Junos® OS Evolved MACsec Cryptographic Library
VERSION 1.2

FIPS 140-3 Non-Proprietary Security Policy

Version 1.1

Last update: 2024-08-29

Prepared by:

atsec information security corporation

4516 Seton Center Parkway, Suite 250

Austin, TX 78759

www.atsec.com

Prepared for:

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 the [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.

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 How this Security Policy was Prepared

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

1.3 Security Levels

The following sections describe the cryptographic module and how it conforms to the FIPS 140-3 specification in each of the required areas.

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic Module Specification	1
3	Cryptographic Module Interfaces	1
4	Roles, Services and Authentication	1
5	Software/Firmware Security	1
6	Operational Environment	1
7	Physical Security	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 Security Level		1

Table 1 - Security Levels

2 CRYPTOGRAPHIC MODULE SPECIFICATION

2.1 Module Embodiment

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 #4776), which provides the integrity check utility that the module uses to check the integrity of the module's software component, and the SP800-90A 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 #4775), which provides the algorithm implementation for integrity test.

For the purpose of the FIPS 140-3 validation, the module is a software-only, multi-chip standalone cryptographic module validated at overall security level 1.

2.2 Module Design, Components, Versions

Figure 1 below shows the cryptographic boundary of the module, and its interfaces with the operational environment.

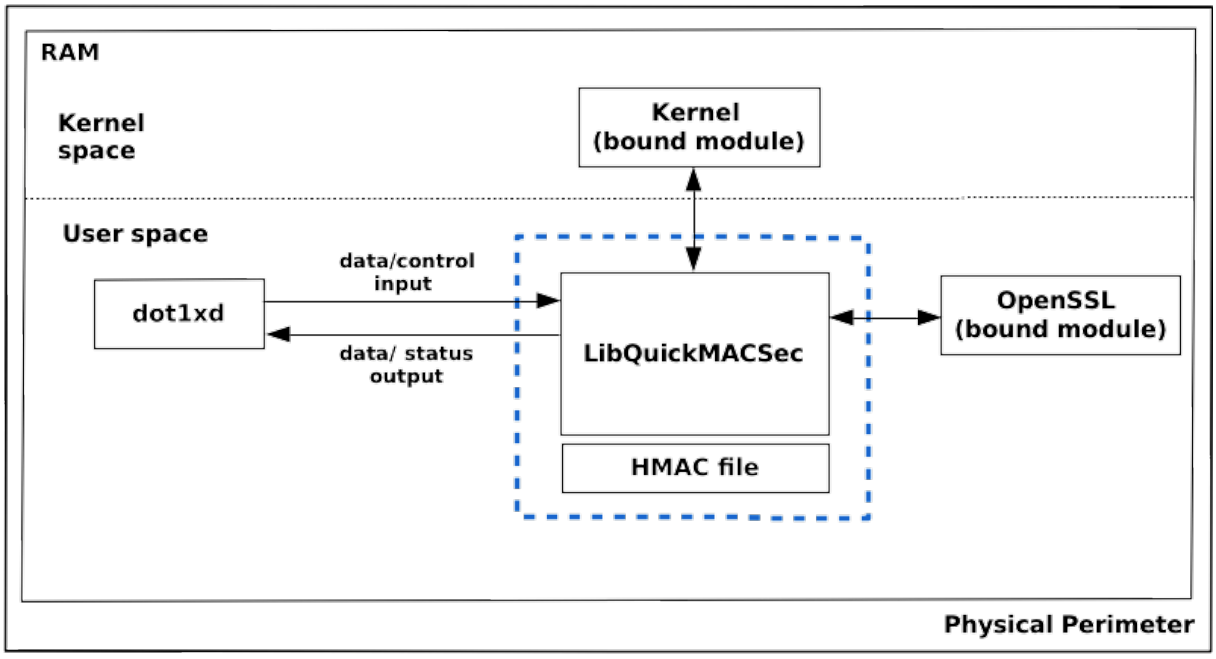

Figure 1 - Cryptographic Boundary

2.2.1 Module Components

Table 2 below enumerates the components that comprise the module with their location in the target platform.

Component Type	Version	Components	Description
Software	**1.2**	**/usr/lib64/libquicksec-macsec.so.1**	Libquick MACsec shared library
		/usr/lib64/.libquicksec-macsec.so.1.hmac	Integrity check HMAC value for the shared library

Table 2 - Cryptographic Module Components

2.3 Tested Operational Enviroments

The module has been tested on the following platforms with the corresponding module variants and configuration options:

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Junos® OS Evolved 22.4	Juniper Networks® Packet Transport Router Model PTX10001-36MR	Intel® Xeon® D-2163IT	Without AES-NI

Table 3 - Tested Operational Environments

2.4 Modes of Operation

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.

The module does not implement a degraded mode of operation.

2.5 Security Functions

2.5.1 Approved Algorithms

Table 4 below lists all security functions of the module, including specific strengths employed for approved services.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
#A4156[1]	AES [FIPS197], [SP800-38B]	ECB	128, 256-bit keys with key strength of 128 or 256 bits	Prerequisite algorithm for AES-CMAC.
#A4156	AES [FIPS197], [SP800-38B]	CMAC	128, 256-bit keys with key strength of 128 or 256 bits	Message authentication code (MAC) generation, Message authentication code (MAC) verification

[1] Not all algorithms tested in the CAVP certificate are used by the module.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
#A4156	AES [FIPS197], [SP800-38F]	KW	128, 256-bit keys with key strength of 128 or 256 bits	Key wrapping, Key unwrapping

Table 4 - Approved Algorithms provided by the module

Table 5 below lists the approved algorithms provided by the bound OpenSSL module that are used by the module to perform integrity tests.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
#A4246, #A4247, #A4248, #A4249	HMAC [FIPS198-1], [SP800-38D]	SHA2-256	256-bit key with key strength of 256 bits	Integrity test

Table 5 - Approved Algorithms provided by the bound OpenSSL module

Table 5 below lists the approved algorithms provided by the bound Kernel module that are used by the module to obtain random numbers.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use
#A3599 #A3600 #A3601 #A3603 #A3604 #A3605	DRBG [SP800-90Arev1]	HMAC with SHA2-256	Key strength of 256 bits	Random number generation

Table 6 - Approved Algorithms provided by the bound Kernel module

2.5.2 Non-Approved Algorithms Allowed in the Approved Mode of Operation

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

2.5.3 Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed

The module does not implement non-approved algorithms.

2.5.4 Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

The module does not implement non-approved algorithms.

3 CRYPTOGRAPHIC MODULE PORTS AND INTERFACES

Ports and interfaces implemented are shown in the following table. The Control Output interface is omitted because the module does not implement it.

All data output via data output interface is inhibited when the module is performing self-tests or zeroization, or when the module is in the error state.

Logical Interface	Physical Port	Data that passes over port/interface
Data Input	None	API input parameters for data.
Data Output	None	API output parameters for data.
Control Input	None	API function calls, API input parameters for control input.
Status Output	None	API return codes, error messages.

Table 7 - Ports and Interfaces

4 ROLES, SERVICES AND AUTHENTICATION

4.1 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.

Role	Service	Input	Output
Crypto Officer (CO)	Key unwrapping	Wrapped key, Key wrapping key	Unwrapped key
	Key wrapping	Key to be wrapped, Key wrapping key	Wrapped key
	Message authentication code (MAC) generation	Message, AES key	MAC tag
	Message authentication code (MAC) verification	Message, AES key, MAC tag	Success/Failure
	Random Number Generation	Number of bytes	Random number
	Show module name and version	None	Module name and version
	Show status	None	SSH_CRYPTO_OK , SSH_CRYPTO_LIBRARY_ERROR
	Self-tests	None	Return code and log message
	Zeroization	Context containing SSPs	None

Table 8 - Role, Service Commands, Input and Output

4.2 Authentication

The module does not implement authentication.

