Juniper Networks, Inc.

Junos® OS Evolved MACsec Cryptographic Library

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 1.2

Last update: 2026-02-26

Prepared by:

Prepared for:

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

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA 94089
www.juniper.net

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 1.2 of the Junos® OS Evolved MACsec Cryptographic Library. 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 a Security Level 1 software module.

This Security Policy has a one-to-one mapping to SP 800-140B starting with section B.2.1 named "General" that maps to section 1 in this document and ending with section B.2.12 named "Mitigation of other attacks" that maps to section 12 in this document.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

1.3 Additional Information

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 Description

Purpose and Use:

The Junos® OS Evolved MACsec Cryptographic Library (hereafter referred to as "the module") is a software module.

The module is composed by a shared library in software, which provides cryptographic services for key wrapping and random number generation.

The module is also bound to the following cryptographic modules:

- Junos® OS Evolved Kernel Cryptographic Module Version 2.0 (validated under FIPS certificate #5399), which provides the integrity check utility that the module uses to check the integrity of the module's software component, and the SP 800-90A compliant random number generation implementation used for the random number generation service.
- Junos® OS Evolved OpenSSL Cryptographic Module Version 3.0.8 (validated under FIPS certificate #5400), which provides the algorithm implementation for integrity test.

Sections of this Security Policy which refer to information from the bound module, also known as the Existing Validated Module (or EVM) are marked by [EVM] as per IG 1.A Resolution 5.

Module Type: : Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

Figure 1 shows a block diagram that represents the design of the module, and its interfaces with the operational environment.

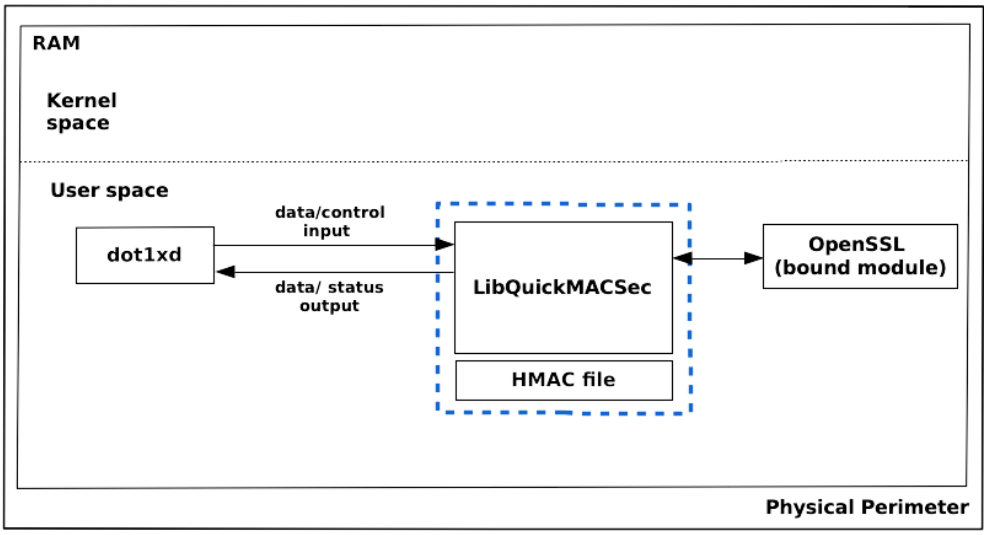

Figure 1: Block Diagram

Tested Operational Environment's Physical Perimeter (TOEPP):

The tested operating environment's physical perimeter is the general-purpose computer on which the module is running.

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
/usr/lib64/libquicksec-macsec.so.1	1.2	N/A	HMAC-SHA2-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Junos® OS Evolved version 22.4	Juniper Networks® Packet Transport Router Model PTX10001-36MR	Intel® Xeon® D-2163IT	No	N/A	1.2

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

2.3 Excluded Components

The module does not have any excluded components.

2.4 Modes of Operation

Modes List and Description:

The module supports only the approved mode of operation. When the module starts up successfully, after passing all the pre-operational self-tests and conditional cryptographic algorithm self-tests (CASTs), the module is operating in the approved mode of operation.

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Service API returns 0

Table 4: Modes List and Description

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CMAC	A4156	Direction - Generation, Verification Key Length - 128, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 8-524288 Increment 8	SP 800-38B
AES-ECB	A4156	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38A
AES-KW	A4156	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 256	SP 800-38F

Algorithm	CAVP Cert	Properties	Reference
		Payload Length - Payload Length: 128-4096 Increment 128	

Table 5: Approved Algorithms -

[EVM]

Algorithm	CAVP Cert	Properties	Reference
HMAC DRBG	A3605	Prediction Resistance - No Mode - SHA2-256	SP 800-90A Rev. 1
HMAC-SHA2-256	A4246	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1

Table 6: Approved Algorithms - [EVM]

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Random number generation	DRBG	[EVM] Random number generation	Hash:SHA2-256 Provided by:Bound Kernel module	HMAC DRBG: (A3605)
Message authentication	MAC	Message authentication		AES-CMAC: (A4156)
Symmetric Encryption	BC-UnAuth	Prerequisite algorithm for AES-CMAC		AES-ECB: (A4156)
Key wrapping	BC-Auth	Key wrapping		AES-KW: (A4156)
Key unwrapping	BC-Auth	Key unwrapping		AES-KW: (A4156)
Integrity test	MAC	[EVM] MAC used solely for integrity testing	Provided by:Bound OpenSSL and Kernel modules	HMAC-SHA2-256: (A4246)

Table 7: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 Key Wrapping

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

2.8 RBG and Entropy

[EVM] The module does not implement any random bit generator. Instead, the module obtains random bits from the Random Number Generation (RNG) service provided by the bound kernel module "Junos OS Evolved Kernel Cryptographic Module".

The bound kernel module uses the Kernel CPU Time Jitter RNG (validated under ESV certificate E50) as an entropy source to seed the DRBG. The entropy source is SP 800-90B compliant and is implemented within the bound kernel module. It resides within the TOEPP of the module and complies with IG 9.3.A Scenario 1b.

The DRBG implemented in the bound kernel module is seeded with 384 bits of seed material (corresponding to 358 bits of entropy) obtained from the entropy source. During reseeding, the DRBG obtains 256 bits of seed material (corresponding to 239 bits of entropy) from the entropy source. The 239 bits of entropy used for reseeding are less than the highest SSP strength generated by the module of 256 bits.

The module generates random strings whose strength is modified by available entropy.

2.9 Key Generation

The module does not provide key generation mechanisms.

2.10 Key Establishment

The module offers authenticated encryption and decryption as a service using AES-KW and AES-KWP. These algorithms can be used to wrap SSPs with a security strength of 128, 192, or 256 bits, depending on the wrapping key size.

2.11 Industry Protocols

The module does not implement any industry-protocol-specific security function or service.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters for data.
N/A	Data Output	API output parameters for data.
N/A	Control Input	API function calls, API input parameters for control input.
N/A	Status Output	API return codes, API output parameters for status output.

Table 8: Ports and Interfaces

The module does not implement a control output interface.

All data output via data output interface is inhibited when the module is performing the pre-operational self-test or zeroization or when the module enters the error state.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication.

4.2 Roles

The module supports the Crypto Officer role only. This sole role is implicitly assumed by the operator of the module when performing a service. The module does not support concurrent operators.

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

Table 9: Roles

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key wrapping	Wraps AES key	ssh_aes_key_wrap() returns 0	Key to wrap, Key wrapping key	Wrapped key	Key wrapping	Crypto Officer - AES key: W,E
Key unwrapping	Unwraps AES key	ssh_aes_key_unwrap() returns 0	Wrapped key, Key wrapping key	Unwrapped key	Key unwrapping	Crypto Officer - AES key: W,E
Message authentication	Compute a MAC tag	ssh_mac*() functions return 0	Message, AES-CMAC key	MAC tag	Message authentication	Crypto Officer - AES key: W,E
[EVM] Random number generation	Generate random bytes (provided by bound	ssh_random_get_bytes() returns SSH_CRYPTO_OK	Entropy of requested bitstring	Random bytes	Random number generation	Crypto Officer - [EVM] DRBG entropy input string:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	Kernel module)					G,W,E - [EVM] HMAC_DRBG internal state: G,W,E - [EVM] HMAC_DRBG seed: G,E
Show status	Return the module status	N/A	N/A	Module Status	None	Crypto Officer
Self-test	Perform the CASTs and the integrity test (integrity test provided by bound OpenSSL and Kernel modules)	N/A	N/A	Pass/fail	Message authentication Symmetric Encryption Key wrapping Key unwrapping Integrity test None	Crypto Officer
Zeroization	Zeroize all SSPs	N/A	N/A	Success/fail	Message authentication Key wrapping Key unwrapping	Crypto Officer - AES key: Z
Show module name and version	Return module name and version information	N/A	None	Module name and version	None	Crypto Officer

Table 10: Approved Services

4.4 Non-Approved Services

The module does not implement any non-approved services.

4.5 External Software/Firmware Loaded

The module does not support the loading of external software/firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is ensured with the HMAC-SHA2-256 value stored in the corresponding /usr/lib64/.libquicksec-macsec.so.1.hmac file that is computed at build time. During Pre-Operational Self-Tests, the module invokes the fips_chk_hmac utility provided by the bound Kernel module (relying on the HMAC service provided by the bound OpenSSL module) to calculate the HMAC value of the shared library, and then compares it with the pre-stored one. If the two HMAC values do not match, the test fails and the module enters the error state.

The integrity of the fips_chk_hmac utility itself is performed before the integrity tests of the module, and ensured with the HMAC-SHA2-256 value stored in the corresponding .hmac file that is computed at build time of the utility. The fips_chk_hmac utility calculates the HMAC value, and then compares it with the pre-stored value. If the two HMAC values do not match, the test fails and the module enters the error state.

The HMAC key is stored within the fips_chk_hmac utility binary.

5.2 Initiate on Demand

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

6 Operational Environment

6.1 Operational Environment Type and Requirements

The module operates in a modifiable operational environment per FIPS 140-3 level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in Table 3.

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

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

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11. If properly installed, 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.

Instrumentation tools like the ptrace system call, gdb and strace utilities, userspace 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-tested 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 11: 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	SFI or Algorithm
API input parameters	Operator calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	

Table 12: SSP Input-Output Methods

The module only supports SSP entry and output to and from the calling application running on the same operational environment. This corresponds to manual distribution (MD), electronic entry/output (EE) ("CM Software to/from App via TOEPP Path") per FIPS 140-3 IG 9.5.A Table 1.

SSPs can be entered into the module via API input parameters, when required by a service. SSPs can also be output from the module via API output parameters, immediately after key wrapping (in encrypted form) and key unwrapping (in plaintext form) services.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Wipe and Free memory block allocated	Zeroizes the SSPs contained within the cipher handle.	Memory occupied by SSPs is overwritten with zeroes and then it is released, which renders the SSP values irretrievable. The completion of the zeroization	By calling the cipher related zeroization API function: macsec_util_zeroize_key() for AES keys

Zeroization Method	Description	Rationale	Operator Initiation
		routine indicates that the zeroization procedure succeeded.	
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
Module Reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed.	By unloading and reloading the module

Table 13: SSP Zeroization Methods

For SSPs that are input or output through the services, it is the responsibility of the calling application to zeroize them by calling macsec_util_zeroize_key() once they are no longer utilized.

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	AES key	128, 256 bits - 128, 256 bits	Symmetric key - CSP			Message authentication Symmetric Encryption Key wrapping Key unwrapping
[EVM] DRBG entropy input string	Entropy input string for DRBG in bound module (IG D.L compliant)	256, 384 bits - 238, 358 bits	Entropy Input - CSP			Random number generation
[EVM] HMAC_DRBG seed	DRBG seed derived from entropy input in bound module (IG D.L compliant)	512 bits - 256 bits	Seed - CSP	Random number generation		Random number generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
[EVM] HMAC_DRBG internal state	Internal state of DRBG in bound module (IG D.L compliant)	1024 bits - 256 bits	Seed - CSP	Random number generation		Random number generation

Table 14: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters	RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Module Reset	
[EVM] DRBG entropy input string		RAM:Plaintext	From service invocation to service completion	Automatic Module Reset	[EVM] HMAC_DRBG seed:Derives
[EVM] HMAC_DRBG seed		RAM:Plaintext	From service invocation to service completion	Automatic Module Reset	[EVM] DRBG entropy input string:Derived From [EVM] HMAC_DRBG internal state:Derives
[EVM] HMAC_DRBG internal state		RAM:Plaintext	From service invocation to service completion	Wipe and Free memory block allocated Automatic Module Reset	[EVM] HMAC_DRBG seed:Derived From

Table 15: SSP Table 2

10 Self-Tests

The module performs the pre-operational self-tests automatically when the module is loaded into memory. These self-tests ensure that the module is not corrupted and that the cryptographic algorithm used in the integrity test works as expected. Conditional cryptographic algorithm self-tests are performed by the module when the library is initialized by the calling application, verifying that all cryptographic algorithms work as expected before their first use.

While the module is executing the pre-operational and the conditional cryptographic algorithms self-tests, services are not available, and input and output are inhibited. The module is not available for use by the calling application until the self-tests are completed successfully. If any of the self-tests fails, an error message is returned and the module transitions to error state.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4246)	SHA2-256	MAC tag verification	SW/FW Integrity	Module becomes operational	[EVM] Integrity test for module

Table 16: Pre-Operational Self-Tests

The module performs a pre-operational software integrity test automatically when the module is powered on before the module transitions into the operational state. The details on the integrity test are specified in Section 5.1.

10.2 Conditional Self-Tests

Table 17 lists the cryptographic algorithm self-tests (CASTs). The CASTs include the KATs for the integrity mechanism that is run prior to performing the integrity test. The details of the integrity test are provided in Section 5.1.

Each KAT includes comparison of the calculated output with the expected known answer, hard coded as part of the test vectors used in the test. Data output through the data output interface is inhibited during the self-tests. If the values do not match, the KAT fails and the module transitions to the error state.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-KW Encrypt	128/256-bit keys, encrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before module becomes operational

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-KW Decrypt	128/256-bit keys, decrypt	KAT	CAST	Module becomes operational	Symmetric operation	Test runs at power-on before module becomes operational
AES-CMAC (A4156)	128-bit key, encrypt	KAT	CAST	Module becomes operational	Message authentication	Test runs at power-on before module becomes operational
HMAC-SHA2-256 (A4246)	SHA2-256	KAT	CAST	Module becomes operational	[EVM] Message authentication	Test runs at power-on before the integrity test
HMAC DRBG (A3605)	HMAC-SHA2-512 without prediction resistance	KAT	CAST	Module becomes operational	[EVM] SP 800-90A Rev. 1 (Instantiate, reseed, generate) health test	Test runs at power-on before module becomes operational

Table 17: Conditional Self-Tests

KATs for the HMAC and DRBG algorithms used in this module are performed by the respective bound modules.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4246)	MAC tag verification	SW/FW Integrity	On Demand	Manually

Table 18: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-KW Encrypt	KAT	CAST	On Demand	Manually
AES-KW Decrypt	KAT	CAST	On Demand	Manually
AES-CMAC (A4156)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A4246)	KAT	CAST	On Demand	Manually
HMAC DRBG (A3605)	KAT	CAST	On Demand	Manually

Table 19: Conditional Periodic Information

On-demand self-tests can be invoked by powering-off and reloading the module which cause the module to run the pre-operational and conditional cryptographic algorithms self-tests.

10.4 Error States

When the module fails any pre-operational or conditional self-test, the module will enter the Error state. Any further cryptographic operation is inhibited. The calling application can obtain the state with the return value of the API function used to initialize the module (after finishing self-tests), or by requesting the get status service (using a dedicated API function).
The Crypto Officer can recover from the Error state by restarting the hardware platform on which the module is running.

Name	Description	Conditions	Recovery Method	Indicator
Error	General-purpose error state	Failure of CAST Failure of integrity tests	Power cycle	Error message written in syslog, SSH_CRYPTO_LIBRARY_ERROR

Table 20: Error States

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The binaries of the module are contained in the base Junos Evolved installation image. The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated as a FIPS 140-3 validated module.

11.2 Administrator Guidance

In order to run in the Approved mode, the module must be operated using the approved services, with their corresponding approved and allowed cryptographic algorithms provided in this Security Policy. In addition, key sizes must comply with [SP800-131A Rev. 2].

The module is already pre-installed on the image file (junos-evo-install-ptx-fixed-x86-64-22.4R2.11-S1-EVO.iso). The crypto officer is responsible to verify the correct installation of the module by executing the following command:

```
cli show security macsec crypto version
```

Verify that the command returns the following name and version of the module:

```
Crypto library version : Junos OS Evolved MACsec Cryptographic Library, version 1.2
```

The Junos OS Evolved OpenSSL Cryptographic Module version 3.0.8 and the Junos OS Evolved Kernel Cryptographic Module are bound modules that shall also be installed and configured as described in section 11.1 of their corresponding Security Policies. The administrator shall follow the steps to install the modules and verify their version.

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory.

12 Mitigation of Other Attacks

The module does not implement any additional mitigation mechanism.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
API	Application Program Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
EE	Electronic Entry
FIPS	Federal Information Processing Standards Publication
HMAC	Hash Message Authentication Code
IG	Implementation Guidance
KAT	Known Answer Test
KW	Key Wrap
MAC	Message Authentication Code
MD	Manual Distribution
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
SSP	Sensitive Security Parameter

Appendix B. References

FIPS 140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS 140-3 IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** October 2022 https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
SP 800-38A	**Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://doi.org/10.6028/NIST.SP.800-38A
SP 800-38B	**Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://doi.org/10.6028/NIST.SP.800-38B
SP 800-38F	**Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://doi.org/10.6028/NIST.SP.800-38F
FIPS 198-1	**The Keyed-Hash Message Authentication Code (HMAC)** July 2008 https://doi.org/10.6028/NIST.FIPS.198-1
SP 800-90A Rev. 1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://doi.org/10.6028/NIST.SP.800-90Ar1