Kernel Cryptography Module for AlmaLinux 9

version: kernel 5.14.0-284.1101.el9_2.tuxcare.7
libkcapi 1.3.1-3.el9

FIPS 140-3 Non-Proprietary Security Policy

document version 1.3

Last update: 2025-04-29

Prepared by:
atsec information security corporation
4516 Seton Center Parkway, Suite 250
Austin, TX 78759
www.atsec.com

Prepared for:
CloudLinux Inc., TuxCare division
2318 Louis Road, Suite B
Palo Alto, CA 94303
www.tuxcare.com

Table of Contents

1 General....5
1.1 Overview....5
1.1.1 How this Security Policy was prepared....5
1.2 Security levels....5
2 Cryptographic module specification....7
2.1 Description....7
2.2 Version Information....7
2.3 Operating Environments....8
2.4 Excluded Components....9
2.5 Modes of Operation....9
2.6 Approved algorithms....10
2.7 RNG and Entropy....12
2.8 SSP Generation....12
2.9 SSP Establishment....12
2.10 Industry Protocols....12
2.11 Design and Rules....13
2.12 Initialization....13
3 Cryptographic Module Interfaces....14
3.1 Description....14
3.2 Trusted Channel Specification....14
3.3 Control Interface Not Inhibited....14
4 Roles, Services, and Authentication....15
4.1 Authentication Methods....15
4.2 Roles....15
4.3 Approved Services....15
4.4 Non-Approved Services....17
4.5 External Software/Firmware Loaded....17
4.6 Bypass Actions and Status....17
4.7 Cryptographic Output Actions and Status....17
5 Software/Firmware Security....18
5.1 Integrity Techniques....18
5.2 Initiate on Demand....18
6 Operational Environment....19
6.1 Operational Environment Type and Requirements....19
6.2 Configurable Settings and Restrictions....19
7 Physical Security....20
8 Non-Invasive Security....21
9 Sensitive Security Parameters Management....22
9.1 Storage Areas....22
9.2 SSP Input-Output Methods....22
9.3 SSP Zeroization Methods....22
9.4 SSPs....23
9.5 Transitions....24

10 Self-Tests 25
10.1 Pre-Operational Self-Tests 25
10.2 Conditional Self-Tests 25
10.3 Periodic Self-Tests 27
10.4 Error States 27
10.5 Operator Initiation 27
11 Life-Cycle Assurance 28
11.1 Startup Procedures 28
11.2 Administrator Guidance 28
11.2.1 AES GCM IV 28
11.2.2 AES XTS 29
11.2.3 RSA 29
11.3 Non-Administrator Guidance 29
11.4 Maintenance Requirements 29
11.5 End of Life 29
12 Mitigation of Other Attacks 30
Appendix A. Glossary and abbreviations 31
Appendix B. References 32

List of Tables

Table 1 - Security Levels....6
Table 2 - Software, Firmware, Hybrid Tested Operating Environments....8
Table 3 - Executable Code Sets....9
Table 4 - Modes List and Description....9
Table 5 - Approved Algorithms....11
Table 6 – Non-Approved, Not Allowed Algorithms....12
Table 7 – Entropy....12
Table 8 - Ports and Interfaces....14
Table 9 – Roles....15
Table 10 – Approved Services....16
Table 11 - Non-Approved Services....17
Table 12 - Storage Areas....22
Table 13 - SSP Input-Output....22
Table 14 - SSP Zeroization Methods....22
Table 15 - SSP Information First....23
Table 16 - SSP Information Second....24
Table 17 - Pre-Operational Self-Tests....25
Table 18 - Conditional Self-Tests....27
Table 19 - Error States....27

List of Figures

Figure 1 – Block Diagram....7

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version kernel 5.14.0-284.1101.el9_2.tuxcare.7; libkcapi 1.3.1-3.el9 of the Kernel Cryptography Module for AlmaLinux 9 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. Other documentation is proprietary to their authors.

1.1.1How this Security Policy was prepared

The vendor has provided the non-proprietary Security Policy of the cryptographic module, which was further consolidated into this document by atsec information security together with other vendor-supplied documentation. 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

Table 1 describes the individual security areas of FIPS 140-3, as well as the security levels of those individual areas.

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 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	Not Applicable
8	Non-invasive Security	Not Applicable
9	Sensitive Security Parameter Management	1
10	Self-tests	1

11	Life-cycle Assurance	1
12	Mitigation of Other Attacks	Not Applicable
Overall		1

Table 1 - Security Levels

2 Cryptographic module specification

2.1 Description

Purpose and Use: The Kernel Cryptography Module for AlmaLinux 9 (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

Module Characteristics: N/A

Cryptographic Boundary: The cryptographic boundary of the module is defined as the kernel binary and the kernel crypto object files, the libkcapi library, and the sha512hmac binary, which is used to verify the integrity of the software components. 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.

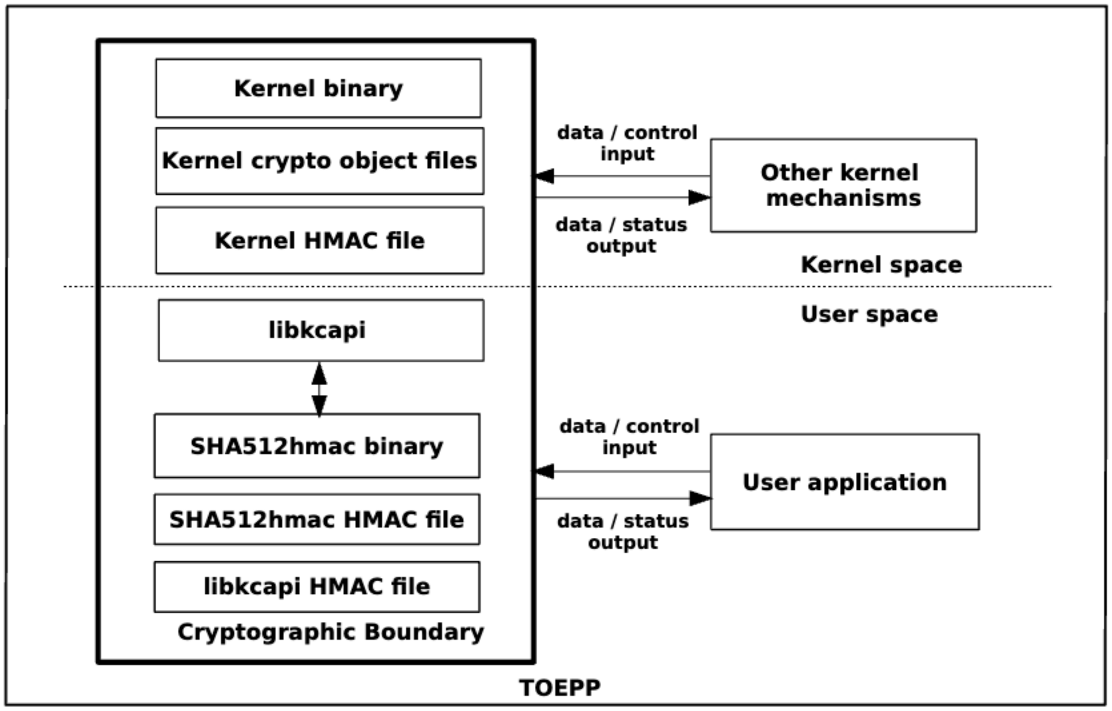

Figure 1 – Block Diagram

2.2 Version Information

Hardware Versions: N/A

Software Versions: kernel 5.14.0-284.1101.el9_2.tuxcare.7; libkcapi 1.3.1-3.el9

Firmware Versions: N/A

2.3 Operating Environments

Hardware Operating Environments: N/A

Software, Firmware, Hybrid Tested Operating Environments:

Operating System	Hardware Platform	Processor(s)	PAA/PAI	Hypervisor and Host OS
AlmaLinux 9.2	Amazon Web Services (AWS) m5.metal	Intel Xeon Platinum 8259CL	AES-NI (PAA)	N/A
AlmaLinux 9.2	Amazon Web Services (AWS) m5.metal	Intel Xeon Platinum 8259CL	None	N/A
AlmaLinux 9.2	Amazon Web Services (AWS) a1.metal	AWS Graviton	Neon / CE, SHA Extensions (PAA)	N/A
AlmaLinux 9.2	Amazon Web Services (AWS) a1.metal	AWS Graviton	None	N/A

Table 2 - Software, Firmware, Hybrid Tested Operating Environments

Executable Code Sets:

Package or File Names	Software/ Firmware Version	Features	Hybrid Hardware Version	Integrity Test
/boot/vmlinuz-5.14.0-284.1101.el9_2.tuxcare.7.x86_64 (for Intel platform) /boot/vmlinuz-5.14.0-284.1101.el9_2.tuxcare.7.aarch64 (for ARM platform)	5.14.0-284.1101.el9_2.tuxcare.7	N/A	N/A	HMAC-SHA2-512
/usr/lib/modules/5.14.0-284.1101.el9_2.tuxcare.7.x86_64/kernel/crypto/*.ko (for Intel platform) /usr/lib/modules/5.14.0-284.1101.el9_2.tuxcare.7.x86_64/kernel/arch/x86/crypto/*.ko (for Intel platform) /usr/lib/modules/5.14.0-284.1101.el9_2.tuxcare.7.aarch64/kernel/crypto/*.ko				RSA signature verification

Package or File Names	Software/ Firmware Version	Features	Hybrid Hardware Version	Integrity Test
(for ARM platform) /usr/lib/modules/5.14.0-284.1101.el9_2.tuxcare.7.aarch64/kernel/arch/arm64/crypto/*.ko (for ARM platform)				
/usr/lib64/libkcapi.so.1.3.1 (for Intel platform) /usr/bin/sha512hmac	1.3.1-3.el9	N/A	N/A	HMAC-SHA2-512
/usr/lib64/libkcapi.so.1.3.1 (for ARM platform) /usr/bin/sha512hmac	1.3.1-3.el9	N/A	N/A	HMAC-SHA2-512

Table 3 - Executable Code Sets

2.4 Excluded Components

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

2.5 Modes of Operation

Modes List and Description:

Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested.	Approved	Equivalent to the indicator of the requested service
Non-approved mode	Automatically entered whenever a non-approved service is requested.	Non-approved	Equivalent to the indicator of the requested service

Table 4 - Modes List and Description

After passing all pre-operational self-tests and cryptographic algorithm self-tests executed on start-up, the module automatically transitions to the approved mode.

Mode change instructions and status indicators:

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.

Degraded Mode Description:

The module does not implement a degraded mode of operation.

2.6 Approved algorithms

Approved Algorithms:

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths[1]	Use / Function
A4025 A4032 A4036 A4037 A4047 A4048 A4049	SHA [FIPS 180-4]	SHA-224, SHA-256, SHA-384, SHA-512	N/A	Message digest
A4026	SHA-3 [FIPS 202]	SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Message digest
A4025 A4027 A4028 A4029 A4030 A4031 A4032 A4033 A4034 A4035 A4036 A4038 A4039 A4040 A4041 A4042 A4043 A4044 A4045 A4046	AES [FIPS 197, SP 800-38A, SP 800-38A Addendum]	ECB, CBC, CBC-CTS-CS3, OFB, CFB128, CTR	128, 192, 256 bits	Encryption Decryption
	AES [FIPS 197, SP 800-38C]	CCM	128, 192, 256 bits	Authenticated encryption Authenticated decryption
A4025 A4030 A4031 A4033 A4034 A4035 A4038 A4039 A4040 A4041 A4042 A4043	AES [FIPS 197, SP 800-38D]	GCM (internal IV)	128, 192, 256 bits	Authenticated encryption
	AES [FIPS 197, SP 800-38D]	GCM (external IV)	128, 192, 256 bits	Authenticated decryption
A4025 A4032 A4033 A4036 A4038 A4041	AES [FIPS 197, SP 800-38E]	XTS	128, 256 bits	Encryption Decryption

[1] Key strengths are identical to key sizes unless indicated otherwise.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths[1]	Use / Function
A4025 A4032 A4033 A4041	AES [FIPS 197, SP 800-38B, SP 800-38D]	CMAC, GMAC	128, 192, 256 bits	Message authentication
A4025 A4032 A4036 A4037 A4047 A4048 A4049	HMAC [FIPS 198-1]	SHA-224, SHA-256, SHA-384, SHA-512	112-524288 bits (112-256 bits)	Message authentication
A4026		SHA3-224, SHA3-256, SHA3-384, SHA3-512		
A4025 A4030 A4031 A4033 A4034 A4035 A4038 A4039 A4040 A4041 A4042 A4043	CTR_DRBG [SP 800-90Ar1]	AES-128, AES-192, AES-256, with derivation function, with/without prediction resistance	128, 192, 256 bits	Random number generation
A4025 A4030 A4031 A4034 A4035 A4038 A4039 A4040 A4041 A4042 A4043 A4047 A4048 A4049	Hash_DRBG [SP 800-90Ar1]	SHA-1, SHA-256, SHA-512 with/without prediction resistance	112, 256 bits	Random number generation
	HMAC_DRBG [SP 800-90Ar1]	SHA-1, SHA-256, SHA-512 with/without prediction resistance	112, 256 bits	Random number generation
A4025 A4047 A4048 A4049	RSA [FIPS 186-4]	PKCS#1 v1.5 with SHA-256	4096 bits (150 bits)	Internal function: Integrity verification

Table 5 - Approved Algorithms

Vendor Affirmed Algorithms:

The module does not implement vendor affirmed algorithms.

Non-Approved, Allowed Algorithms:

The module does not implement non-approved algorithms allowed in the approved mode of operation.

Non-Approved, Allowed Algorithms with No Security Claimed:

The module does not implement non-approved algorithms allowed in the approved mode of operation with no security claimed.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
AES GCM with external IV	Encryption
KBKDF (libkcapi)	Key derivation
HKDF (libkcapi)	Key derivation
PBKDF2 (libkcapi)	Password-based key derivation
RSA	Encryption primitive Decryption primitive
RSA with PKCS#1 v1.5 padding (pre-hashed message)	Signature generation primitive Signature verification primitive

Table 6 – Non-Approved, Not Allowed Algorithms

2.7 RNG and Entropy

Entropy Information:

Name	Type	Operational Environment	Sample Size	Entropy Per Sample	Conditioning Component
AlmaLinux Kernel CPU Time Jitter RNG Entropy Source (ESV cert. #E75)	Non-physical	See Table 2	256 bits	256 bits	SHA3-256

Table 7 – Entropy

RNG Information:

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, using the parameters listed in Table 5. When instantiated, these DRBGs can be used to generate random numbers for external usage.

2.8 SSP Generation

The module does not implement any SSP generation methods.

2.9 SSP Establishment

The module does not implement any SSP establishment methods.

2.10 Industry Protocols

AES GCM with internal IV generation in approved mode is compliant with RFC 4106 and shall only be used in conjunction with the IPsec protocol. No parts of this protocol, other than the AES GCM implementation, have been tested by the CAVP and CMVP.

2.11 Design and Rules

The module operates in the approved mode of operation by default and can only transition into the non-approved mode by calling one of the non-approved services listed in Table 11 of the Security Policy.

In the operational state, the module accepts service requests from calling applications through its logical interfaces. If the Linux kernel is shut down, the module will end its operation.

2.12 Initialization

There are no specific initialization requirements.

3 Cryptographic Module Interfaces

3.1 Description

Physical Port	Logical Interface	Data that passes over the port/interface
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Input	API data input parameters, AF_ALG type sockets.
	Data Output	API output parameters, AF_ALG type sockets.
	Control Input	API function calls, API control input parameters, AF_ALG type sockets, kernel command line.
	Status Output	API return values, AF_ALG type sockets, kernel logs.

Table 8 - Ports and Interfaces

The logical interfaces are the APIs through which the applications request services and AF_ALG type socket that allows the applications running in the user space to request cryptographic services from the module. These logical interfaces are logically separated from each other by the API design.

3.2 Trusted Channel Specification

The module does not implement a trusted channel.

3.3 Control Interface Not Inhibited

The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	N/A

Table 9 – Roles

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed by the operator of the module. No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Message digest	Compute a message digest	crypto_shash_init returns 0	Message	Digest value	SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A
Encryption	Encrypt a plaintext	crypto_skcipher_setkey return 0	AES key, plaintext	Ciphertext	AES ECB, CBC, CBC-CTS-CS3, OFB, CFB128, CTR, XTS	AES key: W, E
Decryption	Decrypt a ciphertext		AES key, ciphertext	Plaintext		
Authenticated encryption	Encrypt a plaintext	For all except AES_GCM: crypto_aead_setkey returns 0 For AES-GCM: the TFM handle has the CRYPTO_TFM_FIPS_COMPLIANCE flag set	AES key, plaintext	Ciphertext, MAC tag	AES CCM, GCM (internal IV) AES CBC or CTR with HMAC-SHA2-256, SHA2-384, or SHA2-512	AES key: W, E HMAC key: W, E
Authenticated decryption	Decrypt a ciphertext		AES key, ciphertext, MAC tag	Plaintext	AES CCM, GCM (external IV) AES CBC or CTR with HMAC-SHA2-256, SHA2-384, or SHA2-512	
Message authentication	Compute a MAC tag	crypto_shash_init returns 0	AES key, message	MAC tag	AES CMAC, GMAC	AES key: W, E
			HMAC key, message		HMAC-SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384,	HMAC key: W, E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					SHA3-512	
Random number generation	Generate random bytes	crypto_rng_get_bytes returns 0	Output length	Random bytes	CTR_DRBG Hash_DRBG HMAC_DRBG	Entropy input: W, E DRBG seed: E, G DRBG Internal state (V, Key), DRBG Internal state (V, C): W, E, G
Error detection code	Compute an EDC (crc32, crct10dif)	None	Message	EDC	N/A	N/A
Compression	Compress data (deflate, lzo, zlib-deflate)	None	Data	Compressed data	N/A	N/A
Generic system call	Use the kernel to perform various non-cryptographic operations	None	Identifier, various arguments	Various return values	N/A	N/A
Show version	Return the module name and version information	None	N/A	Module name and version	N/A	N/A
Show status	Return the module status	None	N/A	Module status	N/A	N/A
Self-test	Perform the CASTs and integrity tests	None	N/A	Pass/fail	SHA SHA-3 AES HMAC CTR_DRBG Hash_DRBG HMAC_DRBG RSA See Table 18 for specifics	N/A
Zeroization	Zeroize all SSPs	None	Any SSP	N/A	N/A	All SSPs: Z

Table 10 – Approved Services

Table 10 lists the approved services. The following convention is used to specify access rights to SSPs:

- **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	Security Functions	Role
AES GCM external IV encryption	Encrypt a plaintext using AES GCM with an external IV	AES GCM with external IV	CO
Key derivation	Derive a key from a key-derivation key or a shared secret	KBKDF (libkcapi) HKDF (libkcapi)	CO
Password-based key derivation	Derive a key from a password	PBKDF2 (libkcapi)	CO
RSA encryption primitive	Compute the raw RSA encryption of a number	RSA	CO
RSA decryption primitive	Compute the raw RSA decryption of a number		CO
RSA signature generation primitive (pre-hashed message)	Generate a digital signature for a pre-hashed message	RSA with PKCS#1 v1.5 padding (pre-hashed message)	CO
RSA signature verification primitive (pre-hashed message)	Verify a digital signature for a pre-hashed message		CO

Table 11 - Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

4.6 Bypass Actions and Status

The module does not implement a bypass capability.

4.7 Cryptographic Output Actions and Status

The module does not implement a self-initiated cryptographic output capability.

5 Software/Firmware Security

5.1 Integrity Techniques

The Linux kernel binary, libkcapi, and sha512hmac software components are integrity tested using an HMAC-SHA2-512 calculation performed by the sha512hmac utility (which utilizes the module's HMAC and SHA-512 implementations). The kernel crypto object files listed in Table 3 are loaded on start-up by the module and verified using RSA signature verification with PKCS#1 v1.5 padding, SHA-256, and a 4096-bit key.

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, which will perform (among others) the software integrity tests.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operating Environment: modifiable: the module executes as part of a general-purpose operating system (AlmaLinux 9.2), which allows modification, loading, and execution of software that is not part of the validated module.

How Requirements are Satisfied: the approved cryptographic algorithms of the module are part of the Linux kernel, which operates in Linux kernel space. This ensures that any SSPs contained within the module are protected by the process isolation and memory separation mechanisms provided by the Linux kernel, and only the module has control over these SSPs. The user space libkcapi and sha512hmac components, though not processing any SSPs, are similarly protected by the operating environment.

6.2 Configurable 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 environment. 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

This module does not implement any non-invasive security mechanism and therefore this section is not applicable.

9 Sensitive Security Parameters Management

9.1 Storage Areas

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

Table 12 - Storage Areas

The module does not perform persistent storage of SSPs. The SSPs are temporarily stored in the RAM in plaintext form. 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	Related SFI
API input parameters	Operator calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	N/A
AF_ALG type sockets (input)						

Table 13 - SSP Input-Output

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	By calling the appropriate API functions
			AES key: crypto_free_skcipher and crypto_free_aead HMAC key: crypto_free_shash and crypto_free_ahash Entropy input, DRBG seed, DRBG Internal state (V, Key), DRBG Internal state (V, C): crypto_free_rng
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	By removing power

Table 14 - 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	AES key used for encryption, decryption, and computing MAC tags	XTS: 128, 256 bits Other modes: 128, 192, 256 bits	Symmetric Key	N/A	N/A	Encryption Decryption Authenticated encryption Authenticated decryption Message authentication
HMAC key	HMAC key	112-524288 bits (112-256 bits)	Authentication key	N/A	N/A	Message authentication
Entropy input	Entropy input used to seed the DRBGs. IG D.L compliant	128-384 bits	Entropy input	Non-Physical Entropy Source See Table 7	N/A	Random number generation
DRBG seed	DRBG seed derived from entropy input. IG D.L compliant	CTR_DRBG: 128, 192, 256 bits Hash_DRBG: 128, 256 bits HMAC_DRBG: 128, 256 bits	Seed	CTR_DRBG Hash_DRBG HMAC_DRBG	N/A	Random number generation
DRBG Internal state (V, Key)	Internal state of CTR_DRBG and HMAC_DRBG instances. IG D.L compliant		Internal state	CTR_DRBG HMAC_DRBG	N/A	Random number generation
DRBG Internal state (V, C)	Internal state of Hash_DRBG instances. IG D.L compliant		Internal state	Hash_DRBG	N/A	Random number generation

Table 15 - SSP Information First

Name	Input - Output	Storage	Storage Duration	Type	Related SSPs
AES key	API input parameters AF_ALG type sockets (input)	RAM	Until cipher handle is freed	CSP	None
HMAC key				CSP	None
Entropy input	N/A		From generation until the DRBG seed is created	CSP	DRBG seed
DRBG seed	N/A		While the DRBG is being instantiated	CSP	Entropy input DRBG Internal state (V, Key) DRBG Internal state (V, C)

Name	Input - Output	Storage	Storage Duration	Type	Related SSPs
DRBG Internal state (V, Key)	N/A		From DRBG instantiation until DRBG termination	CSP	DRBG seed
DRBG Internal state (V, C)	N/A			CSP	DRBG seed

Table 16 - SSP Information Second

9.5 Transitions

The RSA algorithm as implemented by the module conforms to FIPS 186-4, which has been superseded by FIPS 186-5. FIPS 186-4 will be withdrawn on February 3, 2024.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-512	C	128-bit key	Message Authentication	Software integrity	Module becomes operational	Used for kernel binary, libkcapi, and sha512hmac binary
RSA PKCS#1 v1.5	C	4096-bit key with SHA-256	Signature Verification			Used for kernel crypto object files

Table 17 - 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. 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 transitions to the operational state only after the pre-operational self-tests are passed successfully.

10.2 Conditional Self-Tests

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA-224	C, CE, Neon, AVX, AVX2, SSSE3	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA-256							
SHA-384	C, AVX, AVX2, SSSE3						
SHA-512							
SHA3-224	C						
SHA3-256							
SHA3-384							
SHA3-512							
AES ECB	C, CE, AES-NI	128, 192, 256 bit keys				Encryption Decryption (separately)	
AES CBC							
AES CBC-CTS-CS3	C, CE, Neon, AES-NI	128 bit keys					
AES OFB	C, CE, AES-NI	128 bit keys					
AES CFB128		128, 192, 256 bit keys					
AES CTR							
AES CCM	C, CE	128, 192, 256 bit keys					

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
		128-bit IVs					
AES GCM (internal IV)	C, CE, AES-NI	128, 192, 256 bit keys 96-bit IVs				Encryption	
AES GCM (external IV)		128, 192, 256 bit keys				Decryption	
AES XTS		128 and 256 bit keys				Encryption Decryption (separately)	
AES CMAC	C, CE, Neon, AES-NI	128 and 256 bit keys				Message authentication	
HMAC-SHA2-224	C, CE, AVX2	32, 160, 1048 bit keys					
HMAC-SHA2-256		32, 256, 296, 640 bit keys					
HMAC-SHA2-384	C, AVX2	32, 160, 1048 bit keys					
HMAC-SHA2-512		32, 160, 1048 bit keys					
HMAC SHA3-224	C	32, 160, 1048 bit keys					
HMAC SHA3-256		32, 160, 1048 bit keys					
HMAC SHA3-384		32, 160, 1048 bit keys					
HMAC SHA3-512		32, 160, 1048 bit keys					
CTR_DRBG		128, 192, 256 bit keys With/without PR	KAT, Health tests according to section 11.3 of [SP800-90Arev1]			Seed Generate	
Hash_DRBG		SHA-1, SHA-256, SHA-512 With/without PR	KAT, Health tests according to section 11.3 of [SP800-90Arev1]				
HMAC_DRBG		SHA-1, SHA-256, SHA-512 With/without	KAT, Health tests				

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
		PR	according to section 11.3 of [SP800-90Arev1]				
RSA PKCS#1 v1.5		4096-bit key with SHA-256	KAT			Verify	
Non-Physical Entropy Source		1024 time deltas	RCT			Entropy source start-up test	Entropy source initialization
		1024 time deltas	APT				
		Continuously	RCT		Entropy source is operational	Entropy source continuous test	Continuously
		Continuously	APT				

Table 18 - Conditional Self-Tests

When all of the pre-operational self-tests pass successfully, the module automatically performs all cryptographic algorithm self-tests (CASTs) as specified in Table 18. Only if these CASTs also passed successfully, the module transitions to the operational state. No operator intervention is required to reach this point. Services are not available, and data output (via the data output interface) is inhibited during the self-tests. If any of these tests fails, the module transitions to the error state.

10.3 Periodic Self-Tests

The module does not implement any periodic self-tests.

10.4 Error States

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

Table 19 - 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

The software integrity tests, cryptographic algorithm self-tests, and entropy source start-up tests can be invoked on demand by unloading and subsequently re-initializing the module.

11 Life-Cycle Assurance

11.1 Startup Procedures

The module is distributed as a part of the AlmaLinux 9.2 package in the form of the kernel-5.14.0-284.1101.el9_2.tuxcare.7, libkcapi-1.3.1-3.el9, and libkcapi-hmaccalc-1.3.1-3.el9 RPM packages.

Before the packages are installed, the AlmaLinux 9.2 system must operate in approved mode. This can be achieved by:

- Starting the installation in approved mode. Add 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 approved mode after the installation. Execute the `fips-mode-setup --enable` command. Restart the system.

In both cases, the Crypto Officer must verify the AlmaLinux 9.2 system operates in approved mode by executing the `fips-mode-setup --check` command, which should output “FIPS mode is enabled.”

After installation of the kernel-5.14.0-284.1101.el9_2.tuxcare.7, libkcapi-1.3.1-3.el9, and libkcapi-hmaccalc-1.3.1-3.el9 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:

```
Kernel Cryptography module for AlmaLinux 9
```

Then, the Crypto Officer must execute the `cat /proc/sys/crypto/fips_version` and `rpm -q libkcapi` commands. These commands must output the following (one line per output) depending on the platform in which are executed:

Intel:

```
5.14.0-284.1101.el9_2.tuxcare.7.x86_64
```

```
libkcapi-1.3.1-3.el9.x86_64
```

ARM:

```
5.14.0-284.1101.el9_2.tuxcare.7.aarch64
```

```
libkcapi-1.3.1-3.el9.aarch64
```

11.2 Administrator Guidance

The cryptographic boundary consists only of those APIs provided by the Kernel crypto API. If any other API in the Linux kernel is invoked, the user is not interacting with the module specified in this Security Policy.

11.2.1 AES GCM IV

The Crypto Officer shall consider the following requirements and restrictions when using the module.

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 Table 10.

11.2.2 AES XTS

In compliance with FIPS 140-3 IG C.I, the module implements the check to ensure that the two AES keys used in AES XTS algorithm are not identical.

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.

11.2.3 RSA

The module provides RSA signature verification as an internal function compliant with IG C.F. The module supports RSA modulus lengths of 4096 bits for signature verification. The RSA signature verification implementation has been tested for all implemented RSA modulus lengths.

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 Maintenance Requirements

There are no maintenance requirements.

11.5 End of Life

Secure disposal is the customer's responsibility, since the module goes EOL with the operating system.

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-5.14.0-284.1101.el9_2.tuxcare.7, libkcapi-1.3.1-3.el9, and libkcapi-hmaccalc-1.3.1-3.el9 RPM packages (for both Intel and ARM platforms) can be uninstalled from the AlmaLinux 9.2 system.

12 Mitigation of Other Attacks

The module does not offer mitigation of other attacks and therefore this section is not applicable.

Appendix A. Glossary and abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
CTS	Ciphertext Stealing
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ESV	Entropy Source Validation
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
IPsec	Internet Protocol Security
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PCT	Pair-wise Consistency Test
PBKDF2	Password-based Key Derivation Function v2
PKCS	Public-Key Cryptography Standards
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SSP	Sensitive Security Parameter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix B. References

FIPS 140-3
FIPS PUB 140-3 - Security Requirements For Cryptographic Modules
March 2019
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf

FIPS 140-3 IG
Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program
https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements

FIPS 180-4
Secure Hash Standard (SHS)
March 2012
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS 186-4
Digital Signature Standard (DSS)
July 2013
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS 197
Advanced Encryption Standard
November 2001
https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS 198-1
The Keyed Hash Message Authentication Code (HMAC)
July 2008
https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

FIPS 202
SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions
August 2015
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

PKCS#1
Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1
February 2003
https://www.ietf.org/rfc/rfc3447.txt

SP 800-38A
Recommendation for Block Cipher Modes of Operation Methods and Techniques
December 2001
https://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

SP 800-38A Addendum
Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode
October 2010
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.pdf

SP 800-38B
Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
May 2005
https://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

SP 800-38C
Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality
May 2004
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

SP 800-38D
Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
November 2007
https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

SP 800-38E
Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices
January 2010
https://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

SP 800-90Ar1
Recommendation for Random Number Generation Using Deterministic Random Bit Generators
June 2015
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

RFC 4106
The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
June 2005
https://datatracker.ietf.org/doc/html/rfc4106