Microsoft Corporation FIPS 140 Validation

Microsoft SymCrypt Cryptographic Library
FIPS 140-3 Non-Proprietary Security Policy Document

Prepared By	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399
Document Version Number	1.0
Updated On	August 25, 2025

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.

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

The Microsoft SymCrypt Cryptographic Library (the “module”) is a general purpose software cryptographic module that provides cryptographic services.

The module implements FIPS 140-3 approved cryptographic algorithms, and this document is the FIPS 140-3 Security Policy for the module. The 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	N/A
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 Microsoft SymCrypt Cryptographic Library (the “module”) is a general purpose, software cryptographic module that provides cryptographic services used by Microsoft and third-party applications

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

The module is a software library that provides cryptographic services to calling applications.

The module conforms to the definition of a software module in that its cryptographic boundary delimits the software components of the module from the underlying computing platform and operating system, which are external to the module. The elements outside of the module’s cryptographic boundary depicted in Figure 1 are out of scope of this evaluation and do not interfere with the secure operation of the module in its approved mode of operation.

The Tested Operational Environment’s Physical Perimeter (TOEPP) is the physical perimeter of the computer that contains the module. The software component of the module defines its cryptographic boundary as listed below.

Software Component	Description
LIBSYMCRYPT.SO	Binary file that contains the module.

Table 2: Module Software Components

Tested Operational Environment’s Physical Perimeter (TOEPP):

The following diagram illustrates the module Tested Operational Environment’s Physical Perimeter (TOEPP) and cryptographic boundary (SymCrypt block).

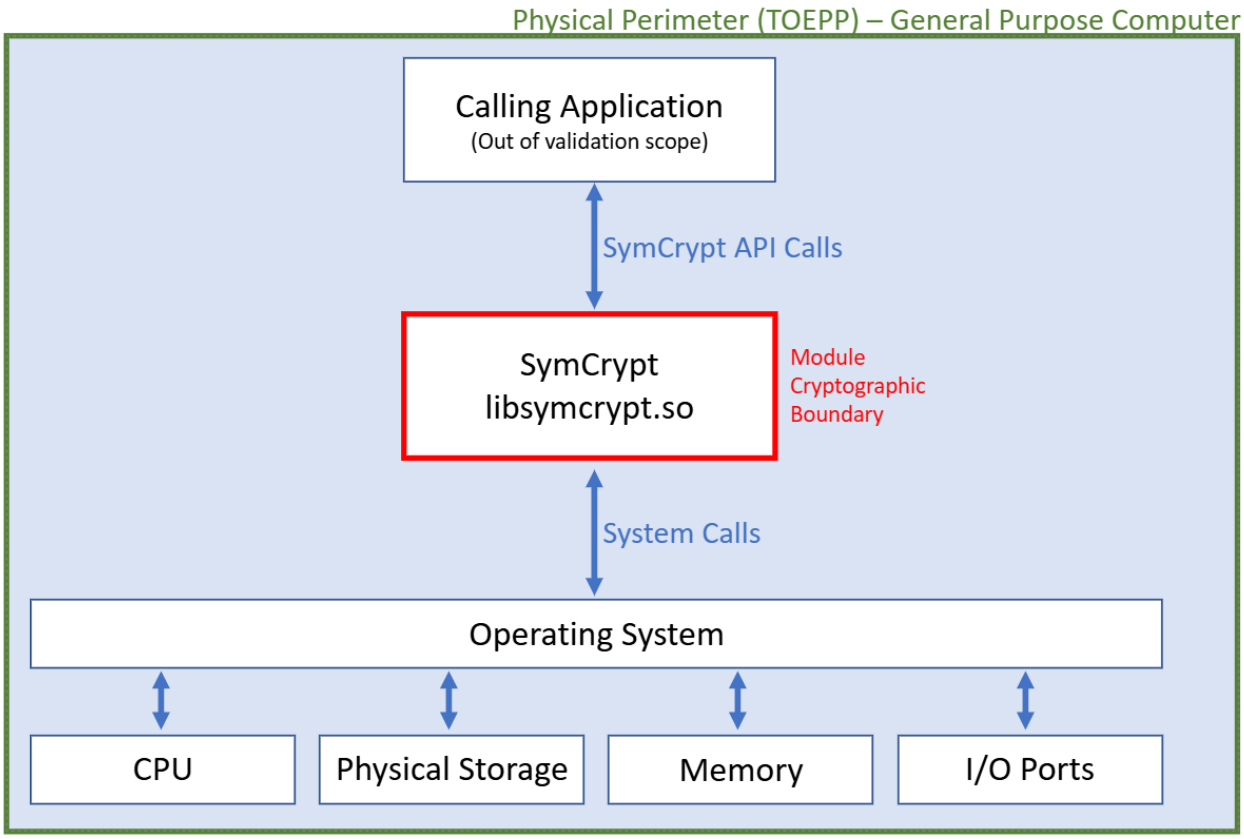

Figure 1 - Module Physical Perimeter and Cryptographic Boundary

2.2 Tested and Vendor Affirmed Module Version and Identification

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

The following table identifies the modules tested, including the executable code sets.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libsymcrypt.so	103.8.0		Message authentication with HMAC-SHA2-256

Table 3: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Operational Environments - Software, Firmware, Hybrid:

The operational environments for the module are the operating systems listed below running on a supported hardware platform.

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
SONiC release 202305	Arista-7060CX-32S	AMD GX-424CC	Yes		103.8.0

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
SONiC release 202305	Nokia 7250	AMD EPYC 3251 8-Core Processor	Yes		103.8.0
SONiC release 202305	Cisco-8102	Intel Xeon CPU D-1530	Yes		103.8.0
SONiC release 202305	Force10-S6100	Intel Atom CPU C2538	Yes		103.8.0
SONiC release 202305	Mellanox-SN2700	Intel Celeron CPU 1047UE	No		103.8.0
Metaswitch Linux 8	AWS c4.2xlarge	1X Intel(R) Xeon(R) CPU E5-2666	Yes		103.8.0
Metaswitch Linux 8	Dell PowerEdge R640	2 x Intel(R) Xeon(R) Gold 6134 CPU @ 3.20GHz	Yes		103.8.0
Windows Server 2022	Microsoft Azure C2071	Intel Xeon E-2288G	Yes		103.8.0
Windows Server 2022	Microsoft Azure C2080	Intel 3rd Generation Xeon Scalable 8370C	Yes		103.8.0
Ubuntu 20.04	Microsoft Azure C2071	Intel Xeon E-2288G	Yes		103.8.0
Ubuntu 20.04	Microsoft Azure C2080	Intel 3rd Generation Xeon Scalable 8370C	Yes		103.8.0
Azure Linux 3.0	Azure Virtual Machine Standard_D4_v5	Intel(R) Xeon(R) Platinum 8370C	Yes		103.8.0
Azure Linux 3.0	Standard_D4ps_v5 ARM64	Ampere Altra [Arm64]	Yes		103.8.0
Azure Linux 3.0	Standard_D4ps_v6 ARM64	Azure Cobalt 100 [Arm64]	Yes		103.8.0

Table 4: Tested Operational Environments - Software, Firmware, Hybrid

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

The following table presents the vendor-affirmed operational environment(s).

Operating System	Hardware Platform
Linux-based operating system with SymCrypt compiled using clang.	UEFI-based computers with a CPU from AMD (x64), Ampere (ARM64), Intel (x64), or Microsoft Cobalt (ARM64)

Table 5: 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 keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

No components within the cryptographic boundary are claimed as excluded.

2.4 Modes of Operation

Modes List and Description:

The following table lists the module's modes of operation.

Mode Name	Description	Type	Status Indicator
Approved Mode	Normal operation of the module.	Approved	API function SymCryptDeprecatedServiceIndicator() returns 0.
Non-Approved Mode	Operation of the module if a non-approved algorithm is invoked.	Non-Approved	API function SymCryptDeprecatedServiceIndicator() returns 1.

Table 6: 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 for. The mode changes to the non-approved mode implicitly if any non-approved algorithm is use. The calling application is responsible for ensuring that CSPs are not shared between approved and non-approved services and modes of operation.

Degraded Mode Description:

N/A as the module does not have a degraded mode of operation.

2.5 Algorithms

Approved Algorithms:

The tables below list the approved algorithms used in the module. Only those algorithms/modes specified in the table below are utilized by the module. The module may not use some of the capabilities described in each CAVP certificate. See Section 13 Standards References for links to the standards referenced in the tables below.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6701	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A6701	Key Length - 128, 192, 256	SP 800-38C
AES-CFB128	A6701	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A6701	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A6701	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A6701	Direction - Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A6701	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A6701	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A6701	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
AES-XTS Testing Revision 2.0	A6701	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A6701	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
cSHAKE-128	A6701	Message Length - Message Length: 0-65536 Increment 8	SP 800-185
cSHAKE-256	A6701	Message Length - Message Length: 0-65536 Increment 8	SP 800-185
DSA PQGVer (FIPS186-4)	A6701	L - 2048, 3072 N - 256 Hash Algorithm - SHA2-256	FIPS 186-4
DSA SigVer (FIPS186-4)	A6701	L - 2048, 3072 N - 256 Hash Algorithm - SHA2-256	FIPS 186-4
ECDSA KeyGen (FIPS186-5)	A6701	Curve - P-256, P-384, P-521 Secret Generation Mode - extra bits	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A6701	Curve - P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A6701	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512 Component - No, Yes	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A6701	Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
HMAC-SHA-1	A6701	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-256	A6701	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-384	A6701	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512	A6701	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA3-256	A6701	Key Length - Key Length: 112-1088 Increment 8	FIPS 198-1
HMAC-SHA3-384	A6701	Key Length - Key Length: 112-832 Increment 8	FIPS 198-1
HMAC-SHA3-512	A6701	Key Length - Key Length: 112-576 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC Sp800-56Ar3	A6701	Domain Parameter Generation Methods - P-256, P-384, P-521 Function - Full Validation, Key Pair Generation, Partial Validation Scheme - ephemeralUnified - KAS Role - Initiator 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	A6701	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC Sp800-56Ar3	A6701	Domain Parameter Generation Methods - 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	A6701	Domain Parameter Generation Methods - ffdhe2048, MODP-2048 Scheme - dhEphem - KAS Role - initiator, responder dhOneFlow - KAS Role - initiator, responder dhStatic - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA OneStep SP800-56Cr2	A6701	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8	SP 800-56C Rev. 2
KDF SP800-108	A6701	KDF Mode - Counter Supported Lengths - Supported Lengths: 160-256 Increment 8	SP 800-108 Rev. 1
KDF SRTP (CVL)	A6701	AES Key Length - 128, 192, 256	SP 800-135 Rev. 1
KDF SSH (CVL)	A6701	Cipher - AES-128, AES-192, AES-256 Hash Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF TLS (CVL)	A6701	TLS Version - v1.0/1.1 Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1
KMAC-128	A6701	Message Length - Message Length: 0-65536 Increment 8 Key Data Length - Key Data Length: 128-65536 Increment 8	SP 800-185
KMAC-256	A6701	Message Length - Message Length: 0-65536 Increment 8 Key Data Length - Key Data Length: 128-65536 Increment 8	SP 800-185
PBKDF	A6701	Iteration Count - Iteration Count: 10-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA Decryption Primitive Sp800-56Br2 (CVL)	A6701	Modulo - 2048	SP 800-56B Rev. 2
RSA KeyGen (FIPS186-5)	A6701	Key Generation Mode - probable Modulo - 2048, 3072, 4096 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A6701	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA Signature Primitive (CVL)	A6701	Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A6701	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-5)	A6701	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
Safe Primes Key Generation	A6701	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, MODP-2048, MODP-3072, MODP-4096, MODP-6144	SP 800-56A Rev. 3
SHA-1	A6701	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A6701	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A6701	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A6701	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-256	A6701	Message Length - Message Length: 0-65336 Increment 8	FIPS 202
SHA3-384	A6701	Message Length - Message Length: 0-65336 Increment 8	FIPS 202
SHA3-512	A6701	Message Length - Message Length: 0-65336 Increment 8	FIPS 202
SHAKE-128	A6701	Output Length - Output Length: 16-65528 Increment 8	FIPS 202
SHAKE-256	A6701	Output Length - Output Length: 16-65528 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A6701	Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1

Table 7: Approved Algorithms

Vendor-Affirmed Algorithms:

The following table presents the vendor-affirmed algorithms.

Name	Properties	Implementation	Reference
CKG [SP800-133rev2]	Key Type: Symmetric and Asymmetric	N/A	Section 4, example 1. The value of U is directly output without XORing V.

Table 8: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

The following table presents the non-approved, allowed algorithms with no security claimed.

Name	Caveat	Use and Function
MD5	Allowed for use with TLS 1.0/1.1 KDF per FIPS 140-3 IG 2.4.A, example scenario 2.a.	Hashing

Table 9: 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
Poly1305	Message Authentication
ChaCha20	Encryption and Decryption
ChaCha20-Poly1305	Authenticated Encryption
Marvin32	Hashing
HKDF	Key Derivation
AES-CBC-MAC	Message Authentication
MD2, MD4 and MD5	Hashing
SHA2-224, SHA2-512/224, SHA2-512/256 and SHA3-224	Hashing
DES and Triple-DES	Encryption and Decryption
AES-KW and AES-KWP	Encryption and Decryption
DSA key generation and DSA PQG generation	Key Pair Generation
DSA signature generation	Digital Signature Generation
NIST SP 800-56A key establishment Diffie-Hellman and EC-Diffie Hellman prior to revision 3	Key Agreement
SHA-1 hash algorithm for signature generation.	Digital Signature Generation
ECDSA with SHA-1 for signature verification.	Digital Signature Verification
HMAC with key sizes less than 112 bits (14 bytes) for HMAC generation.	Message Authentication
HMAC-MD2, HMAC-MD4, HMAC-MD5, HMAC-SHA2-224, HMAC-SHA2-512-224, HMAC-SHA2-512-256 and HMAC-SHA3-224	Message Authentication
RSA 1024-bit signature generation.	Digital Signature Generation
RSA Encrypt and Decrypt	Encryption and Decryption
RC2 and RC4	Encryption and Decryption
ECDSA with the following curves and strengths: nistP192 (96 bits), curve25519 (128 bits), numsP256t1 (128 bits), numsP384t1 (192 bits), numsP512t1 (256 bits). See section 2.7.2 Elliptic Curve Specification Reference for links to the specifications of each curve.	Digital Signature Generation
LMS, XMSS, XMSS^MT, ML-DSA	Key Generation, Signature Generation and Signature Verification
ML-KEM	Key Encapsulation, Key Decapsulation and Key Pair Generation

Table 10: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

The following table lists the security function implementations present in the module, organized according to the approved security function types listed in NIST SP 800-140Cr1. See section 9 Sensitive Security Parameters Management for details on the keys referenced by the security function implementation descriptions

Name	Type	Description	Properties	Algorithms
Asymmetric Key Generation	AsymKeyPair-KeyGen CKG	Asymmetric Key Generation function used by the Key-Pair Generation service.	Standard: NIST SP 800-133 Rev. 2 (Sections 5.1 and 5.2) key generation.	RSA KeyGen (FIPS186-5): (A6701) Modulo: 2048, 3072 and 4096 bits ECDSA KeyGen (FIPS186-5): (A6701) Curves: P-256, P-384 and P-521 Safe Primes Key Generation: (A6701) Safe primes groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, MODP-2048, MODP-3072, MODP-4096 and MODP-6144
Asymmetric Key Verification	AsymKeyPair-KeyVer	Asymmetric Key Pair Verification for DSA and ECDSA		ECDSA KeyVer (FIPS186-5): (A6701) Curves: P-256, P-384 and P-521 DSA PQGVer (FIPS186-4): (A6701) DSA Keys: (2048, 256) and (3072, 256)

Name	Type	Description	Properties	Algorithms
BC1	BC-UnAuth	Symmetric block cipher function used by the Encryption and Decryption service.		AES-CBC: (A6701) Key Length: 128, 192, 256 AES-CFB128: (A6701) Key Length: 128, 192, 256 AES-CFB8: (A6701) Key Length: 128, 192, 256 AES-CTR: (A6701) Key Length: 128, 192, 256 AES-ECB: (A6701) Key Length: 128, 192, 256 AES-XTS Testing Revision 2.0: (A6701) Key Length: 128, 256
Authenticated Encryption/Decryption	BC-Auth	Block Cipher Symmetric Encryption/Decryption Authenticated.		AES-CCM: (A6701) Key Length: 128, 192, 256 AES-GCM: (A6701) Key Length: 128, 192, 256
CKG1	CKG	Section 4: Using the Output of a Random Bit Generator. Section 5.1: Generation of Key Pairs for Digital Signature Schemes. Section 5.2: Generation of Key Pairs for Key Establishment. Section 6.1: Direct Generation of Key Pairs for Symmetric Algorithms. Section 6.2: Derivation of Symmetric Keys.		CKG [SP800-133rev2]: ()

Name	Type	Description	Properties	Algorithms
Digital Signature Generation	DigSig-SigGen	Digital Signature Generation using ECDSA or RSA		ECDSA SigGen (FIPS186-5): (A6701) Curve: P-256, P-384, P-521 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 RSA SigGen (FIPS186-5): (A6701) Signature Type : PKCS 1.5 and PSS Modulo: 2048, 3072, 4096 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 RSA Signature Primitive: (A6701) Modulo: 2048, 3072, 4096 SHA2-256: (A6701) SHA2-384: (A6701) SHA2-512: (A6701)

Name	Type	Description	Properties	Algorithms
Digital Signature Verification	DigSig-SigVer	Digital Signature Verification using ECDSA, RSA or DSA		DSA SigVer (FIPS186-4): (A6701) Key pair: (2048, 256), (3072, 256) Hash Algorithm: SHA2-256 ECDSA SigVer (FIPS186-5): (A6701) Curve: P-256, P-384, P-521 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 RSA SigVer (FIPS186-5): (A6701) Signature Type: PKCS 1.5 and PSS Modulo: 2048, 3072, 4096 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 RSA SigVer (FIPS186-4): (A6701) Signature Type: PKCS 1.5 and PSS Modulo: 1024, 2048, 3072, 4096 Hash Algorithm: SHA-1, SHA2-256, SHA2-384, SHA2-512
DRBG	DRBG	Deterministic random bit generator		Counter DRBG: (A6701) Mode: AES-256
ENT1	ENT-ESV	Non-Physical Entropy Source used by the Random Number Generation service.		
ENT2	ENT-ESV	Physical Entropy Source used by Intel Ice Lake RSEED Entropy.		

Name	Type	Description	Properties	Algorithms
ENT3	ENT-ESV	Physical Entropy Source used by Intel Coffee Lake RSEED Entropy		
KAS-ECC	KAS-SSC/KDF	Full Key Agreement scheme	IG: IG D.F scenario 2, path (2), end-to-end Key confirmation:No Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 128 and 256 encryption	KAS-ECC Sp800-56Ar3: (A6701) Curves: P-256, P-384, P-521
KAS-ECC-SSC	KAS-SSC/KDF	Key Agreement Shared Secret Computation	IG: IG D.F scenario 2, path (2), split. Key confirmation:No Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 112 and 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3: (A6701) Curve: P-256, P-384, P-521

Name	Type	Description	Properties	Algorithms
KAS-FFC	KAS-SSC/KDF	Full Key Agreement scheme	IG: IG D.F scenario 2, path (2), end-to-end. Key confirmation:No Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides between 112 and 152 bits of encryption strength	KAS-FFC Sp800-56Ar3: (A6701) Domain Parameter Generation Methods: ffdhe2048, ffdhe3072, ffdhe4096, MODP-2048, MODP-3072, MODP-4096
KAS-FFC-SSC	KAS-SSC/KDF	Key Agreement Shared Secret Computation	IG: IG D.F scenario 2, path (2), split. Key confirmation:No Key derivation:IG 2.4.B SP 800-135rev1 CVL and KDA (separately tested) Caveat:Key establishment methodology provides 112 bits of encryption strength	KAS-FFC-SSC Sp800-56Ar3: (A6701) Domain Parameter Generation Methods: ffdhe2048, MODP-2048
KDF1	KAS-135KDF	TLS Key Derivation function used by the Secret Agreement and Key Derivation service.		KDF TLS: (A6701) Hash Algorithm: SHA2-256, SHA2-384 TLS v1.2 KDF RFC7627: (A6701) Hash Algorithm: SHA2-256, SHA2-384
KDF2	KAS-135KDF	SSH Key Derivation used by the Secret Agreement and Key Derivation service.		KDF SSH: (A6701)

Name	Type	Description	Properties	Algorithms
KBKDF1	KBKDF	Key Based Derivation Function		KDF SP800-108: (A6701) KDF Mode: Counter MAC Mode: CMAC-AES128, CMAC-AES192, CMAC-AES256, HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512 Supported Lengths: 160-256 Increment 8
Key Derivation 56Crev2	KAS-56CKDF	Key Derivation using SP 800-56Crev2		KDA OneStep SP800-56Cr2: (A6701) Auxiliary Function Methods: SHA-1, SHA2-256, SHA2-384, SHA2-512, HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512, KMAC-128, KMAC-256
MAC1	MAC	Message Authentication Computation with AES CMAC or AES GMAC		AES-CMAC: (A6701) Key Length: 128, 192, 256 AES-GMAC: (A6701) Key Length: 128, 192, 256

Name	Type	Description	Properties	Algorithms
MAC2	MAC	Message Authentication Computation with HMAC or KMAC		HMAC-SHA-1: (A6701) HMAC-SHA2-256: (A6701) HMAC-SHA2-384: (A6701) HMAC-SHA2-512: (A6701) HMAC-SHA3-256: (A6701) HMAC-SHA3-384: (A6701) HMAC-SHA3-512: (A6701) KMAC-128: (A6701) KMAC-256: (A6701)
Message Digest	SHA XOF	Secure Hash		cSHAKE-128: (A6701) cSHAKE-256: (A6701) SHA-1: (A6701) SHA2-256: (A6701) SHA2-384: (A6701) SHA2-512: (A6701) SHA3-256: (A6701) SHA3-384: (A6701) SHA3-512: (A6701) SHAKE-128: (A6701) SHAKE-256: (A6701)

Name	Type	Description	Properties	Algorithms
PBKDF2	PBKDF	PBKDF2 Key Derivation		PBKDF: (A6701) HMAC Algorithm: SHA-1, SHA2-256, SHA2-384, SHA2-512, SHA3-256, SHA3-384, SHA3-512 Iteration Count: 10-10000 Increment 1 Password Length: 8-128 Increment 1 Salt Length: 128-4096 Increment 8
KDF3	KAS-135KDF	SRTP Key Derivation function used by the Secret Agreement and Key Derivation service.		KDF SRTP: (A6701)
RSADP	AsymKeyPair-Decap	Key Transport component function used by the Encryption and Decryption service		RSA Decryption Primitive Sp800-56Br2: (A6701) Modulus Length: 2048 bits

Table 11: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 Elliptic Curve Specification Reference

The table below lists the specification references for each elliptic curve named in the non-approved algorithm tables in section 2.5 Algorithms.

Curve	Specification
Curve25519	https://cr.yp.to/ecdh/curve25519-20060209.pdf
nistP192	http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf
numsP256t1	https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/curvegen.pdf
numsP384t1	https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/curvegen.pdf
numsP512t1	https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/curvegen.pdf

Table 12 - Elliptic Curves References

2.7.2 AES-GCM

The module's implementation of AES-GCM complies with the Implementation Guidance Scenario 1(a) for TLS 1.2 protocol per RFC7627, Scenario 1(d) for SSHv2 protocol per RFC4252, RFC4253 and RFC5647 and Scenario 3 of FIPS 140-3 IG C.H. The module does not implement the TLS and SSH protocols itself, however,

it provides the cryptographic functions required for implementing the protocols. AES GCM encryption is used in the context of SSH and TLS protocols.

When the IV exhausts the maximum number of possible values for a given session key, this results in a failure in encryption and a handshake to establish a new encryption key will be required. It is the responsibility of the user of the module, i.e., the first party, client or server, to encounter this condition, to trigger this handshake in accordance with the TLS/SSH protocol.

The initialization vector (IV) bit size is 96 bits. In the event the module's power is lost and restored, the calling application must ensure that a new key is established for use with AES-GCM key encryption or decryption. In addition to the approved AES-GCM, the module also provides a non-approved AES-GCM encryption service which accepts arbitrary external IVs from the operator.

The Module also supports importing of GCM IVs when an IV is not generated within the Module. In the approved mode, an IV must not be imported for encryption from outside the cryptographic boundary of the Module as this will result in a non-conformance.

2.7.3 AES-XTS

The AES algorithm in XTS mode shall only be used for confidentiality on storage devices, as specified in [SP800-38E]. The module complies with [FIPS 140-3 IG] C.I by implementing a check to ensure that the two AES keys used in AES-XTS algorithm are not equal. This check is performed before using the keys in the AES-XTS algorithm to process data with them. AES-XTS keys (i.e., Key_1 and Key_2) entered into the module shall be generated and/or established independently according to NIST SP 800-133rev2, Section 6.3. for an approved use of AES-XTS.

2.7.4 FIPS 202 Usage

The module is compliant to the [FIPS 140-3 IG] C.C regarding the SHA-3 family algorithms.

All the SHA-3 functions have been tested and validated with the CAVP tool (#A6701), as it is indicated in Table 7. In addition, every higher-level algorithm that use a SHA-3 family algorithm are also validated with the CAVP, together in the same certificate (#A6701).

2.7.5 RSA Usage

The module is compliant with [FIPS 140-3 IG] C.F regarding RSA digital signature.

RSA digital signature generation can be performed with 2048-, 3072- or 4096- bits modulus length. As it is indicated in Table 7, all the RSA signature algorithm implementations are tested with the CAVP (Cert. #A6701), and all the approved modulus length are validated for the signature generation.

For the signature verification, all the different RSA [FIPS 186-5] modulus length implement (2048-, 3072- and 4096- bits) by the module are validated with the CAVP (Cert. #A6701). For the signature verification with RSA [FIPS 186-4] all the possible modulus length (1024-, 2048-, 3072- and 4096- bits) have also been validated with the CAVP (Cert. #A6701).

2.7.6 Key Transport

SymCrypt provides services establishing sensitive security parameters (SSP) between two cryptographic modules; it does not implement a 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.

2.7.7 SHA-1 Usage

SymCrypt provides SHA-1 for use by a calling application as a stand-alone security function for hashing. Internally, use of SHA-1 is limited to the following:

- As underlying algorithm for HMAC-SHA-1.
- As auxiliary function for KDF SSH and KDA OneStep 800-56Cr2.
- To support legacy DSA, ECDSA, RSA digital signature verification functions.

Per SP800-131Ar2, the use of SHA-1 is deprecated for digital signature generation, but is permitted for digital signature verification (legacy use) and all non-digital signature applications through December 31, 2030. After December 31, 2030, any use of SHA-1 will be deprecated.

2.8 RBG and Entropy

The following tables present the entropy certificates and entropy source details for the entropy source used by the random number generation service.

Cert Number	Vendor Name
E257	Microsoft Corporation
E272	Microsoft Corporation
E273	Microsoft Corporation

Table 13: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Jitter Entropy	Non-Physical	SONiC release 202305, Metaswitch Linux 8, Azure Linux 3.0	256	Full Entropy	A6568
Intel Coffee Lake RDSEED Entropy	Physical	Ubuntu 20.04 on Intel Xeon E-2288G	128	Full Entropy	A2151
Intel Ice Lake RSEED Entropy	Physical	Ubuntu 20.04 on Intel 3rd Generation Xeon Scalable 8370C	128	Full Entropy	A2518

Table 14: Entropy Sources

2.8.1 RBG Information and Output

The entropy source uses either the Jitter DRBG or the Intel RDSEED instruction for OpenEnclave operational environments.

The Jitter DRBG entropy source consists of a non-physical noise source and supplemented with a vetted conditioning component. It is based on version 3.3.1 of Jitter RNG. This noise source follows the non-IID track and makes no IID claims. The entropy source does not expose any configuration settings to the user. All parameterization and settings are handled by the system at startup.

Each 256-bit output sample contains 256 bits of entropy, or full entropy. This output is conditioned via SHA3-256 hash function.

In Linux modules, AES-CTR DRBG is used and sources entropy from various origins:

- In OpenEnclave, the RDSEED instruction provides FIPS-compliant entropy, supplemented by additional entropy from the platform via oe_sgx_get_additional_host_entropy.
- For other Linux modules, jitterentropy serves as the FIPS-compliant entropy source, with getrandom providing additional platform entropy.

The Intel RDSEED instructions draw random numbers from an on-die SP800-90A and SP800-90B compliant hardware random number generator, the Digital Random Number Generator (DRNG). It provides 128-bit full-entropy outputs.

These entropy sources are used to initialize the SymCrypt deterministic random number generator; the conditioned entropy is not made accessible to users through an external interface. The entropy source does not provide an external interface to obtain raw noise. The entropy source does not provide an interface to trigger an on-demand health test. Restarting the cryptographic module will restart the entropy source and so it will execute all health tests again.

2.9 Key Generation

For RSA, and ECDSA key pairs, the module implements approved key generation services compliant with [FIPS 186-5] where the key material is directly obtained from approved [SP 800-90Arev1] DRBG according to [SP 800-133rev2].

The public and private key pairs used in the Diffie-Hellman and EC Diffie-Hellman KAS are compliant with NIST [SP800-56Arev3].

The module uses an Approved CTR DRBG specified in [SP800-90Arev1] to generate random cryptographic material. The resulting generated material are the unmodified outputs from the DRBG.

According to FIPS 140-3 Implementation Guidance D.H, a component key generation (CKG) using the unmodified output of an approved DRBG can be used to generate cryptographic material for:

- Direct Generation of symmetric keys per section 6.1 of the [SP800-133rev2].
- Derivation of symmetric keys per section 6.2 of the [SP800-133rev2].
- Asymmetric Key Pair Generation for Digital Signature per section 5.1 of the [SP800-133rev2].
- Asymmetric Key Pair Generation for Key Establishment per section 5.2 of the [SP800-133rev2].

During the SSP generation, services are not available, and input and output are inhibited.

For symmetric keys, the caller supplies random bytes through the pbKey parameter, which are then expanded into the complete symmetric key. These input bytes can be:

- Generated by SymCryptRandom.
- Derived from an approved Key Derivation Function (KDF).

2.10 Key Establishment

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.

The module provides different methods of key agreement and key derivation.

2.10.1 Key Agreement

To meet the requirements of Section 5.6.2 of SP 800-56Ar3, the operator must use the module with a TLS protocol application. During the “Key pair generation” service, the module will internally validate the generated public key. Additionally, the module’s shared secret computation service will validate the peer public key, in line with Sections 5.6.2.2.1 and 5.6.2.2.2 of SP 800-56Ar3.

The module provides Diffie-Hellman and EC Diffie-Hellman key agreement methods. The security strength of the preceding algorithms is as follows:

1. Diffie-Hellman key agreement and the shared secret computation provide between 112 and 152 bits of encryption strength.
2. EC Diffie-Hellman key agreement and the shared secret computation provide between 112 and 256 bits of encryption strength.

Diffie-Hellman and EC Diffie-Hellman key agreement are under scenario 2, path 2 of [FIPS 140-3 IG] D.F.

2.10.2 Key Derivation

The module supports the following SSP Derivation methods:

- [SP800-108rev1] KDF Counter mode.
- Password-Based Key Derivation (PBKDF2) based on [SP800-132] option 1a.
- Protocol-Suite Key Derivation: TLS v1.0/1.1 KDF, TLS v1.2 KDF, SRTP KDF, SSH KDF.
- Key Derivation based on SP800-56Crev2: KDA OneStep.

The module supports the following KDF in the Non-Approved mode:

- Key Derivation based on SP800-56Crev2: HKDF.

For the protocol-suite key derivation functions TLS v1.1 KDF (CVL), TLS v1.2 KDF (CVL), SRTP KDF (CVL) and SSHv2 KDF (CVL); they shall only be used within the context of the TLS, SRTP and SSH protocols.

Regarding PBKDF2, in line with the requirements for [SP800-132], keys generated using the approved PBKDF2 must only be used for storage applications. Any other use of the approved PBKDF2 is non-conformant. The security strength of the derived key is at least 112 bits.

As the module is a general-purpose software module, it is not possible to anticipate all the ways PBKDF2 may be used, however a user of the module should also note that a password should at least contain enough security strength to be unguessable and also contain enough strength to reflect the security strength required for the key being generated. The supported lengths of a password/passphrase used can range between 8 and 128 bytes.

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

For the iteration count, as the functionality of the module relies on the usage that a user performs with the module, it is recommended a minimum of 1.000 iterations. The iteration count values used range from 10 to 10,000 per [SP800-132] Section 5.2 whereby the iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users.

In addition, users are referred to Appendix A, “Security Considerations” in [SP800-132] for further information on password, salt, and iteration count selection.

2.11 Industry Protocols

While SymCrypt does not contain a TLS implementation, a TLS developer can use cryptographic primitives in SymCrypt to construct a TLS 1.2 client or server incorporating any of the ciphersuites specified in section 3.3.1 of [SP800-52].

2.12 Additional Information

1. DSA PQGVer and DSA sigVer can only be used for legacy purposes.
2. RSA signature verification using SHA-1 is used for legacy signature verification only.
3. 1024-bit RSA key is used for legacy signature verification only.
4. The RSA Decryption Primitive SP800-56Brev2 (CVL) shall only be used within the context of a SP 800-56Brev2 key transport service.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

As a software 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. The module logical interface is a C language Application Program Interface (API) that allows the calling application to request services. The module logical interfaces are described in the table below.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Platform API input parameters
N/A	Data Output	Platform API output parameters
N/A	Control Input	Platform API function calls, API control input parameters.
N/A	Status Output	Platform API return codes and error messages

Table 15: Ports and Interfaces

The module does not implement either control output interface or power input interface.

3.2 Trusted Channel Specification

This module does not have a trusted channel, so the requirement is not applicable.

3.3 Control Interface Not Inhibited

The module control interface is always inhibited when the module in its error state, so this requirement is not applicable.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module is not required to authenticate for a Level 1 validation, so this requirement is not applicable.

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 16: Roles

4.3 Approved Services

The following tables present the approved services of the module. Due to the number of columns specified in SP 800-140Br1 for the approved services table, it is presented below in two parts. The first table includes the service names, descriptions, indicators, inputs, and outputs; the second table lists the SFIs, roles, and roles SSP access for each service.

The service indicator is invoked by calling the function SymCryptDeprecatedStatusIndicator() and it returns a list of SymCrypt functions and services. The list identifies specifically which functions are approved (list as “approved” in the service indicator return). The return string from the SymCryptDeprecatedStatusIndicator() allows the user of the module to determine which algorithms are approved or non-approved, and what parameters/flags to a security function result in an approved or non-approved algorithm execution. An additional API function, SymCryptDeprecatedServiceIndicator(),allows to indicate specific arguments depending on the service to check. This API call will return a 0 if it is an approved service or a non-zero value for non-approved services.

Please note that the Sensitive Security Parameters (SSPs) listed below indicate the type of access using the following notation:

- G – Generate: The SSP is generated or derived.
- R – Read: The SSP is read from the module.
- W – Write: The SSP is updated, imported, or written to the module.
- E – Execute: The SSP is used within an Approved security function.
- Z – Zeroize: The SSP is zeroized.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Initialization	Perform the initialization of the module	None	Module Default Entry Point	None	None	Cryptographic Officer (CO)
Self-Tests	Perform pre-operational and conditional self-tests	Return code from function call	Power cycle, and API call parameters	Success is implicit in module availability. If any self-test fails, the module returns an error code	BC1 Authenticated Encryption/Decryption Digital Signature Generation Digital Signature Verification DRBG KAS-ECC-SSC KAS-FFC-SSC KDF1 KDF2 KBKDF1 Key Derivation 56Crev2 MAC1 MAC2 Message Digest PBKDF2 KDF3	Cryptographic Officer (CO)
Show Status	Provide module status	None	This service is fully automatic and occurs when any function is called. If function SymCryptDeprecatedServiceIndicator() is used, inputs are the last algorithm invoked together with the parameters used.	0 for Approved mode or 1 for non-Approved mode.	None	Cryptographic Officer (CO)

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show Version	Display the module name and version	None	API call parameters	Module Name and Version	None	Cryptographic Officer (CO)

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Asymmetric Key Generation/Verification	Generate asymmetric key pairs (RSA, ECDSA, DH, ECDH)	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: Curve, modulus, group, key length. Key pair to be verified	Status, Generated private and public key pair	Asymmetric Key Generation Asymmetric Key Verification CKG1	Cryptographic Officer (CO) - DH Private Key: G,R,E - DH Public Key: G,R,E - ECDH Public Key: G,R,E - ECDH Private Key: G,R,E - ECDSA Public Keys: G,R,E - ECDSA Private Keys: G,R,E - RSA Public Keys: G,R,E - RSA Private Keys: G,R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Encryption and Decryption	Encrypts and decrypts a block of data or decrypting data encrypted by an RSA private key as part of RSA Decryption Primitive	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: key, IV, plaintext/ciphertext	Status, ciphertext/plaintext	BC1 Authenticated Encryption/Decryption RSADP	Cryptographic Officer (CO) - AES Keys: W,E - RSA Private Keys: W,E
Key Derivation	Derive keying material using PBKDF, SP800-56Crev2 OneStep KDA, SSHv2, TLS 1.1/1.2 KDFs, SRTP, SP 800-108 KDF.	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: Shared Secret, additional info depending on the algorithm used.	Status and derived keying material.	CKG1 KDF1 KDF2 KBKDF1 Key Derivation 56Crev2 PBKDF2 KDF3	Cryptographic Officer (CO) - PBKDF2 password: W,E - PBKDF2 salt: W,E - Keying Material: G - Shared Secret: W,E - TLS Pre-master secret: W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Keyed Hash	Generate or verify data integrity with HMAC/KMAC	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: HMAC/KMAC key, message, keyed hash value (for verification)	Status, Keyed Hash value (Generation), True or False (Verification)	MAC2	Cryptographic Officer (CO) - HMAC Keys: W,E - KMAC Key: W,E
Message Authentication	Generation or verify data integrity with CMAC and GMAC	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: key, data, authenticated message digest to verify.	Status, authenticated message digest.	MAC1	Cryptographic Officer (CO) - AES Keys: W,E
Message Digest	Compute and return a message digest using SHS and SHA-3 algorithms	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: message	Status, hash value	Message Digest	Cryptographic Officer (CO)

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Random Number Generation	Fills a buffer with random bytes using the AES-256 CTR mode DRBG.	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: number of bits to be generated.	Status and random bitstring	DRBG ENT1 ENT2 ENT3	Cryptographic Officer (CO) - AES-CTR DRBG Entropy Input: W,E,Z - AES-CTR DRBG Seed: W,E,Z - AES-CTR DRBG V: G,E - AES-CTR DRBG Key: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Secret Agreement	Provides key agreement.	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: private key, counter public key	Status, key components, key	KAS-ECC KAS-ECC-SSC KAS-FFC KAS-FFC-SSC KDF1 Key Derivation 56Crev2	Cryptographic Officer (CO) - DH Private Key: W,E - DH Public Key: W,E - ECDH Public Key: W,E - ECDH Private Key: W,E - Shared Secret: G

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Digital Signature Generation and Verification	Generate or verify RSA and ECDSA digital signatures. Verification of DSA signatures.	Return value from SymCryptDeprecatedServiceIndicator()	API call parameters: key pair, message, signature (for verification)	Status, signature	Digital Signature Generation Digital Signature Verification	Cryptographic Officer (CO) - DSA Public Key: W,E - ECDSA Public Keys: W,E - ECDSA Private Keys: W,E - RSA Public Keys: W,E - RSA Private Keys: W,E

Zeroization	Zeroizes cryptographic material.	Return code from function call	Memory pointer	None	None	Cryptographic Officer (CO) - AES Keys: Z - AES-CTR DRBG Entropy Input: Z - AES-CTR DRBG Seed: Z - AES-CTR DRBG V: Z - AES-CTR DRBG Key: Z - DSA Public Key: Z - DH Private Key: Z - DH Public Key: Z - ECDH Public Key: Z - ECDH Private Key: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- ECDSA Public Keys: Z - ECDSA Private Keys: Z - HMAC Keys: Z - KMAC Key: Z - PBKDF2 password: Z - PBKDF2 salt: Z - Keying Material: Z - RSA Public Keys: Z - RSA Private Keys: Z - Shared Secret: Z - TLS Pre-master secret: Z

Table 17: Approved Services

4.4 Non-Approved Services

The following table identifies the non-approved 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, sub-section Non-Approved, Not Allowed Algorithms.

Name	Description	Algorithms	Role
Non-Approved Secret Agreement	Secret Agreement using one of the non-approved algorithms listed to the right.	NIST SP 800-56A key establishment Diffie-Hellman and EC-Diffie Hellman prior to revision 3 ML-KEM	CO
Non-Approved Key Derivation	Key derivation using a non-approved algorithm.	HKDF	CO
Non-Approved Key-Pair Generation	Key-pair generation using non-approved algorithms.	DSA key generation and DSA PQG generation ECDSA with the following curves and strengths: nistP192 (96 bits), curve25519 (128 bits), numsP256t1 (128 bits), numsP384t1 (192 bits), numsP512t1 (256 bits). See section 2.7.2 Elliptic Curve Specification Reference for links to the specifications of each curve. LMS, XMSS, XMSS^MT, ML-DSA ML-KEM	CO
Non-Approved Hashing	Hashing using non-approved algorithms.	Marvin32 MD2, MD4 and MD5 SHA2-224, SHA2-512/224, SHA2-512/256 and SHA3-224	CO
Non-Approved Message Authentication	Message authentication using non-approved algorithms.	Poly1305 AES-CBC-MAC HMAC with key sizes less than 112 bits (14 bytes) for HMAC generation. HMAC-MD2, HMAC-MD4, HMAC-MD5, HMAC-SHA2-224, HMAC-SHA2-512-224, HMAC-SHA2-512-256 and HMAC-SHA3-224	CO
Non-Approved Encryption and Decryption	Encryption and decryption using non-approved algorithms.	ChaCha20 ChaCha20-Poly1305 DES and Triple-DES AES-KW and AES-KWP RSA Encrypt and Decrypt RC2 and RC4	CO
Non-Approved Signing and Verification	Signing and verification using non-approved algorithms.	DSA signature generation SHA-1 hash algorithm for signature generation. ECDSA with SHA-1 for signature verification. RSA 1024-bit signature generation. ECDSA with the following curves and strengths: nistP192 (96 bits), curve25519 (128 bits), numsP256t1 (128 bits), numsP384t1 (192 bits), numsP512t1 (256 bits). See section 2.7.2 Elliptic Curve Specification Reference for links to the specifications of each curve. LMS, XMSS, XMSS^MT, ML-DSA	CO

Table 18: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not load external software or firmware. The operating system environment enforces process isolation, including memory (where keys and intermediate key data are stored) and CPU scheduling.

5 Software/Firmware Security

5.1 Integrity Techniques

When the module is compiled the build process a random HMAC key and places it into the executable. During runtime initialization, the module verifies the integrity of the executable by computing an HMAC-SHA2-256 digest for the executable and then comparing with the HMAC-SHA2-256 digest computed at build time. If the digests match, the conditional self-tests are performed.

5.2 Initiate on Demand

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

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The module is a software module. It is operated in a modifiable operational environment per FIPS 140-3 level 1 specifications. This modifiable operational environment for the SymCrypt module is listed in section 2.2 Tested and Vendor Affirmed Module Version and Identification. The SymCrypt module is loaded into process memory for a single application. Concurrent operators are explicitly excluded. The “single operator” for the module is the identity associated with the process that executes libsymcrypt.so. The calling application is responsible for ensuring that CSPs are not shared between approved and non-approved services and modes of operation.

The OS manages all cryptographic keys and SSPs, ensuring the protection of CSPs from unauthorized access, use, disclosure, modification, and substitution, as well as safeguarding PSPs from alteration and replacement. The OS allocates a dedicated process address space for each running process, with the module functioning entirely within the process address space of the calling application. Within this allocated memory, the module manages its own SSPs and provides access to them through a well-defined API. The module is unable to spawn new processes and does not support concurrent operators.

7 Physical Security

As a software module, the Physical Security requirements are not applicable for this module.

8 Non-Invasive Security

The module does not claim any non-invasive attack mitigation techniques to protect the unprotected SSPs from non-invasive attacks.

9 Sensitive Security Parameters Management

9.1 Storage Areas

The module does not directly persist SSPs. 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
RAM	Volatile SSPs are temporarily stored in the computer's memory (see: RAM in the block diagram).	Dynamic

Table 19: Storage Areas

9.2 SSP Input-Output Methods

Each time an application links with the SymCrypt module, the .so file is instantiated in the context of the application's process, therefore no keys exist within the application's process space. The user application is responsible for importing keys into the module and using the module's functions to generate keys.

Keys may be exported out of and imported into the module via the SymCrypt*GetValue and SymCrypt*SetValue functions. Symmetric key entry outputs the established key as a byte sequence to the user, which can then be imported as any other symmetric key.

To prevent the inadvertent output of sensitive information, the following internal actions take place in order to output any plaintext CSP: 1) allocating memory of the necessary context to request the service, 2) processing the service request which outputs CSPs using the created context.

The following API functions are used for importing or exporting keys:

- Key Import:
  - SymCryptDlkeySetValue
  - SymCryptEckeySetValue
  - SymCryptRsakeySetValue
- Key Export:
  - SymCryptDlkeyGetValue
  - SymCryptEckeyGetValue
  - SymCryptRsakeyGetValue

The table below provides additional details on key input and output.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
SSP Input	App via TOEPP path	RAM	Plaintext	Manual	Electronic	
SSP Output	RAM	App via TOEPP path	Plaintext	Manual	Electronic	

Table 20: SSP Input-Output Methods

9.3 SSP Zeroization Methods

The module zeroizes SSPs using the method described in the table below. During the zeroization process, services are unavailable, and the input and output interfaces are disabled. Zeroization of the SSPs begins immediately after the zeroization command is invoked. These techniques take effect instantly, preventing any compromise of plaintext secrets, private keys, and CSPs.

Zeroization Method	Description	Rationale	Operator Initiation
API call	Operators may choose to zeroize all CSPs by executing a specific API call.	Zeroization is performed by overwriting the memory area with zeros using a forced inline function to minimize execution time. Because the memory word is explicitly set to '0', it is not necessary to do a read-back test.	By invocation through API call (SymCryptWipe).
Power Cycle	Operators may choose to zeroize all CSPs by rebooting, reformatting, and overwriting the storage media for the operating system	Zeroization is performed by overwriting the memory area with zeros using a forced inline function to minimize execution time. Because the memory word is explicitly set to '0', it is not necessary to do a read-back test.	Restart the module

Table 21: SSP Zeroization Methods

9.4 SSPs

The following tables present the details of the SSPs used by the module. Due to the number of columns, the information is split between two tables. The Used By column references services named in 4.3 Approved Services, the Inputs / Outputs column references 9.2 SSP Input-Output Methods, the Storage column references 9.1 Storage Areas, and the Zeroization column references 9.3 SSP Zeroization Methods.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Keys	Symmetric keys used for AES encryption/decryption	128, 192 or 256 bits - 128, 192 or 256 bits	Symmetric Key - CSP			BC1 Authenticated Encryption/Decryption MAC1
AES-CTR DRBG Entropy Input	Entropy material for AES_CTR DRBG.	256 bits - 256 bits	DRBG Material - CSP	ENT1 ENT2 ENT3		DRBG
AES-CTR DRBG Seed	Seed material for AES_CTR DRBG.	384 bits - 384 bits	DRBG Material - CSP	ENT1 ENT2 ENT3		DRBG
AES-CTR DRBG V	DRBG material	128 bits - 128 bits	DRBG Material - CSP	DRBG		DRBG
AES-CTR DRBG Key	DRBG material	256 bits - 256 bits	DRBG Material - CSP	DRBG		DRBG
DSA Public Key	Used for digital signature verification	2048 or 3072 bits - 112 or 128 bits	Asymmetric Key - PSP			Asymmetric Key Verification Digital Signature Verification
DH Private Key	DH Private Key	From 2048 to 6144 bits - 112, 128, 152 and 184 bits	Asymmetric Key - CSP	Asymmetric Key Generation		KAS-FFC KAS-FFC-SSC
DH Public Key	DH Public Key	From 2048 to 6144 bits - 112, 128, 152 and 184 bits	Asymmetric Key - PSP	Asymmetric Key Generation		KAS-FFC KAS-FFC-SSC
ECDH Public Key	ECDH Public Key	Curve sizes: P-256, P-384, or P-521. - From 128 to 256 bits	Asymmetric Key - PSP	Asymmetric Key Generation		KAS-ECC KAS-ECC-SSC
ECDH Private Key	ECDH Private Key	Curve sizes: P-256, P-384, or P-521 - From 128 to 256 bits	Asymmetric Key - CSP	Asymmetric Key Generation		KAS-ECC KAS-ECC-SSC

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
ECDSA Public Keys	Used for digital signature verification and asymmetric key verification	P-256, P-384, or P-521 - From 128 to 256 bits	Asymmetric Key - PSP	Asymmetric Key Generation		Asymmetric Key Verification Digital Signature Verification
ECDSA Private Keys	Used for digital signature generation	P-256, P-384, or P-521 - From 128 to 256 bits	Asymmetric Key - CSP	Asymmetric Key Generation		Digital Signature Generation
HMAC Keys	Used for message authentication	112 bits or greater - 112 bits or greater	HMAC Key - CSP			MAC2
KMAC Key	KMAC Key used for message authentication	128 bits or greater - 128 bits or greater.	KMAC Key - CSP			MAC2
Keying Material	Keying material derived from key derivation function (SP 800-108rev1 KBKDF, SP 800-132 PBKDF, KDA, SP 800-135rev1 KDFs).	Keying material bitstring - Keying material bitstring	Bitstring - CSP	KDF1 KDF2 KBKDF1 Key Derivation 56Crev2 PBKDF2 KDF3		KDF1 KDF2 KBKDF1 Key Derivation 56Crev2 PBKDF2 KDF3
PBKDF2 password	PBKDF2 password	From 8 to 128 characters (bytes) with increments of one character. - From 8 to 128 characters (bytes) with increments of one character.	Password - CSP			PBKDF2
PBKDF2 salt	PBKDF2 salt	From 128 to 4096-bit salt bitstring with increment of 8-bits. - From 128 to 4096-bit salt bitstring with increment of 8-bits.	Bitstring - CSP			PBKDF2
RSA Public Keys	RSA Public Key used for digital signature verification	1024, 2048, 3072, or 4096 bits. - 80, 112, 128, 152 bits.	Asymmetric Key - PSP	Asymmetric Key Generation		Digital Signature Verification

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
RSA Private Keys	RSA Private Key used for digital signature generation	2048, 3072, or 4096 bits. - 112, 128, 152 bits	Asymmetric Key - CSP	Asymmetric Key Generation		Digital Signature Generation
Shared Secret	Shared secret	112 bits or greater - 112 bits or greater	Bitstring - CSP		KAS-ECC-SSC KAS-FFC-SSC	KDF1 KDF2 KBKDF1 Key Derivation 56Crev2 KDF3
TLS Pre-master secret	Used for TLS key derivation	384 bits - 384 bits	Secret key - CSP		KAS-ECC-SSC KAS-FFC-SSC	KDF1

Table 22: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Keys	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
AES-CTR DRBG Entropy Input	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	AES-CTR DRBG Seed:Used With AES-CTR DRBG V:Used With AES-CTR DRBG Key:Used With
AES-CTR DRBG Seed	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	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	Ephemeral	API call Power Cycle	AES-CTR DRBG Entropy Input:Used With AES-CTR DRBG Seed:Used With AES-CTR DRBG Key:Used With
AES-CTR DRBG Key		RAM:Plaintext	Ephemeral	API call Power Cycle	AES-CTR DRBG Entropy Input:Used With AES-CTR DRBG Seed:Used With AES-CTR DRBG V:Used With
DSA Public Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
DH Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	DH Public Key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DH Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	DH Private Key:Paired With
ECDH Public Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDH Private Key:Paired With
ECDH Private Key	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDH Public Key:Paired With
ECDSA Public Keys	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDSA Private Keys:Paired With
ECDSA Private Keys	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	ECDSA Public Keys:Paired With
HMAC Keys	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
KMAC Key	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	
Keying Material	SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	PBKDF2 password:Used With PBKDF2 salt:Used With
PBKDF2 password	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	PBKDF2 salt:Used With Keying Material:Used With
PBKDF2 salt	SSP Input	RAM:Plaintext	Ephemeral	API call Power Cycle	PBKDF2 password:Used With Keying Material:Used With
RSA Public Keys	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	RSA Private Keys:Paired With
RSA Private Keys	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	RSA Public Keys:Paired With
Shared Secret	SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	DH Private Key:Derived From DH Public Key:Derived From ECDH Public Key:Derived From ECDH Private Key:Derived From
TLS Pre-master secret	SSP Input SSP Output	RAM:Plaintext	Ephemeral	API call Power Cycle	

Table 23: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

The module performs pre-operational tests automatically when the module is loaded into memory, without operator intervention. The pre-operational self-tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected. The module transitions to the operational state only after the pre-operational self-tests (and the cryptographic algorithm self-tests, which in this module are executed automatically after the pre-operational self-tests) pass successfully.

Before the pre-operational integrity test is executed, the module performs a conditional self-test for the cryptographic algorithm used in the integrity test (HMAC-SHA2-256). Once the HMAC CAST is passed, the module executes the pre-operational integrity self-test.

During the execution of pre-operational self-tests, services are unavailable, and the input and output interfaces are disabled. The module is not available for use by the calling application until the pre-operational and conditional self-tests are completed successfully. See section 5.1 Integrity Techniques for details on how the module's integrity is checked.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A6701)	256-bit key hardcoded.	KAT	SW/FW Integrity	Continue with the Conditional Algorithm Self-Tests	256-bit module integrity key

Table 24: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Conditional self-tests are performed by the cryptographic module when conditions specified for the following tests occurs: Cryptographic Algorithm Self-Tests, Pair-Wise Consistency Test and Critical Function Tests.

The module does not implement any functions requiring a Software/Firmware Load Test, Manual Entry Tests nor Conditional Bypass Test; therefore, these tests are not performed by the module.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A6701)	256-bit key	KAT	CAST	Module available	HMAC CAST before module integrity test.	Power-up and On-demand.
AES-ECB Encrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Encrypt	Run when module is loaded via the default entry point and on-demand.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB Decrypt (Inverse Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Decrypt	Run when module is loaded via the default entry point and on-demand.
AES-CBC Encrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Encrypt	Run when module is loaded via the default entry point and on-demand.
AES-CBC Decrypt (Inverse Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Decrypt	Run when module is loaded via the default entry point and on-demand.
AES-CCM Authenticated Encrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Encrypt	Run when module is loaded via the default entry point and on-demand.
AES-CCM Authenticated Decrypt (Forward Cipher Function)(A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Decrypt	Run when module is loaded via the default entry point and on-demand.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CMAC Authenticated Encrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Encrypt	Run when module is loaded via the default entry point and on-demand.
AES-CMAC Authenticated Decrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Decrypt	Run when module is loaded via the default entry point and on-demand.
AES-CTR Encrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Encrypt	Run when module is loaded via the default entry point and on-demand.
AES-CTR Decrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Decrypt	Run when module is loaded via the default entry point and on-demand.
AES-GCM Authenticated Encrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Encrypt	Run when module is loaded via the default entry point and on-demand.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM Authenticated Decrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Decrypt	Run when module is loaded via the default entry point and on-demand.
AES-XTS Testing Revision 2.0 Encrypt (Forward Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Encrypt	Run when module is loaded via the default entry point and on-demand.
AES-XTS Testing Revision 2.0 Encrypt (Inverse Cipher Function) (A6701)	128-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Decrypt	Run when module is loaded via the default entry point and on-demand.
HMAC-SHA-1 (A6701)	128-bit key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Message authentication	Run when module is loaded via the default entry point and on-demand.
HMAC-SHA2-384 (A6701)	128-bit key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Message authentication	Run when module is loaded via the default entry point and on-demand.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-512 (A6701)	128-bit key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Message authentication	Run when module is loaded via the default entry point and on-demand.
HMAC-SHA3-256 (A6701)	128-bit key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Message authentication	Run when module is loaded via the default entry point and on-demand.
HMAC-SHA3-384 (A6701)	128-bit key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Message Authentication	On demand by calling the SymCryptHmacSha3_384Selftest API function.
HMAC-SHA3-512 (A6701)	128-bit key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Message Authentication	On demand by calling the SymCryptHmacSha3_512Selftest API function.
KDF TLS (A6701)	Derived key material	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Derivation	Run when module is loaded via the default entry point and on-demand.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
TLS v1.2 KDF RFC7627 (A6701)	HMAC-SHA2-512	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Derivation	Run when module is loaded via the default entry point and on-demand.
KDF SP800-108 (A6701)	Self-test with HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384 and HMAC-SHA2-512	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Derivation	Run when module is loaded via the default entry point and on-demand.
PBKDF (A6701)	HMAC-SHA-1	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Derivation	Run when module is loaded via the default entry point and on-demand.
KDF SRTP (A6701)	Derived Key material	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Derivation	Run when module is loaded via the default entry point and on-demand.
KDF SSH (A6701)	Self-test with SHA2-256 and SHA2-512	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Derivation	Run when module is loaded via the default entry point and on-demand.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDA OneStep SP800-56Cr2 (A6701)	HMAC-SHA2-512	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Key Derivation	Run when module is loaded via the default entry point and on-demand.
RSA SigGen (FIPS186-5) (A6701)	2048 bits modulo with SHA2-256	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Signature Generation	Prior to first use of RSA algorithm and on-demand.
RSA SigVer (FIPS186-5) (A6701)	2048 bits modulo with SHA2-256	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Signature Verification	Prior to first use of RSA algorithm and on-demand.
DSA SigVer (FIPS186-4) (A6701)	2048-bit DSA key with SHA2-256	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Signature Verification	Prior to first use of DSA algorithm and on-demand.
ECDSA SigGen (FIPS186-5) (A6701)	P-256 curve with SHA2-256	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Signature Generation	Prior to first use of ECDSA algorithm and on-demand.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA SigVer (FIPS186-5) (A6701)	P-256 curve with SHA2-256	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Signature Verification	Prior to first use of ECDSA algorithm and on-demand.
KAS-ECC-SSC Sp800-56Ar3 (A6701)	P-256 curve	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Shared Secret computation	Prior to first use of KAS algorithm and on-demand.
KAS-FFC-SSC Sp800-56Ar3 (A6701)	2048-bit key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Shared Secret computation	Prior to first use of KAS algorithm and on-demand.
Counter DRBG (A6701)	256-bit AES key	KAT	CAST	Cryptographic functions execute and a status is returned via the interface	Random number generation: Instantiate, Reseed and Generate functions	Run when module is loaded via the default entry point and on-demand.
ECDSA KeyGen (FIPS186-5) (A6701)	Generated curve	PCT	PCT	Cryptographic functions execute and a status is returned via the interface	Perform a Pairwise Consistency Test (PCT) on key generation and key import.	Key Generation/Key Import

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA KeyGen (FIPS186-5) (A6701)	Generated key size	PCT	PCT	Cryptographic functions execute and a status is returned via the interface	Perform a Pairwise Consistency Test (PCT) on key generation and key import.	Key Generation/Key Import
KAS-ECC-SSC Sp800-56Ar3 Assurances (A6701)	SP800-56Arev3 assurances	Critical function	Critical Function	Cryptographic functions execute and a status is returned via the interface	Perform ECDH assurances (including pairwise consistency tests) according to NIST SP 800-56Arev3	Prior the first use of KAS-ECC
KAS-FFC-SSC Sp800-56Ar3 Assurances (A6701)	SP800-56Arev3 assurances	Critical function	Critical Function	Cryptographic functions execute and a status is returned via the interface	Perform DH assurances (including pairwise consistency tests) according to NIST SP 800-56Arev3	Prior the first use of KAS-FFC
DRBG Health Checks	DRBG Health Checks	Health Checks	Critical Function	Cryptographic functions execute and a status is returned via the interface	DRBG health checks: DRBG Instantiate, Generate and Reseed Tests	Run when module is loaded via the default entry point.
SHA-1 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptSha1Selftest API function.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA2-256 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptSha256Selftest API function.
SHA2-384 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptSha384Selftest API function.
SHA2-512 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptSha512Selftest API function.
SHA3-256 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptSha3_256Selftest API function.
SHA3-384 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptSha3_384Selftest API function.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA3-512 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptSha3_512Selftest API function.
SHAKE-128 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptShake128Selftest API function.
SHAKE-256 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptShake256Selftest API function.
cSHAKE-128 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptCShake128Selftest API function.
cSHAKE-256 (A6701)	Generation	KAT	CAST	Cryptographic functions execute and a status is returned via the interface.	Message Digest Generation.	On demand by calling the SymCryptCShake256Selftest API function.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Jitter Entropy Source (E257)	Entropy source health tests	Fault detection test	CAST	The entropy source is instantiated and a status is returned via the interface.	RCT, APT, lag prediction test, stuck test and clock resolution test.	Run continuously and at instantiation of the entropy source.
Intel-based Entropy Source (E272)	Entropy source health tests	Fault detection test	CAST	Status returned via internal status register and carry flag.	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 (E273)	Entropy source health tests	Fault detection test	CAST	Status returned via internal status register and carry flag.	Start-up and continuous noise source health tests and start-up logic integrity self-test.	Run at start-up and continuously.

Table 25: Conditional Self-Tests

In addition to the pre-operational self-tests, the module performs self-tests on Approved cryptographic algorithms supported in the approved mode of operation, using the tests shown in (and indicated as CASTs). These CASTs are performed prior to the first operational use of each cryptographic algorithm.

Data output through the data output interface is inhibited during the self-tests.

The cryptographic algorithm self-tests are performed in the form of Known Answer Tests (KATs), in which the calculated output is compared with the expected known answers that are hardcoded in the module. A failed comparison causes a failure of the self-test. If any of these self-tests fails, then SymCrypt invokes the SymCryptFatal function and the module transitions to Critical Error state to halt execution.

The Pairwise Conditional Self-tests are run when an asymmetric key pair is generated. In case of failure the module enters in Critical Error state for RSA and Non-Critical Error state for ECDSA.

For the DRBG health checks, if any fails, the module will enter in a Critical Error state.

While the module is executing the conditional self-tests, services are not available, and input and output are inhibited. The module is not available for use by the calling application until these tests are completed successfully.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6701)	KAT	SW/FW Integrity	On demand	Automatic

Table 26: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6701)	KAT	CAST	On Demand	Programmatically
AES-ECB Encrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-ECB Decrypt (Inverse Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-CBC Encrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-CBC Decrypt (Inverse Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-CCM Authenticated Encrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-CCM Authenticated Decrypt (Forward Cipher Function)(A6701)	KAT	CAST	On Demand	Programmatically
AES-CMAC Authenticated Encrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-CMAC Authenticated Decrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-CTR Encrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-CTR Decrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM Authenticated Encrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-GCM Authenticated Decrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-XTS Testing Revision 2.0 Encrypt (Forward Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
AES-XTS Testing Revision 2.0 Encrypt (Inverse Cipher Function) (A6701)	KAT	CAST	On Demand	Programmatically
HMAC-SHA-1 (A6701)	KAT	CAST	On Demand	Programmatically
HMAC-SHA2-384 (A6701)	KAT	CAST	On Demand	Programmatically
HMAC-SHA2-512 (A6701)	KAT	CAST	On Demand	Programmatically
HMAC-SHA3-256 (A6701)	KAT	CAST	On Demand	Programmatically
HMAC-SHA3-384 (A6701)	KAT	CAST	On demand	Programmatically
HMAC-SHA3-512 (A6701)	KAT	CAST	On demand	Programmatically
KDF TLS (A6701)	KAT	CAST	On Demand	Programmatically
TLS v1.2 KDF RFC7627 (A6701)	KAT	CAST	On Demand	Programmatically
KDF SP800-108 (A6701)	KAT	CAST	On Demand	Programmatically
PBKDF (A6701)	KAT	CAST	On Demand	Programmatically
KDF SRTP (A6701)	KAT	CAST	On Demand	Programmatically
KDF SSH (A6701)	KAT	CAST	On Demand	Programmatically
KDA OneStep SP800-56Cr2 (A6701)	KAT	CAST	On demand	Programmatically
RSA SigGen (FIPS186-5) (A6701)	KAT	CAST	On demand	Programmatically
RSA SigVer (FIPS186-5) (A6701)	KAT	CAST	On demand	Programmatically

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
DSA SigVer (FIPS186-4) (A6701)	KAT	CAST	On demand	Programmatically
ECDSA SigGen (FIPS186-5) (A6701)	KAT	CAST	On demand	Programmatically
ECDSA SigVer (FIPS186-5) (A6701)	KAT	CAST	On demand	Programmatically
KAS-ECC-SSC Sp800-56Ar3 (A6701)	KAT	CAST	On demand	Programmatically
KAS-FFC-SSC Sp800-56Ar3 (A6701)	KAT	CAST	On demand	Programmatically
Counter DRBG (A6701)	KAT	CAST	On demand	Programmatically
ECDSA KeyGen (FIPS186-5) (A6701)	PCT	PCT	On demand	Programmatically
RSA KeyGen (FIPS186-5) (A6701)	PCT	PCT	On demand	Programmatically
KAS-ECC-SSC Sp800-56Ar3 Assurances (A6701)	Critical function	Critical Function	On demand	Programmatically
KAS-FFC-SSC Sp800-56Ar3 Assurances (A6701)	Critical function	Critical Function	On demand	Programmatically
DRBG Health Checks	Health Checks	Critical Function	On demand	Programmatically
SHA-1 (A6701)	KAT	CAST	On demand	Programmatically
SHA2-256 (A6701)	KAT	CAST	On demand	Programmatically
SHA2-384 (A6701)	KAT	CAST	On demand	Programmatically
SHA2-512 (A6701)	KAT	CAST	On demand	Programmatically
SHA3-256 (A6701)	KAT	CAST	On demand	Programmatically
SHA3-384 (A6701)	KAT	CAST	On demand	Programmatically
SHA3-512 (A6701)	KAT	CAST	On demand	Programmatically
SHAKE-128 (A6701)	KAT	CAST	On demand	Programmatically
SHAKE-256 (A6701)	KAT	CAST	On demand	Programmatically
cSHAKE-128 (A6701)	KAT	CAST	On demand	Programmatically
cSHAKE-256 (A6701)	KAT	CAST	On demand	Programmatically
Jitter Entropy Source (E257)	Fault detection test	CAST	On demand	Programmatically

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
Intel-based Entropy Source (E272)	Fault detection test	CAST	On demand	Programmatically
Intel-based Entropy Source (E273)	Fault detection test	CAST	On demand	Programmatically

Table 27: Conditional Periodic Information

10.4 Error States

The two error states for the module are Critical Error State and Non-Critical Error State. When the module is in these error states, services are unavailable, and the input and output interfaces are disabled. If any self-tests described in Sections 10.1, 10.2, and 10.3 (except from ECDSA PCT) fail, the module enters a Critical Error state and must be reloaded to perform any cryptographic services. In the Critical Error State, no cryptographic services are available, and data output is prohibited. The only way to recover from the Critical Error state is to restart the process, which reloads the module into memory and initiates the pre-operational software integrity test and Conditional CASTs.

In addition, if the ECDSA PCT or the AES XTS “key1 != key2” check fail, the module will flow to Non-Critical Error State, and will wait for the next cryptographic operation, not allowing any cryptographic service and no data output or input until the error is cleared.

The table below shows the different causes that lead to the error states and the status indicator reported.

Name	Description	Conditions	Recovery Method	Indicator
Critical Error	Failure of pre-operational self-tests, cryptographic algorithms self-tests, RSA pairwise consistency test and critical security function tests.	Failure of pre-operational self-tests, cryptographic algorithms self-tests, RSA pairwise consistency test and critical security function tests.	Power Cycle	Return code set to: *0x46695053 (FIPS) when the Integrity test, ECDSA Self-test, RSA Self-test, DH self-test, ECDH self-test and RSA PCT fail. *0x68736832 (aci2), 0x61637232 (acr2), 0x61636732 (acg2) when CTR DRBG self-test for instantiate, reseed and generate fails. *0x616573 (aes) when AES self-test fails. *0x68736835 (hsh5) when AES CMAC or HMAC SHA2-512 self-test fails. *0x63636d31 (ccm1) and 0x63636d32 (ccm2) for AES CCM (encryption or decryption) self-test failure. *0x67636d31 (gcm1) and 0x67636d32 (gcm2) for AES GCM (encryption or decryption) self-test failure. *0x78747361 (xtsa) for AES XTS self-test failure. *0x68536831 (hSh1), 0x68736832 (hsh2) for HMAC SHA-1 and HMAC SHA2-256, 0x68736833 (hsh3) for HMAC SHA2-384 and HMAC SHA3-256 and 0x68736835 (hsh5) for HMAC SHA2-512 self-test failure. *0x746c3131 (tl11) and 0x746c3132 (tl12) for TLS 1.1 and TLS 1.2 KDF self-test failure. *0x38313038 (8108) for KDF SP800-108 self-test failure. *0x50626b32 (Pbk2) for PBKDF2 self-test failure. *0x73727470 (srtp) for SRTP KDF self-test failure. *0x7373686b (sshk) for SSH KDF self-test failure. *0x73736b64 (sskd) for SSKDF self-test failure. *0x00008000e for DH and ECDH assurances failure.
Non-Critical Error	When ECDSA PCT fails or AES XTS check fails.	ECDSA PCT failure or AES XTS check failure (Key1 == Key2)	The module clears the error automatically	Return code set to: -0x00008009 for AES XTS check failure. -0x00008010 for ECDSA PCT error.

Table 28: Error States

10.5 Operator Initiation of Self-Tests

To perform the module self-tests on demand, the user may restart the computer or execute any of the specific self-test API functions.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

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 operating system. No additional configuration is necessary.

Module initialization occurs automatically as part of the OS boot process. The finite state model diagram below visualizes the initialization process along with other module states.

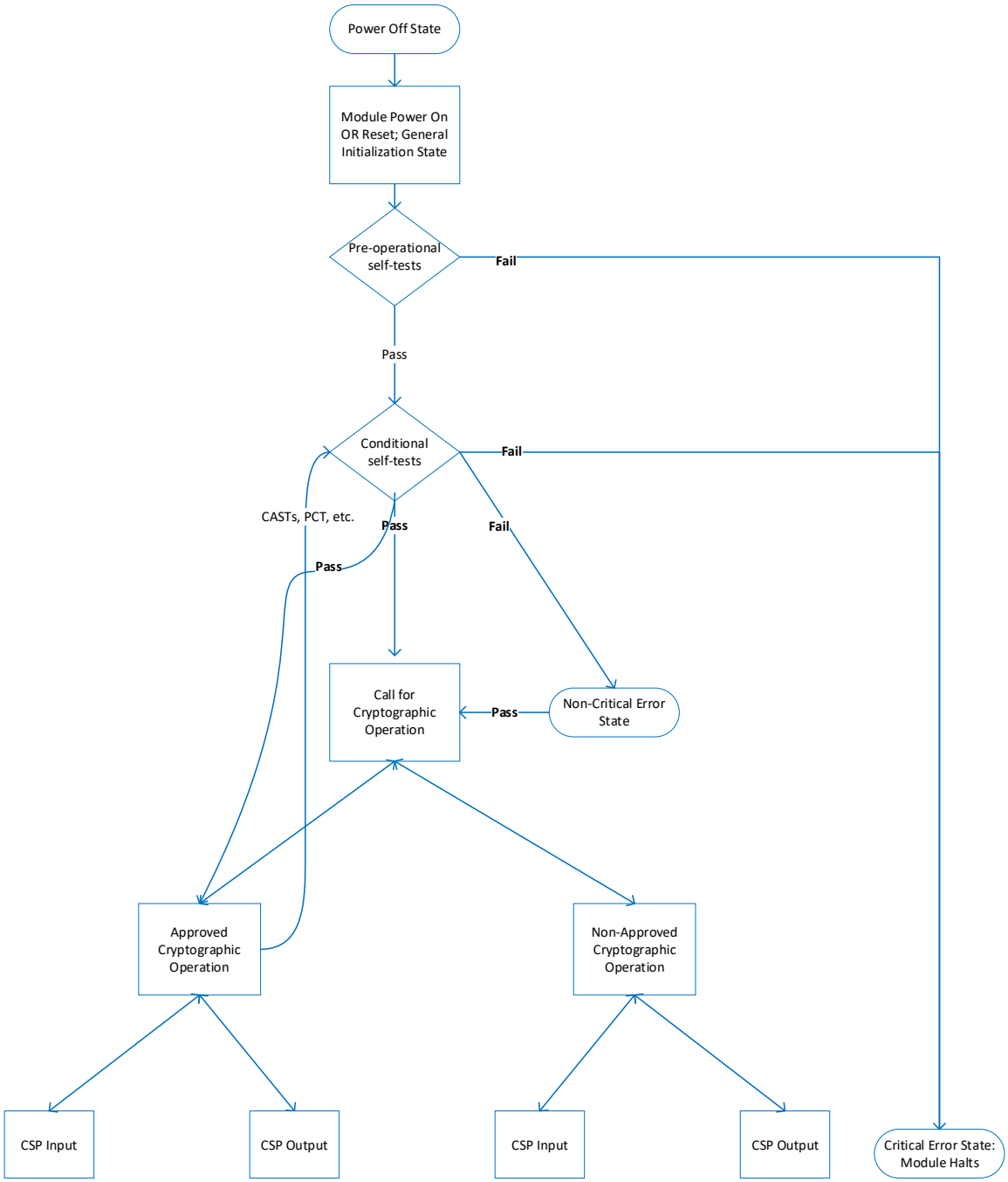

Figure 2 - FSM Diagram

11.2 Administrator Guidance

To ensure the installed version of SymCrypt matches the validated version:

- Run the following command: `strings libsymcrypt.so | grep -E v[[:digit:]]+\.[[:digit:]]+\.[[:digit:]]+` to confirm the SymCrypt version is 103.8.0. The full output of the command is:

  "`v103.8.0_main_2025-01-28T00:44:15+00:00_53be637_2025-02-06T21:59:47`

  `SymCrypt v103.8.0`"
- SymCrypt version 103.8.0 is associated with git commit ID #53be637, located at https://github.com/microsoft/SymCrypt/commit/53be637dab201a4c9d95e1cf58040c85d71cf3c2.

In case of an issue or compromise, and for the secure sanitization of the module:

- Reformat the hard drive to ensure complete removal of any potential threats.

By following these steps, administrators can ensure the integrity and security of the SymCrypt module.

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 software cryptographic module that provides cryptographic services within the Operational Environments listed in 2.2 Tested and Vendor Affirmed Module Version and Identification. No user installation or maintenance is required. To invoke the approved mode of operation, the user must abide by the administrator guidance above. 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 module.

Algorithm	Protected Against	Mitigation
SHA1	Timing Analysis Attack	Constant time implementation.
	Cache Attack	Memory access pattern is independent of any confidential data.
SHA2	Timing Analysis Attack	Constant time implementation.
	Cache Attack	Memory access pattern is independent of any confidential data.
AES	Timing Analysis Attack	Constant time implementation.
	Cache Attack	Memory access pattern is independent of any confidential data. Protected against cache attacks only when used with AES NI.

Table 29 - 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 800-52 Rev.2, Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations, https://csrc.nist.gov/pubs/sp/800/52/r2/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