Amazon Linux 2023 Kernel Cryptographic API

Versions:
Kernel 6.1.41-64.118.amzn2023, 6.1.41-64.118.fips.amzn2023
Libkcapi 1.4.0-105.amzn2023

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 1.2

Document Date: 2024-09-20

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

Table of Contents

1 General 6
1.1 Overview 6
1.2 Security Levels 6
1.2.1 Additional Information 6
2 Cryptographic Module Specification 8
2.1 Description 8
2.2 Tested and Vendor Affirmed Version and Identification 8
2.3 Excluded Components 9
2.4 Modes of Operation 9
2.5 Algorithms 10
2.6 Security Function Implementations 12
2.7 Algorithm Specific Information 13
2.7.1 AES GCM IV 13
2.7.2 AES XTS 13
2.7.3 RSA 13
2.7.4 SP 800-56Ar3 Assurances 13
2.7.5 Legacy Use 13
2.8 RBG and Entropy 14
2.9 Key Generation 14
2.10 Key Establishment 14
2.11 Industry Protocols 15
3 Cryptographic Module Interfaces 16
3.1 Ports and Interfaces 16
4 Roles, Services, and Authentication 17
4.1 Authentication Methods 17
4.2 Roles 17
4.3 Approved Services 17
4.4 Non-Approved Services 18
4.5 External Software/Firmware Loaded 19
5 Software/Firmware Security 20
5.1 Integrity Techniques 20
5.2 Initiate On-Demand Integrity Test 20
6 Operational Environment 21
6.1 Operational Environment Type and Requirements 21
6.2 Configurable Settings and Restrictions 21
7 Physical Security 22
8 Non-Invasive Security 23
9 Sensitive Security Parameters Management 24
9.1 Storage Areas 24
9.2 SSP Input-Output Methods 24
9.3 SSP Zeroization Methods 24
9.4 SSPs 24
9.5 Transitions 26

10 Self-Tests 27
10.1 Pre-Operational Self-Tests 27
10.2 Conditional Self-Tests 27
10.3 Periodic Self-Tests 28
10.4 Error States 28
10.5 Operator Initiation of Self-Tests 29
11 Life-Cycle Assurance 30
11.1 Installation, Initialization, and Startup Procedures 30
11.2 Administrator Guidance 30
11.3 Non-Administrator Guidance 30
11.4 End of Life Procedures 30
12 Mitigation of Other Attacks 31
Appendix A. Glossary and Abbreviations 32
Appendix B. References 34

List of Tables

Table 1 - Security Levels 6
Table 2 - Tested Module Identification 9
Table 3 - Tested Operational Environments 9
Table 4 - Software, Hardware, Hybrid Vendor Affirmed Operational Environment 9
Table 5 - Modes List and Description 10
Table 6 - Approved Algorithms 11
Table 7 - Vendor Affirmed Algorithms 12
Table 8 - Non-Approved, Not Allowed Algorithms 12
Table 9 - Security Function Implementations 13
Table 10 - Entropy Certificates 14
Table 11 - Entropy 14
Table 12 - Ports and Interfaces 16
Table 13 - Roles 17
Table 14 - Approved Services 18
Table 15 - Non-Approved Services 19
Table 16 - Storage Areas 24
Table 17 - SSP Input-Output Methods 24
Table 18 - SSP Zeroization Methods 24
Table 19 - SSP Information First 25
Table 20 - SSP Information Second 26
Table 21 - Pre-Operational Self-Tests 27
Table 22 - Conditional Self-Tests 28
Table 23 - Error States 29

List of Figures

Figure 1 – Block Diagram 8

Copyrights and Trademarks

Amazon is a registered trademark of Amazon Web Services, Inc. or its affiliates.

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for Amazon Linux 2023 Kernel Cryptographic API versions:

- **Kernel:** 6.1.41-64.118.amzn2023 (Amazon Linux 2023) and 6.1.41-64.118.fips.amzn2023 (SnowOS 1.0)
- **Libkcapi:** 1.4.0-105.amzn2023

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.

1.2 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 Subsections	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

1.2.1 Additional Information

This Security Policy describes the features and design of the module named Amazon Linux 2023 Kernel Cryptographic API using the terminology contained in the FIPS 140-3 specification. The FIPS 140-3 Security Requirements for Cryptographic Module specifies the security requirements that will be satisfied by a cryptographic module utilized within a security system protecting sensitive but unclassified information. The NIST/CCCS Cryptographic Module Validation Program (CMVP) validates cryptographic module to FIPS 140-3. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

This FIPS 140-3 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.

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.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use: The Amazon Linux 2023 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

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.

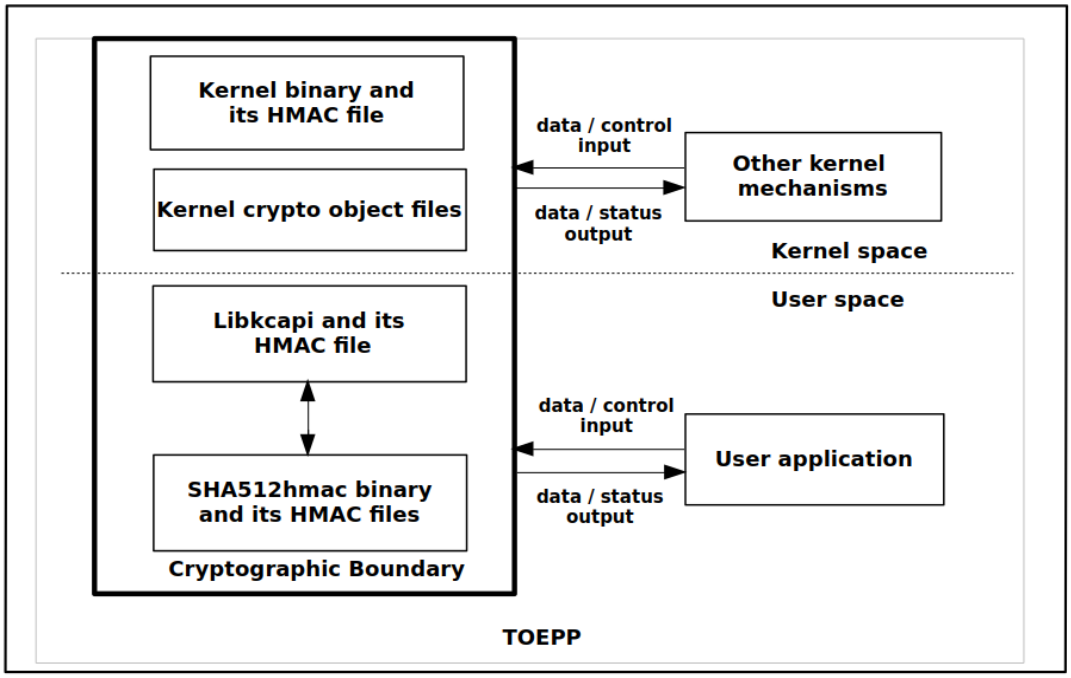

Figure 1 – Block Diagram

2.2 Tested and Vendor Affirmed Version and Identification

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

Package/File Names	Software/ Firmware Version	Integrity Test Implemented
EC2 c7g.metal: /boot/vmlinuz-6.1.41-64.118.amzn2023.aarch64	**EC2 c7g.metal, EC2 c6i.metal:** 6.1.41-64.118.amzn2023 **AWS Snowball, AWS Snowblade, AWS Snowcone:** 6.1.41-64.118.fips.amzn2023	HMAC-SHA-512
EC2 c6i.metal: /boot/vmlinuz-6.1.41-64.118.amzn2023.x86_64		
AWS Snowball, AWS Snowblade, AWS Snowcone: /boot/vmlinuz-6.1.41-64.118.fips.amzn2023.x86_64		

EC2 c7g.metal: *.ko and *.ko.xz files in /usr/lib/modules/6.1.41-64.118.amzn2023.aarch64/kernel/crypto/ *.ko.xz files in /usr/lib/modules/6.1.41-64.118.amzn2023.aarch64/kernel/arch/aarch64/crypto		RSA Signature Verification
EC2 c6i.metal: *.ko and *.ko.xz files in /usr/lib/modules/6.1.41-64.118.amzn2023.x86_64/kernel/crypto/ *.ko.xz files in /usr/lib/modules/6.1.41-64.118.amzn2023.x86_64/kernel/arch/x86/crypto/		
AWS Snowball, AWS Snowblade, AWS Snowcone: *.ko and *.ko.xz files in /usr/lib/modules/6.1.41-64.118.fips.amzn2023.x86_64/kernel/crypto/ *.ko.xz files in /usr/lib/modules/6.1.41-64.118.fips.amzn2023.x86_64/kernel/arch/x86/crypto		
/usr/lib64/libkcapi.so.1.4.0 /usr/lib/sha512hmac	1.4.0-105.amzn2023	HMAC-SHA-512

Table 2 - Tested Module Identification

Tested Operational Environments - Software, Firmware, Hybrid: The module has been tested on the following platforms with the corresponding module variants and configuration options with and without PAA:

Operating System	Hardware Platform	Processor(s)	PAA/PAI	Version(s)
Amazon Linux 2023	EC2 c7g.metal	AWS Graviton3	Neon, Cryptography Extensions (PAA)	6.1.41-64.118.amzn2023 and 1.4.0-105.amzn2023
Amazon Linux 2023	EC2 c6i.metal	Intel Xeon Platinum 8375C	AES-NI (PAA)	
SnowOS 1.0	AWS Snowball	AMD EPYC 7702		6.1.41-64.118.fips.amzn2023 and 1.4.0-105.amzn2023
SnowOS 1.0	AWS Snowblade	Intel Xeon Gold 6314U		
SnowOS 1.0	AWS Snowcone	Intel Atom C3558		
Amazon Linux 2023	EC2 c7g.metal	AWS Graviton3	None	6.1.41-64.118.amzn2023 and 1.4.0-105.amzn2023
Amazon Linux 2023	EC2 c6i.metal	Intel Xeon Platinum 8375C		
SnowOS 1.0	AWS Snowball	AMD EPYC 7702		6.1.41-64.118.fips.amzn2023 and 1.4.0-105.amzn2023
SnowOS 1.0	AWS Snowblade	Intel Xeon Gold 6314U		
SnowOS 1.0	AWS Snowcone	Intel Atom C3558		

Table 3 - Tested Operational Environments

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid: The vendor affirms the following platforms with the corresponding module variants and configuration options with and without PAA:

Operating System	Hardware Platform
Bottlerocket v1.20.0	EC2 c7g.metal with Intel Xeon Platinum 8375C (PAA: AES-NI)
Bottlerocket v1.20.0	EC2 c6i.metal with AWS Graviton3 processor (PAA: Neon, Cryptography Extensions)

Table 4 - Software, Hardware, Hybrid Vendor Affirmed Operational Environment

2.3 Excluded Components

There are no components excluded from the requirements of the FIPS 140-3 standard.

2.4 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 5 - 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. No operator intervention is required to reach this point.

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.

2.5 Algorithms

Approved Algorithms: Table 6 lists all approved cryptographic algorithms of the module, including specific key lengths employed for approved services (Table 14), and implemented modes or methods of operation of the algorithms.

Algorithm Name	CAVP Cert	Algorithm Capabilities	OE (Implementation)	Reference
AES-CBC	A4551, A4554, A4557, A4558, A4561, A4563, A4566	Key size: 128, 192, 256 bits	Amazon Linux 2023 on EC2 c7g.metal Amazon Linux 2023 on EC2 c6i.metal SnowOS 1.0 on AWS Snowball SnowOS 1.0 on AWS Snowblade SnowOS 1.0 on AWS Snowcone	FIPS 197 SP 800-38A
AES-CBC-CS3	A4551, A4554, A4557, A4558, A4561, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38A Addendum
AES-CCM	A4551, A4554, A4557, A4558, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38C
AES-CFB128	A4551, A4554, A4558, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38A
AES-CMAC	A4551, A4554, A4557, A4558, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38B
AES-CTR	A4551, A4554, A4557, A4558, A4561, A4563, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38A
AES-ECB	A4551, A4552, A4553, A4554, A4555, A4556, A4557, A4558, A4559, A4560, A4561, A4563, A4564, A4565, A4566, A4567, A4568	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38A
AES-GCM	A4551, A4552, A4553, A4554, A4555, A4556, A4558, A4559, A4560, A4563, A4564, A4565, A4566, A4567, A4568	Key size: 128, 192, 256 bits IV Generation: Internal (encryption) & External (decryption) IV Generation Mode: 8.2.2		FIPS 197 SP 800-38D
AES-GMAC	A4551, A4554, A4558, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38D
AES-KW	A4551, A4554, A4558, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38F

Algorithm Name	CAVP Cert	Algorithm Capabilities	OE (Implementation)	Reference
AES-OFB	A4551, A4554, A4558, A4566	Key size: 128, 192, 256 bits		FIPS 197 SP 800-38A
AES-XTS	A4551, A4554, A4557, A4558, A4561, A4563, A4566	Key size: 128, 256 bits		FIPS 197 SP 800-38E
CTR_DRBG	A4551, A4552, A4553, A4554, A4555, A4556, A4558, A4559, A4560, A4563, A4564, A4565, A4566, A4567, A4568	Size: AES-128, AES-192, AES-256 Without derivation function With/without prediction resistance		SP 800-90Ar1
ECDSA	A4551	Key Pair Generation Mode: Testing Candidates (Appendix B.4.2) Curves: P-256, P-384		FIPS 186-4
Hash_DRBG	A4551, A4569, A4570, A4571	Hashes: SHA-1, SHA-256, SHA-512 With/without prediction resistance		SP 800-90Ar1
HMAC_DRBG	A4551, A4569, A4570, A4571	Hashes: SHA-1, SHA-256, SHA-512 With/without prediction resistance		SP 800-90Ar1
HMAC	A4551, A4557, A4569, A4570, A4571	SHA-1 Key size: 112-524288 bits		FIPS 198-1 FIPS 180-4
	A4551, A4557, A4561, A4562, A4569, A4570, A4571	SHA-224, SHA-256 Key size: 112-524288 bits		FIPS 198-1 FIPS 180-4
	A4551, A4557, A4562, A4569, A4570, A4571	SHA-384, SHA-512 Key size: 112-524288 bits		FIPS 198-1 FIPS 180-4
	A4551, A4557	SHA3-224, SHA3-256, SHA3-384, SHA3-512 Key size: 112-524288 bits		FIPS 198-1 FIPS 202
KAS-ECC-SSC	A4551	Scheme: Ephemeral Unified Model Roles: initiator, responder Curves: P-256, P-384		SP 800-56Ar3
KAS-FFC-SSC	A4551	Scheme: dhEphem Roles: initiator, responder Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192		
RSA	A4551, A4569, A4570, A4571	Signature Verification Padding: PKCS#1 v1.5 Hashes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Modulus: 4096 bits		FIPS 186-4
Safe Primes	A4551	Key Pair Generation Mode: Testing Candidates (Appendix 5.6.1.1.4) Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192		SP 800-56Ar3
SHA-1	A4551, A4557, A4569, A4570, A4571	N/A		FIPS 180-4
SHA-224	A4551, A4557, A4561, A4562, A4569, A4570, A4571	N/A		
SHA-256				
SHA-384	A4551, A4557, A4562, A4569, A4570, A4571	N/A		
SHA-512				
SHA3-224	A4551, A4557	N/A		FIPS 202
SHA3-256				
SHA3-384				
SHA3-512				

Table 6 - Approved Algorithms

Vendor Affirmed Algorithms:

Algorithm Name	Algorithm Capabilities	OE (Implementation)	References
Cryptographic Key Generation (CKG)	Key Pair Generation using Safe Primes and EC	Amazon Linux 2023 on EC2 c7g.metal	SP 800-133r2 Section 4, 5.1, and 5.2
		Amazon Linux 2023 on EC2 c6i.metal	
		SnowOS 1.0 on AWS Snowball	
		SnowOS 1.0 on AWS Snowblade	
		SnowOS 1.0 on AWS snowcone	

Table 7 - 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: Table 8 lists all non-approved cryptographic algorithms of the module employed by the non-approved services in Table 15.

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	Signature Generation (pre-hashed message) Signature Verification (pre-hashed message)
	Key Encapsulation Key Un-encapsulation

Table 8 - Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	SF Capabilities	Algorithms
KAS-ECC-SSC	KAS	EC Diffie-Hellman Shared Secret Computation	Security strength: 128, 192 bits Compliant with SP 800-56Ar3 and Scenario 2 (1) of FIPS 140-3 IG D.F	KAS-ECC-SSC (SP 800-56Ar3)
KAS-FFC-SSC	KAS	Diffie-Hellman Shared Secret Computation	Security strength: 112-200 bits Compliant with SP 800-56Ar3 and Scenario 2 (1) of FIPS 140-3 IG D.F	KAS-FFC-SSC (SP 800-56Ar3)
AES-KW	KTS	Key wrapping/unwrapping	Security strength: 128, 192, 256 bits	AES-KW (SP 800-38F)
AES-CCM	KTS	Key wrapping/unwrapping using authenticated encryption (as permitted by IG D.G)	Security strength: 128, 192, 256 bits	AES-CCM (SP 800-38C)
AES-GCM/WRAP	KTS	Key wrapping using authenticated encryption	IV generated internally Security strength: 128, 192, 256 bits	AES-GCM (SP 800-38D)

		(as permitted by IG D.G)		
AES-GCM/UNWRAP	KTS	Key unwrapping using authenticated encryption (as permitted by IG D.G)	IV provided externally Security strength: 128, 192, 256 bits	AES-GCM (SP 800-38D)
AES-CBC with HMAC	KTS	Key wrapping/unwrapping using “combination” mode encryption (as permitted by IG D.G)	Security strength: 128, 192, 256 bits Hashes: SHA-1, SHA-256, SHA-384, SHA-512	AES-CBC (SP 800-38A) HMAC (FIPS 198-1)
AES-CTR with HMAC	KTS			AES-CTR (SP 800-38A) HMAC (FIPS 198-1)

Table 9 - Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 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 14.

2.7.2 AES XTS

In accordance with IG C.I, the module implements a check to ensure that the two AES keys used in the 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.

2.7.3 RSA

For RSA signature verification, all supported, approved modulus sizes 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 and elliptic curve Diffie-Hellman shared secret computation algorithms with the NVMe and Bluetooth related protocols. Additionally, the module’s approved key pair generation service (see Table 14) must be used to generate ephemeral Diffie-Hellman or EC 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, and the partial public key validation of the peer EC public key, complying with Section 5.6.2.2.2 of SP 800-56Ar3.

2.7.5 Legacy Use

Digital signature verification using SHA-1 is allowed for legacy use only.

2.8 RBG and Entropy

Vendor Name	Certificate Number
Amazon	Cert. E105

Table 10 - Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy Per Sample	Conditioning Component
Amazon Kernel CPU Time Jitter RNG Entropy Source	Non-physical	See Table 3	256 bits	256 bits	SHA3-256 (A4551)

Table 11 - Entropy

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 6. When instantiated, these DRBGs can be used to generate random numbers for external usage.

Additionally, the module employs a specific HMAC-SHA-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 (corresponding to 384 bits of entropy) and reseeded with 256 output bits from the entropy source (corresponding to 256 bits of entropy).

The module complies with the Public Use Document for ESV certificate E105 seeding the aforementioned DRBG using the jent_kcapi_random function, which corresponds to the GetEntropy() function. The operational environment of the module is identical to the one listed on the ESV certificate. There are no maintenance requirements for the entropy source.

The following is the link to the Public Use Document of Amazon Kernel CPU Time Jitter RNG Entropy Source:

https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E105_PublicUse.pdf

2.9 Key Generation

The module implements Cryptographic Key Generation (CKG, vendor affirmed), 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 following methods are implemented:

- Safe Primes key pair generation: compliant with SP 800-133r2, Section 5.2, which maps to SP 800-56Ar3. The method described in Section 5.6.1.1.4 of SP 800-56Ar3 (“Testing Candidates”) is used.
- EC key pair generation: compliant with SP 800-133r2, Section 5.1, which maps to FIPS 186-4. The method described in Appendix B.4.2 of FIPS 186-4 (“Testing Candidates”) is used. Note that this generation method is also used to generated ECDH key pairs.

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 SSP agreement and SSP transport methods as listed in Table 9.

The module implements the P-256 and P-384 curves.

For P-256, N is 256 bits and s is 128 bits.
For P-384, N is 384 bits and s is 192 bits.

The module implements the ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, and ffdhe8192 safe prime groups.

For ffdhe2048, N is 2048 bits and s is 112 bits
For ffdhe3072, N is 3072 bits and s is 128 bits
For ffdhe4096, N is 4096 bits and s is 152 bits
For ffdhe6144, N is 6144 bits and s is 176 bits
For ffdhe8192, N is 8192 bits and s is 200 bits

(N is the bit length of the private key and s is the maximum security strength supported)

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 and EC Diffie-Hellman shall only be used with the NVMe and Bluetooth related protocols.

No other parts of the NVMe, Bluetooth, 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	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 12 - Ports and Interfaces

The logical interfaces are the APIs through which the applications request services. These logical interfaces are logically separated from each other by the API design. 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 13 - 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	Roles	SSP Access
Message Digest	Compute a message digest	crypto_shash_init returns 0	Message	Digest value	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	CO	N/A
Encryption	Encrypt a plaintext	crypto_skcipher_setkey returns 0	AES key, IV, plaintext	Ciphertext	AES CBC, CBC-CS3, CFB128, CTR, ECB, KW, OFB, XTS		AES key: W, E
Decryption	Decrypt a ciphertext		AES key, IV, ciphertext	Plaintext			
Authenticated Encryption	Encrypt 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, IV, plaintext	Ciphertext, MAC tag	AES CCM, GCM (internal IV) AES CBC or CTR with HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-512		AES key: W, E HMAC key: W, E
Authenticated Decryption	Decrypt a ciphertext		AES key, IV, ciphertext, MAC tag	Plaintext	AES CCM, GCM (external IV) AES CBC or CTR with HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-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-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512		HMAC key: W, E
Shared Secret Computation	Compute a shared secret	crypto_kpp_compute_shared_secret returns 0	DH private key, DH public key	Shared secret	KAS-FFC-SSC		DH private key: W, E DH public key: W, E Shared secret: G, R
			EC private key, EC public key		KAS-ECC-SSC		EC private key: W, E EC 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	DH private key, DH public key	Safe Primes Key Pair Generation		DH private key: G, R DH public key: G, R Intermediate key generation value: G,

Name	Description	Indicator	Inputs	Outputs	Security Functions	Roles	SSP Access
							E, Z
			Curve	EC private key, EC public key	EC Key Pair Generation		EC private key: G, R EC public key: G, R Intermediate key generation value: G, E, Z
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 Internal state: E, G
Error Detection Code	Compute an EDC (crc32, crct10dif)	None	Message	EDC	N/A		N/A
Compression	Compress data (deflate, lz4, lz4hc, lzo, zlib-deflate, zstd)	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, KAS-FFC-SSC, KAS-ECC-SSC, CTR_DRBG, Hash_DRBG, HMAC_DRBG, RSA See Table 22 for specifics		N/A
Zeroization	Zeroize all SSPs	None	Any SSP	N/A	N/A		All SSPs: Z

Table 14 - 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.
- **N/A**: The module does not access any SSP or key during its operation.

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 (external IV)	CO
Key Derivation	Derive a key from a key-derivation key or a shared secret	KBKDF (libkcapi)	

Name	Description	Security Functions	Role
		HKDF (libkcapi)	
Password-Based Key Derivation	Derive a key from a password	PBKDF2 (libkcapi)	
Encryption Primitive	Compute the raw RSA encryption of a plaintext/ciphertext	RSA	
Decryption Primitive	Compute the raw RSA decryption of a plaintext/ciphertext		
Signature Generation (pre-hashed message)	Generate a digital signature for a pre-hashed message	RSA with PKCS#1 v1.5 padding	
Signature Verification (pre-hashed message)	Verify a digital signature for a pre-hashed message		
Key Encapsulation	Encapsulate a secret key using RSA with PKCS#1 v1.5 padding		
Key Un-encapsulation	Un-encapsulate a secret key using RSA with PKCS#1 v1.5 padding		

Table 15 - 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

The Linux kernel binary is integrity tested using an HMAC-SHA-512 calculation performed by the sha512hmac utility (which utilizes the module's HMAC and SHA-512 implementations) which compares the computed HMAC value with a precomputed HMAC value. An HMAC-SHA-512 calculation is also performed on the sha512hmac utility and the libkcapi library to verify their integrity by comparing the computed HMAC value with a precomputed HMAC value. 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-512, and a 4096-bit key.

5.2 Initiate On-Demand Integrity Test

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: The module operates in a modifiable operational environment. The module runs on commercially available general-purpose operating systems (Amazon Linux 2023 and SnowOS 1.0), which allows modification, loading, and execution of software that is not part of the validated module.

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 Configurable Settings and Restrictions

The module shall be installed as stated in Section 11.

Instrumentation tools like the `ptrace` system call, `gdb` and `strace`, user space live patching, 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 16 - 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
API input parameters	Operator calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic
AF_ALG_type sockets (input)					
API output parameters	Cryptographic module	Operator calling application (TOEPP)			
AF_ALG type sockets (output)					

Table 17 - SSP Input-Output Methods

9.3 SSP Zeroization Methods

Method	Description	Rationale	Operator Initiation Capability
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 zeroization functions: **AES Key:** crypto_free_skcipher and crypto_free_aead **HMAC Key:** crypto_free_shash and crypto_free_ahash **Internal State (V, Key), Internal State (V, C):** crypto_free_rng **DH Public Key & DH Private Key**: crypto_free_kpp **EC Public Key & EC Private Key:** crypto_free_kpp **RSA Public Key:** public_key_free
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
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 18 - SSP Zeroization Methods

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size	Strength	Type	Generated By	Established By
AES Key	AES key used for encryption, decryption, and computing MAC tags	XTS: 256, 512 bits Other modes: 128, 192, 256 bits	XTS: 128, 256 bits Other modes: 128, 192, 256 bits	Symmetric key	N/A	N/A

Name	Description	Size	Strength	Type	Generated By	Established By
HMAC Key	HMAC key used for computing MAC tags	112-524288 bits	112-256 bits	Symmetric key	N/A	N/A
Shared Secret	Shared secret generated by (EC) Diffie-Hellman	P-256, P-384	128, 192 bits	Shared secret	N/A	SP 800-56Ar3 (KAS-ECC-SSC)
		ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	112-200 bits			SP 800-56Ar3 (KAS-FFC-SSC)
Entropy input	Entropy input used to seed the DRBGs (IG D.L)	256-384 bits	256-384 bits	Entropy input	Entropy Source See Table 11	N/A
DRBG Seed	DRBG seed derived from entropy input (IG D.L)	CTR_DRBG: 256, 320, 384 bits Hash_DRBG: 440, 888 bits HMAC_DRBG: 160, 256, 512 bits	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
Internal State (V, Key)	Internal state of CTR_DRBG and HMAC_DRBG instances (IG D.L)	CTR_DRBG: 256, 320, 348 bits HMAC_DRBG: 320, 512, 1024 bits		Internal state	CTR_DRBG, HMAC_DRBG	N/A
Internal State (V, C)	Internal state of Hash_DRBG instances (IG D.L)	Hash_DRBG: 880, 1776 bits		Internal state	Hash_DRBG	N/A
DH Public Key	Public key used for Diffie-Hellman	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	112-200 bits	Public key	SP 800-56Ar3 (Safe Primes) Section 5.6.1.1.4 Testing Candidates	N/A
DH Private Key	Private key used for Diffie-Hellman		112-200 bits	Private key		
EC Public Key	Public key used for ECDH	P-256, P-384	128, 192 bits	Public key	FIPS 186-4 Appendix B.4.2 Testing Candidates	N/A
EC Private Key	Private key used for ECDH			Private key		
Intermediate Key Generation Value	Temporary value generated during Key Pair Generation services	2048-8192 bits	112-200 bits	Intermediate value	CKG	N/A

Table 19 - SSP Information First

Name	Used By	Inputs/Outputs	Storage	Temporary Storage Duration	Zeroization	Category	Related SSPs
AES Key	Encryption Decryption Authenticated Encryption Authenticated Decryption Message Authentication	API input parameters AF_ALG type sockets (input) No output	RAM	For the duration of the service	Free cipher handle Remove power from the module	CSP	None
HMAC Key	Message Authentication Authenticated Encryption Authenticated Decryption					CSP	None
Shared Secret	Shared Secret Computation	No input API output parameters, AF_ALG type sockets (output)			Automatic Remove power from the module	CSP	DH Public Key, DH Private Key, EC Public Key, EC Private Key
Entropy Input	Random Number Generation	No input No output		From generation until DRBG Seed is created		CSP	DRBG Seed

DRBG Seed		No input No output		While the DRBG is instantiated		CSP	Entropy Input, Internal State
Internal State (V, Key)		No input No output		From DRBG instantiation until DRBG termination	Free cipher handle Remove power from the module	CSP	DRBG Seed
Internal State (V, C)		No input No output				CSP	DRBG Seed
DH public Key	Shared Secret Computation Key Pair Generation	API input parameters AF_ALG type sockets (input) API output parameters AF_ALG type sockets (output)		For the duration of the service		PSP	DH Private Key, Shared Secret
DH private Key	Shared Secret Computation Key Pair Generation					CSP	DH Public Key, Shared Secret
EC Public Key	Shared Secret Computation Key Pair Generation					PSP	EC Private Key, Shared Secret
EC Private Key	Shared Secret Computation Key Pair Generation					CSP	EC Public Key, Shared Secret
Intermediate Key Generation Value	Key Pair Generation	No input No output			Automatic	CSP	DH Private Key, DH Public Key, EC Private Key, EC Public Key

Table 20 - SSP Information Second

9.5 Transitions

The SHA-1 algorithm as implemented by the module will be non-approved for all purposes, starting January 1, 2030.

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-SHA-512	AVX2, C, CE	128-bit key	Message Authentication	Software integrity	Module becomes operational	Integrity test for vmlinuz and libkcapi components
RSA	C	PKCS#1 v1.5 with SHA-512 4096-bit key	Signature Verification			Integrity test for kernel object files

Table 21 - 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	Type	Indicator	Details	Condition
AES-CBC	AESNI, C, CE, NEON	128, 192, 256-bit keys	KAT	CAST	Module is operational	Encryption Decryption	Module initialization
AES-CBC-CS3	AESNI, C, CE, NEON	128-bit keys					
AES-CCM	AESNI, C, CE	128, 192, 256-bit keys 128-bit IVs					
AES-CFB128	AESNI, C, CE	128, 192, 256-bit keys					
AES-CMAC	AESNI, C, CE, NEON	128, 256-bit keys				Message Authentication	
AES-CTR	AESNI, C, CE, NEON	128, 192, 256-bit keys				Encryption Decryption	
AES-ECB	AESNI, C, CE, NEON	128, 192, 256-bit keys					
AES-GCM (internal IV)	AESNI, C	128, 192, 256-bit keys 96-bit IVs				Encryption	
AES-GCM (external IV)	AESNI, C	128, 192, 256-bit keys				Decryption	
AES-OFB	AESNI, C, CE	128-bit keys				Encryption Decryption	
AES-XTS	AESNI, C, CE, NEON	128, 256-bit keys					
CTR_DRBG	C	AES-128, AES-192, AES-256 without prediction resistance AES-128 with prediction resistance				Instantiate Seed Reseed Generate (compliant to SP 800-90A Section 11.3)	
Hash_DRBG	C	SHA-256 with/without prediction resistance					
HMAC_DRBG	C	SHA-256, SHA-512 without prediction resistance SHA-256 with prediction resistance					
HMAC-SHA-1	C, CE	32-64-bit keys				Message Authentication	
HMAC-SHA-224	C, CE	32-1048-bit keys					
HMAC-SHA-256	C, CE	32-64-bit keys					Before integrity test
HMAC-SHA-384	AVX2, C, CE	32-1048-bit keys					Module

Algorithm	Implementation	Test Properties	Test Method	Type	Indicator	Details	Condition
HMAC-SHA-512	AVX2, C, CE, SSSE3	32-1048-bit keys					initialization
HMAC-SHA3-224	C, CE	32-1048-bit keys					
HMAC-SHA3-256	C, CE	32-1048-bit keys					
HMAC-SHA3-384	C, CE	32-1048-bit keys					
HMAC-SHA3-512	C, CE	32-1048-bit keys					
KAS-ECC-SSC	C	P-256, P-384				Shared Secret Computation	
KAS-FFC-SSC	C	ffdhe2048					
SHA-1	AVX, AVX2, C, CE, SSSE3	0–8184-bit messages				Message Digest	
SHA-224	ARM64, AVX, AVX2, C, CE, NEON, SSSE3						
SHA-256							
SHA-384	ARM64, AVX, AVX2, C, CE, SSSE3						
SHA-512							
SHA3-224	C, CE						
SHA3-256	C, CE						
SHA3-384	C, CE						
SHA3-512	C, CE						
RSA	C	PKCS#1 v1.5 with SHA-256 4096-bit key				Signature Verification	
Safe Primes	C	N/A	PCT	PCT	Key Pair Generation is successful	SP 800-56Ar3 Section 5.6.2.1.4	Key Pair Generation
EC	C	N/A					
Entropy Source	C	Cutoff C = 61 1024 samples	RCT	CAST	Entropy source is operational	Entropy source start-up test	Entropy source initialization
		Cutoff C = 355 Window W = 512 1024 samples	APT				
		Intermittent cutoff C = 31 Permanent cutoff C = 61	RCT		jent_kcapi_random returns 0	Entropy source continuous test	Continuously
		Intermittent cutoff C = 325 Permanent cutoff C = 355 Window W = 512	APT				

Table 22 - Conditional Self-Tests

The module performs self-tests on all approved cryptographic algorithms as part of the approved services supported in the approved mode of operation, using the tests shown in Table 22.

Upon generation of a DH or EC key pair, the module will perform a pair-wise consistency test (PCT) as shown in Table 22, which provides some assurance that the generated key pair is well formed. This test consists of the PCT described in Section 5.6.2.1.4 of SP 800-56Ar3.

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-Tests

The module does not implement periodic self-tests.

10.4 Error States

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

Table 23 - Error States

If the module fails any of the self-tests, the module enters the error state. 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 Amazon Linux 2023 and SnowOS 1.0 package in the form of the kernel-6.1.41-64.118.amzn2023.rpm, kernel-6.1.41-64.118.fips.amzn2023.rpm, and libkcapi-hmaccalc-1.4.0-105.amzn2023.0.1.rpm packages.

11.2 Administrator Guidance

Before the RPM packages are installed, the Amazon Linux 2023 or SnowOS 1.0 system must operate in the approved mode. This can be achieved by switching the system into the approved mode after the installation. Execute the fips-mode-setup --enable command. Restart the system. More information can be found at the vendor documentation.

The Crypto Officer must verify the Amazon Linux 2023 or SnowOS 1.0 system operates in the approved mode by executing the fips-mode-setup --check command, which should output “FIPS mode is enabled.”

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:

```
Amazon Linux 2023 Kernel Cryptographic API
```

Then, the Crypto Officer must execute the “cat /proc/sys/crypto/fips_version” and “rpm -q libkcapi-hmaccalc” commands. These commands must output the following (one line per output):

EC2 c7g.metal:

```
6.1.41-64.118.amzn2023.aarch64
libkcapi-hmaccalc-1.4.0-105.amzn2023.0.1.aarch64
```

EC2 c6i.metal:

```
6.1.41-64.118.amzn2023.x86_64
libkcapi-hmaccalc-1.4.0-105.amzn2023.0.1.x86_64
```

AWS Snowball, AWS Snowblade, AWS Snowcone:

```
6.1.41-64.118.fips.amzn2023.x86_64
libkcapi-hmaccalc-1.4.0-105.amzn2023.0.1.x86_64
```

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 End of Life Procedures

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 RPM packages can be uninstalled from the Amazon Linux 2023 or SnowOS 1.0 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
CE	Cryptography Extensions
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
CTS	Ciphertext Stealing
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
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
KAS	Key Agreement Scheme
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
KTS	Key Transport Scheme

KW	Key Wrap
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PBKDF2	Password-based Key Derivation Function v2
PCT	Pair-Wise Consistency Test
PKCS	Public-Key Cryptography Standards
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
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 186-5	Digital Signature Standard (DSS) February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.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
RFC 4106	The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP) June 2005 https://datatracker.ietf.org/doc/html/rfc4106
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 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP 800-56Ar3	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.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
SP 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP 800-108r1	NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions August 2022 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf
SP 800-131Ar2	Transitioning the Use of Cryptographic Algorithms and Key Lengths March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP 800-133r2	Recommendation for Cryptographic Key Generation June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP 800-140Br1	CMVP Security Policy Requirements November 2023 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140Br1.pdf