4.3 Services

The module only provides approved services, which are shown in Table 9.

4.3.1 Service Indicator

The module provides an approved service indicator as specified in the “Indicator” column in Table 9.

4.3.2 Approved Services

The table below shows the services available in the Approved mode. For each service, the table lists the associated cryptographic algorithm(s), the role to perform the service, the cryptographic keys or SSPs involved, and their access type(s). The following convention is used to specify access rights to a SSP:

- **G = Generate**: The module generates or derives the SSP.
- **R = Read**: The SSP is read from the module (e.g. the SSP is output).
- **W = Write**: The SSP is updated, imported, or written to the module.
- **E = Execute**: The module uses the SSP in performing a cryptographic operation.
- **Z = Zeroize**: The module zeroizes the SSP.
- **N/A**: the calling application does not access any SSP or key during its operation.

The details of the approved cryptographic algorithms including the CAVP certificate numbers can be found in Section 2.5.1.

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Cryptographic Module Services						
Key unwrapping	Unwraps AES key	AES-KW	AES key	CO	W, E	ssh_aes_key_unwrap() returns SSH_CRYPTO_OK
Key wrapping	Wraps AES key	AES-KW	AES key	CO	W, E	ssh_aes_key_wrap() returns SSH_CRYPTO_OK
Message authentication code (MAC) generation	Compute a MAC tag	AES CMAC	AES key	CO	W, E	ssh_mac*() functions return SSH_CRYPTO_OK
Message authentication code (MAC) verification	Verify a MAC tag	AES CMAC	AES key	CO	W, E	ssh_mac*() functions return SSH_CRYPTO_OK
Random number generation	Generate random bytes	HMAC_DRBG from Kernel bound module	Entropy input DRBG seed DRBG internal state (V, C)	CO	W, E E, G W, E, G	ssh_random_get_bytes() returns SSH_CRYPTO_OK
Other services						
Show status	Return the module status	N/A	None	CO	N/A	None
Self-test	Perform the CASTs and the integrity test	AES-KW, AES-CMAC	None	CO	N/A	None
Zeroization	Zeroize all SSPs	N/A	All SSPs	CO	Z	None
Show module name and version	Return module name and version information	N/A	None	CO	N/A	None

Table 9 - Approved Services

4.3.3 Non-Approved Services

The module does not implement non-approved services.

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 for the shared library. During Pre-Operational Self-Tests, the module invokes the fips_chk_hmac utility provided by the bound Kernel module (and whose cryptographic functionality is provided by the bound OpenSSL module) to calculate the HMAC value of the shared library, and then compares it with the prestored value. 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.

5.2 On-demand Integrity Tests

Integrity tests are performed as part of the Pre-Operational Self-Tests, and can be invoked by reloading the module which will cause the module to run the power-up tests again.

5.3 Executable Code

The module consists of the software component that implements the cryptographic algorithms, in the form of a shared library as stated in Table 2.

6 OPERATIONAL ENVIRONMENT

6.1 Applicability

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.

6.2 Policy and Requirements

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.

The module does not have the capability of loading software or firmware from an external source.

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

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

9 SENSITIVE SECURITY PARAMETER MANAGEMENT

Table 10 summarizes the Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module.

Key /SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import /export	Establishment	Storage	Zeroization	Use & Related Keys
AES key	128, 256 bits	AES-KW, AES-CMAC #4156	N/A	**Import**: CM from TOEPP Path. Passed into the module via API parameters in plaintext (P) form. **Export:** N/A	N/A	RAM	macsec_util_zeroize_key()	**Use:** Key wrapping; Key unwrapping; Message authentication code (MAC) generation; Message authentication code (MAC) verification; **Related SSPs:** none

Table 10 – SSPs

The bound Kernel module is used for obtaining random numbers using a DRBG implemented in the Kernel; all CSPs related to the DRBG are managed internally by the bound module and do not constitute CSPs for this module.

9.1 Random Bit Generators

The module provides a random number generator service using the Deterministic Random Bit Generator (DRBG) provided by the bound Kernel cryptographic module, which is based on [SP800-90Arev1]. The bound Kernel cryptographic module provides the DRBG using HMAC_DRBG with SHA2-256 without prediction resistance, and uses the Kernel CPU Time Jitter RNG as an entropy source to seed the DRBG.

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

9.2 SSP Generation

The module does not support any key generation or derivation methods.

9.3 Key Agreement

The module does not support any approved key agreement methods.

9.4 Key Transport

The module the following key transport mechanisms:

- AES key wrapping using AES-KW.

According to Table 2: Comparable strengths in [SP 800-57rev5], the key sizes of AES provide the following security strength in the approved mode of operation:

- AES key wrapping in KW mode provides 128 or 256 bits of encryption strength.

9.5 SSP Entry and Output

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.

9.6 SSP Storage

The module does not perform persistent storage of SSPs. The SSPs are temporarily stored in the RAM in plaintext form; the module does not support storage of any cryptographically protected SSP.

SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

9.7 SSP Zeroization

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The module calls internally the appropriate zeroization functions (i.e. memset) before returning to the calling application. The zeroization functions overwrite the memory occupied by SSPs with "zeros" and deallocate the memory with the regular memory deallocation operating system call. The completion of a zeroization routine(s) will indicate that a zeroization procedure succeeded.

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.

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.

See Section 10.4 for descriptions of possible self-test errors and recovery procedures.

10.1 Pre-operational 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 Tests

Table 11 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	Power-Up Tests
AES	• KAT AES KW mode with 128 and 256 bit keys, encryption and decryption (separately tested).
CMAC	• KAT AES CMAC with 128-bit key, MAC generation.

Table 11 – Conditional Cryptographic Algorithms Self-Tests performed by the module

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

10.2.1 Pairwise Consistency Tests

The module does not perform any pairwise consistency test.

10.3 Periodic/On-demand Self-tests

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 State

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.

Error State	Cause of Error	Status Indicator
Error state	Failure of CAST	Error message written in syslog. SSH_CRYPTO_LIBRARY_ERROR
	Failure of integrity tests	Error message ”libquicksec_macsec Integrity check FAILED” is written in syslog. SSH_CRYPTO_LIBRARY_ERROR

Table 12 - Error States

11 LIFE-CYCLE ASSURANCE

11.1 Delivery and Operation

11.1.1 Module Installation

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.

11.1.2 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.

11.2 Crypto Officer 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 (see Section 4.3). In addition, key sizes must comply with [SP800-131A].

11.2.1 Verification of the Module Installation

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
```

12 MITIGATION OF OTHER ATTACKS

The module does not implement any mitigation mechanism.

13 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
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
GCM	Galois Counter Mode
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
PR	Prediction Resistance
PCT	Pair-wise Consistency Test
SHA	Secure Hash Algorithm
TOEPP	Tested Operational Environment's Physical Perimeter

14 APPENDIX B – REFERENCES

FIPS140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** October 2022 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.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
KERNEL-SP	**Junos® OS Evolved Kernel Cryptographic Module version 2.0 - FIPS 140-3 Non-Proprietary Security Policy** August 2023 https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp4776.pdf
OPENSSL-SP	**Junos® OS Evolved OpenSSL Cryptographic Module version 3.0.8 - FIPS 140-3 Non-Proprietary Security Policy** August 2023 https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp4775.pdf
SP800-38A	**NIST Special Publication 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
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
SP800-38F	**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

SP800-57	**NIST Special Publication 800-57 Part 1 Revision 4 - Recommendation for Key Management Part 1: General** January 2016 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.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 - Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP800-140B	**NIST Special Publication 800-140B - CMVP Security Policy Requirements** **March 2020** https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf