Ctrl IQ, Inc.

Rocky Linux 9 Kernel Cryptographic API

FIPS 140-3 Non-Proprietary Security Policy

Prepared by:

atsec information security corporation
4516 Seton Center Pkwy, Suite 250
Austin, TX 78759
www.atsec.com

Document version: 1.0
Last update: 2026-07-22

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 kernel rocky9.20260721; libkcapi-1.4.0-2.el9 of the Rocky Linux 9 Kernel Cryptographic API 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.

This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice.

1.1.1 How this Security Policy was Prepared

In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

1.2 Security Levels

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

2.1 Description

Purpose and Use:

The Rocky Linux 9 Kernel Cryptographic API (hereafter referred to as "the module") provides a C language application program interface (API) for use by other (kernel space and user space) processes that require cryptographic functionality. The module operates on a general-purpose computer as part of the Linux kernel. Its cryptographic functionality can be accessed using the Linux Kernel Crypto API.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

The cryptographic boundary of the module is defined as the kernel binary and the kernel crypto object files, the libkcapi shared library, and the sha512hmac binary, which is used to verify the integrity of the kernel binary. In addition, the cryptographic boundary contains the .hmac files which store the expected integrity values for each of the software components.

Tested Operational Environment's Physical Perimeter (TOEPP):

The TOEPP of the module is defined as the general-purpose computer on which the module is installed. The PAA provided by the processor is located within the module's physical perimeter and outside the module's cryptographic boundary. The TOEPP is indicated by the large thin border in Figure 1.

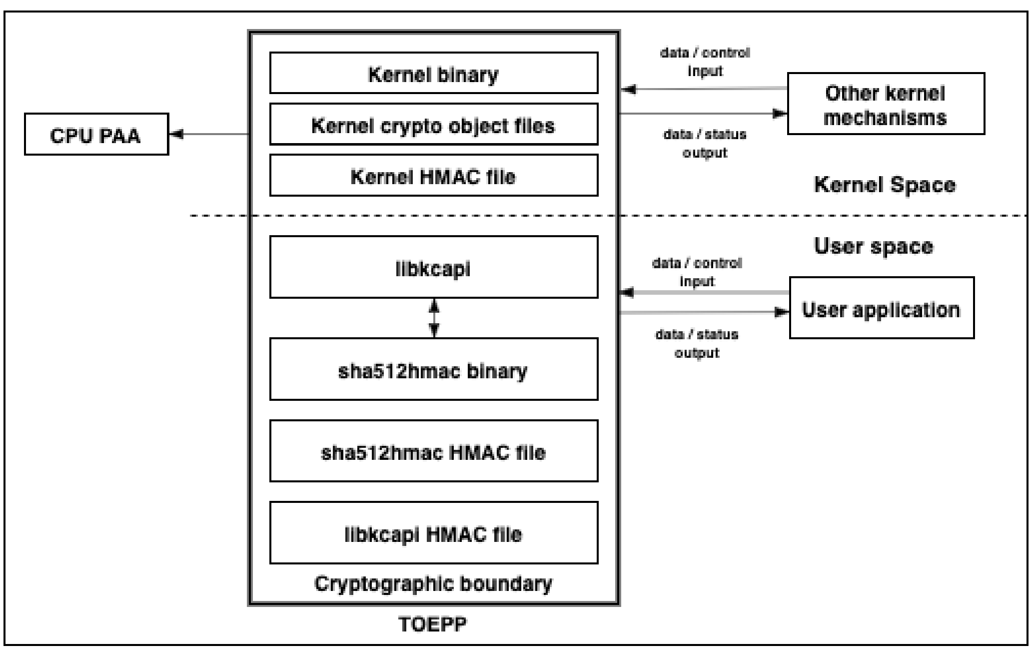

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
/boot/vmlinuz-5.14.0-570.39.1.el9_6_ciqfips.3.1.x86_64; /usr/lib/modules/5.14.0-570.39.1.el9_6_ciqfips.3.1.x86_64/kernel/crypto/{,*/}*.ko.xz; /usr/lib/modules/5.14.0-570.39.1.el9_6_ciqfips.3.1.x86_64/kernel/arch/x86/crypto/*.ko.xz; /lib64/libkcapi.so.1.4.0; /usr/bin/sha512hmac	kernel: rocky9.20260721; libkcapi: 1.4.0-2.el9	N/A	HMAC-SHA2-256 (libkcapi.so); HMAC-SHA2-512 (vmlinuz, sha512hmac); RSA signature verification (*.ko.xz files)

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	Yes	N/A	kernel: rocky9.20260721; libkcapi: 1.4.0-2.el9
Rocky Linux 9	SuperMicro SuperServer 5039MS	Intel Kaby Lake Xeon E3-1270 v6	No	N/A	kernel: rocky9.20260721; libkcapi: 1.4.0-2.el9

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

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

N/A for this module.

2.3 Excluded Components

There are no components within the cryptographic boundary excluded from the FIPS 140-3 requirements.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Mapped to approved service indicator in Section 4.3 for all approved algorithms except AES GCM: respective approved service function returns indicator 0; For AES GCM: crypto_aead_get_flags(tfm) has the CRYPTO_TFM_FIPS_COMPLIANCE flag set
Non-approved mode	Automatically entered whenever a non-approved service is requested	Non-Approved	No service indicator required for non-approved services per IG 2.4.C

Table 4: Modes List and Description

After the module is powered on, it performs all the pre-operational self-tests and cryptographic algorithm self-tests without operator intervention. Only after all the self-tests are successful, the module automatically transitions to the approved mode.

Mode Change Instructions and Status:

The module automatically switches between the approved and non-approved modes depending on the services requested by the operator. The status indicator of the mode of operation is equivalent to the indicator of the service that was requested.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A7403, A7407, A7410	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS3	A7403, A7407, A7410	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A7403, A7410	Key Length - 128, 192, 256	SP 800-38C
AES-CFB128	A7403, A7410	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A7403, A7410	Direction - Generation Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A7403, A7407, A7410	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A7403, A7405, A7406, A7407, A7408, A7409, A7410, A7411, A7412	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-GCM	A7403, A7406, A7407, A7409, A7410, A7412	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D
AES-GCM	A7405, A7408, A7411	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A7403, A7410	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D
AES-OFB	A7403, A7410	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A7403, A7407, A7410	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A7403, A7405, A7406, A7407, A7408, A7409, A7410, A7411, A7412	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
Hash DRBG	A7403, A7413, A7414, A7415	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC DRBG	A7403, A7413, A7414, A7415	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC-SHA-1	A7403, A7413, A7414, A7415	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A7403, A7413, A7414, A7415	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A7403, A7413, A7414, A7415	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A7403, A7413, A7414, A7415	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A7403, A7413, A7414, A7415	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A7403	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A7403	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A7403	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A7403	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-FFC-SSC Sp800-56Ar3	A7403	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
RSA SigVer (FIPS186-4)	A7403	Signature Type - PKCS 1.5 Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-5)	A7403	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5	FIPS 186-5
Safe Primes Key Generation	A7403	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	SP 800-56A Rev. 3
SHA-1	A7403, A7413, A7414, A7415	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 180-4
SHA2-224	A7403, A7413, A7414, A7415	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 180-4
SHA2-256	A7403, A7413, A7414, A7415	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 180-4
SHA2-384	A7403, A7413, A7414, A7415	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 180-4
SHA2-512	A7403, A7413, A7414, A7415	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 180-4
SHA3-224	A7403	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 202
SHA3-256	A7403	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 202
SHA3-384	A7403	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 202
SHA3-512	A7403	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2	FIPS 202

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

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

Table 6: Vendor-Affirmed Algorithms

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:

Name	Use and Function
AES-GCM with external IV	Encryption with external IV (not compliant to FIPS 140-3 IG C.H)
KBKDF in libkcapi	Key derivation with implementation not tested by CAVP
HKDF in libkcapi	Key derivation with implementation not tested by CAVP
PBKDF2 in libkcapi	Password-based Key derivation with implementation not tested by CAVP
RSA PKCS#1 v1.5 with pre-hashed message	Signature generation(pre-hashed message) primitive; Signature verification(pre-hashed message) primitive
RSA PKCS#1 v1.5	Key encapsulation / un-encapsulation
RSA primitive	Encryption primitive; Decryption primitive (not compliant to SP 800-56Br2)
ECDSA with pre-hashed message	Signature generation(pre-hashed message) primitive; Signature verification(pre-hashed message) primitive

Table 7: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Encryption	BC-UnAuth	Encrypt a plaintext		AES-CBC: (A7403, A7407, A7410) AES-CBC-CS3: (A7403, A7407, A7410) AES-CFB128: (A7403, A7410) AES-CTR: (A7403, A7407, A7410) AES-ECB: (A7403, A7405, A7406, A7407, A7408, A7409, A7410, A7411, A7412) AES-OFB: (A7403, A7410) AES-XTS Testing Revision 2.0: (A7403, A7407, A7410)

Name	Type	Description	Properties	Algorithms
Decryption	BC-UnAuth	Decrypt a plaintext		AES-CBC: (A7403, A7407, A7410) AES-CBC-CS3: (A7403, A7407, A7410) AES-CFB128: (A7403, A7410) AES-CTR: (A7403, A7407, A7410) AES-ECB: (A7403, A7405, A7406, A7407, A7408, A7409, A7410, A7411, A7412) AES-OFB: (A7403, A7410) AES-XTS Testing Revision 2.0: (A7403, A7407, A7410)
Authenticated encryption	BC-Auth	Encrypt and authenticate a plaintext		AES-CCM: (A7403, A7410) AES-GCM: (A7403, A7405, A7406, A7407, A7408, A7409, A7410, A7411, A7412)
Authenticated decryption	BC-Auth	Decrypt and authenticate a ciphertext		AES-CCM: (A7403, A7410) AES-GCM: (A7403, A7405, A7406, A7407, A7408, A7409, A7410, A7411, A7412)
Message digest	SHA	Compute a message digest		SHA-1: (A7403, A7413, A7414, A7415) SHA2-224: (A7403, A7413, A7414, A7415) SHA2-256: (A7403, A7413, A7414, A7415) SHA2-384: (A7403, A7413, A7414, A7415)

Name	Type	Description	Properties	Algorithms
				SHA2-512: (A7403, A7413, A7414, A7415) SHA3-224: (A7403) SHA3-256: (A7403) SHA3-384: (A7403) SHA3-512: (A7403)
Message authentication	MAC	Compute a MAC tag		AES-CMAC: (A7403, A7410) AES-GMAC: (A7403, A7410) HMAC-SHA-1: (A7403, A7413, A7414, A7415) HMAC-SHA2-224: (A7403, A7413, A7414, A7415) HMAC-SHA2-256: (A7403, A7413, A7414, A7415) HMAC-SHA2-384: (A7403, A7413, A7414, A7415) HMAC-SHA2-512: (A7403, A7413, A7414, A7415) HMAC-SHA3-224: (A7403) HMAC-SHA3-256: (A7403) HMAC-SHA3-384: (A7403) HMAC-SHA3-512: (A7403)
Random Number Generation with CTR_DRBG	DRBG	Generate random bytes		Counter DRBG: (A7403, A7405, A7406, A7407, A7408, A7409, A7410, A7411, A7412)
Random Number Generation with HMAC_DRBG	DRBG	Generate random bytes		HMAC DRBG: (A7403, A7413, A7414, A7415)
Random Number Generation with Hash_DRBG	DRBG	Generate random bytes		Hash DRBG: (A7403, A7413, A7414, A7415)

Name	Type	Description	Properties	Algorithms
Shared secret computation	KAS-SSC	Compute a shared secret	IG:IG D.F, Scenario 2 path (1) Caveat:Key establishment methodology provides between 112 and 200 bits of security strength	KAS-FFC-SSC Sp800-56Ar3: (A7403)
Signature verification with RSA	DigSig-SigVer	Verify a digital signature		RSA SigVer (FIPS186-5): (A7403)
Signature verification with RSA (legacy use)	DigSig-SigVer	Verify a digital signature using RSA with SHA-1		RSA SigVer (FIPS186-4): (A7403)
Key pair generation	CKG	Generate a key pair		Safe Primes Key Generation: (A7403) Asymmetric Cryptographic Key Generation (CKG): () Key type: Asymmetric

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-GCM IV

For IPsec, the module offers the AES-GCM implementation and uses the context of Scenario 1 of FIPS 140-3 IG C.H. The mechanism for IV generation is compliant with RFC 4106. IVs generated using this mechanism may only be used in the context of AES-GCM encryption within the IPsec protocol.

The module does not implement IPsec. The module's implementation of AES GCM is used together with an application that runs outside the module's cryptographic boundary. This application must use RFC 7296 compliant IKEv2 to establish the shared secret SKEYSEED from which the AES-GCM encryption keys are derived.

The design of the IPsec protocol implicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values for a given session key. In the event the module's power is lost and restored, the consuming application must ensure that a new key for use with the AES-GCM key encryption or decryption under this scenario shall be established.

The module also provides a non-approved AES-GCM encryption service which accepts arbitrary external IVs from the operator. This service can be requested by invoking the crypto_aead_encrypt API function with an AES-GCM handle. When this is the case, the API will not set an approved service indicator, as described in this document.

2.7.2 AES-XTS

The length of a single data unit encrypted or decrypted with AES-XTS shall not exceed 2^{20} AES blocks, that is 16MB, of data per XTS instance. An XTS instance is defined in Section 4 of SP 800-38E. The XTS mode shall only be used for the cryptographic protection of data on storage devices. It shall not be used for other purposes, such as the encryption of data in transit.

To meet the requirement stated in IG C.I, the module implements a check to ensure that the two AES keys used in AES XTS mode are not identical. As the module does not implement symmetric key generation, this check is performed when the keys are input by the operator. Key_1 and Key_2 shall be generated and/or established independently according to the rules for component symmetric keys from NIST SP 800-133r2, Section 6.3.

2.7.3 RSA

All supported modulus sizes for RSA signature verification have been CAVP tested.

2.7.4 SP 800-56Ar3 Assurances

To comply with the assurances found in Section 5.6.2 of SP 800-56Ar3, the operator must use the Diffie-Hellman shared secret computation algorithms with the NVMe and Bluetooth related protocols. Additionally, the module's approved key pair generation service must be used to generate ephemeral Diffie-Hellman key pairs, or the key pairs must be obtained from another FIPS-validated module. As part of this service, the module will internally perform the full public key validation of the generated public key.

The module's shared secret computation service will internally perform the full public key validation of the peer DH public key, complying with Section 5.6.2.2.2 of SP 800-56Ar3.

2.7.5 SHA-3

The module implements the SHA-3 functions as both standalone functions and as part of higher-level algorithms (in compliance with FIPS 140-3 IG C.C). As detailed in Section 2.6 Security Function Implementations with corresponding certificates, the cryptographic algorithms that use SHA-3 functions include HMAC.

2.7.6 Shared Secret Computation

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

2.7.7 Authenticated Encryption and Decryption

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

2.7.8 Legacy Use

Digital signature verification using SHA-1 is allowed for legacy use only. Algorithms designated as "Legacy" can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M.

2.8 RBG and Entropy

Cert Number	Vendor Name
E205	Ctrl IQ, Inc.

Table 9: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Rocky Linux Kernel CPU Time Jitter RNG Entropy Source	Non-Physical	Rocky Linux 9 on Intel Kaby Lake Xeon E3-1270 v6	256 bits	256 bits	SHA3-256 (A5837)

Table 10: Entropy Sources

The module implements three different Deterministic Random Bit Generator (DRBG) implementations based on SP 800-90Ar1: CTR_DRBG, Hash_DRBG, and HMAC_DRBG. Each of these DRBG implementations can be instantiated by the operator of the module. When instantiated, these DRBGs can be used to generate random numbers for external usage.

Additionally, the module employs a specific HMAC-SHA2-512 DRBG implementation for internal purposes (e.g. to generate asymmetric key pairs). This DRBG is initially seeded with 384 output bits from the entropy source (384 bits of entropy) and reseeded with 256 output bits from the entropy source (256 bits of entropy).

The module complies with the Public Use Document for ESV certificate E205 by reading entropy data from the jent_kcapi_random() function, which corresponds to the GetEntropy() conceptual interface. This function outputs 256 bits of full entropy.

The operational environment on the ESV certificate is identical to the operating system described in this document, and the entropy source is implemented inside the cryptographic boundary. Thus, the module is compliant with scenario 1 of IG 9.3.A. There are no maintenance requirements for the entropy source.

2.9 Key Generation

The module implements asymmetric key pair generation compliant with SP 800-133r2. When random values are required, they are directly obtained as output from the SP 800-90Ar1 approved DRBG, compliant with Section 4 of SP 800-133r2 (without XOR). The methods of asymmetric key pair generation implemented are stated in Section 2.6.

Intermediate key generation values are not output from the module and are explicitly zeroized after processing the service.

2.10 Key Establishment

The module implements shared secret computation methods as listed in the Security Function Implementations table in Section 2.6.

2.11 Industry Protocols

AES-GCM with internal IV generation in the approved mode is compliant with RFC 4106 and shall only be used in conjunction with the IPsec protocol.

Diffie-Hellman shall only be used with the NVMe related protocol.

No parts of the NVMe or IPSec protocols, other than those mentioned above, have been tested by the CAVP and CMVP.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API data input parameters, AF_ALG type input sockets
N/A	Data Output	API data output parameters, AF_ALG type output sockets, /proc/sys/crypto virtual files
N/A	Control Input	API function calls, API control input parameters, AF_ALG type input sockets, kernel command line arguments
N/A	Status Output	API return values, AF_ALG type output sockets, kernel logs

Table 11: Ports and Interfaces

The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement any authentication methods.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	Crypto Officer	None

Table 12: Roles

No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Encryption	Encrypt a plaintext	crypto_skcipher_setkey returns 0	AES key, plaintext, IV (if required)	Ciphertext	Encryption	Crypto Officer - AES key: W,E
Decryption	Decrypt a ciphertext	crypto_skcipher_setkey returns 0	AES key, ciphertext, IV (if required)	Plaintext	Decryption	Crypto Officer - AES key: W,E
Authenticated encryption	Encrypt and authenticate a plaintext	For all except AES GCM: crypto_aead_setkey returns 0; For AES GCM: crypto_aead_get_flags(tfm) has the CRYPTO_TFM_FIPS_COMPLIANCE flag set	AES key, plaintext, IV (CCM/GCM)	Ciphertext, MAC tag (CCM/GCM)	Authenticated encryption	Crypto Officer - AES key: W,E - GCM IV: G,R,E
Authenticated decryption	Decrypt and authenticate a ciphertext	crypto_aead_setkey returns 0	AES key, ciphertext, IV (CCM/GCM), MAC tag (CCM/GCM)	Plaintext or failure	Authenticated decryption	Crypto Officer - AES key: W,E - GCM IV: W,E
Message digest	Compute a message digest	crypto_shash_init returns 0	Message	Digest value	Message digest	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Message authentication	Compute a MAC tag	crypto_shash_init returns 0	AES key or HMAC key, message	MAC tag	Message authentication	Crypto Officer - AES key: W,E - HMAC key: W,E
Random number generation	Generate random bytes	crypto_rng_get_bytes returns 0	Output length	Random bytes	Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG	Crypto Officer - Entropy input: G,E,Z - CTR_DRBG Seed (IG D.L): G,E - Hash_DRBG Seed (IG D.L): G,E - HMAC_DRBG Seed (IG D.L): G,E - CTR_DRBG Internal State (V, Key) (IG D.L): G,W,E - Hash_DRBG Internal State (V, C) (IG D.L): G,W,E - HMAC_DRBG Internal State (V, Key) (IG

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						D.L): G,W,E
Random number generation using entropy source	Generate entropy from a dedicated instance of Jitter RNG	crypto_rng_get_bytes returns 0	Output length	Random bytes		Crypto Officer
Shared secret computation	Compute a shared secret	crypto_kpp_compute_shared_secret returns 0	Owner private key, peer public key	Shared secret	Shared secret computation	Crypto Officer - DH private key: W,E - DH public key: W,E - Shared secret: G,R
Key pair generation	Generate a key pair	crypto_kpp_set_secret and crypto_kpp_generate_public_key return 0	Group	Key pair	Key pair generation	Crypto Officer - DH private key: G,R - DH public key: G,R - Intermediate key generation value: G,E,Z
Error detection code	Compute an EDC (crc32, crc32c, crct10dif)	None	Message	EDC	None	Crypto Officer
Compression	Compress data (deflate, lz4, lz4hc, lzo, zlib-deflate, zstd)	None	Data	Compressed data	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Generic system call	Use the kernel to perform various non-cryptographic operations	None	Identifier, various arguments	Various return values	None	Crypto Officer
Show version	Return the module name and version information	None	N/A	Module name and version	None	Crypto Officer
Show status	Return the module status	None	N/A	Module status	None	Crypto Officer
Self-test	Perform the CASTs and integrity tests	None	N/A	Pass/fail	Encryption Decryption Authenticated encryption Authenticated decryption Message digest Message authentication Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Random Number Generation with Hash_DRBG Shared secret computation Signature verification with RSA Signature verification with RSA (legacy use) Key pair generation	
Zeroization	Zeroize SSPs	None	Any SSP	N/A	None	Crypto Officer - AES key: Z - HMAC key: Z - Shared secret: Z - Entropy input: Z - CTR_DRBG Seed (IG D.L): Z - Hash_DRBG Seed (IG D.L): Z - HMAC_DRBG Seed (IG D.L): Z -

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						CTR_DRBG Internal State (V, Key) (IG D.L): Z - Hash_DRBG Internal State (V, C) (IG D.L): Z - HMAC_DRBG Internal State (V, Key) (IG D.L): Z - DH public key: Z - DH private key: Z - Intermediate key generation value: Z

Table 13: Approved Services

For the above table, the convention below applies when specifying the access permissions (types) that the service has for each SSP.

- **Generate (G):** The module generates or derives the SSP.
- **Read (R):** The SSP is read from the module (e.g. the SSP is output).
- **Write (W):** The SSP is updated, imported, or written to the module.
- **Execute (E):** The module uses the SSP in performing a cryptographic operation.
- **Zeroize (Z):** The module zeroizes the SSP.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
AES-GCM with external IV encryption	Encrypt and authenticate a plaintext using AES-GCM with an external IV	AES-GCM with external IV	Crypto Officer

Name	Description	Algorithms	Role
Key derivation	Derive a key from a key-derivation key, shared secret, or password	KBKDF in libkcapi HKDF in libkcapi PBKDF2 in libkcapi	Crypto Officer
Pre-hashed message signature generation	Generate a digital signature for a pre-hashed message	RSA PKCS#1 v1.5 with pre-hashed message ECDSA with pre-hashed message	Crypto Officer
Pre-hashed message signature verification	Verify a digital signature for a pre-hashed message	RSA PKCS#1 v1.5 with pre-hashed message ECDSA with pre-hashed message	Crypto Officer
Key encapsulation	Key encapsulation using RSA PKCS#1 v1.5	RSA PKCS#1 v1.5	Crypto Officer
Key un-encapsulation	Key un-encapsulation using RSA PKCS#1 v1.5	RSA PKCS#1 v1.5	Crypto Officer
Encryption primitive	Compute the RSA encryption primitive	RSA primitive	Crypto Officer
Decryption primitive	Compute the RSA decryption primitive	RSA primitive	Crypto Officer

Table 14: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

On system boot, the kernel crypto object files are loaded and verified. Each kernel object file has a signature appended to it which is verified using RSA signature generation with PKCS#1 v1.5 padding, SHA2-256, and a hardcoded 3072-bit public key embedded in the kernel. If the signature verification fails for any module, the boot process terminates. After successful signature verification, all loaded cryptographic algorithms undergo cryptographic algorithm self-tests. Once the kernel object files are loaded and operational, the sha512hmac utility and its supporting libkcapi library are integrity tested using the module's implementation of HMAC-SHA2-256. These tests use a key hardcoded in the sha512hmac binary to recompute the MAC tags and verify they match the values stored in the pre-computed .hmac files. Finally, the kernel binary itself is integrity tested using HMAC-SHA2-512. The sha512hmac utility computes the HMAC of the kernel image file and compares it to the pre-computed file in the corresponding .hmac file. The cryptographic algorithm used for the integrity test of the kernel object files (RSA SigVer) is executed before the object file integrity tests are performed.

5.2 Initiate on Demand

Integrity tests are performed as part of the pre-operational self-tests, which are executed when the module is initialized. The integrity tests can be invoked on demand by unloading and subsequently re-initializing the module (i.e., rebooting the system), which will perform (among others) the software integrity tests.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data and uncontrolled access to the data of other processes is prevented.

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1.

Instrumentation tools like the ptrace system call, gdb and strace, as well as other tracing mechanisms offered by the Linux environment such as ftrace or systemtap, shall not be used in the operational environments. The use of any of these tools implies that the cryptographic module is running in a non-validated operational environment.

7 Physical Security

The module is comprised of software only and therefore this section is not applicable.

8 Non-Invasive Security

The module does not implement any non-invasive security mechanisms.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
Module RAM	Temporary storage for SSPs used by the module as part of service execution	Dynamic

Table 15: Storage Areas

The module does not perform persistent storage of SSPs; SSPs in use by the module exist in volatile memory only. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters	Operator calling application (TOEPP)	Module RAM	Plaintext	Manual	Electronic	
AF_ALG type input sockets	Operator calling application (TOEPP)	Module RAM	Plaintext	Manual	Electronic	
API output parameters	Module RAM	Operator calling application (TOEPP)	Plaintext	Manual	Electronic	
AF_ALG type output sockets	Module RAM	Operator calling application (TOEPP)	Plaintext	Manual	Electronic	

Table 16: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Free cipher handle	Zeroizes the SSPs contained within the cipher handle	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that the zeroization procedure succeeded.	By calling the appropriate zeroization functions: AES key: crypto_free_skcipher and crypto_free_aead; HMAC key: crypto_free_shash and crypto_free_ahash; Entropy input: crypto_free_rng; DRBG seed: crypto_free_rng; DRBG internal state: crypto_free_rng; DH public key & DH private key: crypto_free_kpp

Zeroization Method	Description	Rationale	Operator Initiation
Remove power from the module	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed. Module power off indicates that the zeroization procedure succeeded.	By removing power
Automatic	Automatically zeroized by the module when no longer needed	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	N/A

Table 17: SSP Zeroization Methods

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	Symmetric key used for AES operations	128, 256 bits (AES-XTS); 128, 192, 256 bits (others) - 128, 256 bits (AES-XTS); 128, 192, 256 bits (others)	Symmetric key - CSP			Encryption Decryption Authenticated encryption Authenticated decryption Message authentication
HMAC key	Symmetric key used for HMAC operations	112-524288 bits - 112-256 bits	Authentication key - CSP			Message authentication
Shared secret	Shared secret generated by Diffie-Hellman	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	Shared secret - CSP		Shared secret computation	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		- 112-200 bits				
Entropy input	Entropy input used to seed DRBGs	128-384 bits - 128-384 bits	Entropy input - CSP			Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG
CTR_DRBG Seed (IG D.L)	DRBG seed derived from Entropy Input	256, 320, 384 bits - 128, 192, 256 bits	Seed - CSP	Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG		Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG
Hash_DRBG Seed (IG D.L)	DRBG seed derived from Entropy Input	440, 888 bits - 128, 256 bits	Seed - CSP	Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG		Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				Random Number Generation with Hash_DRBG		Random Number Generation with Hash_DRBG
HMAC_DRBG Seed (IG D.L)	DRBG seed derived from Entropy Input	440, 888 bits - 128, 256 bits	Seed - CSP	Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG		Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG
CTR_DRBG Internal State (V, Key) (IG D.L)	Internal state of Counter DRBG instance	256, 320, 384 bits - 128, 192, 256 bits	Internal state - CSP	Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG		Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG
Hash_DRBG Internal State (V, C) (IG D.L)	Internal state of Hash DRBG instance	880, 1776 bits - 128, 256 bits	Internal state - CSP	Random Number Generation with CTR_DRBG Random Number Generation		Random Number Generation with CTR_DRBG Random Number Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				with HMAC_DRBG Random Number Generation with Hash_DRBG		with HMAC_DRBG Random Number Generation with Hash_DRBG
HMAC_DRBG Internal State (V, Key) (IG D.L)	Internal state of HMAC DRBG instance	320, 512, 1024 bits - 128, 256 bits	Internal state - CSP	Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG		Random Number Generation with CTR_DRBG Random Number Generation with HMAC_DRBG Random Number Generation with Hash_DRBG
DH private key	Private key used for Diffie-Hellman	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 - 112-200 bits	Private key - CSP	Key pair generation		Shared secret computation
DH public key	Public key used for Diffie-Hellman	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	Public key - PSP	Key pair generation		Shared secret computation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		- 112-200 bits				
Intermediate key generation value	Temporary value generated during key pair generation services	2048-8192 bits - 112-200 bits	Intermediate value - CSP	Key pair generation		Key pair generation
GCM IV	IV used for Authenticated encryption; Authenticated decryption	96 bit - N/A	Initialization vector - PSP	Authenticated encryption		Authenticated encryption Authenticated decryption

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters AF_ALG type input sockets	Module RAM:Plaintext	Until cipher handle is freed or module is powered off	Free cipher handle Remove power from the module	
HMAC key	API input parameters AF_ALG type input sockets	Module RAM:Plaintext	Until cipher handle is freed or module is powered off	Free cipher handle Remove power from the module	
Shared secret	API output parameters AF_ALG type output sockets	Module RAM:Plaintext	Until cipher handle is freed or module is powered off	Free cipher handle Remove power from the module	DH private key:Derived From DH public key:Derived From
Entropy input		Module RAM:Plaintext	From generation until DRBG seed is created	Automatic	CTR_DRBG Seed (IG D.L):Derives Hash_DRBG Seed (IG D.L):Derives HMAC_DRBG Seed (IG D.L):Derives
CTR_DRBG Seed (IG D.L)		Module RAM:Plaintext	From service invocation to service completion	Automatic Remove power from the module	Entropy Input (IG D.L):Derived From CTR_DRBG Internal State (V,

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Key) (IG D.L):Derives
Hash_DRBG Seed (IG D.L)		Module RAM:Plaintext	From service invocation to service completion	Automatic Remove power from the module	Entropy Input (IG D.L):Derived From Hash_DRBG Internal State (V, C) (IG D.L):Derives
HMAC_DRBG Seed (IG D.L)		Module RAM:Plaintext	From service invocation to service completion	Automatic Remove power from the module	Entropy Input (IG D.L):Derived From HMAC_DRBG Internal State (V, Key) (IG D.L):Derives
CTR_DRBG Internal State (V, Key) (IG D.L)		Module RAM:Plaintext	From service invocation to service completion	Free cipher handle Remove power from the module	CTR_DRBG Seed (IG D.L):Derived From
Hash_DRBG Internal State (V, C) (IG D.L)		Module RAM:Plaintext	From service invocation to service completion	Free cipher handle Remove power from the module	Hash_DRBG Seed (IG D.L):Derived From
HMAC_DRBG Internal State (V, Key) (IG D.L)		Module RAM:Plaintext	From service invocation to service completion	Free cipher handle Remove power from the module	HMAC_DRBG Seed (IG D.L):Derived From
DH private key	API input parameters API output parameters AF_ALG type input sockets AF_ALG type output sockets	Module RAM:Plaintext	Until cipher handle is freed or module is powered off	Free cipher handle Remove power from the module	DH public key:Paired With Shared secret:Derives Intermediate key generation value:Derived From
DH public key	API input parameters API output parameters AF_ALG type input	Module RAM:Plaintext	Until cipher handle is freed or module is powered off	Free cipher handle Remove power from the module	DH private key:Paired With Shared secret:Derives Intermediate key generation

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	sockets AF_ALG type output sockets				value:Derived From
Intermediate key generation value		Module RAM:Plaintext	For the duration of the service	Automatic	DH private key:Derives DH public key:Derives
GCM IV	API input parameters API output parameters	Module RAM:Plaintext	From service invocation to service completion	Free cipher handle Automatic Remove power from the module	

Table 19: SSP Table 2

9.5 Transitions

The SHA-1 algorithm as implemented by the module will be non-approved for all purposes, starting January 1, 2031. SHA-1 from Message Digest is only approved for non-digital-signature uses (see Table 8 of SP 800-131Arev2).

10 Self-Tests

While the module is executing the self-tests, services are not available, and data output (via the data output interface) is inhibited until the tests are successfully completed. The module does not return control to the calling application until the tests are completed. If any of the self-tests fails, the module immediately transitions to the error state.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
RSA SigVer (FIPS186-5) (A7403)	3072-bit key with SHA2-256	Signature verification with RSA	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for kernel object files
HMAC-SHA2-256 (A7403)	128-bit key	Message authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for sha512hmac binary and libkcapi binary
HMAC-SHA2-512 (A7403)	128-bit key	Message authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for kernel binary

Table 20: Pre-Operational Self-Tests

The pre-operational software integrity tests are performed automatically when the module is powered on, before the module transitions into the operational state.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CBC	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CBC (A7407)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CBC (A7407)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CBC (A7410)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CBC (A7410)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC-CS3	128-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CBC-CS3	128-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CBC-CS3 (A7407)	128-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CBC-CS3 (A7407)	128-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CCM	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CCM	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CCM (A7410)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CCM (A7410)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CFB128	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CFB128	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CFB128 (A7410)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CFB128 (A7410)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CTR	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CTR	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-CTR (A7407)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-CTR (A7407)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-ECB	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-ECB	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-ECB (A7407)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-ECB (A7407)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-GCM	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-GCM (A7409)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-GCM (A7409)	128, 192, 256-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-OFB	128-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-OFB	128-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-OFB (A7410)	128-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-OFB (A7410)	128-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-XTS Testing Revision 2.0	256, 512-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-XTS Testing Revision 2.0	256, 512-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
AES-XTS Testing Revision 2.0 (A7407)	256, 512-bit keys	KAT	CAST	Module is operational	Encryption	Module initialization
AES-XTS Testing Revision 2.0 (A7407)	256, 512-bit keys	KAT	CAST	Module is operational	Decryption	Module initialization
SHA-1 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA-1 (A7413)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA-1 (A7414)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA-1 (A7415)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-224 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-224 (A7413)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-224 (A7414)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-224 (A7415)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA2-256 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-256 (A7413)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-256 (A7414)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-256 (A7415)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-384 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-384 (A7413)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-384 (A7414)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-384 (A7415)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-512 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-512 (A7413)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-512 (A7414)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-512 (A7415)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA3-224 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA3-256 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA3-384 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA3-512 (A7403)	0-8184-bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
AES-CMAC	128, 256-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
AES-CMAC (A7410)	128, 256-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA-1	32-64-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA-1 (A7415)	32-64-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA2-224	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-224 (A7415)	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA2-256	32-64-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA2-256 (A7415)	32-64-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA2-384	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA2-384 (A7415)	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA2-512	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA2-512 (A7415)	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA3-224 (A7403)	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA3-256 (A7403)	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA3-384 (A7403)	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
HMAC-SHA3-512 (A7403)	32-1048-bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
Counter DRBG	AES-128, AES-192, AES-256 with/without prediction resistance	KAT	CAST	Module is operational	SP 800-90Ar1 (instantiate, reseed, generate) health test	Module initialization
Hash DRBG	SHA-1, SHA2-256, SHA2-512 with/without prediction resistance	KAT	CAST	Module is operational	SP 800-90Ar1 (instantiate, reseed, generate) health test	Module initialization
HMAC DRBG	SHA-1, SHA2-256, SHA2-512 with/without prediction resistance	KAT	CAST	Module is operational	SP 800-90Ar1 (instantiate, reseed, generate) health test	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-FFC-SSC Sp800-56Ar3 (A7403)	ffdhe2048	KAT	CAST	Module is operational	Shared secret computation	Module initialization
RSA SigVer (FIPS186-5) (A7403)	PKCS#1 v1.5 with SHA-512 and 4096-bit key	KAT	CAST	Module is operational	Signature verification with RSA	Module initialization
Safe Primes Key Generation (A7403)	N/A	PCT	PCT	Key pair generation is successful	SP 800-56Ar3 Section 5.6.2.1.4	Key pair generation
Entropy source	Cutoff C = 325; Windows size = 512	APT	CAST	Entropy source is operational	Entropy source start-up test on 1024 samples	Entropy source initialization
Entropy source	Cutoff C = 31	RCT	CAST	Entropy source is operational	Entropy source start-up test on 1024 samples	Entropy source initialization
Entropy source	Cutoff C = 355; Windows size = 512	APT	CAST	jent_kcapi_random returns 0	Entropy source continuous test	Continuously as entropy is requested
Entropy source	Cutoff C = 61	RCT	CAST	jent_kcapi_random returns 0	Entropy source continuous test	Continuously as entropy is requested

Table 21: Conditional Self-Tests

Data output through the data output interface is inhibited during the conditional self-tests. The module does not return control to the calling application until the tests are completed. If any of these tests fails, the module transitions to the error state (Section 10.4).

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-5) (A7403)	Signature verification with RSA	SW/FW Integrity	On demand	Manually
HMAC-SHA2-256 (A7403)	Message authentication	SW/FW Integrity	On demand	Manually
HMAC-SHA2-512 (A7403)	Message authentication	SW/FW Integrity	On demand	Manually

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC	KAT	CAST	On demand	Manually
AES-CBC	KAT	CAST	On demand	Manually
AES-CBC (A7407)	KAT	CAST	On demand	Manually
AES-CBC (A7407)	KAT	CAST	On demand	Manually
AES-CBC (A7410)	KAT	CAST	On demand	Manually
AES-CBC (A7410)	KAT	CAST	On demand	Manually
AES-CBC-CS3	KAT	CAST	On demand	Manually
AES-CBC-CS3	KAT	CAST	On demand	Manually
AES-CBC-CS3 (A7407)	KAT	CAST	On demand	Manually
AES-CBC-CS3 (A7407)	KAT	CAST	On demand	Manually
AES-CCM	KAT	CAST	On demand	Manually
AES-CCM	KAT	CAST	On demand	Manually
AES-CCM (A7410)	KAT	CAST	On demand	Manually
AES-CCM (A7410)	KAT	CAST	On demand	Manually
AES-CFB128	KAT	CAST	On demand	Manually
AES-CFB128	KAT	CAST	On demand	Manually
AES-CFB128 (A7410)	KAT	CAST	On demand	Manually
AES-CFB128 (A7410)	KAT	CAST	On demand	Manually
AES-CTR	KAT	CAST	On demand	Manually
AES-CTR	KAT	CAST	On demand	Manually
AES-CTR (A7407)	KAT	CAST	On demand	Manually
AES-CTR (A7407)	KAT	CAST	On demand	Manually
AES-ECB	KAT	CAST	On demand	Manually
AES-ECB	KAT	CAST	On demand	Manually
AES-ECB (A7407)	KAT	CAST	On demand	Manually
AES-ECB (A7407)	KAT	CAST	On demand	Manually
AES-GCM	KAT	CAST	On demand	Manually
AES-GCM	KAT	CAST	On demand	Manually
AES-GCM (A7409)	KAT	CAST	On demand	Manually
AES-GCM (A7409)	KAT	CAST	On demand	Manually
AES-OFB	KAT	CAST	On demand	Manually
AES-OFB	KAT	CAST	On demand	Manually
AES-OFB (A7410)	KAT	CAST	On demand	Manually
AES-OFB (A7410)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-XTS Testing Revision 2.0 (A7407)	KAT	CAST	On demand	Manually
AES-XTS Testing Revision 2.0 (A7407)	KAT	CAST	On demand	Manually
SHA-1 (A7403)	KAT	CAST	On demand	Manually
SHA-1 (A7413)	KAT	CAST	On demand	Manually
SHA-1 (A7414)	KAT	CAST	On demand	Manually
SHA-1 (A7415)	KAT	CAST	On demand	Manually
SHA2-224 (A7403)	KAT	CAST	On demand	Manually
SHA2-224 (A7413)	KAT	CAST	On demand	Manually
SHA2-224 (A7414)	KAT	CAST	On demand	Manually
SHA2-224 (A7415)	KAT	CAST	On demand	Manually
SHA2-256 (A7403)	KAT	CAST	On demand	Manually
SHA2-256 (A7413)	KAT	CAST	On demand	Manually
SHA2-256 (A7414)	KAT	CAST	On demand	Manually
SHA2-256 (A7415)	KAT	CAST	On demand	Manually
SHA2-384 (A7403)	KAT	CAST	On demand	Manually
SHA2-384 (A7413)	KAT	CAST	On demand	Manually
SHA2-384 (A7414)	KAT	CAST	On demand	Manually
SHA2-384 (A7415)	KAT	CAST	On demand	Manually
SHA2-512 (A7403)	KAT	CAST	On demand	Manually
SHA2-512 (A7413)	KAT	CAST	On demand	Manually
SHA2-512 (A7414)	KAT	CAST	On demand	Manually
SHA2-512 (A7415)	KAT	CAST	On demand	Manually
SHA3-224 (A7403)	KAT	CAST	On demand	Manually
SHA3-256 (A7403)	KAT	CAST	On demand	Manually
SHA3-384 (A7403)	KAT	CAST	On demand	Manually
SHA3-512 (A7403)	KAT	CAST	On demand	Manually
AES-CMAC	KAT	CAST	On demand	Manually
AES-CMAC (A7410)	KAT	CAST	On demand	Manually
HMAC-SHA-1	KAT	CAST	On demand	Manually
HMAC-SHA-1 (A7415)	KAT	CAST	On demand	Manually
HMAC-SHA2-224	KAT	CAST	On demand	Manually
HMAC-SHA2-224 (A7415)	KAT	CAST	On demand	Manually
HMAC-SHA2-256	KAT	CAST	On demand	Manually
HMAC-SHA2-256 (A7415)	KAT	CAST	On demand	Manually
HMAC-SHA2-384	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A7415)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-512	KAT	CAST	On demand	Manually
HMAC-SHA2-512 (A7415)	KAT	CAST	On demand	Manually
HMAC-SHA3-224 (A7403)	KAT	CAST	On demand	Manually
HMAC-SHA3-256 (A7403)	KAT	CAST	On demand	Manually
HMAC-SHA3-384 (A7403)	KAT	CAST	On demand	Manually
HMAC-SHA3-512 (A7403)	KAT	CAST	On demand	Manually
Counter DRBG	KAT	CAST	On demand	Manually
Hash DRBG	KAT	CAST	On demand	Manually
HMAC DRBG	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A7403)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A7403)	KAT	CAST	On demand	Manually
Safe Primes Key Generation (A7403)	PCT	PCT	On demand	Manually
Entropy source	APT	CAST	On demand	Manually
Entropy source	RCT	CAST	On demand	Manually
Entropy source	APT	CAST	On demand	Manually
Entropy source	RCT	CAST	On demand	Manually

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error	The Linux kernel immediately stops executing	Any self-test failure	Restart of the module	Kernel panic

Table 24: Error States

In the error state, the output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running).

10.5 Operator Initiation of Self-Tests

The software integrity tests, CASTs and entropy source start-up tests can be invoked on demand by unloading and subsequently re-initializing the module. The PCTs can be invoked on demand by requesting the key pair generation service.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is distributed as a part of the Rocky Linux 9 distribution, in the form of the kernel-5.14.0-570.39.1.el9_6.ciqfips.3.1.x86_64, libkcapi-1.4.0-2.el9, and libkcapi-hmaccalc-1.4.0-2.el9 RPM packages.

The module can achieve FIPS validated configuration by:

- Adding the fips=1 option to the kernel command line during the system installation. During the software selection stage, do not install any third-party software.
- Switching the system into the validated configuration after installation. Execute the "fips-mode-setup --enable" command. Restart the system.

In both cases, the Crypto Officer must verify the system operates in the validated configuration by executing the "fips-mode-setup --check" command, which should output "FIPS mode is enabled."

11.2 Administrator Guidance

After installation of the RPM packages, the Crypto Officer must execute the "cat /proc/sys/crypto/fips_name" command. The Crypto Officer must ensure that the proper name is listed in the output as follows:

Rocky Linux 9 - Kernel Cryptographic API

Then, the Crypto Officer must execute the "cat /proc/sys/crypto/fips_version" and "rpm -qa | grep kcapi" commands. These commands must output the following (one line per output):

rocky9.20260721
libkcapi-hmaccalc-1.4.0-2.el9.x86_64
libkcapi-1.4.0-2.el9.x86_64

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 Design and Rules

Not applicable.

11.5 Maintenance Requirements

Not applicable.

11.6 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory. Then, if desired, the kernel-5.14.0-570.39.1.el9_6.ciqfips.3.1_x86_64, libkcapi-1.4.0-2.el9, and libkcapi-hmaccalc-1.4.0-2.el9 RPM packages can be uninstalled from the Rocky Linux 9 system.

12 Mitigation of Other Attacks

The module does not implement security mechanisms to mitigate other attacks.

A Glossary and Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CBC-CS3	Cipher Block Chaining with Ciphertext Stealing 3
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECDSA	Elliptic Curve Digital Signature Algorithm
ECB	Electronic Code Book
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HMAC	Keyed-Hash Message Authentication Code
IPsec	Internet Protocol Security
IG	Implementation Guidance
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PCT	Pair-wise Consistency Test
PSP	Public Security Parameter
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment's Physical Perimeter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

B References

FIPS 140-3	**Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS 140-3 IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf September 2020; Updated September 2025
FIPS 180-4	**Secure Hash Standard (SHS)** August 2015 https://doi.org/10.6028/NIST.FIPS.180-4
FIPS 186-4	**Digital Signature Standard (DSS)** https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf July 2013; Withdrawn February 2024
FIPS 186-5	**Digital Signature Standard (DSS)** February 2023 https://doi.org/10.6028/NIST.FIPS.186-5
FIPS 197	**Advanced Encryption Standard (AES)** November 2001; Updated May 2023 https://doi.org/10.6028/NIST.FIPS.197-upd1
FIPS 198-1	**The Keyed-Hash Message Authentication Code (HMAC)** July 2008 https://doi.org/10.6028/NIST.FIPS.198-1
FIPS 202	**SHA-3 Standard: Permutation-Based Hash and Extendable- Output Functions** August 2015 https://doi.org/10.6028/NIST.FIPS.202
PKCS#1	**PKCS #1: RSA Cryptography Specifications Version 2.2** November 2016 https://doi.org/10.17487/RFC8017
RFC 4106	**The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)** June 2005 https://doi.org/10.17487/RFC4106
SP 800-38A	**Recommendation for Block Cipher Modes of Operation: Methods and Techniques** December 2001 https://doi.org/10.6028/NIST.SP.800-38A
SP 800-38A-Add	**Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode** October 2010 https://doi.org/10.6028/NIST.SP.800-38A-Add
SP 800-38B	**Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication** May 2005; Updated October 2016 https://doi.org/10.6028/NIST.SP.800-38B

SP 800-38C	**Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004; Updated July 2007 https://doi.org/10.6028/NIST.SP.800-38C
SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://doi.org/10.6028/NIST.SP.800-38D
SP 800-38E	**Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices** January 2010 https://doi.org/10.6028/NIST.SP.800-38E
SP 800-56Ar3	**Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP 800-90Ar1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://doi.org/10.6028/NIST.SP.800-90Ar1
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B
SP 800-133r2	**Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP 800-140Br1	**Cryptographic Module Validation Program (CMVP) Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B** November 2023 https://doi.org/10.6028/NIST.SP.800-140Br1