Vendor name
DigiCert, Inc.

Module Name
DigiCert TrustCore Cryptographic Loadable Kernel Module

FIPS 140-3 Non-Proprietary Security Policy

Software Version: 7.0.0f

Document Version: 7.0.0f_1.3
Date: 11/07/2025

DigiCert, Inc.

2801 North Thanksgiving Way
Suite 500
Lehi, UT 84043
+1 800-896-7973

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 7.0.0f of the DigiCert TrustCore Cryptographic Loadable Kernel Module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module.

1.2 Security Levels

The FIPS 140-3 security levels for the Module are as follows in the table below:

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	N/A
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

This DigiCert, Inc. (DigiCert) DigiCert TrustCore Cryptographic Loadable Kernel Module, is hereafter denoted as the Module. The Module is the cryptographic engine of DigiCert's TrustCore development platform.

2.1 Description

Purpose and Use:

The primary purpose of the Module is to provide Approved cryptographic routines to consuming applications via an Application Programming Interface (API). The Module is intended for use by US Federal agencies or other markets that require FIPS 140-3 validated Security Level 1 software modules. The Module is intended to be used in dedicated purpose IOT (Internet of Things) devices and general-purpose computer systems.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

The physical form of the Module is depicted in Figure 1. The Module is software, with a multi-chip standalone embodiment. The cryptographic boundary is comprised of the kernel module (moc_crypto.ko) and the integrity check signature file (moc_crypto.ko.sig), and the CPU when PAA is enabled.

Tested Operational Environment's Physical Perimeter (TOEPP)

The TOEPP is bound by the General Purpose Computer and includes the DigiCert TrustCore Cryptographic Loadable Kernel Module, the CPU with PAA when PAA is enabled, and API calls from calling applications running within the same process as the Module.

Figure 1 shows the Module, interfaces with the Tested Operational Environment (TOEPP), and the delimitation of its cryptographic boundary, shown shaded in blue. The Cryptographic Boundary components are described in Table 1.

Component *	Description
moc_crypto.ko	Kernel object for cryptographic algorithms
moc_crypto.ko.sig	Integrity Check HMAC value for the kernel object

Table 2: Cryptographic Module Components

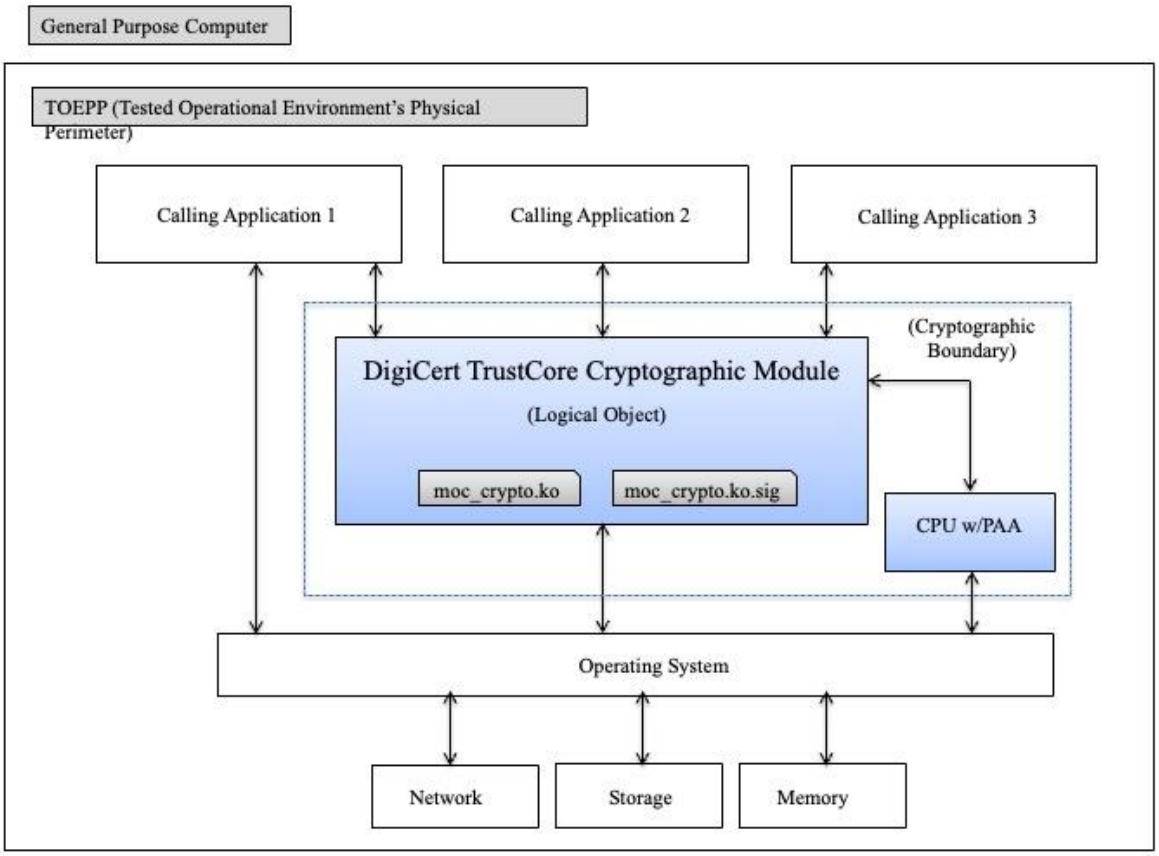

Figure 1: Logical [cryptographic] boundary [and physical perimeter if combined]

2.2 Tested and Vendor Affirmed Module Version and Identification

Package or File Name	Software/ Firmware Version	Features	Integrity Test
E3950-moc_crypto.ko	7.0.0f	Xerox Explorer 6.5 with Intel Atom E3950 without PAA	HMAC-SHA2-256
X6413E-moc_crypto.ko	7.0.0f	Xerox Explorer 8.0 with Intel Atom x6413E without PAA	HMAC-SHA2-256
ARM-A53-moc_crypto.ko	7.0.0f	Xerox Alexandra Platform with ARM Cortex A53 without PAA	HMAC-SHA2-256

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

Tested Operational Environments - Software, Firmware, Hybrid:

The DigiCert TrustCore Cryptographic Loadable Kernel Module is tested on the following operational environments:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Yocto Linux 3.1 (64-bit)	Xerox Explorer 6.5	Intel Atom E3950	No	N/A	7.0.0f
Yocto Linux 3.1 (64-bit)	Xerox Explorer 8.0	Intel Atom x6413E	No	N/A	7.0.0f
Yocto Linux 3.1 (64-bit)	Xerox Alexandra Platform	ARM Cortex A53	No	N/A	7.0.0f

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

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

The DigiCert TrustCore Cryptographic Loadable Kernel Module is tested on the following operational environments.

Operating System	Hardware Platform
Ubuntu Linux 4.15 (64-bit)	Intel NUC with i7-8650U processor with and without PAA

Table 5: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

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 are excluded from [140-3] requirements.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Only approved or allowed security functions with sufficient key security strength can be used	Approved	FIPS_eventLog, FIPS_EventType
Non-Approved Mode	When non-approved security functions or approved security functions with insufficient key security strength are used.	Non-Approved	FIPS_eventLog, FIPS_EventType

Table 6: Modes List and Description

The Module enters Approved mode after pre-operational self-tests have successfully completed. Once the Module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys. To provide an indicator of the current mode of operation, an asynchronous callout event mechanism is provided.

The application using the Module can register for a FIPS_eventLog call-out function to be used as the approved-mode/non-approved-mode indicator. The Module uses the scenario of an indicator for multiple services, per IG 2.4.C example #3. It uses a dedicated status output interface that is a software callback function. The calling application using the Module registers a callback function to be called at the beginning and end of all services and algorithm implementations. The application's FIPS_eventLog function will be called from the Module at the beginning and end of each Approved or non-Approved service or algorithm. This FIPS_eventLog and the FIPS_EventType enumeration value provided as a parameter serves as a thread-safe status indicator of the current Approved or non-Approved mode of operation.

Mode Change Instructions and Status:
The Approved mode of operation is configured at instantiation of the Module by the Cryptographic Officer role by execution of an application or protocol operating system process that uses the Module's cryptographic functions.

The Module transitions to the non-Approved mode of operation when one of the non-Approved security functions is utilized. The Module can transition back to the Approved mode of operation by utilizing an Approved security function.

Keys and CSPs are not shared between the Approved and non-Approved mode of operation.

Degraded Mode Description:
N/A

2.5 Algorithms

Approved Algorithms:
The Module implements the Approved cryptographic algorithms listed in the table below:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A845	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A845	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56-104 Increment 8 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0-524288 Increment 8	SP 800-38C
AES-CFB128	A845	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A845	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 32-128 Increment 8 Message Length - Message Length: 0-65536 Increment 8	SP 800-38B
AES-CTR	A845	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
		Supports Counter larger than maximum value - No Incremental Counter - Yes Counter Tests Performed - Yes	
AES-ECB	A845	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A846	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 8-1024 Increment 8 Payload Length - Payload Length: 8-65536 Increment 8 AAD Length - AAD Length: 0-65536 Increment 8	SP 800-38D
AES-GCM	A847	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 8-1024 Increment 8 Payload Length - Payload Length: 8-65536 Increment 8 AAD Length - AAD Length: 0-65536 Increment 8	SP 800-38D
AES-GMAC	A846	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 8-1024 Increment 8 AAD Length - AAD Length: 0-4096 Increment 8	SP 800-38D
AES-GMAC	A847	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 8-1024 Increment 8 AAD Length - AAD Length: 0-4096 Increment 8	SP 800-38D
AES-OFB	A845	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A845	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 128 Tweak Mode - Hex Data Unit Length Matches Payload Length - Yes	SP 800-38E
Counter DRBG	A845	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - AES-128, AES-192, AES-256	SP 800-90A Rev. 1

Algorithm	CAVP Cert	Properties	Reference
		Derivation Function Enabled - No, Yes Additional Input - Additional Input: 0-16777216 Increment 8, Additional Input: 0-256 Increment 8, Additional Input: 0-320 Increment 8, Additional Input: 0-384 Increment 8 Entropy Input - Entropy Input: 128-16777216 Increment 8, Entropy Input: 192-16777216 Increment 8, Entropy Input: 256, Entropy Input: 256-16777216 Increment 8, Entropy Input: 320, Entropy Input: 384 Nonce - Nonce: 0, Nonce: 128-16777216 Increment 8, Nonce: 64-16777216 Increment 8, Nonce: 96-16777216 Increment 8 Personalization String Length - Personalization String Length: 0-16777216 Increment 8, Personalization String Length: 0-256 Increment 8, Personalization String Length: 0-320 Increment 8, Personalization String Length: 0-384 Increment 8 Returned Bits - 512	
HMAC-SHA-1	A845	MAC - MAC: 32-160 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA2-224	A845	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA2-256	A845	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA2-384	A845	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA2-512	A845	MAC - MAC: 32-512 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA3-224	A845	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA3-256	A845	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA3-384	A845	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
HMAC-SHA3-512	A845	MAC - MAC: 32-512 Increment 8 Key Length - Key Length: 112-65536 Increment 8	FIPS 198-1
KDF SP800-108	A845	KDF Mode - Feedback MAC Mode - HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512 Supported Lengths - Supported Lengths: 8-4096 Increment 8 Fixed Data Order - After Fixed Data Counter Length - 8 Supports Empty IV - Yes Requires Empty IV - No Custom Key In Length - 0	SP 800-108 Rev. 1
SHA-1	A845	Message Length - Message Length: 160, 0-65536 Increment 8	FIPS 180-4
SHA2-224	A845	Message Length - Message Length: 224, 0-65536 Increment 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-256	A845	Message Length - Message Length: 256, 0-65536 Increment 8	FIPS 180-4
SHA2-384	A845	Message Length - Message Length: 384, 0-65536 Increment 8	FIPS 180-4
SHA2-512	A845	Message Length - Message Length: 512, 0-65536 Increment 8	FIPS 180-4
SHA3-224	A845	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A845	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A845	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A845	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A845	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A845	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-65536 Increment 8	FIPS 202

Table 7: Approved Algorithms

Vendor-Affirmed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

The Module implements the Non-Approved, Not Allowed cryptographic algorithms listed below:

Name	Use and Function
AES-EAX (NC)	Authentication and Encryption
AES GCM 256-bit (NC)	256-bit Encryption/Decryption for 256-bit state table implementation
AES GMAC 256-bit (NC)	256-bit Message Authentication for 256-bit state table implementation

Name	Use and Function
AES XCBC (NC)	Message Authentication
DES (NC)	Encryption/Decryption
HMAC (NC)	HMAC generation with key size less than 112 bits
HMAC-MD5 (NC)	Message Authentication
MD2, MD4, MD5 (NC)	Message Digest
RNG (NC)	FIPS 186-2 Random Number Generation
Triple-DES (NC)	Encryption/Decryption

Table 8: Non-Approved, Not Allowed Algorithms

Note: All the various AES modes (e.g., EAX, XCBC, XTS, etc.) use the same underlying AES implementation as the Approved AES cert.

