Amazon Linux 2 Kernel Crypto API Cryptographic Module

Software Module Version: 1.0

FIPS 140-2 Non-Proprietary Security Policy

Document Version: 1.3

Document Date: 2023-10-10

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

List of Tables

List of Figures

Copyrights and Trademarks

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

1 Introduction

This document is the non-proprietary FIPS 140-2 Security Policy for version 1.0 of the Amazon Linux 2 Kernel Crypto API Cryptographic Module. It contains the security rules under which the module must be operated and describes how this module meets the requirements as specified in FIPS 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 module.

1.1 Purpose of the Security Policy

There are three major reasons that a security policy is needed:

- It is required for FIPS 140 2 validation,
- It allows individuals and organizations to determine whether a cryptographic module, as implemented, satisfies the stated security policy, and
- It describes the capabilities, protection and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will meet their security requirements.

1.2 Target Audience

This document is part of the package of documents that are submitted for FIPS 140 2 conformance validation of the module. It is intended for the following audience:

- Developers.
- FIPS 140-2 testing lab.
- The Cryptographic Module Validation Program (CMVP).
- Customers using or considering integration of Amazon Linux 2 Kernel Crypto API Cryptographic Module.

1.3 How 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.

2 Cryptographic Module Specification

2.1 Module Overview

The Amazon Linux 2 Kernel Crypto API Cryptographic Module (hereafter referred to as the “module”) is a software module supporting FIPS 140-2 Approved cryptographic algorithms. The module provides a C language application program interface (API) for use by other processes that require cryptographic functionality.

2.2 FIPS 140-2 Validation Scope

Table 1 shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard.

Table 1: FIPS 140-2 Security Requirements.

Security Requirements Section		Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles and Services and Authentication	1
4	Finite State Machine Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A
Overall Level		1

2.3 Definition of the Cryptographic Module

The Amazon Linux 2 Kernel Crypto API Cryptographic Module is defined as a Multi-chip Standalone module per the requirements within FIPS 140-2. The logical cryptographic boundary of the module consists of the static kernel binary, the sha512hmac application, their respective integrity HMAC files, and the kernel loadable components, which are delivered through the Amazon Linux 2 yum core repositories (ID amz2-core/2/x86_64 and amz2-core/2/aarch64) from the RPM file with version kernel-4.14.146-119.123.amzn2.x86_64 for the x86_64 platform and kernel-4.14.146-120.123.amzn2.aarch64 for the aarch64 platform:

- kernel loadable components:
  x86_64: /lib/modules/4.14.146-119.123.amzn2.x86_64/kernel/crypto/*.ko
  aarch64: /lib/modules/4.14.146-120.123.amzn2.aarch64 /kernel/crypto/*.ko
- kernel loadable components:
  x86_64: /lib/modules/4.14.146-119.123.amzn2.x86_64/kernel/arch/x86/crypto/*.ko
  aarch64: /lib/modules/4.14.146-120.123.amzn2.aarch64/kernel/arch/x86/crypto/*.ko

- static kernel binary (vmlinuz):
  x86_64: /boot/vmlinuz-4.14.146-119.123.amzn2.x86_64
  aarch64: /boot/vmlinuz-4.14.146-120.123.amzn2.aarch64
- static kernel binary (vmlinuz) HMAC file:
  x86_64: /boot/.vmlinuz-4.14.146-119.123.amzn2.x86_64.hmac
  aarch64: /boot/vmlinuz-4.14.146-120.123.amzn2.aarch64.hmac
- sha512hmac binary file for performing the integrity checks:
  usr/bin/sha512hmac
- sha512hmac binary HMAC file:
  /usr/lib64/hmaccalc/sha512hmac.hmac

The module instantiation is provided by the dracut-fips RPM package with the file versions:

x86_x64:
dracut-fips-033-535.amzn2.1.3.x86_64.rpm

aarch64:
dracut-fips-033-535.amzn2.1.3.aarch64.rpm

The AES-NI configuration of the kernel is provided by the dracut-fips RPM package with the file versions:

x86_64:
dracut-fips-aesni-033-535.amzn2.1.3.x86_64.rpm

aarch64:
dracut-fips-aesni-033-535.amzn2.1.3.aarch64.rpm

The module shall be instantiated by the dracut-fips package with the RPM file version specified above. The dracut-fips RPM package is only used for the installation and instantiation of the module. This code is not active when the module is operational and does not provide any services to users interacting with the module. Therefore, the dracut-fips RPM package is outside the module's logical boundary.

The bound module Amazon Linux 2 NSS Cryptographic Module with FIPS 140-2 Certificate #4565 (hereafter referred to as the “NSS bound module” or “NSS module”) is also required for the Kernel Crypto API to operate. The NSS bound module provides the HMAC-SHA2-512 algorithm used by the sha512hmac binary file to verify the integrity of both the sha512hmac file and the vmlinuz (static kernel binary). This HMAC-SHA2-512 algorithm from the bound module is otherwise not exposed to a user of the Amazon Linux 2 Kernel Crypto API Cryptographic Module.

Figure 1 shows the logical block diagram of the module executing in memory on the host system.

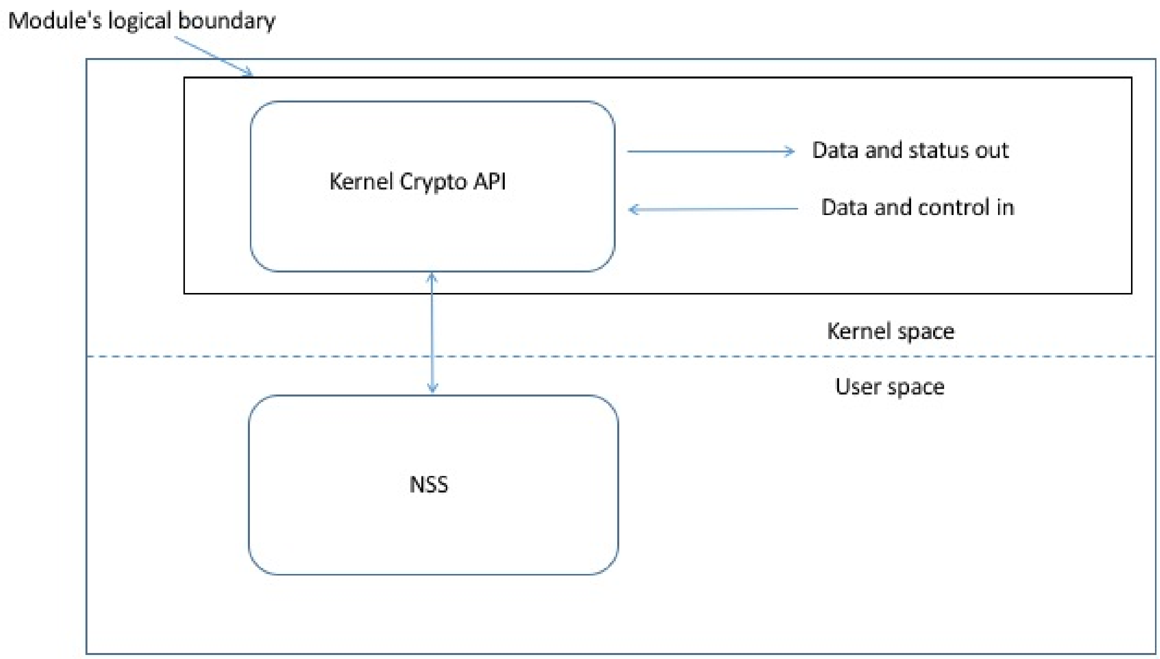

Figure 1: Logical cryptographic boundary.

2.4 Definition of the Physical Cryptographic Boundary

The physical cryptographic boundary of the module is defined as the hard enclosure of the host system on which the module runs. Figure 2 depicts the hardware block diagram. The physical hard enclosure is indicated by the dashed colored line. No components are excluded from the requirements of FIPS 140-2.

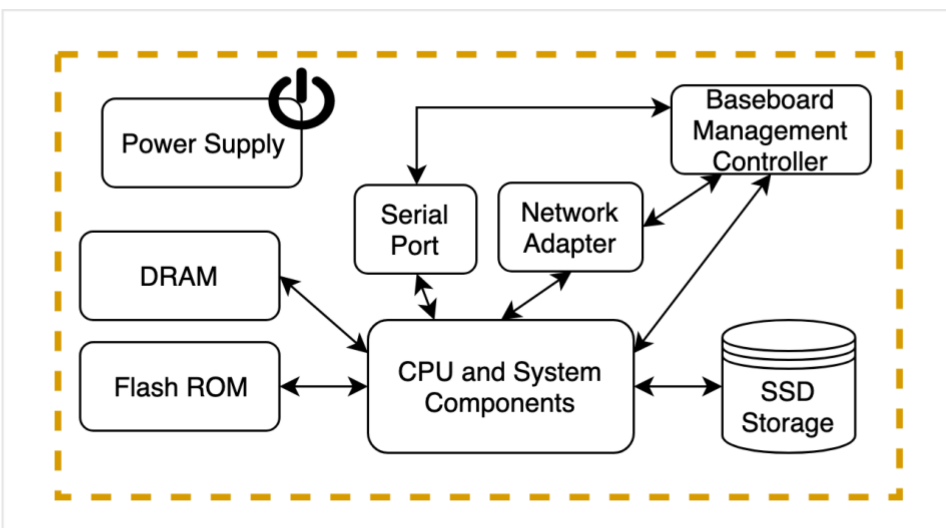

Figure 2: Hardware block diagram.

2.5 Tested Environments

The module was tested on the environments/platforms listed in Table 2. The tested operational environment is not a virtualized cloud service, and was controlled such that the laboratory had full and exclusive access to the environment and module during the testing procedures.

Table 2: Tested operational environments.

Operating System	Processor	Hardware
Amazon Linux 2	Intel Xeon E5-2686 x86_64 bits with PAA	Amazon EC2 i3.metal 512 GiB system memory 13.6 TiB SSD storage + 8 GiB SSD boot disk 25 Gbps Elastic Network Adapter
Amazon Linux 2	Intel Xeon E5-2686 x86_64 bits without PAA	Amazon EC2 i3.metal 512 GiB system memory 13.6 TiB SSD storage + 8 GiB SSD boot disk 25 Gbps Elastic Network Adapter
Amazon Linux 2	Graviton 2 aarch64 bit with PAA (i.e., Neon and Crypto Extension (EC))	Amazon EC2 c6g.metal 128 GiB system memory 200 GiB SSD storage 25 Gbps Elastic Network Adapter
Amazon Linux 2	Graviton 2 aarch64 bit without PAA (i.e., Neon and Crypto Extension (EC))	Amazon EC2 c6g.metal 128 GiB system memory 200 GiB SSD storage 25 Gbps Elastic Network Adapter

2.6 Modes of Operation

The module supports two modes of operation.

- In "**FIPS mode**" (the Approved mode of operation), only approved or allowed security functions with sufficient security strength are offered by the module.
- In "**non-FIPS mode**" (the non-Approved mode of operation), non-approved security functions are offered by the module.

The module enters the operational mode after Power-On Self-Tests (POST) succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength[1] of the cryptographic keys chosen for the service.

If the POST or the Conditional Tests fail (Section 9), the module goes into the error state. The status of the module can be determined by the availability of the module. If the module is available, then it had passed all self-tests. If the module is unavailable, it is because any self-test failed, and the module has transitioned to the error state.

Keys and Critical Security Parameters (CSPs) used or stored in FIPS mode shall not be used in non-FIPS mode, and vice versa.

[1] See Section 5.6.1 in [SP800-57] for a definition of "security strength".

3 Module Ports and Interfaces

As a Software module, the module does not have physical ports. The operator can only interact with the module through the API provided by the module. Thus, the physical ports within the physical boundary are interpreted to be the physical ports of the hardware platform on which the module runs and are directed through the logical interfaces provided by the software.

The logical interfaces are the API through which applications request services and receive output data through return values or modified data referenced by pointers. Table 3 summarizes the logical interfaces.

Table 3: Ports and interfaces.

Logical Interface	Description
Data Input	API input parameters from kernel system calls, AF_ALG type socket.
Data Output	API output parameters from kernel system calls, AF_ALG type socket.
Control Input	API function calls, API input parameters for control from kernel system calls, AF_ALG type socket, kernel command line.
Status Output	API return codes, AF_ALG type socket, kernel logs.
Power Input	N/A

4 Roles, Services and Authentication

4.1 Roles

The module supports the following roles:

- **User role**: performs all services (in both FIPS mode and non-FIPS mode of operation), except module installation and configuration. This role is assumed by the calling application accessing the module.
- **Crypto Officer (CO) role**: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed depending on the service requested.

4.2 Services

The module provides services to calling applications that assume the user role, and human users assuming the Crypto Officer role. Table 4 and Table 5 depict all services, which are described with more detail in the user documentation.

The tables use the following convention when specifying the access permissions that the service has for each CSP or key.

- **Create (C)**: the service can create a new CSP.
- **Read (R)**: the service can read the CSP.
- **Update (U)**: the service can write a new value to the CSP.
- **Zeroize (Z)**: the service can zeroize the CSP.
- **N/A**: the service does not access any CSP or key during its operation.

For the “Role” column, U indicates the User role, and CO indicates the Crypto Officer role. An X marks which role has access to that service.

Note: The bound NSS module does not offer any service to a user of the Amazon Linux 2 Kernel Crypto API Cryptographic Module.

4.2.1 Services in the FIPS-Approved Mode of Operation

Table 4 provides a full description of FIPS Approved services and the non-Approved but Allowed services provided by the module in the FIPS-approved mode of operation and lists the roles allowed to invoke each service.

Table 4: Services in the FIPS-approved mode of operation.

Service	Service Description and Algorithms	Role		Keys and CSPs	Access Types
		User	CO		
Symmetric Encryption/Decryption	Encrypts or decrypts a block of data using AES	X		AES Keys	R
	Encrypts or decrypts a block of data using 3-Key Triple-DES			Triple-DES Keys	
RSA Digital Signature Verification	Verify signature operations for RSA PKCS#1v1.5	X		RSA public/private keys	R
Key Wrapping	Encrypt and decrypt a key using RSA primitives	X		RSA public/private keys	R
	Wrap and unwrap keys with:	X		AES keys	R

Service	Service Description and Algorithms	Role		Keys and CSPs	Access Types
		User	CO		
	▪ AES-KW ▪ AES-CCM ▪ AES-GCM ▪ AES (ECB, CBC, CTR) and HMAC				
Message Authentication Code (MAC)	Authenticate and verify authentication of data using HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	X		HMAC Key	R
	Authenticate and verify authentication of data using CMAC with AES	X		AES Key	R
	Authenticate and verify authentication of data using CMAC with Triple-DES	X		Triple-DES Key	R
Message Digest	Hash a block of data with: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 SHA3-224, SHA3-256, SHA3-384, SHA3-512	X		None	N/A
Random Number Generation	Generate random numbers based on the SP 800-90A DRBG	X		Entropy input string and internal state	C, R, U
Other FIPS-related Services					
Show Status	Show status of the module state	X		None	N/A
Self-Tests	Initiate power-on self-tests	X		None	N/A
Zeroization	Zeroize all critical security parameters	X		All keys and CSPs	Z
Module Installation	Installation of the module		X	None	N/A
Module Configuration	Configuration of the module		X	None	N/A

4.2.2 Services in the Non-FIPS-Approved Mode of Operation

Table 5 presents the services only available in non-FIPS-approved mode of operation.

Table 5: Services in the non-FIPS approved mode of operation.

Service	Service Description and Algorithms	Role		Keys	Access Types
		User	CO		
AES encryption/decryption	XTS block chaining mode	X		192-bit AES key	C, R, U
	CTS block chaining mode			128/192/256-bit key	
	OFB block chaining mode				
	GCM with external IV generation				
Triple-DES encryption/decryption	2-key Triple-DES encryption/decryption	X		112-bit Triple-DES key	C, R, U
HMAC MAC generation	Keyed message digest	X		HMAC key smaller than 112 bits	C, R, U
CBC-MAC MAC generatoin	CBC-MAC operation outside the CCM operation	X		Triple-DES or AES key	C, R, U
GHASH message digest generation	GHASH operation outside the GCM operation	X		AES key	C, R, U
RSA operations	Encrypts or decrypts using non-Approved RSA key size	X		RSA public/private key	C, R, U
	Sign or verify primitives				
Diffie-Hellman shared secret computation	Shared secret computation KAS FFC	X		Diffie-Hellman public/private keys	C, R, U
				Shared secret	C, R, U
EC Diffie-Hellman shared secret computation	Shared secret computation KAS ECC	X		EC Diffie-Hellman public/private keys	R

4.3 Algorithms

The module implements cryptographic algorithms that are used by the services provided by the module. The cryptographic algorithms that are approved to be used in the FIPS mode of operation are tested and validated by the CAVP.

The module provides assembler implementations for AES, Triple-DES and SHS; support for the AVX2, AVX, AESNI and SSSE3 instruction sets from the CPU (per the tested operational environment in Table 2) for AES and SHS; CLMUL for the GHASH algorithm used in the GCM block mode; and C-language generic implementations for the rest of the algorithms.

Table 6, Table 7 and Table 8 present the cryptographic algorithms in specific modes of operation. These tables include the CAVP certificates for different implementations, the algorithm name, respective standards, the available modes and key sizes wherein applicable, and usage. Information from certain columns may be applicable to more than one row.

4.3.1 FIPS-Approved

Table 6 lists the cryptographic algorithms that are approved to be used in the FIPS mode of operation. Appendix B brings a list of the names contained in the algorithm certificates (e.g., aesni_iiv), which refer to the specific algorithm implementations, and their description.

Note: some algorithms for which a CAVP certificate was issued may be not implemented by the module as FIPS approved algorithms.

Table 6: FIPS-approved cryptographic algorithms.

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert#
AES	[FIPS197] [SP800-38A]	CBC, CTR[2], ECB,	128, 192 and 256 bits	Data Encryption and Decryption	A3671 (DH_C) A3672 (ECDH_C) A3673 (C_C) A3675 (IIV_C_C) A3676 (CTI_C) A3677 (IIV_CTI_C) A3678 (ARM64_CE) A3679 (ARM_64_CE_C) A3680 (IIV_ARM64_CE_C) C911 (aesasm) C912 (aesasm-iiv)[5] C913 (aesgen) C914 (aesgen_iiv)[5] C915 (aesni) C917 (aesni_iiv)
	[FIPS197] [SP800-38B]	CMAC	128, 192 and 256 bits	MAC Generation and Verification	A3673 (C_C) A3676 (CTI_C) A3678 (ARM64_CE) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)

[2] AES-CTR tested for encryption only.

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert#
	[FIPS197] [SP800-38C]	CCM	128, 192 and 256 bits	Data Encryption and Decryption	A3673 (C_C) A3676 (CTI_C) A3678 (ARM64_CE) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)
	[FIPS197] [SP800-38D]	GCM (internal IV, mode 8.2.2)	128, 192 and 256 bits	Data Encryption and Decryption	A3675 (IIV_C_C) A3677 (IIV_CTI_C) A3680 (IIV_ARM64_CE_C) C912 (aesasm-iiv)[6] C914 (aesgen_iiv)[6] C917 (aesni_iiv)[3]
		GCM (external IV)	128, 192 and 256 bits	Data Decryption	A3673 (C_C) A3676 (CTI_C) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)
		GMAC	128, 192 and 256 bits	Message Authentication Code	A3673 (C_C) A3676 (CTI_C) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)
	[FIPS197] [SP800-38E]	XTS	128 and 256 bits	Data Encryption and Decryption	A3673 (C_C) A3676 (CTI_C) A3678 (ARM64_CE) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)

[3] AES-GCM tested for encryption only.

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert#
	[FIPS197] [SP800-38F]	KW	128, 192 and 256 bits	Key Wrapping	A3673 (C_C) A3676 (CTI_C) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)
DRBG	[SP800-90A]	Hash_DRBG SHA-1, SHA2-256, SHA2-384, SHA2-512 with/without PR	n/a	Random Number Generation	A3671 (DH_C) A3672 (ECDH_C) A3673 (C_C) A3676 (CTI_C) C921 (shassse3)[4] C920 (shagen) C919 (shaavx2) C918 (shaavx)[8]
		HMAC_DRBG SHA-1, SHA2-256, SHA2-384, SHA2-512 with/without PR	n/a	Random Number Generation	A3671 (DH_C) A3672 (ECDH_C) A3673 (C_C) A3676 (CTI_C) C921 (shassse3)[8] C920 (shagen) C919 (shaavx2) C918 (shaavx)[8]
		CTR_DRBG AES128, AES192, AES256 with DF, with/without PR	n/a	Random Number Generation	A3671 (DH_C) A3672 (ECDH_C) A3673 (C_C) A3676 (CTI_C) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)

[4] The SHASSSE3 and SHAAVX implementations do not include SHA2-224, SHA2-384, HMAC-SHA2-224, HMAC-SHA2-384, Hash_DRBG with SHA2-384 and HMAC_DRBG with SHA2-384.

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert#
HMAC	[FIPS198-1]	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	112 bits or greater	Message Authentication Code	A3671 (DH_C) A3672 (ECDH_C) A3673 (C_C) A3678 (ARM64_CE) A3681 (ARM64_ASM) C921 (shassse3)[8] C920 (shagen) C919 (shaavx2) C918 (shaavx)[8]
		SHA3-224, SHA3-256, SHA3-384, SHA3-512			A3674 (SHA_C_C) C923
RSA	[FIPS186-4]	PKCS#1 v1.5 with SHA2-256, SHA2-512	2048 and 3072 bits	Signature Verification	A3673 (C_C) C923 C921 (shassse3) C920 (shagen) C919 (shaavx2) C918 (shaavx)
SHS	[FIPS180-4]	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	N/A	Message Digest	A3671 (DH_C) A3672 (ECDH_C) A3673 (C_C) A3678 (ARM64_CE) A3681 (ARM64_ASM) C921 (shassse3)[8] C920 (shagen) C919 (shaavx2) C918 (shaavx)[8]
SHA-3	[FIPS202]	SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Message Digest	A3674 (SHA3_C_C) C923
Triple-DES	[SP800-67] [SP800-38A]	ECB, CBC, CTR,	192 bits	Data Encryption and Decryption	A3673 (C_C) C923
	[SP800-67] [SP800-38B]	CMAC	192 bits	MAC Generation and Verification	

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert#
KTS	[SP800-38F]	AES-KW	128, 192, 256 bits	Key Wrapping	A3673 (C_C) A3676 (CTI_C) A3679 (ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni)
		AES-[GCM, CCM]	128, 192, 256 bits	Key Wrapping	A3673 (C_C) A3675 (IIV_C_C) A3676 (CTI_C) A3677 (IIV_CTI_C) A3678 (ARM64_CE) A3679 (ARM64_CE_C) A3680 (IIV_ARM64_CE_C) C911 (aesasm) C913 (aesgen) C915 (aesni) C917 (aesni_iiv) [6] C912 (aesasm-iiv)[6] C914 (aesgen_iiv)[6]

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert#
		AES-[CBC, CTR, ECB] and HMAC	128, 192, 256 bits	Key Wrapping	A3671 (DH_C) A3672 (ECDH_C) A3673 (C_C) A3675 (IIV_C_C) A3676 (CTI_C) A3677 (IIV_CTI_C) A3678 (ARM64_CE) A3679 (ARM_64_CE_C) A3680 (IIV_ARM64_CE_C) C917 (aesni_iiv) C911 (aesasm) C912 (aesasm-iiv)[5] C913 (aesgen) C914 (aesgen_iiv)[5] C915 (aesni) C921 (shassse3)[8] C920 (shagen) C919 (shaavx2) C918 (shaavx)
Algorithms used from the bound NSS Module					
HMAC	[FIPS198-1]	SHA2-512	112 bits or greater	Message Authentication Code	A3741 C803

4.3.3 Non-Approved-but-Allowed

Table 7 lists the non-Approved-but-Allowed cryptographic algorithms provided by the module that are allowed to be used in the FIPS mode of operation.

Table 7: Non-Approved-but-allowed cryptographic algorithms.

Algorithm	Usage
RSA	Encrypt/Decrypt with keys between 2048 bits up to 15,360 or more
NDRNG	Used for seeding NIST SP 800-90A DRBG.

4.3.4 Non-Approved

Table 8 lists the cryptographic algorithms that are not allowed to be used in the FIPS mode of operation. Use of any of these algorithms (and corresponding services in Table 5) will implicitly switch the module to the non-Approved mode.

Table 8: Non-FIPS approved cryptographic algorithms.

Algorithm	Usage
AES	XTS with 192-bit keys CTS with 128/192/256-bit keys OFB with 128/192/256-bit keys GCM encryption with external IV generation with 128/192/256-bit keys
Triple-DES	2-key Triple-DES
HMAC	HMAC with keys smaller than 112 bits
CBC-MAC	CBC-MAC not part of CCM
GHASH	GHASH not part of GCM
Diffie-Hellman	Shared secret computation
EC Diffie-Hellman	Shared secret computation
RSA	Encrypt/Decrypt with keys smaller than 2048 bits Sign/Verify primitive operation (i.e., the computation of the hash of the message if performed externally to the primitive, and the hash must be provided to the primitive) for any key size

4.4 Operator Authentication

The module does not support operator authentication mechanisms. The role of the operator is implicitly assumed based on the service requested.

5 Physical Security

The module is comprised of software only and thus this Security Policy does not claim any physical security.

6 Operational Environment

6.1 Applicability

The module operates in a modifiable operational environment per FIPS 140-2 Security Level 1 specifications. The module runs on the Amazon Linux 2 operating system executing on the hardware specified in Section 2.5.

6.2 Policy

The operating system is restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded by the operating system).

The application that makes calls to the modules is the single user of the module, even when the application is serving multiple clients.

7 Cryptographic Key Management

Table 9 summarizes the keys and other CSPs that are used by the cryptographic services implemented in the module. All CSPs are zeroized when the cypher handle is freed (Section 7.4).

Table 9: Lifecycle of keys and other Critical Security Parameters (CSPs).

Name	Use	Generation	Entry and Output
AES Key	Encryption, decryption. MAC generation and verification for CMAC.	N/A	Entered via API input parameter. No output.
Triple-DES Key			
HMAC Key	MAC generation and verification		
RSA public and private key	Key Wrapping	N/A	
Entropy input string	Entropy input strings used in seeds to the DRBG.	Obtained from NDRNG.	N/A
DRBG Internal state (V, C, Key)	V and key are used internally by HMAC and CTR DRBGs. V and C are used internally by HASH DRBG. Used to generate random bits.	Derived from Entropy input as defined in NIST SP 800-90A	

7.1 Random Number Generation

The module provides a DRBG compliant with [SP800-90A] for random number generation. The DRBG implements a Hash_DRBG, CTR_DRBG, and HMAC_DRBG mechanisms and performs the health tests for the SP800-90A DRBG as defined per Section 11.3 of SP800-90A. The DRBG is initialized during module initialization and seeded from the in-kernel NDRNG. The NDRNG is within the module's physical boundary but outside the module's logical boundary. The NDRNG provides at least 256 bits of entropy to the DRBG.

The module performs continuous random number generator tests (CRNGT) on the output of NDRNG to ensure that consecutive random numbers do not repeat.

7.2 Key Generation

The module does not support key generation, manual key entry or intermediate key generation output. In addition, the module does not produce key output outside its physical boundary. The keys can be entered or output from the module in plaintext form via API parameters, to and from the calling application only.

7.3 Key/CSP Storage

All keys and CSPs keys are provided to the module by the calling process and are destroyed when released by the appropriate API function calls. The module does not perform persistent storage of keys.

7.4 Key/CSP Zeroization

The application is responsible for calling the appropriate destruction functions from the Kernel Crypto API. The destruction functions then overwrite the memory occupied by keys with zeros and deallocates the memory with the free() call. In case of abnormal termination, the keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.

7.5 Key Establishment

The module provides AES key wrapping per [SP800-38F] and RSA key wrapping or encapsulation as allowed by [FIPS140-2_IG] D.9.

The module provides approved key transport methods as permitted by IG D.9. These key transport methods are provided either by using an approved key wrapping algorithm, an approved authenticated encryption mode, or a combination method of AES encryption and HMAC-SHA-1 or HMAC-SHA-2.

Table 6 and Table 7 specify the key sizes allowed in the FIPS mode of operation. According to “Table 2: Comparable strengths” in [SP800-57], the key sizes of AES key wrapping and RSA key wrapping provide the following security strengths:

- AES-KW key wrapping; key establishment methodology provides between 128 and 256 bits of encryption strength.
- AES-GCM and AES-CCM authenticated encryption key wrapping: key establishment methodology provides between 128 and 256 bits of encryption strength.
- Combination method of approved AES block mode (CTR, CBC, ECB) and message authentication code (HMAC-SHA-1/SHA-2) key wrapping: key establishment methodology provides between 128 and 256 bits of encryption strength.
- RSA key wrapping: key establishment methodology provides between 112 and 256 bits of encryption strength.

8 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms listed in Table 2 have been tested and found to conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, FCC PART 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (i.e., Business use). These devices are designed to provide reasonable protection against harmful interference when the devices are operated in a commercial environment.

9 Self-Tests

9.1 Power-Up Self-Tests

The module performs power-up tests when the module is loaded into memory, without operator intervention. Power-up tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

While the module is executing the power-up tests, services are not available, and input and output are inhibited. The module will not return the control to the calling application until the power-up tests are completed successfully.

The module verifies its integrity through the following mechanisms:

- All kernel object (*.ko) files are signed with a RSA and SHA2-512. Before these kernel objects are loaded into memory, the module performs RSA signature verification by using the RSA public key from the X.509 certificates that are compiled into the module's binary. If the signature cannot be verified, the kernel panics to indicate that the test fails and the module enters the error state.
- The integrity of the static kernel is ensured with the HMAC-SHA2-512 value stored in the .hmac file that was computed at build time. At run time, the module invokes the sha512hmac utility to calculate the HMAC value of the static kernel binary file, and then compares it with the pre-stored one. If the two HMAC values do not match, the kernel panics to indicate that the test fails, and the module enters the error state.
- The Integrity of the sha512hmac utility (i.e., /usr/bin/sha512hmac) is ensured with the HMAC-SHA2-512 value stored in the .hmac file (i.e. /usr/bin/.sha512hmac.hmac) that was computed at build time. At run time, the utility itself calculates the HMAC value of the utility, and then compares it with the pre-stored one. If the two HMAC values do not match, the kernel panics to indicate that the test fails, and the module enters the error state. The HMAC-SHA2-512 algorithm is provided by the bound NSS module and is KAT tested before the NSS module makes itself available to the sha512hmac application.

Both the RSA signature verification and HMAC-SHA2-512 algorithms are approved algorithms implemented in the Kernel Crypto API module and the bound NSS module.

Table 10 details the self-tests that are performed on the FIPS-approved cryptographic algorithms supported in the FIPS-approved mode of operation, using the Known-Answer Tests[9] (KATs) and Pairwise Consistency Tests (PCTs). For algorithms that have more than one implementation in the module (per Table 6), the module performs the self-tests independently for each of these implementations.

Table 10: Self-tests.

Algorithm	Test
AES	• KAT (CBC) with 128/192/256-bit keys, encryption and decryption • KAT (GCM) with 128/192/256-bit key, encryption and decryption • KAT (CMAC) with 128/256-bit keys, encryption • KAT (XTS) with 128/256-bit keys, encryption and decryption
Triple-DES	• KAT (CBC) with 192-bit key, encryption and decryption • KAT (CMAC) with 192-bit key, encryption

[9] The module also implements Diffie-Hellman and EC Diffie-Hellman "Z" computation KATs. These algorithms are non-approved for this module.

Algorithm	Test
RSA	• KAT RSA PKCS#1v1.5 signature verification with 2048-bit key and using SHA2-256 and SHA2-512 • KAT RSA encryption and decryption primitives with 2048-bit key
DRBG	• KAT (CTR) using AES-128/256 with DF, without PR • KAT (HMAC) using SHA2-256 with and without PR • KAT (Hash) using SHA2-256 with and without PR
HMAC	• KAT HMAC-SHA-1 • KAT HMAC-SHA2-224 • KAT HMAC-SHA2-256 • KAT HMAC-SHA2-384 • KAT HMAC-SHA2-512 • KAT HMAC-SHA3-224 • KAT HMAC-SHA3-256 • KAT HMAC-SHA3-384 • KAT HMAC-SHA3-512
SHS	• KAT SHA-1 • KAT SHA2-224 • KAT SHA2-256 • KAT SHA2-384 • KAT SHA2-512
SHA3	• KAT SHA3-224 • KAT SHA3-256 • KAT SHA3-384 • KAT SHA3-512
Module Integrity	• HMAC-SHA2-512 provided by Amazon Linux 2 NSS Crypto Module

9.2 Conditional Self-Tests

Conditional tests are performed during operational state of the module when the respective crypto functions are used. If any of the conditional tests fails, module transitions to error state.

Table 11 lists the conditional self-tests performed by the functions.

Table 11: Conditional self-tests.

Algorithm	Test
NDRNG	Continuous Random Number Generator Test (CRNGT)

9.3 On-Demand self-tests

The module provides the Self-Test service to perform self-tests on demand. On demand self-tests can be invoked by powering-off and reloading the module. This service performs the same cryptographic algorithm tests executed during power-up. During the execution of the on-demand self-tests, cryptographic services are not available, and no data output or input is possible.

10 Guidance

This section provides guidance for the Crypto Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

10.1 Crypto-Officer Guidance

The RPM files containing the FIPS validated module referenced in Section 2.3 must be installed according to this guidance.

The bound NSS module shall be installed according to its own documentation and Security Policy.

For proper operation of the in-module integrity verification, the prelink shall be disabled. This can be done by setting PRELINKING=no in the /etc/sysconfig/prelink configuration file. If the libraries were already prelinked, the prelink shall be undone on all the system files using the 'prelink –u -a' command.

To configure the operating environment to support FIPS perform the following steps:

1. Install the dracut-fips package.

   x86_64:

   ```
   # yum install dracut-fips-033-535.amzn2.1.3.x86_64.rpm
   ```

   aarch64:

   ```
   # yum install dracut-fips-033-535.amzn2.1.3.aarch64.rpm
   ```

2. Recreate the INITRAMFS image.

   ```
   # dracut -f
   ```

After regenerating the initramfs, the Crypto Officer shall append the following string to the kernel command line by changing the setting in the boot loader:

```
fips=1
```

If /boot or /boot/efi resides on a separate partition, the kernel parameter boot=<partition of /boot or /boot/efi> must be supplied. The partition can be identified with the command "df /boot" or "df /boot/efi".

For example:

```
$ df /boot
Filesystem  1K-blocks   Used  Available   Use%  Mounted on
/dev/sda1   233191      30454       190296      14%   /boot
```

The partition of /boot is located on /dev/sda1 in the example above. Therefore, the following string needs to be appended to the kernel command line:

```
"boot=/dev/sda1"
```

When supporting other formats such as boot=UUID/LABEL, please refer to the FIPS section of the 'dracut.cmdline' man page.

Reboot to apply these settings. After the reboot, the file /proc/sys/crypto/fips_enabled will contain 1. If the file does not exist or does not contain “1”, the operational environment is not configured to support FIPS and the module will not operate as a FIPS validated module.

After performing the configuration described above, the Crypto Officer shall proceed for module installation with the version of the RPM package listed in Section 2.3. The integrity of the RPM is automatically verified during the installation of the modules and the Crypto Officer shall not install the RPM file if the RPM tool indicates an integrity error

10.2 User Guidance

For detailed description of the Linux Kernel Crypto API, please refer to the user documentation [KC API Architecture].

In order to run in FIPS mode, the module must be operated using the FIPS Approved services, with their corresponding FIPS Approved and FIPS allowed cryptographic algorithms provided in this Security Policy (see section 4.2). In addition, key sizes must comply with [SP800-131A].

10.2.1 AES GCM IV

In case the module's power is lost and then restored, the key used for the AES-GCM encryption or decryption shall be redistributed.

When a GCM IV is used for encryption, the module complies with IG A.5 Scenario 2 [FIPS140-2_IG], in which the GCM IV is generated internally at its entirety randomly.

When a GCM IV is used for decryption, the responsibility for the IV generation lies with the party that performs the AES-GCM encryption therefore there is no restriction on the IV generation.

10.2.2 Triple-DES Data Encryption

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES (64-bit) blocks, in accordance to [SP800-67] and IG A.13 in [FIPS140-2-IG]. The user of the module is responsible for ensuring the module's compliance with this requirement.

10.2.3 AES-XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E]. In addition, the length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks, that is, 16 MiB of data.

In addition, to meet the requirement in [FIPS140-2_IG] A.9, the module implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

Note: AES-XTS shall be used with 128 and 256-bit keys only. AES-XTS with 192-bit keys is not an Approved service.

10.2.4 Key Usage and Management

In general, a single key shall be used for only one purpose (e.g., encryption, integrity, authentication, key wrapping, random bit generation, or digital signatures) and be disjoint between the modes of operations of the module. Thus, if the module is switched between its FIPS mode and non-FIPS mode or vice versa, the following procedures shall be observed:

- The DRBG engine shall be reseeded.
- CSPs and keys shall not be shared between security functions of the two different modes.

10.3 Handling Self-Test Errors

When the module fails any self-test, it will panic the kernel and the operating system will not load. Errors occurred during the self-tests transition the module into the error state. The only way to recover from this error state is to reboot the system. If the failure persists, the module must be reinstalled by the Crypto Officer following the instructions as specified in section 10.1.

The kernel dumps self-test success and failure messages into the kernel message ring buffer. The user can use **dmesg** to read the contents of the kernel ring buffer. The format of the ring buffer (dmesg) output for self-test status is:

```
alg: self-tests for %s (%s) passed
```

Typical messages are similar to "alg: self-tests for xts(aes) (xts(aes)) passed" for each algorithm/sub-algorithm type.

The only way to recover from the error state is to reload the module and restart the application. If failures persist, the module must be reinstalled.

11 Mitigation of Other Attacks

The module does not implement mitigation of other attacks.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Program Interface
CAVP	Cryptographic Algorithm Validation Program
CAVS	Cryptographic Algorithm Validation System
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CRNGT	Continuous Random Number Generator Test
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DF	Derivation Function
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
EMI/EMC	Electromagnetic Interference/Electromagnetic Compatibility
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
IG	Implementation Guidance
KAT	Known Answer Test
KW	Key Wrap
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NDRNG	Non-Deterministic Random Number Generator
PAA	Processor Algorithm Acceleration
PCT	Pair-wise Consistency Test
PR	Prediction Resistance
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSSE3	Supplemental Streaming SIMD Extensions 3
XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing

Appendix B. Algorithm Implementations

Table 12 describes the names utilized in the algorithm certificates and the implementations to which they refer for the test platforms. For instance, CAVP Cert. C918 refers to shaavx, which is the SHA algorithms implemented using AVX instructions.

Table 12: Algorithm implementations and their names in the CAVP certificates.

Cert.	Name	Description
C918	shaavx	SHA implementation using AVX instructions
C919	shaavx2	SHA implementation using AVX2 instructions
C920	shagen	SHA implementation using generic C
C921	shassse3	SHA implementation using SSSE3 instructions
C911	aesasm	AES implementation using assembler, and GCM with external IV
C912	aesasm_iiv	AES implementation using assembler, and GCM with internal IV
C913	aesgen	AES implementation using C, and GCM with external IV
C914	aesgen_iiv	AES implementation using C, and GCM with internal IV
C915	aesni	AES implementation with AESNI, and GCM with external IV
C917	aesni_iiv	AES implementation using AESNI, and GCM with internal IV
C923	TDES	C implementations

Appendix C. References

FIPS140-2 **FIPS PUB 140-2 - Security Requirements For Cryptographic Modules**
May 2001
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS140-2_IG **Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program**
October 23, 2019
http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
July 2013
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

KC API Architecture **Kernel Crypto API Architecture**
2016
http://www.chronox.de/crypto-API/crypto/architecture.html

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

SP800-38A **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
December 2001
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

SP800-38B **NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication**
May 2005
http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

SP800-38C **NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality**
May 2004
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-67	**NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** January 2012 http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf
SP800-90A	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-131A	**NIST Special Publication 800-131A Revision 2- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf