of operation will cause Virtual Trusted Platform Module to operate in a non-approved mode of operation:

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

2.1 Cryptographic Boundary

The software-hybrid cryptographic boundary for Virtual TPM consists of disjoint software and hardware components within the same physical boundary of the host platform. The software components are defined as the binary TPMEngUM.dll, and the hardware components are the CPUs running on each host platform.

2.2 FIPS 140-2 Approved Algorithms

VTPM implements the following FIPS-140-2 Approved algorithms.¹

Table 2 Approved Algorithms

Algorithm	Windows Server 2019 build 10.0.17763.10021	Windows Server 2019 build 10.0.17763.10127
FIPS 186-4 ECDSA with NIST Curves P-256 and P-384 (Key Pair Generation)	#C1587	#C2053
FIPS 186-4 ECDSA with NIST Curves P-256 and P-384 (Public Key Validation, Signature Generation, Signature Verification)	#C1585	#C2051
FIPS 186-4 RSA PKCS#1 (v1.5) and RSASSA-PSS digital signature generation and verification with 1024², 2048, and 3072 moduli; supporting SHA-1, SHA-256, and SHA-384	#C1587	#C2053
FIPS 186-4 RSA key-pair generation with 2048 and 3072 moduli	#C1585	#C2051
FIPS 180-4 SHS SHA-1, SHA-256, and SHA-384	#C1577	#C2044
FIPS 197 AES-128, AES-192, and AES-256 encryption and decryption in ECB mode	#C1577	#C2044
FIPS 197 AES-128, AES-192, and AES-256 encryption and decryption in CBC, CFB128, OFB and CTR modes	#C1587	#C2053
FIPS PUB 198-1 HMAC-SHA-1³, HMAC-SHA-256, and HMAC-SHA-384	#C1587	#C2053

¹ This module may not use some of the capabilities described in each CAVP certificate.

² RSA 1024 is only applicable for signature verification.

³ For HMAC, only key sizes that are \geq 112 bits in length are used by the module in FIPS mode.

SP 800-56A KAS ECC with parameter EC (P-256 with SHA-256) and ED (P-384 with SHA-384)	#C1587	#C2053
SP 800-90A CTR DRBG (AES-256)	#C1587	#C2053
SP 800-108 KDF with HMAC (SHA-1, SHA-256, SHA-384)	#C1587	#C2053
SP 800-56B RSADP component (CVL)	#C1585	#C2051
SP 800-133r2 (sections 5.1,5.2, 6.1, and 6.2) Cryptographic Key Generation	Vendor affirmed	Vendor affirmed
SP 800-56B (Key Transport using RSA-OAEP, RSA Key Generation)	Vendor affirmed	Vendor affirmed

2.3 Non-Approved Algorithms

VTPM implements the following non-approved algorithms.

Non-Approved algorithms allowed in the Approved mode of operation:

- SHA-1 hash, which is disallowed for use in digital signature generation. It can be used for legacy digital signature verification. Its use is Acceptable for non-digital signature generation applications.

Non-Approved algorithms disallowed in the Approved mode of operation:

- RSA 1024-bits for digital signature generation, which is disallowed.
- ECDAA for object creation and approved actions on keys that are non-FIPS compliant.
- XOR obfuscation used as a hash-based stream cipher.
- MGF1 RSAES_OAEP mask generation.
- EC Schnorr for signing and verifying signatures that are non-FIPS compliant.
- AES CFB128 for key wrapping during key import⁴

As a matter of security policy, non-FIPS Approved algorithms shall not be used for any form of data protection while in FIPS mode.

⁴ Disallowed since 2017 per FIPS 140-2 IG D.9. Using this algorithm for key import qualifies as 'obfuscation' per FIPS 140-2 IG 1.23 and is considered equivalent to plaintext.

2.4 FIPS 140-2 Approved and Allowed Algorithms from Bounded Modules

A bounded module is a FIPS 140 module which provides cryptographic functionality that is relied on by a downstream module. As described in the [Integrity Chain of Trust](#) section, the Virtual TPM depends on the following modules and algorithms:

When Memory Integrity, called HVCI in previous Windows Server versions, is not enabled, Code Integrity (module certificate [#4602](#)) provides:

- CAVP certificates #C1577 and #C2044 for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificates #C1577 and #C2044 for FIPS 180-4 SHS SHA-256

When Memory Integrity, called HVCI in previous Windows Server versions, is enabled, Secure Kernel Code Integrity (module certificate [#4640](#)) provides:

- CAVP certificates #C1577 and #C2044 for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificates #C1577 and #C2044 for FIPS 180-4 SHS SHA-256

The Virtual TPM depends on the Kernel Mode Cryptographic Primitives Library (module certificate [#4670](#)) for:

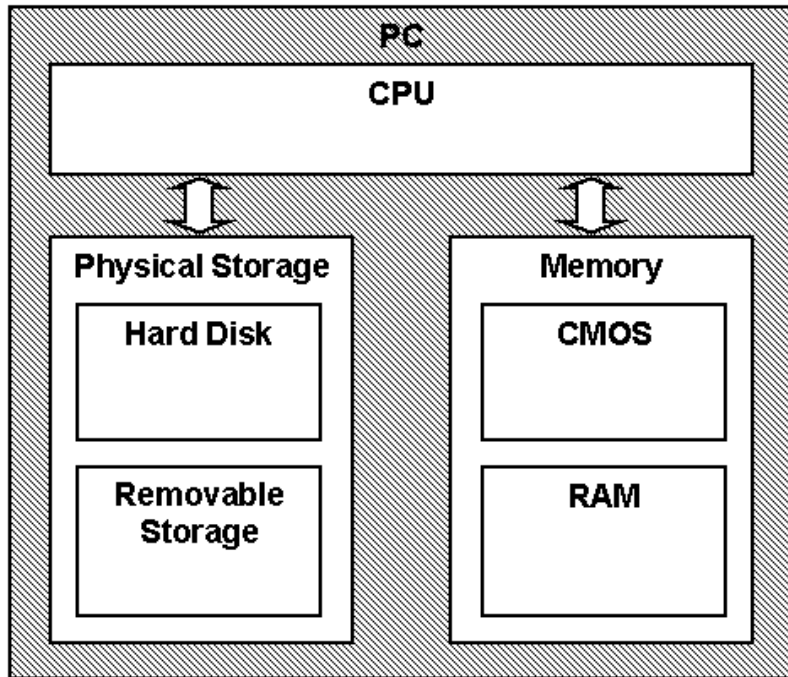
- AES-CTR DRBG (CAVP certificates #C1577 and #C2044) to provide the entropy input for the Virtual TPM's DRBG instance
- AES (CAVP certificates #C1577 and #C2044) in CTR mode as a prerequisite for the DRBG of the bounded module
- A non-Approved but Allowed Non-deterministic Random Number Generator (NDRNG) to seed the DRBG of the bounded module

2.5 Cryptographic Bypass

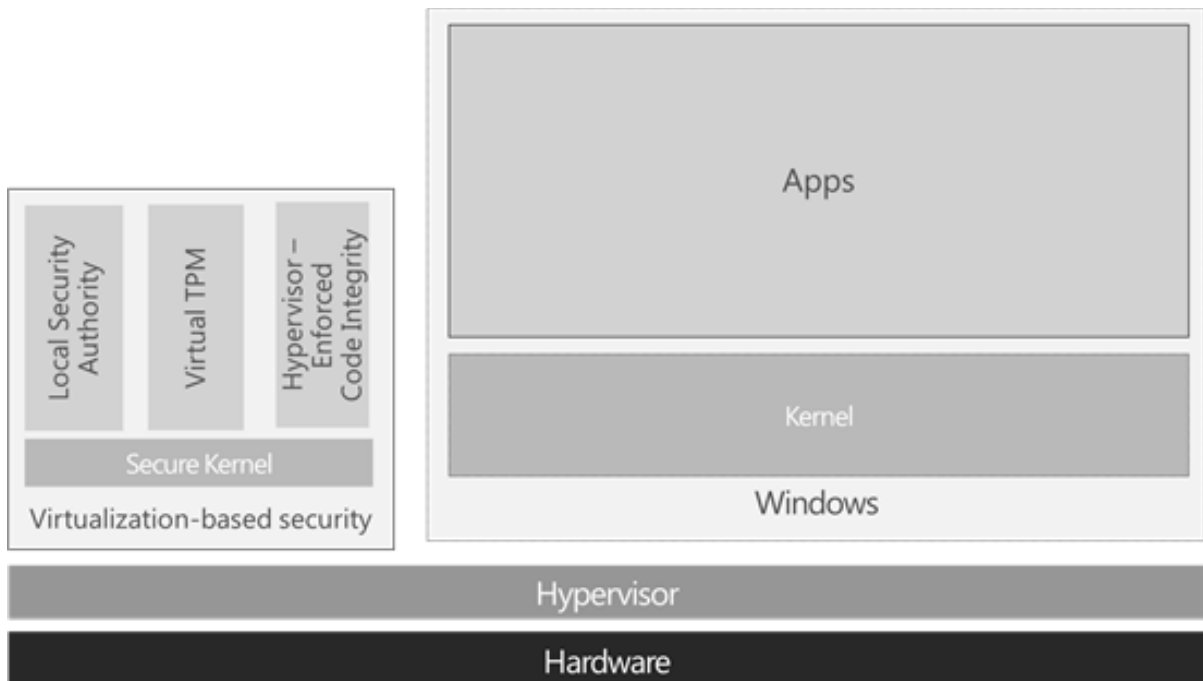
Cryptographic bypass is not supported by VTPM.

2.6 Hardware Components of the Cryptographic Module

The physical boundary of the module is the physical boundary of the computer that contains the module. The following diagrams illustrate the hardware components used by the Virtual TPM module:



Note: The CPU provides Processor Algorithm Accelerator (PAA)



3 Cryptographic Module Ports and Interfaces

3.1 VTPM export functions

The following list contains all the functions exported by VTPM. These functions are explained further in the subsequent subsections.

- VtpmColdInitWithPersistentState
- VtpmWarmInit
- VtpmShutdown
- VtpmExecuteCommand
- VtpmSetCancelFlag
- VtpmGetRuntimeState

3.1.1 VtpmColdInitWithPersistentState

VtpmColdInitWithPersistentState() is the function exported by VTPM to initialize an instance of VTPM with an existing persistent state. A pointer to the existing state is passed as one of the parameters. If that pointer is NULL, a new VTPM instance will be created. The return code S_OK means the function was successful. Any other return code indicates failure. Includes a memory reference to the runtime state for the Virtual TPM, which includes the vTPM's CSPs.

3.1.2 VtpmWarmInit

VtpmWarmInit() is an exported VTPM function to initialize an instance of a VTPM by the caller providing both a persistent state and a run-time state. A pointer to the existing persistent state and a pointer to the existing run-time state are passed in as parameters. Neither pointer can be NULL. The resulting VTPM instance is a copy of the source VTPM. The resulting VTPM instance can be used where the source instance left off. No additional initialization of the TPM engine is required. The return code S_OK means the function was successful. Any other return code indicates failure. Includes a memory reference to the runtime state for the Virtual TPM, which includes the vTPM's CSPs.

3.1.3 VtpmShutdown

VtpmShutdown() is the function exported by VTPM to shut down an instance of a VTPM. The return code S_OK means the function was successful. Any other return code indicates failure.

3.1.4 VtpmExecuteCommand

VtpmExecuteCommand() is an exported VTPM function to execute TPM 2.0 commands. Pointers and size parameters for the command and response buffers are provided by the caller. The return code S_OK means the function was successful. Any other return code indicates failure.

The TPM 2.0 library specification for the commands is *Trusted Platform Module Library Part 3: Commands*, Family "2.0", Level 00 Revision 01.16, dated October 30, 2014. It is available from the Trusted Computing Group (TCG) at:

http://www.trustedcomputinggroup.org/resources/tpm_library_specification

A high-level list of TPM 2.0 commands is provided here. Please refer to the TPM 2.0 library specification for details.

- Start-up Commands
 - _TPM_Init
 - TPM2_Startup
 - TPM2_Shutdown
- Testing Commands
 - TPM2_SelfTest
 - TPM2_IncrementalSelfTest
 - TPM2_GetTestResult
- Session Commands
 - TPM2_StartAuthSession
 - TPM2_PolicyRestart
- Object Commands
 - TPM2_Create
 - TPM2_Load
 - TPM2_LoadExternal
 - TPM2_ReadPublic
 - TPM2_ActivateCredential
 - TPM2_MakeCredential
 - TPM2_Unseal
 - TPM2_ObjectChangeAuth
- Duplication Commands
 - TPM2_Duplicate
 - TPM2_Rewrap
 - TPM2_Import
- Asymmetric Primitives
 - TPM2_RSA_Encrypt
 - TPM2_RSA_Decrypt
 - TPM2_ECDH_KeyGen
 - TPM2_ECDH_Zgen
 - TPM2_ECC_Parameters
 - TPM2_Zgen_2Phase
- Symmetric Primitives
 - TPM2_EncryptDecrypt
 - TPM2_Hash
 - TPM2_HMAC
- Random Number Generator
 - TPM2_GetRandom
 - TPM2_StirRandom
- Hash/HMAC/Event Sequences
 - TPM2_HMAC_Start
 - TPM2_HashSequenceStart

- TPM2_SequenceUpdate
- TPM2_SequenceComplete
- TPM2_EventSequenceComplete
- Attestation Commands
 - TPM2_Certify
 - TPM2_CertifyCreation
 - TPM2_Quote
 - TPM2_GetSessionAuditDigest
 - TPM2_GetCommandAuditDigest
 - TPM2_GetTime
- Ephemeral EC Keys
 - TPM2_Commit
 - TPM2_EC_Ephemeral
- Signing and Signature Verification
 - TPM2_VerifySignature
 - TPM2_Sign
- Command Audit
 - TPM2_SetCommandCodeAuditStatus
- Integrity Collection (PCR)
 - TPM2_PCR_Extend
 - TPM2_PCR_Event
 - TPM2_PCR_Read
 - TPM2_PCR_Allocate
 - TPM2_PCR_SetAuthPolicy
 - TPM2_PCR_SetAuthValue
 - TPM2_PCR_Reset
 - _TPM_Hash_Start
 - _TPM_Hash_Data
 - _TPM_Hash_End
- Enhanced Authorization (EA) Commands
 - TPM2_PolicySigned
 - TPM2_PolicySecret
 - TPM2_PolicyTicket
 - TPM2_PolicyOR
 - TPM2_PolicyPCR
 - TPM2_PolicyLocality
 - TPM2_PolicyNV
 - TPM2_PolicyCounterTimer
 - TPM2_PolicyCommandCode
 - TPM2_PolicyPhysicalPresence
 - TPM2_PolicyCpHash

- TPM2_PolicyNameHash
- TPM2_PolicyDuplicationSelect
- TPM2_PolicyAuthorize
- TPM2_PolicyAuthValue
- TPM2_PolicyPassword
- TPM2_PolicyGetDigest
- TPM2_PolicyNvWritten
- Hierarchy Commands
 - TPM2_CreatePrimary
 - TPM2_HierarchyControl
 - TPM2_SetPrimaryPolicy
 - TPM2_ChangePPS
 - TPM2_ChangeEPS
 - TPM2_Clear
 - TPM2_ClearControl
 - TPM2_HierarchyChangeAuth
- Dictionary Attack Functions
 - TPM2_DictionaryAttackLockReset
 - TPM2_DictionaryAttackParameters
- Miscellaneous Management Functions
 - TPM2_PP_Commands
 - TPM2_SetAlgorithmSet
- Field Upgrade
 - TPM2_FieldUpgradeStart
 - TPM2_FieldUpgradeData
 - TPM2_FirmwareRead
- Context Management
 - TPM2_ContextSave
 - TPM2_ContextLoad
 - TPM2_FlushContext
 - TPM2_EvictControl
- Clocks and Timers
 - TPM2_ReadClock
 - TPM2_ClockSet
 - TPM2_ClockRateAdjust
- Capability Commands
 - TPM2_GetCapability
 - TPM2_TestParms
- Non-volatile Storage
 - TPM2_NV_DefineSpace
 - TPM2_NV_UndefineSpace

- TPM2_NV_UndefineSpaceSpecial
- TPM2_NV_ReadPublic
- TPM2_NV_Write
- TPM2_NV_Increment
- TPM2_NV_Extend
- TPM2_NV_SetBits
- TPM2_NV_WriteLock
- TPM2_NV_GlobalWriteLock
- TPM2_NV_Read
- TPM2_NV_ReadLock
- TPM2_NV_ChangeAuth
- TPM2_NV_Certify

3.1.5 VtpmSetCancelFlag

VtpmSetCancelFlag() is an exported VTPM function to set or reset the cancel flag. When the cancel flag is set, the TPM library will opportunistically abort the command being executed.

3.1.6 VtpmGetRuntimeState

VtpmGetRuntimeState() is a function exported by VTPM to return the VTPM current run-time state. The caller passes a pointer to the buffer to receive the run-time state and a pointer to the variable containing the buffer size. The return code S_OK means the function was successful. Any other return code indicates failure. Includes a memory reference to the runtime state for the Virtual TPM, which includes the vTPM's CSPs.

3.2 Control Input Interface

The Control Input Interface for VTPM consists of the export functions. Options for control operations are passed as input parameters to the export functions.

3.3 Status Output Interface

The Status Output Interface for VTPM also consists of the export functions. The status information is returned to the caller as the return value of each function or to log files.

3.4 Data Input Interface

The Data Input Interface for VTPM also consists of the export functions. 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.

3.5 Data Output Interface

The Data Output Interface for VTPM also consists of the export functions. Data is returned to the function's caller via output parameters.

3.6 Non-Security Relevant Interfaces

Many TPM 2.0 commands are not cryptographic functions. For details, see the TPM 2.0 library specification.

4 Roles, Services and Authentication

4.1 Roles

VTPM exposes all interfaces to the user launching the application that loaded the module.

FIPS 140 validations define formal “User” and “Cryptographic Officer” roles. Both roles can use any VTPM service.

4.2 Services

Virtual Trusted Platform Module services are described below:

1. **TPM 2.0 Services** – The list of exported VTPM functions and TPM 2.0 commands in Section [Cryptographic Module Ports and Interfaces](#) describes the TPM services offered by this module.
2. **Show Status** – The module provides a show status service that is automatically executed by the module to provide the status response of the module either via output to the computer monitor as the return value of each function or to log files.
3. **Self-Tests** – The module provides a power-up self-tests service that is automatically executed when the module is loaded into memory.
4. **Zeroizing Cryptographic Material** – This service is executed as part of the module shutdown. See [Key Zeroization](#).

The list of exported VTPM functions and TPM 2.0 commands in Section [Cryptographic Module Ports and Interfaces](#) contains all the services available to an operator.

4.2.1 Mapping of Services, Algorithms, and Critical Security Parameters

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

Table 3 Services

Service	Algorithms	CSPs
Start-up Commands (TPM 2.0 Services)	None	None
Testing Commands (Self-Tests)	See Section Self-Tests for the list of algorithms	None
Session Commands (TPM 2.0 Services)	AES, RSA, ECC, ECDSA / EC Schnorr [non-Approved], SP 800-133	Symmetric Keys Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys

	Cryptographic Key Generation ⁵ , SP 800-56B RSA-OAEP AES-256 CTR DRBG AES CFB key import [non-Approved] XOR [non-Approved]	AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Object Commands (TPM 2.0 Services)	AES, RSA, ECC, HMAC, SP800-108 KDF, ECDSA / EC Schnorr [non-Approved], SP 800-133 Cryptographic Key Generation ⁶ , SP 800-56B RSA-OAEP AES-256 CTR DRBG AES CFB key import [non-Approved] MGF1 [non-Approved]	Symmetric Keys Symmetric Keys (for HMAC) Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys Storage Root Key (SRK) Hierarchy Proofs Hierarchy Authorization/Policy keys AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Duplication Commands (TPM 2.0 Services)	AES, HMAC, SP 800-56B RSA-OAEP AES CFB key import [non-Approved] MGF1 [non-Approved]	Symmetric Keys Symmetric Keys (for HMAC) Asymmetric RSA Public Keys Asymmetric RSA Private Keys
Asymmetric Primitives (TPM 2.0 Services)	RSA, ECDH, ECC, RSADP, ECDSA / EC Schnorr [non-Approved], SP 800-133 Cryptographic Key Generation ⁷ , SP 800-56B RSA-OAEP AES-256 CTR DRBG MGF1 [non-Approved]	Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys ECDH Private and Public values AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Symmetric Primitives (TPM 2.0 Services)	AES (CTR, OFB, CBC, CFB, ECB) and HMAC AES CFB key import [non-Approved]	Symmetric Keys Symmetric Keys (for HMAC) Hierarchy Proofs
Random Number Generator (TPM 2.0 Services)	AES-256 CTR DRBG	AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V

⁵ Generated directly from the output of a DRBG.

⁶ Generated directly from the output of a DRBG.

⁷ Generated directly from the output of a DRBG.

		AES-CTR DRBG Key
Hash/HMAC/Event Sequences (TPM 2.0 Services)	FIPS 180-4 SHS SHA-1, SHA-256, and SHA-384; FIPS 180-4 SHA-1, SHA-256, SHA-384 HMAC; AES CFB key import [non-Approved] XOR [non-Approved]	Symmetric Keys (for HMAC) Hierarchy Proofs
Attestation Commands (TPM 2.0 Services)	FIPS 186-4 RSA (RSASSA-PKCS1-v1_5 and RSASSA-PSS) digital signature generation and verification; supporting SHA-1 ⁸ , SHA-256, and SHA-384 FIPS 186-4 ECDSA, ECDSA / EC Schnorr [non-Approved], SP 800-56B RSA-OAEP MGF1 [non-Approved]	Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys Endorsement Key (EK) Hierarchy Proofs AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Ephemeral EC Keys (TPM 2.0 Services)	ECC, ECDSA / EC Schnorr [non-Approved], SP 800-133 Cryptographic Key Generation ⁹ AES-256 CTR DRBG	Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Signing and Signature Verification (TPM 2.0 Services)	FIPS 186-4 RSA (RSASSA-PKCS1-v1_5 and RSASSA-PSS) digital signature generation and verification; supporting SHA-1 ¹⁰ , SHA-256, and SHA-384 FIPS 186-4 ECDSA / EC Schnorr [non-Approved] MGF1 [non-Approved]	Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys Hierarchy Proofs AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Command Audit (TPM 2.0 Services)	FIPS 180-4 SHS SHA-1, SHA-256, and SHA-384	Hierarchy Authorization/Policy
Integrity Collection (PCR) (TPM 2.0 Services)	FIPS 180-4 SHS SHA-1, SHA-256, and SHA-384	Hierarchy Authorization/Policy keys

⁸ SHA-1 is only acceptable for signature verification.

⁹ Generated directly from the output of a DRBG.

¹⁰ SHA-1 is only acceptable for signature verification.

Enhanced Authorization (EA) Commands (TPM 2.0 Services)	FIPS 180-4 SHS SHA-1, SHA-256, and SHA-384; FIPS 186-4 RSA (RSASSA-PKCS1-v1_5 and RSASSA-PSS) digital signature generation and verification; supporting SHA-1 ¹¹ , SHA-256, and SHA-384 FIPS 186-4 ECDSA / EC Schnorr [non-Approved] MGF1 [non-Approved]	Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys Hierarchy Proofs Hierarchy Authorization/Policy keys AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Hierarchy Commands (TPM 2.0 Services)	AES-256 CTR DRBG	Storage Root Key (SRK) Storage Primary Seed (SPS) Hierarchy Proofs Hierarchy Authorization/Policy keys Endorsement Primary Seed (EPS) Platform Primary Seed (PPS) AES-CTR DRBG Seed AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Dictionary Attack Functions (TPM 2.0 Services)	None	Hierarchy Authorization/Policy keys
Miscellaneous Management Functions (TPM 2.0 Services)	None	Hierarchy Authorization/Policy keys
Field Upgrade (TPM 2.0 Services)	Not Implemented	None
Context Management (TPM 2.0 Services)	AES, HMAC, SP800-108 KDF AES CFB key import [non-Approved]	Symmetric Keys (for HMAC) Hierarchy Proofs Hierarchy Authorization/Policy keys
Clocks and Timers (TPM 2.0 Services)	None	Hierarchy Authorization/Policy keys
Capability Commands (TPM 2.0 Services)	None	None
Non-volatile Storage (TPM 2.0 Services)	FIPS 186-4 RSA (RSASSA-PKCS1-v1_5 and RSASSA-PSS) digital signature generation and verification; supporting SHA-1 ¹² , SHA-256, and SHA-384	Asymmetric RSA Public Keys Asymmetric RSA Private Keys Asymmetric ECDSA Public keys Asymmetric ECDSA Private keys Hierarchy Authorization/Policy keys AES-CTR DRBG Seed

¹¹ SHA-1 is only acceptable for signature verification.

¹² SHA-1 is only acceptable for signature verification.

	FIPS 186-4 ECDSA / EC Schnorr [non-Approved], SP 800-56B RSA-OAEP	AES-CTR DRBG Entropy Input AES-CTR DRBG V AES-CTR DRBG Key
Show Status	None	None
Self-Tests	See Section Self-Tests for the list of algorithms	None
Zeroizing Cryptographic Material	None	All keys and CSPs

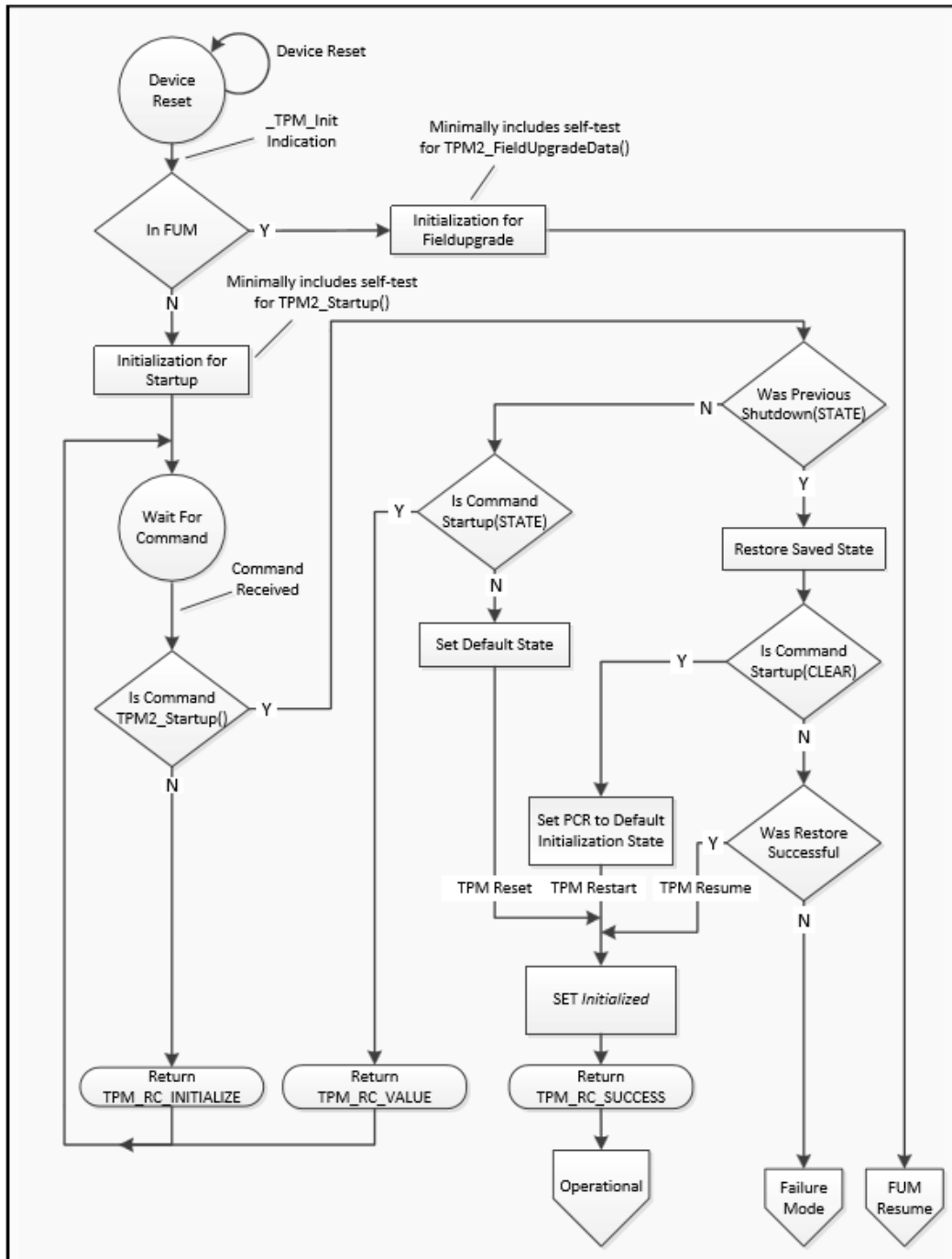
4.3 Authentication

VTPM does not provide authentication of users. Roles are implicitly assumed based on the services that are executed.

5 Finite State Model

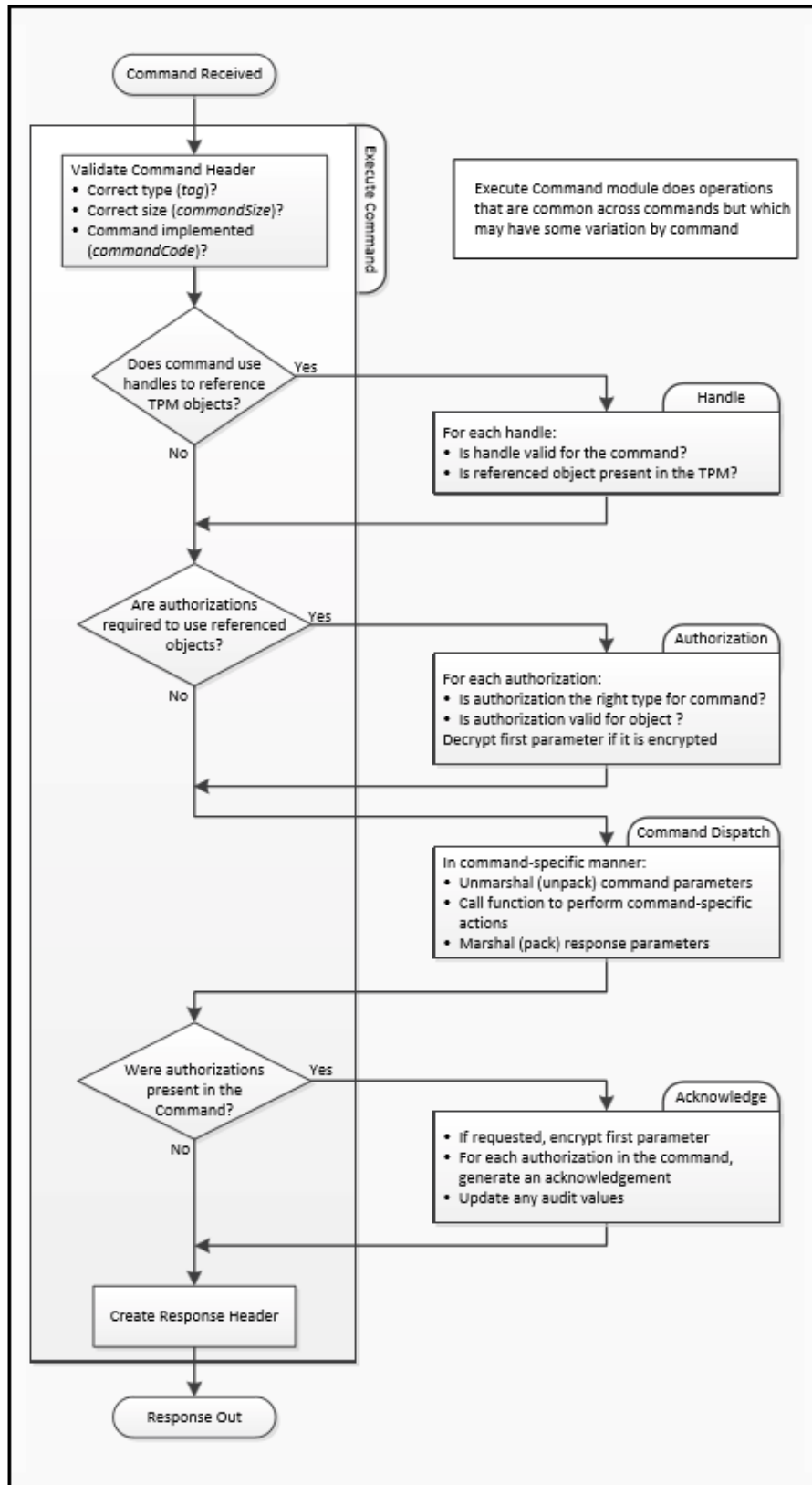
The Trusted Computing Group TPM specification, Part 1 “Architecture” contains Finite state models for TPM Startup and Command Processing. They are reproduced below.

5.1 Startup



For self-testing, if a command is received that requires return of a value that depends on untested functions, the TPM shall test the required functions before completing the command. The Virtual TPM executes all of the self-tests directly from its “DllMain” function, which occurs prior to “_TPM_Init” in the diagram. Furthermore, the “Device Reset” section of the diagram refers to when the Virtual TPM library is reloaded.

5.2 Command Processing



6 Operational Environment

The operational environment for VTPM is the Windows Server operating system running on a supported hardware platform listed section 1.2.

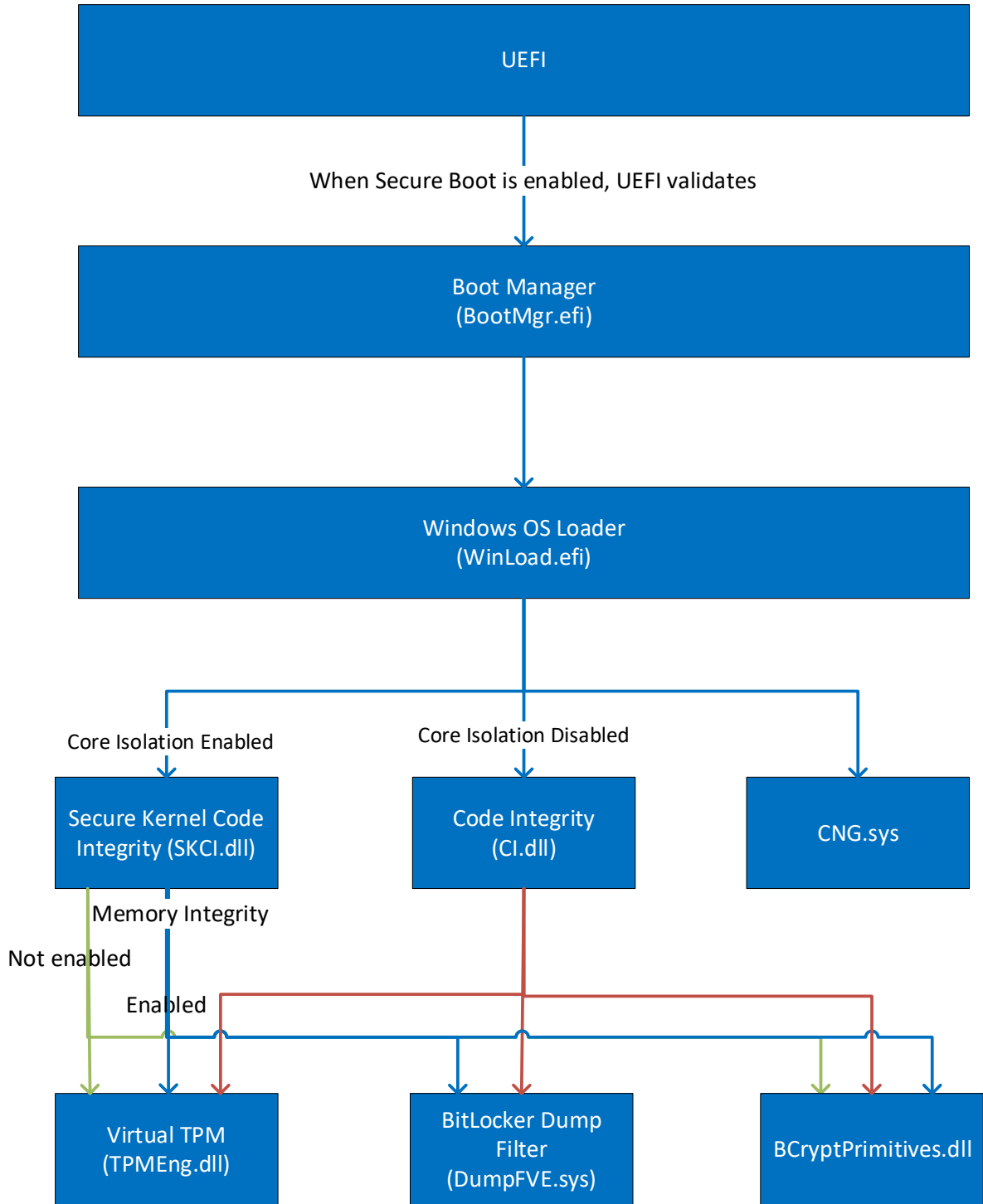
6.1 Cryptographic Isolation

In the Windows operating system environment, all DLLs, including TPMEngUM.dll, are loaded into a process, on-demand, for the services offered by that DLL. The OS environment enforces process isolation including memory (where keys and intermediate key data are stored) and CPU scheduling.

6.2 Integrity Chain of Trust

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

- Windows Server 2019 build 10.0.17763.10021 and 10.0.17763.10127



The integrity of the Virtual Trusted Platform Module (TPMEngUM.dll) is checked by Code Integrity (or Secure Kernel Code Integrity on VSM configurations).

7 Cryptographic Key Management

VTPM uses the following security relevant data items:

Table 4 Security Relevant Data Items

Security Relevant Data Item	Description
Symmetric keys	Keys used for AES encryption/decryption. Key sizes for AES are 128, 192, and 256 bits
Symmetric keys (for HMAC)	Keys used for HMAC-SHA1, HMAC-SHA256, and HMAC-SHA384. Key sizes are variable length of at least 112 bits.
Asymmetric ECDSA Public Keys	Keys used for the verification of ECDSA digital signatures. Curve sizes are P-256 and P-384.
Asymmetric ECDSA Private Keys	Keys used for the calculation of ECDSA digital signatures. Curve sizes are P-256 and P-384.
Asymmetric RSA Public Keys	Keys used for the verification of RSA digital signatures. Key size is 2048 bits. These keys can be produced using RSA Key Generation.
Asymmetric RSA Private Keys	Keys used for the calculation of RSA digital signatures. Key size is 2048 bits. These keys can be produced using RSA Key Generation.
AES-CTR DRBG Seed	A 384 bit secret value maintained internal to the module that provides the seed material for AES-CTR DRBG output
AES-CTR DRBG Entropy Input	A secret value that is at least 256 bits and maintained internal to the module that provides the entropy material for AES-CTR DRBG output
AES-CTR DRBG V	A 128 bit secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output
AES-CTR DRBG key	A 256 bit secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output
ECDH Private and Public values	Private and public values used for EC Diffie-Hellman key establishment. Curve sizes are P-256 and P-384.
Storage Root Key (SRK)	A 2048-bit RSA public/private keypair used to seal data to the TPM.
Endorsement Key (EK)	A 2048-bit RSA public/private keypair used to uniquely identify and attest the TPM instance.
Endorsement Primary Seed (EPS)	Random seed material used to derive EK. Created as part of 32 bytes of primary seed values.
Platform Primary Seed (PPS)	Random seed material used to derive firmware hierarchies. Created as part of 32 bytes of primary seed values.
Storage Primary Seed (SPS)	Random seed material used to derive SRK. Created as part of 32 bytes of primary seed values.

Hierarchy Proofs (phProof, shProof, ehProof, nullProof)	Symmetric HMAC keys used to validate previous cryptographic operations. Created as part of 32 bytes of secret values.
Hierarchy Authorization/Policy (platformAuth/Policy, ownerAuth/Policy, endorsementAuth/Policy, lockoutAuth/Policy)	Keys used for authorization. Variable in length up to 384 bits.

7.1 Access Control Policy

VTPM allows controlled access to security relevant data items contained within it. The following table defines the access that a service has to each item. The permissions are categorized as a set of three separate permissions: read [®], write (w), and delete (d). If no permission is listed, the service has no access to the item. The User and Cryptographic Officer roles have the same access to keys so roles are not distinguished in the table.

Table 5 Access Control Policy

Service	Security Relevant Data Item	Permissions
Start-up Commands	None	
Testing Commands (Self-Tests)	None	
Session Commands	Symmetric Keys	rw
	Asymmetric RSA Public Keys	rw
	Asymmetric RSA Private Keys	rw
	Asymmetric ECDSA Public keys	rw
	Asymmetric ECDSA Private keys	rw
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG V	rw
Object Commands	AES-CTR DRBG Key	rw
	Symmetric Keys	rw
	Symmetric Keys (for HMAC)	rw
	Asymmetric RSA Public Keys	rw
	Asymmetric RSA Private Keys	rw
	Asymmetric ECDSA Public keys	rw
	Asymmetric ECDSA Private keys	rw
	Storage Root Key (SRK)	r
	Hierarchy Proofs	r
	Hierarchy Authorization/Policy keys	r
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
AES-CTR DRBG V	rw	

	AES-CTR DRBG Key	rw
Duplication Commands	Symmetric Keys	rw
	Symmetric Keys (for HMAC)	rw
	Asymmetric RSA Public Keys	rw
	Asymmetric RSA Private Keys	rw
Asymmetric Primitives	Asymmetric RSA Public Keys	rw
	Asymmetric RSA Private Keys	rw
	Asymmetric ECDSA Public keys	rw
	Asymmetric ECDSA Private keys	rw
	ECDH Private and Public values	rw
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG V	rw
	AES-CTR DRBG Key	rw
Symmetric Primitives	Symmetric Keys	r
	Symmetric Keys (for HMAC)	r
	Hierarchy Proofs	r
Random Number Generator	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG V	rw
	AES-CTR DRBG Key	rw
Hash/HMAC/Event Sequences	Symmetric Keys (for HMAC)	r
	Hierarchy Proofs	r
Attestation Commands	Asymmetric RSA Public Keys	r
	Asymmetric RSA Private Keys	r
	Asymmetric ECDSA Public keys	r
	Asymmetric ECDSA Private keys	r
	Endorsement Key	r
	Hierarchy Proofs	r
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG V	rw
	AES-CTR DRBG Key	rw
Ephemeral EC Keys	Asymmetric RSA Public Keys	rw
	Asymmetric RSA Private Keys	rw
	Asymmetric ECDSA Public keys	rw
	Asymmetric ECDSA Private keys	rw
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG V	rw
	AES-CTR DRBG Key	rw

Signing and Signature Verification	Asymmetric RSA Public Keys	r
	Asymmetric RSA Private Keys	r
	Asymmetric ECDSA Public keys	r
	Asymmetric ECDSA Private keys	r
	Hierarchy Proofs	r
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG V	rw
	AES-CTR DRBG Key	rw
Command Audit	Hierarchy Authorization/Policy keys	r
Integrity Collection (PCR)	Hierarchy Authorization/Policy keys	r
Enhanced Authorization (EA) Commands	Asymmetric RSA Public Keys	r
	Asymmetric RSA Private Keys	r
	Asymmetric ECDSA Public keys	r
	Asymmetric ECDSA Private keys	r
	Hierarchy Proofs	r
	Hierarchy Authorization/Policy keys	r
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG Key	rw
Hierarchy Commands	Storage Root Key	rwd
	Storage Primary Seed	rwd
	Hierarchy Proofs	rwd
	Hierarchy Authorization/Policy keys	rwd
	Endorsement Primary Seed	rwd
	Platform Primary Seed	rwd
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG Key	rw
Dictionary Attack Functions	Hierarchy Authorization/Policy keys	r
Miscellaneous Management Functions	Hierarchy Authorization/Policy keys	r
Field Upgrade	None	
Context Management	Symmetric Keys (for HMAC)	r
	Hierarchy Proofs	r
	Hierarchy Authorization/Policy keys	r
Clocks and Timers	Hierarchy Authorization/Policy keys	r
Capability Commands	None	
Non-volatile Storage	Asymmetric RSA Public Keys	r

	Asymmetric RSA Private Keys	r
	Asymmetric ECDSA Public keys	r
	Asymmetric ECDSA Private keys	r
	Hierarchy Authorization/Policy keys	r
	AES-CTR DRBG Seed	rw
	AES-CTR DRBG Entropy Input	rw
	AES-CTR DRBG V	rw
	AES-CTR DRBG Key	rw
Show Status	None	
Self-Tests	None	
Zeroizing Cryptographic Material	All keys and CSPs	wd

Note that the Endorsement Key is not directly accessible by any service and never leaves the logical boundary of the VTPM.

7.2 Key Material

See Section [Key Storage for keys](#) that persist in VTPM. The user application is responsible for using VTPM commands to generate or import other cryptographic keys.

7.3 Key Generation

VTPM can generate (create) and use cryptographic keys for the following FIPS Approved algorithms: AES, ECDSA, RSA, and HMAC.

Random keys can be generated by this module using a variety of methods as described below in the list of services. Random data is provided by the SP 800-90A CTR DRBG for the generation of symmetric keys.

The following VTPM services can be used to generate cryptographic keys:

- Session Commands
- Object Commands
- Asymmetric Primitives
- Ephemeral EC Keys

Keys generated while not operating in the FIPS mode of operation cannot be used in FIPS mode, and vice versa.

7.4 Key Agreement and Derivation

VTPM implements pair-wise key establishment in accordance with NIST SP 800-56A KAS ECC.

VTPM implements key derivation using SP 800-108 KDF.

7.5 Key Entry and Output

Cryptographic keys may be imported into and exported out of VTPM via TPM commands as described in the TPM 2.0 specification. Keys can be imported and exported in plaintext, AES-CFB encrypted, and RSA-OAEP encrypted forms. The module also offers other APIs, such as VTpmColdInitWithPersistentState, VTpmWarmInit, and VTpmGetRuntimeState (See section 3.1) that include a memory reference to the runtime state for the virtual TPM, which includes the vTPM's CSPs.

7.6 Key Storage

VTPM supports the Storage Root Key (SRK), Endorsement Key (EK), and Primary Seeds as defined by the Trusted Computing Group in the TPM 2.0 specification. All of the vTPM state, including the CSPs, is stored in memory as a blob for TPMEngUM.dll to process. If there is a need to persist one of these keys it is the calling application's (VMSP.exe) responsibility to persist the blob. The Hypervisor worker process (VMSP.exe) encrypts and decrypts the VM state as needed using the Cryptographic Primitives library (BCRYPTPRIMITIVES.DLL) for AES 256 GCM.

The SRK is used to protect TPM keys created by applications. The EK is used in many scenarios including the generation of TPM-bound signing keys.

7.7 Key Archival

VTPM does not directly archive cryptographic keys. A cryptographic key may be exported, but management of the secure archival of that key is the responsibility of the user/calling application such as VMSP.exe.

7.8 Key Zeroization

TPMEngUM.dll will destroy keys by zeroizing them after use.

All CSPs in memory are zeroized when the virtual machine is shutdown.

The persisted state of a Virtual TPM as described in [Key Storage](#), is deleted when the virtual machine is deleted. To zeroize the persistent state from the disk, users can reformat and overwrite the hard drive.

8 Self-Tests

8.1 Power-On Self-Tests

VTPM performs the following power-on (startup) self-tests:

- AES 128, 192, and 256 and encrypt/decrypt ECB Known Answer Tests
- AES 128, 192, and 256 encrypt/decrypt CBC Known Answer Tests
- AES 128, 192, and 256 encrypt/decrypt OFB Known Answer Tests
- AES 128, 192, and 256 encrypt/decrypt CFB Known Answer Tests
- AES 128, 192, and 256 encrypt/decrypt CTR Known Answer Tests
- HMAC (SHA-1, SHA-256, SHA-384) Known Answer Tests

- RSA PKCS#1 (v1.5) sign and verify Known Answer Test with 2048-bit key and SHA-384 message digest
- RSA PSS sign and verify Pairwise Consistency Test with 2048-bit key and SHA-384 message digest
- ECDSA sign and verify Pairwise Consistency Test with P-256 curve and SHA-384 message digest
- ECDH shared secret computation Known Answer Test with P-256 curve
- SP 800-108 Key Derivation with HMAC-SHA-384 as the PRF
- SP 800-90A AES-256 counter mode DRBG Known Answer Tests (instantiate, generate and reseed)

8.2 Conditional Self-Tests

VTPM performs the following conditional self-tests:

- CRNGT for SP 800-90A AES-CTR DRBG
- Health tests for SP 800-90A AES-CTR DRBG
- Pair-wise consistency checks for ECDH, ECDSA and RSA key generations

9 Design Assurance

The secure installation, generation, and startup procedures of this cryptographic module are part of the overall operating system secure installation, configuration, and startup procedures for the Windows Server operating system.

The Windows Server 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 Server version downloaded from Windows Update.

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

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

For Windows Updates, the client only accepts binaries signed by Microsoft certificates. The Windows Update client only accepts content whose SHA-2 hash matches the SHA-2 hash specified in the metadata. All metadata communication is done over a Secure Sockets Layer (SSL) port. Using SSL ensures that the client is communicating with the real server and so prevents a spoof server from sending the client harmful requests. The version and digital signature of new cryptographic module releases must be verified to match the version that was validated. See Appendix A for details on how to do this.

10 Mitigation of Other Attacks

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

Table 6 Mitigation of Other Attacks

Algorithm	Protected Against	Mitigation	Comments
SHA1	Timing Analysis Attack	Constant Time Implementation	
	Cache Attack	Memory Access pattern is independent of any confidential data	
SHA2	Timing Analysis Attack	Constant Time Implementation	
	Cache Attack	Memory Access pattern is independent of any confidential data	
AES	Timing Analysis Attack	Constant Time Implementation	
	Cache Attack	Memory Access pattern is independent of any confidential data	Protected Against Cache attacks only when used with AES NI

11 Security Levels

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

Table 7 Security Levels

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

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

12 Additional Details

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

<http://windows.microsoft.com>

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

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

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

13.1 How to Verify Windows Versions

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

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

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

13.2 How to Verify Windows Digital Signatures

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

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