2.6 Security Function Implementations

The table below shows the Security Function Implementations that the Module implements:

Name	Type	Description	Properties	Algorithms
SFI-AES-UnAuth-Encrypt	BC-UnAuthEncrypt	Block Cipher Encryption		AES-CBC: (A845) AES-ECB: (A845) AES-CTR: (A845) AES-CFB128: (A845) AES-OFB: (A845) AES-XTS Testing Revision 2.0: (A845)
SFI-AES-UnAuth-Decrypt	BC-UnAuthDecrypt	Block Cipher Decryption		AES-CBC: (A845) AES-ECB: (A845) AES-CTR: (A845) AES-CFB128: (A845) AES-OFB: (A845) AES-XTS Testing Revision 2.0: (A845)
SFI-AES-CCM-Encrypt	BC-AuthEncrypt	Block Cipher Encryption		AES-CCM: (A845)
SFI-AES-CCM-Decrypt	BC-AuthDecrypt	Block Cipher Decryption		AES-CCM: (A845)
SFI-AES-GCM-Encrypt	BC-AuthEncrypt	Block Cipher Encryption		AES-GCM: (A846, A847)
SFI-AES-GCM-Decrypt	BC-AuthDecrypt	Block Cipher Decryption		AES-GCM: (A846, A847)
SFI-SHS	SHA	Secure Hash Standard	Publications:IG C.B	SHA-1: (A845) SHA2-224: (A845) SHA2-256: (A845)

Name	Type	Description	Properties	Algorithms
				SHA2-384: (A845) SHA2-512: (A845)
SFI-SHA3	SHA	Secure Hash Standard	Publications: IG C.B, IG C.C	SHA3-224: (A845) SHA3-256: (A845) SHA3-384: (A845) SHA3-512: (A845)
SFI-SHAKE	SHA	SHAKE Extendable Output Function	Publications: IG C.C	SHAKE-128: (A845) SHAKE-256: (A845)
SFI-AES-CMAC	MAC	Message Authentication Generation		AES-CMAC: (A845)
SFI-AES-GMAC	MAC	Message Authentication Generation		AES-GMAC: (A846, A847)
SFI-HMAC	MAC	Message Authentication Generation		HMAC-SHA-1: (A845) HMAC-SHA2-224: (A845) HMAC-SHA2-256: (A845) HMAC-SHA2-384: (A845) HMAC-SHA2-512: (A845) HMAC-SHA3-224: (A845) HMAC-SHA3-256: (A845) HMAC-SHA3-384: (A845) HMAC-SHA3-512: (A845)
SFI-HMAC-KDF	KBKDF	Key-Based Key Derivation	Publications:[IG D.F]	KDF SP800-108: (A845) HMAC-SHA-1: (A845) HMAC-SHA2-224: (A845) HMAC-SHA2-256: (A845) HMAC-SHA2-384: (A845) HMAC-SHA2-512: (A845)

Name	Type	Description	Properties	Algorithms
SFI-DRBG-Generate	DRBG	Random Number Generation		Counter DRBG: (A845)
SFI-DRBG-ReSeed	DRBG	Random Number ReSeed		Counter DRBG: (A845)

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

AES GCM IV Uniqueness: FIPS 140-3 IG C.H., Option 1
The AES GCM implementation generates GCM IVs deterministically as specified in SP800-38D Section 8.2.1 using the following protocols:

TLS 1.2 Protocol IV generation for GCM Cipher Suites: The Module supports TLS 1.2 GCM Cipher Suites for TLS, as described in RFCs 5116, 5246, 5288 and 5289 and shall only be used for the TLS protocol version 1.2 to be compliant with FIPS140-3 IG C.H, Option 1.

Per [IG] C.H. technique 1.a. TLS 1.2 GCM Cipher Suites for TLS method was used for testing during operational testing. During operational testing, the Module was tested against an independent version of TLS and found to behave correctly.

The counter portion of the IV is set by the Module within its cryptographic boundary.

The nonce explicit part of the IV is incremented each time an AES GCM computation is performed. The Module establishes a new session key when the nonce explicit part of the IV exhausts the maximum number of possible values (2^{32}-1).

In case the Module's power is lost and then restored, a new key for use with the AES GCM encryption/decryption shall be established.

Protocol specific KDF listed in SP 800-135rev1: FIPS-140-3 IG D.C Option 4
The cryptographic module supports HMAC-KDF used by upper-level protocols, but does not implement protocol specific KDFs as defined in SP-800-135rev1. This module does not implement, use, or depend upon protocols listed in SP 800-135rev1.

AES XTS Requirements on the Key: FIPS 140-3 IG C.I.

Per [IG] C.I. the XTS algorithm implementation includes a check to ensure Key_1 ≠ Key_2.

2.8 RBG and Entropy

The Module does not have a specific entropy source. Entropy must be provided by the calling application through the API.

The Module implements a CTR-based DRBG per SP800-90Ar1 for creation of symmetric and asymmetric keys.

The Module accepts input from entropy sources external to the cryptographic boundary for use as seed material for the Module's Approved DRBGs. External entropy can be added via several APIs available to the cryptographic module client application.

The calling application of the Module shall use entropy sources that meet the security strength required for the random bit generation mechanism as shown in NIST SP800-90Ar1 Table 3 (CTR_DRBG). A minimum of 384 bits of entropy must be provided by the calling application. The calling application shall provide full entropy for 256-bit keys. When the CTR_DRBG is used without a derivation function full entropy must be provided, per SP 800.90Ar1, IG D.L. Due to the entropy being provided by an external source, the following caveat applies: There is no assurance of the minimum strength of generated SSPs (e.g. keys).

The Module performs DRBG health tests (Instantiate, Generate, ReSeed) as defined in section 11.3 of SP800-90Ar1.

2.9 Key Generation

For Key Generation, see Section 2.5 and Section 2.6 above.

2.10 Key Establishment

Key Agreement Information

The Module does not implement Key Agreement.

Key Transport Information

The Module does not implement Key Transport.

2.11 Industry Protocols

The Module does not implement any Industry Protocols.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

The Module's ports and associated defined logical interface categories are listed below. The Module's logical interface (API) provides logical separation of the input and output interfaces.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Input parameters of API function calls
N/A	Data Output	Output parameters of API function calls
N/A	Control Input	API Function Calls
N/A	Control Output	Output of the FIPS_EventType enumeration value and an indication of approved or non-approved mode of operation.
N/A	Status Output	For Approved mode, function calls returning status information and return code provided by API function calls
N/A	Power	None

Table 10: Ports and Interfaces

4 Roles, Services, and Authentication

4.1 Authentication Methods

Note: The Module is Level 1 and does not implement any Authentication techniques.

N/A for this module.

4.2 Roles

The Module supports one distinct operator role, Cryptographic Officer (CO).

The Roles Table below lists all operator roles supported by the Module.

The Module does not support concurrent operators, bypass capability, or a maintenance role. The Cryptographic Officer role is implicitly identified by the service that is requested.

Name	Type	Operator Type	Authentication Methods
CO	Role	Cryptographic Officer	None

Table 11: Roles

4.3 Approved Services

All Approved services implemented by the Module are listed in the table below:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
AES Encrypt	Perform encryption on a block of data using the shared key Modes: AES-CBC AES-ECB AES-CTR AES-CFB128 AES-OFB AES-XTS Testing Revision 2	Approved	Encrypt command with AES, input data, input size, key, key size, mode of operation	Ciphertext and return status of OK or error condition	SFI-AES-UnAuth-Encrypt	CO - AES Keys: W,E
AES Decrypt	Perform decryption on a block of data using the shared key Modes: AES-CBC AES-ECB AES-CTR	Approved	Decrypt command with AES, input data, input size, key, key size, mode of operation	Plaintext and return status of OK or error condition	SFI-AES-UnAuth-Decrypt	CO - AES Keys: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	AES-CFB128 AES-OFB AES-XTS Testing Revision 2					
AES-CCM Encrypt	Perform encryption on a block of data using the shared key and CCM message authentication code	Approved	Encrypt command with AES, input data, input size, key, key size, mode of operation	Ciphertext and return status of OK or error condition	SFI-AES-CCM-Encrypt	CO - AES Keys: W,E
AES-CCM Decrypt	Perform decryption on a block of data using the shared key and CCM message authentication code	Approved	Decrypt command with AES, input data, input size, key, key size, mode of operation	Plaintext and return status of OK or error condition	SFI-AES-CCM-Decrypt	CO - AES Keys: W,E
AES-GCM Encrypt	Perform encryption on a block of data using the shared key and GCM message authentication code	Approved	Encrypt command with AES, input data, input size, key, key size, mode of operation	Ciphertext and return status of OK or error condition	SFI-AES-GCM-Encrypt	CO - AES Keys: W,E
AES-GCM Decrypt	Perform decryption on a block of data using the shared key and GCM message authentication code	Approved	Decrypt command with AES, input data, input size, key, key size, mode of operation	Plaintext and return status of OK or error condition	SFI-AES-GCM-Decrypt	CO - AES Keys: W,E
SHS	Generation a SHA-1 or SHA-2 message digest	Approved	Message	Message Digest	SFI-SHS	CO
SHA3	Generation a SHA-3 message digest	Approved	Message	Message Digest	SFI-SHA3	CO
SHA3-SHAKE	Generation a SHA-3 Extendable	Approved	Message	Message Digest	SFI-SHAKE	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	Output Function (XOF) message digest					
CMAC MACGen	Generate a keyed-hash message authentication code with AES-CMAC	Approved	AES key, message	Keyed hash with return status of OK or error condition	SFI-AES-CMAC	CO - AES Keys: W,E
GMAC MACGen	Generate a keyed-hash message authentication code with AES-GCM	Approved	AES key, message	Keyed hash with return status of OK or error condition	SFI-AES-GMAC	CO - AES Keys: W,E
HMAC MACGen	Generate a keyed-hash message authentication code	Approved	HMAC key, message	Keyed hash with return status of OK or error condition	SFI-SHS SFI-SHA3 SFI-HMAC	CO - HMAC Key: W,E
KDF-HMAC	Extract input key material and expand into additional keys	Approved	Pseudorandom key material	Key material and return status of OK or error condition	SFI-SHS SFI-SHA3 SFI-HMAC SFI-HMAC-KDF	CO - KBKDF Pseudorandom Keys: W,E - KBKDF Output Key: G,R
AES-CTR-DRBG Gen	Generate Psuedo random numbers	Approved	Generate command	Random number with status of OK or error condition	SFI-DRBG-Generate	CO - DRBG Entropy Input: W - Nonce Values: W - DRBG V: G - DRBG Key: G - DRBG Reseed Counter: G
AES-CTR-DRBG Reseed	Re-seed the DRBG	Approved	Reseed command with input of entropy	Status of OK or error condition	SFI-DRBG-ReSeed	CO - DRBG Entropy Input: W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- Nonce Values: W - DRBG Reseed Counter: G
Integrity Verify	Perform integrity test and return the status	None	Integrity Command	Return status of OK or error condition	None	CO
Self-tests	Initiate self-tests (Software Integrity Check, DRBG KAT, SHA2-256 KAT, HMAC-SHA2-256 KAT)	Approved	Command with list of CASTs to be performed	Return Code - OK or error condition	None	CO
Show Status	Return the status of the module state, exit codes, kernel log (dmesg)	None	Status Command	Return Code - OK or error condition	None	CO
Show Version	Return module version information	None	Version Command	SW Version	None	CO
Zeroize	Destroy/Zeroize all SSPs	Approved	Zeroize command	Return status of OK or error condition	None	CO - DRBG Entropy Input: Z - Nonce Values: Z - DRBG V: Z - DRBG Key: Z - DRBG Reseed Counter: Z - AES Keys: Z - HMAC Key: Z - KBKDF Pseudorandom Keys: Z - KBKDF Output Key: Z - HMAC SHA2-256: Z

Table 12: Approved Services

4.4 Non-Approved Services

All Approved services implemented by the Module are listed in the table below:

Name	Description	Algorithms	Role
NC Keyed Message Digest GMAC	AES-GMAC message authentication for 256-bit state table implementations	AES GMAC 256-bit (NC)	CO
NC Keyed Message Digest HMAC	HMAC generation with key size less than 112 bits	HMAC (NC)	CO
NC Keyed Message Digest HMAC MD5	HMAC generation with MD5	HMAC-MD5 (NC)	CO
NC Message Digest	Generate an MD2, MD4, or MD5 message digest	MD2, MD4, MD5 (NC)	CO
NC Random Number Generation	FIPS 186-2 Random Number Generation	RNG (NC)	CO
NC Symmetric Encryption/Decryption AES	Compute the cipher for encryption and decryption	AES-EAX (NC) AES XCBC (NC)	CO
NC Symmetric Encryption/Decryption DES	Compute the cipher for encryption and decryption	DES (NC) Triple-DES (NC)	CO
NC authenticated encryption/decryption	AES-GCM encryption and decryption for 256-bit state table implementations	AES GCM 256-bit (NC)	CO

Table 13: Non-Approved Services

4.5 External Software/Firmware Loaded

NOTE: There is no External Software/Firmware loaded.

5 Software/Firmware Security

5.1 Integrity Techniques

The Module is composed of the following software component(s):

- moc_crypto.ko: executable - binary - The kernel object that contains the cryptographic module code, data, and constants
- moc_crypto.ko.sig: - data - The integrity check signature file that contains an HMAC-SHA2-256 of the cryptographic module.

The software components are protected with the HMAC-SHA2-256 authentication technique.

The HMAC for the kernel module is calculated during the manufacturing (build) process of the kernel module. This HMAC value is stored either within the resulting “moc_crypto.ko” kernel object or as a separate “moc_crypto.ko.sig” file dependent upon the development tools and target operating system constraints.

During the load of the kernel object, the integrity check of the library code and constants occurs in the module startup function. It verifies the integrity of the kernel module by executing the HMAC-SHA2-256 fingerprint algorithm on the moc_crypto.ko file and comparing the result with the signature. This integrity check is performed as part of the function FIPS_powerupSelfTest(). This function is called automatically by the host O/S upon loading the kernel module into memory as shown below:

```
static int __init
 mss_crypto_init(void)
{
     int status = 0;
     PRINTDEBUG(“moc_crypto_init. \n”);

#ifdef __ENABLE_FIPS_POWERUP_TEST__
    if (OK > (status = FIPS_powerupSelfTest()))
    {
       PRINTDEBUG(“powerup test failed! \n”);
       goto cleanup;
     }
    else
    {
       PRINTDEBUG(“powerup test passed! \n”);
       goto cleanup;
     }
#else
     PRINTDEBUG(“powerup test disabled! \n”);
#endif

cleanup:
       return status;

}
```

```
static void  __exit
mss_crypto_fini(void)
{
```

5.2 Initiate on Demand

The operator can initiate the integrity test on demand by reloading the Module or by calling the API function: FIPS_StartupSelftestIntegrity(void).

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

The Module has a modifiable operational environment under the FIPS 140-3 definitions. The tested operational environments are listed in Section 2.2 Tested and Vendor Affirmed Module Version and Identification above. In addition, DigiCert claims that the Module can be ported on the Vendor Affirmed Operational Environment(s); no statement is made regarding the correct operation of the Module on the Vendor Affirmed Operational Environments.

For each process, session management through the operating system provides role association, process and session isolation, and memory protection. Each process has control over its own data while the operating system prevents uncontrolled access to the data and other processes. The Module does not support concurrent operators.

A software handle between the consuming application (i.e., entity) and the cryptographic module's key structure provides the key to entity association in the Module. The software handle is specific to the consuming application and is contained within its own process (e.g., handles are not shared between multiple consuming applications).

6.2 Configuration Settings and Restrictions

No operational environment restrictions are required for the operation of the Module. The operating system of the host device prevents unauthorized access to SSPs during execution of the Module. The Module allows access to SSPs only through specific APIs.

7 Physical Security

The Module is a software module at Level 1.

8 Non-Invasive Security

8.1 Mitigation Techniques

The Module does not implement any mitigation method against non-invasive attack.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
Memory (S1)	Only stored in volatile memory (RAM)	Dynamic

Table 14: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input in plaintext (IO2)	Memory (S1)	Memory (S1)	Plaintext	Manual	Electronic	
Output in plaintext (IO3)	Memory (S1)	Memory (S1)	Plaintext	Manual	Electronic	

Table 15: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Z1	Zeroized by the zeroization service by overwriting with a fixed pattern of zeros.	The application is responsible for calling the appropriate destruction functions from the API. These functions overwrite the memory with zeros and de-allocate the memory. In case of abnormal termination, the Linux kernel overwrites the keys in physical memory before the physical memory is allocated to another process.	"Zeroize" service.

Table 16: SSP Zeroization Methods

9.4 SSPs

All usage of these SSPs by the Module are described in the services detailed in Section 4.3.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG Entropy Input	Used to seed the DRBG for key generation	128-2^24 bits - 128 ≤ s ≤ 256	Entropy - CSP			SFI-DRBG-ReSeed
Nonce Values	Used to seed the DRBG for key generation	0-2^24 bits - N/A	Entropy - CSP			SFI-DRBG-ReSeed
AES Keys	Used during AES Encryption, Decryption, CMAC, and GMAC operations	128, 192, 256 bits - 128 to 256 bits	Symmetric - CSP			SFI-AES-UnAuth-Encrypt SFI-AES-UnAuth-Decrypt SFI-AES-CCM-Encrypt SFI-AES-CCM-Decrypt SFI-AES-GCM-Encrypt SFI-AES-GCM-Decrypt SFI-AES-CMAC SFI-AES-GMAC
HMAC Key	Used during HMAC-SHA operations	112-65536 - 128 to 256 bits	Symmetric - CSP			SFI-HMAC
KBKDF Pseudorandom Keys	Used in deriving other keys per SP800-108	112-4096 - 128 to 256 bits	Symmetric - CSP			SFI-HMAC-KDF
DRBG V	Internal State Value of V	128 to 256 bits - 128 ≤ s ≤ 256 bits	Internal State Critical Value - CSP	SFI-DRBG-Generate		SFI-DRBG-Generate

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG Reseed Counter	Internal State Value of Reseed Counter	64 bits - N/A	Internal State Critical Value - CSP	SFI-DRBG-ReSeed		SFI-DRBG-ReSeed
DRBG Key	Internal State Value of Key	128 to 256 bits - 128 ≤ s ≤ 256 bits	Internal State Critical Value - CSP	SFI-DRBG-ReSeed		SFI-DRBG-Generate
KBKDF Output Key	Output key derived per SP800-108	128 to 256 bits - 128 to 256 bits	Symmetric - CSP	SFI-HMAC-KDF		SFI-HMAC-KDF
HMAC SHA2-256	Used during the integrity check	256 bits - 256 bits	Symmetric - Neither			SFI-HMAC

Table 17: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG Entropy Input	Input in plaintext (IO2)	Memory (S1):Plaintext	Call lifetime	Z1	Nonce Values:Used With DRBG V:Derived From DRBG Key:Derived From
Nonce Values	Input in plaintext (IO2)	Memory (S1):Plaintext	Call lifetime	Z1	DRBG Entropy Input:Used With
AES Keys	Input in plaintext (IO2)	Memory (S1):Plaintext	Call lifetime	Z1	
HMAC Key	Input in plaintext (IO2)	Memory (S1):Plaintext	Call lifetime	Z1	
KBKDF Pseudorandom Keys	Input in plaintext (IO2)	Memory (S1):Plaintext	Call lifetime	Z1	KBKDF Output Key:Used to derive
DRBG V		Memory (S1):Plaintext	Call lifetime (module up time for internal DRBG)	Z1	DRBG Entropy Input:Used With Nonce Values:Used With DRBG Key:Used With DRBG Reseed Counter:Used With
DRBG Reseed Counter		Memory (S1):Plaintext	Call lifetime (module up time for internal DRBG)	Z1	DRBG Entropy Input:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Nonce Values:Used With DRBG V:Used With DRBG Key:Used With
DRBG Key		Memory (S1):Plaintext	Call lifetime (module up time for internal DRBG)	Z1	DRBG Entropy Input:Used With Nonce Values:Used With DRBG V:Used With DRBG Reseed Counter:Used With
KBKDF Output Key	Output in plaintext (IO3)	Memory (S1):Plaintext	Call lifetime	Z1	KBKDF Pseudorandom Keys:Derived From
HMAC SHA2-256	Input in plaintext (IO2)	Memory (S1):Plaintext	Call lifetime	Z1	

Table 18: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

The Module performs self-tests to ensure the proper operation of the Module. Per FIPS 140-3 these are categorized as either pre-operational self-tests or conditional self-tests.

Pre-operational self-tests are available on demand by power cycling the Module or reloading the Module into memory. The Module is available to perform services only after successfully completing the pre-operational self-tests.

The Module performs the following pre-operational self-tests in table below:

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256	Key Length = 256	KAT	SW/FW Integrity	Crypto module enabled upon return status of OK. ES1 error status upon KAT failure	HMAC-SHA2-256 integrity check is performed. Result is compared against the hash value in the signature .sig file

Table 19: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

The Module performs the conditional self-tests listed in the table below:

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC-Enc	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-CBC
AES-CBC-Dec	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error	Decryption	Before first use of AES-CBC

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				status upon KAT failure		
AES-CCM-Enc	Key length = 128 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-CCM
AES-CCM-Dec	Key length = 128 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-CCM
AES-CFB128-Enc	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-CFB128
AES-CFB128-Dec	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-CFB128
AES-CTR-Enc	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-CTR
AES-CTR-Dec	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-CTR

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB-Enc	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-ECB
AES-ECB-Dec	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-ECB
AES-OFB-Enc	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-OFB
AES-OFB-Dec	Key length = 256 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-OFB
AES-XTS Testing Revision 2.0-Enc	Key length = 128 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-XTS
AES-XTS Testing Revision 2.0-Dec	Key length = 128 bits	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-XTS
AES-CMAC-Gen	Key length = 128 bits	KAT	CAST	Crypto module enabled upon return	Generate	Before first use of AES-CMAC

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				status of OK. ES2 error status upon KAT failure		
AES-GCM-Enc (A847)	Key length = 128 bits for 4k modules	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-GCM
AES-GCM-Dec (A847)	Key length = 128 bits for 4k modules	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-GCM
AES-GCM-Enc (A846)	Key length = 128 bits for 64k modules	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Encryption	Before first use of AES-GCM
AES-GCM-Dec (A846)	Key length = 128 bits for 64k modules	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Decryption	Before first use of AES-GCM
AES-GMAC-Gen (A847)	Key length = 128 bits for 4k modules	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Generate	Before first use of AES-GMAC
AES-GMAC-Ver (A847)	Key length = 128 bits for 64k modules	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error	Verify	Before first use of AES-GMAC

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				status upon KAT failure		
Counter DRBG (A845)	256 Bits with and without df	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Instantiation, Generation, Reseed	Initialization
HMAC-SHA-1	SHA-1	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA-1
HMAC-SHA2-224	SHA2-224	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA2-224
HMAC-SHA2-256	SHA2-256	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA2-256. Before pre-operational integrity test.
HMAC-SHA2-384	SHA2-384	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA2-384
HMAC-SHA2-512	SHA2-512	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA2-512

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA3-224	SHA3-224 Feedback Mode	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA3-224
HMAC-SHA3-256	SHA3-256 Feedback Mode	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA3-256
HMAC-SHA3-384	SHA3-384 Feedback Mode	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA3-384
HMAC-SHA3-512	SHA3-512 Feedback Mode	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash-based Authentication	Before first use of algorithm: HMAC-SHA3-512
KDF SP800-108-SHA2-224	HMAC-SHA2-224	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA2-224
KDF SP800-108-SHA2-256	HMAC-SHA2-256	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA2-256

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDF SP800-108-SHA2-384	HMAC-SHA2-384	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA2-384
KDF SP800-108-SHA2-512	HMAC-SHA2-512	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA2-512
KDF SP800-108-SHA3-224	HMAC-SHA3-224	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA3-224
KDF SP800-108-SHA3-256	HMAC-SHA3-256	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA3-256
KDF SP800-108-SHA3-384	HMAC-SHA3-384	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA3-384
KDF SP800-108-SHA3-512	HMAC-SHA3-512	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA3-512
SHA-1	SHA-1	KAT	CAST	Crypto module enabled upon return	Hash	Before first use of algorithm: SHA-1

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				status of OK. ES2 error status upon KAT failure		
SHA2-224	SHA2-224	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHA2-224
SHA2-256	SHA2-256	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHA2-256
SHA2-384	SHA2-384	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHA2-384
SHA2-512	SHA2-512	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHA2-512
SHA3-224	SHA3-224	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHA3-224
SHA3-256	SHA3-256	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error	Hash	Before first use of algorithm: SHA3-256

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				status upon KAT failure		
SHA3-384	SHA3-384	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHA3-384
SHA3-512	SHA3-512	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHA3-512
SHAKE-128	SHAKE-128	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHAKE-128
SHAKE-256	SHAKE-256	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Hash	Before first use of algorithm: SHAKE-256
KDF SP800-108-SHA-1	HMAC-SHA-1	KAT	CAST	Crypto module enabled upon return status of OK. ES2 error status upon KAT failure	Key Derivation	Before first use of algorithm: KDF HMAC-SHA-1
AES-XTS Key Comparison	Key length = 128 bits	AES-XTS key comparison	Critical Function	OK or ES3	IG C.I key comparison test is performed to verify Key1 != Key2.	Before first use
AES-CMAC-Ver	Key length = 128 bits	KAT	CAST	OK or ES2	Verify	Before first use

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GMAC-Gen (A846)	Key length = 128 bits for 64k modules	KAT	CAST	OK or ES2	Generate	Before first use
AES-GMAC-Ver (A846)	Key length = 128 bits for 64k modules	KAT	CAST	OK or ES2	Verify	Before first use

Table 20: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256	KAT	SW/FW Integrity	On demand	By power cycling or reloading the Module into memory

Table 21: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC-Enc	KAT	CAST	On demand	Manually
AES-CBC-Dec	KAT	CAST	On demand	Manually
AES-CCM-Enc	KAT	CAST	On demand	Manually
AES-CCM-Dec	KAT	CAST	On demand	Manually
AES-CFB128-Enc	KAT	CAST	On demand	Manually
AES-CFB128-Dec	KAT	CAST	On demand	Manually
AES-CTR-Enc	KAT	CAST	On demand	Manually
AES-CTR-Dec	KAT	CAST	On demand	Manually
AES-ECB-Enc	KAT	CAST	On demand	Manually
AES-ECB-Dec	KAT	CAST	On demand	Manually
AES-OFB-Enc	KAT	CAST	On demand	Manually
AES-OFB-Dec	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0-Enc	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-XTS Testing Revision 2.0-Dec	KAT	CAST	On demand	Manually
AES-CMAC-Gen	KAT	CAST	On demand	Manually
AES-GCM-Enc (A847)	KAT	CAST	On demand	Manually
AES-GCM-Dec (A847)	KAT	CAST	On demand	Manually
AES-GCM-Enc (A846)	KAT	CAST	On demand	Manually
AES-GCM-Dec (A846)	KAT	CAST	On demand	Manually
AES-GMAC-Gen (A847)	KAT	CAST	On demand	Manually
AES-GMAC-Ver (A847)	KAT	CAST	On demand	Manually
Counter DRBG (A845)	KAT	CAST	Initialization and on demand	Manually
HMAC-SHA-1	KAT	CAST	On demand	Manually
HMAC-SHA2-224	KAT	CAST	On demand	Manually
HMAC-SHA2-256	KAT	CAST	On demand	Manually
HMAC-SHA2-384	KAT	CAST	On demand	Manually
HMAC-SHA2-512	KAT	CAST	On demand	Manually
HMAC-SHA3-224	KAT	CAST	On demand	Manually
HMAC-SHA3-256	KAT	CAST	On demand	Manually
HMAC-SHA3-384	KAT	CAST	On demand	Manually
HMAC-SHA3-512	KAT	CAST	On demand	Manually
KDF SP800-108-SHA2-224	KAT	CAST	On demand	Manually
KDF SP800-108-SHA2-256	KAT	CAST	On demand	Manually
KDF SP800-108-SHA2-384	KAT	CAST	On demand	Manually
KDF SP800-108-SHA2-512	KAT	CAST	On demand	Manually
KDF SP800-108-SHA3-224	KAT	CAST	On demand	Manually
KDF SP800-108-SHA3-256	KAT	CAST	On demand	Manually
KDF SP800-108-SHA3-384	KAT	CAST	On demand	Manually
KDF SP800-108-SHA3-512	KAT	CAST	On demand	Manually
SHA-1	KAT	CAST	On demand	Manually
SHA2-224	KAT	CAST	On demand	Manually
SHA2-256	KAT	CAST	On demand	Manually
SHA2-384	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA2-512	KAT	CAST	On demand	Manually
SHA3-224	KAT	CAST	On demand	Manually
SHA3-256	KAT	CAST	On demand	Manually
SHA3-384	KAT	CAST	On demand	Manually
SHA3-512	KAT	CAST	On demand	Manually
SHAKE-128	KAT	CAST	On demand	Manually
SHAKE-256	KAT	CAST	On demand	Manually
KDF SP800-108-SHA-1	KAT	CAST	On demand	Manually
AES-XTS Key Comparison	AES-XTS key comparison	Critical Function	On demand	Manually
AES-CMAC-Ver	KAT	CAST	On demand	Manually
AES-GMAC-Gen (A846)	KAT	CAST	On demand	Manually
AES-GMAC-Ver (A846)	KAT	CAST	On demand	Manually

Table 22: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
ES1	The Module enters the disable Crypto Module "Error State"	The Module fails the software integrity pre-operational self-test.	Reboot/Power cycle the module	Outputs status of ERR_FIPS_INTEGRITY_FAIL, otherwise it indicates successful completion by returning the OK status.
ES2	The Module enters the disable Crypto Module "Error State"	The Module fails the software CAST test with a specified error number.	Reboot/Power cycle the module	Outputs a specific error status; otherwise, it indicates successful completion by enabling the Crypto Module with OK status.
ES3	The Module enters the disable Crypto Module "Error State"	The Module fails all other self-tests not listed above.	Reboot/Power cycle the module	Outputs a specific error status; otherwise, it indicates successful completion by enabling the Crypto Module with OK status.

Table 23: Error States

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

Installation is performed by placing the Module in the target file system during the OEM or ISV's manufacturing process. The Module initialization is performed automatically by the operating system's loader when a calling application is loaded into memory. Operation of the Module is controlled by the calling application's use of the Module's API functions.

Installation and Initialization:

The following steps must be performed in order to securely install, initialize, and start up the DigiCert TrustCore Cryptographic Loadable Kernel Module in the FIPS 140-3 Approved mode of operation:

The Module shall be installed within the operating system confines and structures consistent with DigiCert's operating environment's specific documentation. The Cryptographic Officer will install the Module and associated signature of the Module into the proper location within the computer system. For example, the moc_crypto.ko kernel module and signature file will be installed in the file system in /lib or /lib64, or it may be specified in the operating environment specific documentation to be installed in the /usr/local/lib directory, which is protected by Linux access control mechanisms. The Module is protected from modification by the integrity self-test performed during start-up. The Module is initialized by the operating system upon loading the Module into memory for use by calling applications.

The Module must be operated in the Approved mode to ensure that FIPS 140-3 validated cryptographic algorithms and security functions are used.

In addition, the security rules defined in the Rules of Operation section shall apply to the operating system.

11.2 Administrator Guidance

The Module is provided with supporting documentation which includes an API Reference document and Operating Environment document.

11.3 Non-Administrator Guidance

The Module supports the Cryptographic Officer (CO) operator role and does not support non-administrators or non-administrative roles.

11.4 Design and Rules

(Random Number Generation)
The Module implements a CTR-based DRBG per SP800-90Ar1 for creation of symmetric and asymmetric keys.
The Module accepts input from entropy sources external to the cryptographic boundary for use as seed material for the Module's Approved DRBG. External entropy can be added via the AES-CTR-DRBG Gen and the AES-CTR-DRBG ReSeed service available to the cryptographic module client application.

The calling application of the Module shall use entropy sources that meet the security strength required for the random bit generation mechanism as shown in NIST SP 800-90Ar1 Table 3 (CTR_DRBG). A minimum of 384 bits of entropy must be provided by the calling application. The calling application shall provide full entropy for 256-bit keys. Due to the entropy being provided by an external source, the following caveat applies: There is no assurance of the minimum strength of generated SSPs (e.g., keys). The Module performs DRBG health tests (Instantiate, Generate, ReSeed) as defined in section 11.3 of SP800-90Ar1.

(Key Management)
The application that uses the Module is responsible for appropriate destruction and zeroization of the keys. The Module provides API calls for key allocation and destruction. These API calls overwrite the memory occupied by the key information with zeros before that memory is deallocated. See Key Destruction Service below.

(Key/CSP Authorized Access and Use)
An authorized application has access to all key data generated during the operation of the Module.

(Key/CSP Storage)
Private and public keys are provided to the Module by the calling process and are destroyed when released by the appropriate API function calls. The Module does not perform persistent storage of keys.

(Key/CSP Zeroization)
The application is responsible for calling the appropriate destruction functions from the API. These functions overwrite the memory with zeros and deallocate the memory. In case of abnormal termination, the Linux kernel overwrites the keys in physical memory before the physical memory is allocated to another process.

(Key Destruction Service)
A context structure is associated with every cryptographic algorithm available in the Module. Context structures hold sensitive information such as cryptographic keys. These context structures must be zeroized when the application software no longer needs to use a specific algorithm. This API call will zeroize all sensitive information before freeing the dynamically allocated memory. This will occur while the application process is still in memory, but no longer needs the specific algorithm, which protects the sensitive information from compromise. See the *Cryptographic API Reference* for additional information.

Rules of Operation

1. The Module provides one operator role: Cryptographic Officer.
2. The Module does not provide any operator authentication.
3. An operator does not have access to any cryptographic services prior to assuming an authorized role.
4. The Module allows the operator to initiate power-up self-tests by power cycling or reloading the Module into memory.
5. All self-tests do not require any operator action.
6. Data and Control outputs are inhibited during key generation, self-tests, zeroization, and error states. Because the logical interface is defined as the API of the Module and the API of the Module is single-threaded, key generation or zeroization must be complete before the API returns control to the calling application.

7. Status information does not contain CSPs or sensitive data that, if misused, could lead to a compromise of the Module.
8. There are no restrictions on which keys or SSPs are zeroized by the zeroization service.
9. The Module does not support concurrent operators.
10. The Module does not support a maintenance interface or role.
11. The Module does not support a manual SSP establishment method.
12. The Module does not have any proprietary external input/output devices used for entry/output of data.
13. The Module does not enter or output plaintext SSPs, except to/from the calling application via API parameters. The Module does not support the entry or output of encrypted SSPs.
14. The Module does not store any plaintext SSPs. SSPs provided to the Module by the calling processes are destroyed when released by the appropriate API function calls.
15. The Module does not output intermediate key values.
16. The Module does not provide bypass services or ports/interfaces.
17. AES GCM IV uniqueness: The AES GCM implementation meets Option 1 of IG C.H. The Module supports TLS 1.2 GCM Cipher Suites for TLS, as described in RFCs 5116, 5288 and 5289. The counter portion of the IV is set by the Module within its cryptographic boundary.
18. When the nonce explicit (counter) part of the IV exhausts the maximum number of possible values for a given session key this condition triggers a handshake to establish a new encryption key per RFC 5246. During operational testing, the Module was tested against an independent version of TLS and found to behave correctly.
    AES GCM keys are zeroized when the Module is power-cycled and for each new TLS session, a new AES GCM key is established.
19. AES XTS is to be used only for storage purposes, per SP800-38E.

11.5 Maintenance Requirements

The Module currently does not have any maintenance requirements.

11.6 End of Life

For secure sanitization all SSPs shall first be zeroized and the calling application shall be closed. SSP zeroization is performed through the Key Destruction service that is described below. API calls will overwrite the memory occupied by the key information with zeros before that memory is deallocated. If the calling application is terminated prior to zeroization, the Linux kernel overwrites the keys in physical memory before the physical memory is allocated to another process. The key zeroization process is performed in a sufficient time to prevent compromise of SSPs, taking only a few milliseconds.

Then, for actual deprecation, the Module shall be upgraded to a newer version that is FIPS 140-3 validated. Since the Module does not possess persistent storage of SSPs, no further sanitization steps are needed.

12 Mitigation of Other Attacks

The Module does not implement any mitigation method against other attacks.

References and Definitions

The following standards are referred to in this Security Policy.

Table 24: References

Abbreviation*	Full Specification Name
[FIPS140-3]	*Security Requirements for Cryptographic Modules*, March 22, 2019
[ISO19790]	*International Standard, ISO/IEC 19790, Information technology — Security techniques — Test requirements for cryptographic modules, Third edition, March 2017*
[ISO24759]	*International Standard, ISO/IEC 24759, Information technology — Security techniques — Test requirements for cryptographic modules, Second and Corrected version, 15 December 2015*
[IG]	*Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program, <date>*
[108]	*NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions (Revised), October 2009*
[131A]	*Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, Revision 2, March 2019*
[132]	*NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation, Part 1: Storage Applications, December 2010*
[133]	*NIST Special Publication 800-133r2, Recommendation for Cryptographic Key Generation, Revision 2, June 2020*
[135]	*National Institute of Standards and Technology, Recommendation for Existing Application-Specific Key Derivation Functions, Special Publication 800-135rev1, December 2011.*
[186]	*National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-4, July 2013.*
[197]	*National Institute of Standards and Technology, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, November 26, 2001*
[198]	*National Institute of Standards and Technology, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198-1, July, 2008*
[180]	*National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standards Publication 180-4, August, 2015*
[202]	*FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, August 2015*
[38A]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation, Methods and Techniques, Special Publication 800-38A, December 2001*

Abbreviation*	Full Specification Name
[38B]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication 800-38B, May 2005*
[38C]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, Special Publication 800-38C, May 2004*
[38D]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, Special Publication 800-38D, November 2007*
[38E]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, Special Publication 800-38E, January 2010*
[38F]	*National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, Special Publication 800-38F, December 2012*
[56Ar3]	*NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, April 2018*
[56Br2]	*NIST Special Publication 800-56B Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Finite Field Cryptography, March 2019*
[56Cr2]	*NIST Special Publication 800-56C Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, August 2020*
[67]	*National Institute of Standards and Technology, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication 800-67, May 2004*
[90A]	*National Institute of Standards and Technology, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication 800-90A, Revision 1, June 2015.*
[90B]	*National Institute of Standards and Technology, Recommendation for the Entropy Sources Used for Random Bit Generation, Special Publication 800-90B, January 2018.*

Table 25: Acronyms and Definitions

Acronym*	Definition
AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Program Interface
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CMAC	Cipher-based Message Authentication Code

Acronym*	Definition
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
FIPS	Federal Information Processing Standard
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
IG	Implementation Guidance
KAT	Known Answer Test
KDF	Key Derivation Function
KVM	Kernel-based Virtual Machine
NC	Non-Compliant
PAA	Processor Algorithm Acceleration
PCT	Pair-wise Consistency Test
RNG	Random Number Generator
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing