Apple Inc.

Apple corecrypto Module v11.1
[Intel, Kernel, Software]
FIPS 140-3 Non-Proprietary Security Policy

document version 1.4

November, 2022

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Trademarks

Apple's trademarks applicable to this document are listed in https://www.apple.com/legal/intellectual-property/trademark/appletmlist.html. Other company, product, and service names may be trademarks or service marks of others.

Table of Contents

List of Tables

1. General

This document is the non-proprietary FIPS 140-3 Security Policy for Apple corecrypto Module v11.1 [Intel, Kernel, Software] cryptographic module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for a Security Level 1 module.

This document provides all tables and diagrams (when applicable) required by NIST SP 800-140B. The column names of the tables follow the template tables provided in NIST SP 800-140B.

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

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic Module Specification	1
3	Cryptographic Module Interfaces	1
4	Roles, Services, and Authentication	1
5	Software/Firmware Security	1
6	Operational Environment	1
7	Physical Security	Not Applicable
8	Non-invasive Security	Not Applicable
9	Sensitive Security Parameter Management	1
10	Self-tests	1
11	Life-cycle Assurance	1
12	Mitigation of Other Attacks	Not Applicable

Table 1 - Security Levels

2. Cryptographic Module Specification

The Apple corecrypto Module v11.1 [Intel, Kernel, Software] cryptographic module (hereafter referred to as "the module") is a software module running on a multi-chip standalone general-purpose computing platform. The version of module is 11.1, written as v11.1. The module provides implementations of low-level cryptographic primitives to the Host OS's (macOS Big Sur 11.0.1) Security Framework and Common Crypto. The module has been tested by atsec CST lab on the following platforms with and without AES-NI:

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	macOS Big Sur 11.0.1	MacBook Air	Intel i5-8210Y (Amber Lake)	AES-NI
2	macOS Big Sur 11.0.1	MacBook Air	Intel i7-1060NG7 (Ice Lake)	AES-NI
3	macOS Big Sur 11.0.1	MacBook Pro	Intel i7-8850H (Coffee Lake)	AES-NI
4	macOS Big Sur 11.0.1	MacBook Pro	Intel i9-9880H (Coffee Lake)	AES-NI
5	macOS Big Sur 11.0.1	iMac Pro	Xeon W-2140B (Sky Lake)	AES-NI
6	macOS Big Sur 11.0.1	Mac Pro	Xeon W-3223 (Cascade Lake)	AES-NI

Table 2 - Tested Operational Environments

In addition to the platforms listed in Table 2, Apple Inc. has also tested the module on the following platforms and claims vendor affirmation on them:

#	Operating System	Hardware Platform	Processor	Release Year
1	macOS Big Sur 11.0.1	MacBook Pro	i5 (Ice Lake)	2020
2	macOS Big Sur 11.0.1	MacBook Pro	i5 (Coffee Lake)	2020, 2019, 2018
3	macOS Big Sur 11.0.1	MacBook Pro	i7 (Amber Lake)	2019, 2018
4	macOS Big Sur 11.0.1	MacBook Pro	i7 (Coffee Lake)	2020, 2019, 2018
5	macOS Big Sur 11.0.1	MacBook Pro	i7 (Ice Lake)	2020
6	macOS Big Sur 11.0.1	MacBook Pro	i9 (Coffee Lake)	2019, 2018
7	macOS Big Sur 11.0.1	MacBook Air	i5 (Ice Lake)	2020
8	macOS Big Sur 11.0.1	MacBook Air	i7 (Ice Lake)	2020
9	macOS Big Sur 11.0.1	MacBook Air	i5 (Amber Lake)	2019, 2018
10	macOS Big Sur 11.0.1	MacBook Air	i7 (Amber Lake)	2018
11	macOS Big Sur 11.0.1	Mac mini	i5 (Coffee Lake)	2018
12	macOS Big Sur 11.0.1	Mac mini	i7 (Coffee Lake)	2018
13	macOS Big Sur 11.0.1	iMac	i5 (Comet Lake)	2020
14	macOS Big Sur 11.0.1	iMac	i7 (Comet Lake)	2020

15	macOS Big Sur 11.0.1	iMac	i9 (Comet Lake)	2020
16	macOS Big Sur 11.0.1	iMac	i5 (Coffee Lake)	2019
17	macOS Big Sur 11.0.1	iMac	i7 (Coffee Lake)	2019
18	macOS Big Sur 11.0.1	iMac	i9 (Coffee Lake)	2019

Table 3 - Vendor Affirmed Operational Environments

The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

The table below lists all Approved or Vendor-affirmed security functions of the module, including specific key size(s) employed for approved services, and implemented modes of operation. The module is in the Approved mode of operation when the module utilizes the services that use the security functions listed in the table below. The Approved mode of operation is configured in the system by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved services listed in Table 8 - Non-Approved Services. If the device starts up successfully, then the module has passed all self-tests and is operating in the Approved mode.

CAVP Cert.	Algorithm and Standard	Mode / Method	Description / Key Size(s)	Use / Function
A943 (vng_asm) A945 (c_asm) A976 (c-aesni) A973 (vng_aesni)	CTR_DRBG [SP800-90A]	AES-128, AES-256 Derivation Function Enabled No Prediction Resistance	Key Length: 128, 256	Random Number Generation
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	HMAC_DRBG [SP800-90A]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 No Prediction Resistance	Key Length: 112 bits or greater	Random Number Generation
A945 (c_asm) A976 (c_aesni)	AES [FIPS 197] [SP 800-38A]	CBC, ECB, CFB128, CFB8, OFB, CTR	Key Length: 128, 192, 256	Symmetric Encryption and Decryption
A978 (asm_aesni) A977 (asm_x86)	AES [FIPS 197] [SP 800-38A] [SP 800-38E]	CBC, ECB, XTS	Key Length: 128, 192, 256 XTS (128 and 256-bits key size only)	Symmetric Encryption and Decryption
A943 (vng_asm) A973 (vng_aesni)	AES [FIPS 197] [SP 800-38A] [SP 800-38C] [SP 800-38D]	ECB CCM, CTR, GCM	Key Length: 128, 192, 256	Symmetric Encryption and Decryption
A945 (c_asm) A976 (c_aesni)	KTS (AES) [FIPS 197] [SP 800-38F]	AES- KW	Key Length: 128, 192, 256	Key Wrapping and Unwrapping
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	RSA [FIPS 186-4]	Key Generation (ANSI X9.31) (CKG using method in Sections 4 and 5.1 [SP 800-133])	Modulus: 2048, 3072, 4096	Asymmetric Key Generation
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	RSA [FIPS 186-4]	Signature Generation (PKCS#1 v1.5) and (PKCS PSS)	Modulus: 2048, 3072, 4096	Digital Signature Generation
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	RSA [FIPS 186-4]	Signature Verification (PKCS#1 v1.5) and (PKCS PSS)	Modulus: 1024, 2048, 3072, 4096	Digital Signature Verification
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	ECDSA ANSI X9.62 [FIPS 186-4]	Key Pair Generation (CKG using method in Sections 4 and 5.1 [SP 800-133])	Curve: P-224, P-256, P-384, P-521	Asymmetric Key Generation

CAVP Cert.	Algorithm and Standard	Mode / Method	Description / Key Size(s)	Use / Function
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	ECDSA ANSI X9.62 [FIPS 186-4]	Public Key Validation (PKV)	Curve: P-224, P-256, P-384, P-521	Asymmetric Key Validation
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	ECDSA ANSI X9.62 [FIPS 186-4]	Signature Generation	Curve: P-224, P-256, P-384, P-521	Digital Signature Generation
A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	ECDSA ANSI X9.62 [FIPS 186-4]	Signature Verification	Curve: P-224, P-256, P-384, P-521	Digital Signature Verification
A990 (c_ssse3) A979 (vng_Intel) A974 (c_avx2) A975 (c_avx)	SHS [FIPS 180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256 (except for A979)	N/A	Message Digest
A979 (vng_Intel) A974 (c_avx2) A975(c_avx) A990 (c_ssse3)	HMAC [FIPS 198]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 SHA-512/256 (except for A979)	112 bits or greater	Keyed Hash
A990 (c_ssse3)	KBKDF [SP800-108]	KDF Mode: Counter and Feedback MAC Mode: HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA251	Supported Lengths: 8-4096 Increment 8 Fixed Data Order: Before Fixed Data Counter Length: 32	Key Derivation
N/A	ENT (P) [SP800-90B] ENT (NP) [SP800-90B]	N/A	Seeding for the DRBG. (is provided by the underlying operational environment)	Random Number Generation

Table 4 - Approved Algorithms

This module does not have any non-Approved algorithms used in the Approved mode of operation (with or without security claimed).

The table below lists the non-Approved algorithms and security functions that are used in the non-Approved mode of operation:

Algorithm/Functions	Use/Function	Notes
RSA Key Pair Generation	ANSI X9.31 Key Pair Generation Key Size < 2048	Non-Approved
RSA Signature Generation	PKCS#1 v1.5 and PSS Signature Generation Key Size < 2048	Non-Approved
RSA Signature Verification	PKCS#1 v1.5 and PSS Signature Verification Key Size < 1024	Non-Approved

RSA Key Wrapping	OAEP, PKCS#1 v1.5 and -PSS schemes	Non-Approved
Ed25519	Key Agreement Key Generation Signature Generation Signature Verification	Non-Approved
ANSI X9.63 KDF	Hash based Key Derivation Function	Non-Approved
RFC6637	Key Derivation Function	Non-Approved
HKDF [SP800-56C]	Key Derivation Function	Non-Approved
DES	Encryption / Decryption Key Size 56-bits	Non-Approved
CAST5	Encryption / Decryption Key Sizes 40 to 128-bits in 8-bit increments	Non-Approved
RC4	Encryption / Decryption Key Sizes 8 to 4096-bits	Non-Approved
RC2	Encryption / Decryption Key Sizes 8 to 1024-bits	Non-Approved
MD2	Message Digest Digest size 128-bit	Non-Approved
MD4	Message Digest Digest size 128-bit	Non-Approved
MD5	Message Digest Digest size 128-bit	Non-Approved
RIPEMD	Message Digest Digest size 160-bits	Non-Approved
ECDSA	PKG: Curve P-192 PKV: Curve P-192 Signature Generation: Curve P-192 Signature Verification: Curve P-192	Non-Approved due to the small curve size
	Key Pair Generation for compact point representation of points	Non-Approved
Integrated Encryption Scheme on elliptic curves (ECIES)	Encryption / Decryption	Non-Approved
Blowfish	Encryption / Decryption	Non-Approved
OMAC (One-Key CBC MAC)	MAC generation	Non-Approved
Triple-DES [SP 800-67]	ECB, CBC	Encryption/Decryption Note The module itself does not enforce the limit of 2^{16} encryptions with the same Triple-DES key, as required by FIPS 140-3 IG C.G.

Table 5 - Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

The Apple corecrypto Module v11.1 [Intel, Kernel, Software] executes within the kernel space of the computing platforms and operating systems listed in Table 2. Figure 1 below shows the logical block diagram representing the following information:

- The location of the logical object of the module with respect to the operating system, other supporting applications, and the cryptographic boundary so that all the logical and physical layers between the logical object and the cryptographic boundary are clearly defined; and
- The interactions of the logical object of the module with the operating system and other supporting applications resident within the cryptographic boundary.

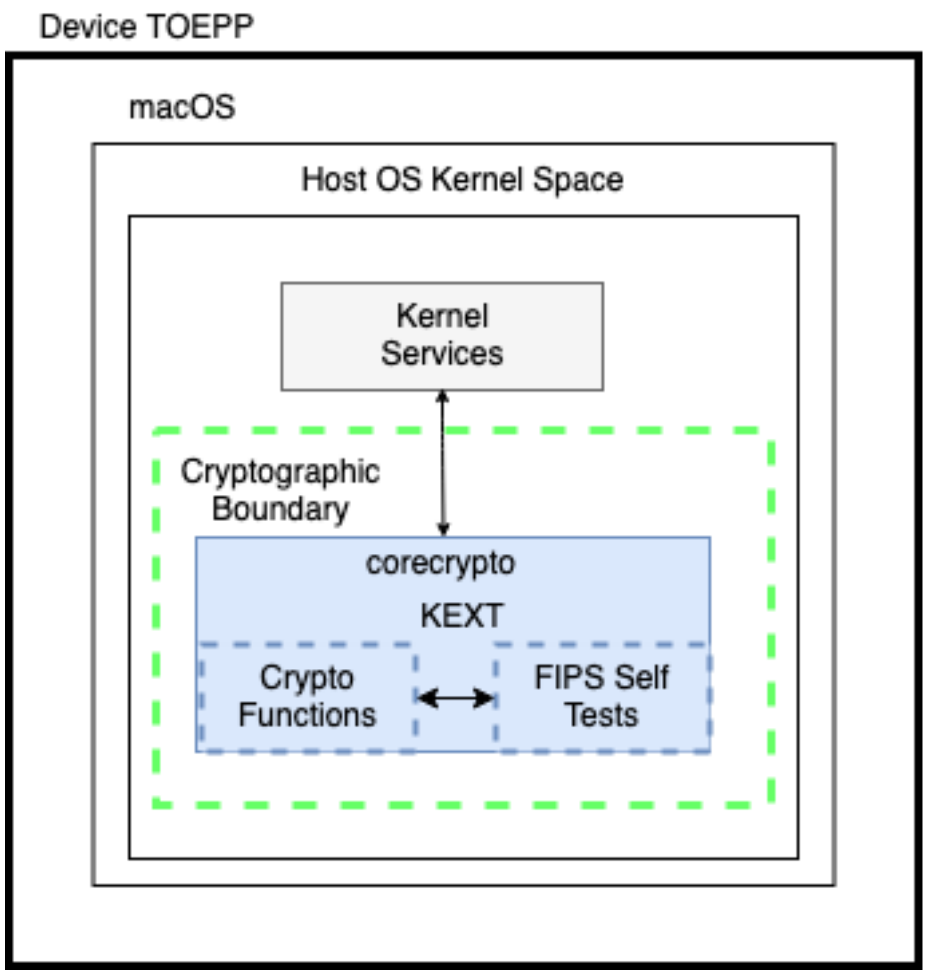

Figure 1 - Logical block diagram. Kernel extension (KEXT) is a bundle that performs low-level tasks. KEXTs run in kernel space, which gives them elevated privileges and the ability to perform tasks that user-space apps cannot.

3. Cryptographic Module Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-3 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the application program interface (API) through which applications request services and the Operating System calls that the module invokes.

The underlying logical interfaces of the module are the C language Kernel Interfaces (KPIs). In detail these interfaces are described in (Table 6):

Logical Interface	Data that passes over port/interface
Data Input	Data inputs are provided in the variables passed in the KPI and callable service invocations, generally through caller-supplied buffers
Data Output	Data outputs are provided in the variables passed in the KPI and callable service invocations, generally through caller-supplied buffers
Control Input	Control inputs which control the mode of the module are provided through dedicated parameters, namely the kernel module plist whose information is supplied to the module by the kernel module loader.
Control Output	Not Applicable
Status Output	Status output is provided in return codes and through messages. Documentation for each KPI lists possible return codes. A complete list of all return codes returned by the C language KPIs within the module is provided in the header files and the KPI documentation. Messages are also documented in the KPI documentation.

Table 6 - Interfaces

The module is optimized for library use within the macOS kernel space and does not contain any terminating assertions or exceptions. It is implemented as a macOS dynamically loadable library. The dynamically loadable library is loaded into the macOS kernel, and its cryptographic functions are made available to macOS kernel services only. Any internal error detected by the module is reflected back to the caller with an appropriate return code. The calling macOS kernel service must examine the return code and act accordingly.

The module communicates any error status synchronously through the use of its documented return codes, thus indicating the module's status. It is the responsibility of the caller to handle exceptional conditions in a FIPS 140-3 appropriate manner.

Caller-induced or internal errors do not reveal any sensitive material to callers. Cryptographic bypass capability is not supported by the module.

4. Roles, services, and authentication

The module supports a single instance of one authorized role: The Crypto Officer. No support is provided for multiple concurrent operators or a Maintenance Operator.

FIPS 140-3 does not require an authentication mechanism for level 1 modules. Therefore, the module does not implement an authentication mechanism for Crypto Officer. The Crypto Officer role is authorized to access all services provided by the module (see Table 7 - Approved Services and Table 8 - Non-Approved Services below).

The module implements a dedicated KPI function to indicate if a requested service utilizes an approved security function. For services listed in Table 7 - Approved Services, the indicator function returns 1. For services listed in Table 8 - Non-Approved Services, the indicator function returns 0.

The table below lists all approved services that can be used in the approved mode of operation. The abbreviations of the access rights to keys and SSPs have the following interpretation:

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 = **Zeroise**: The module zeroises the SSP.

N/A= The service does not access any SSP during its operation

Service	Description and Input/ Output	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
AES Encryption / Decryption	Input for Encryption: key and plain text Output for Encryption: cipher text Input for Decryption: key and cipher text Output for Decryption: plain text	AES-CBC, AES-ECB, AES-CFB128, AES-CFB8, AES-OFB, AES-CTR, AES-XTS, AES-GCM, AES-CCM	AES key	CO	W, E	1
AES Key Wrapping	Input: key encryption key and key to be wrapped Output: wrapped key	AES-KW	AES key-encryption key	CO	W, E	1
Secure Hash Generation	Input: message Output: Hash value	Message Digest SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256	none	CO	N/A	1

Service	Description and Input/ Output	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
HMAC generation and verification	Generation: Input: HMAC key, message Output: keyed Hash value Verification: Input: HMAC key, message, keyed hash value Output: True or False	HMAC Keyed Hash	HMAC key	CO	W, E	1
RSA signature generation and verification	Input for signature generation: RSA private key, message Output: signature Input for signature verification: RSA public key, signature Output: True or False	RSA signature generation, RSA signature verification	RSA key pair	CO	W, E	1
ECDSA signature generation and verification	Input for signature generation: ECDSA private key and message Output: signature Input for signature verification: ECDSA public key and signature Output: True or False	ECDSA signature generation, ECDSA signature verification	ECDSA key pair	CO	W, E	1
Random number generation	Input: Entropy input string, nonce Output: Random numbers	HMAC_DRBG, CTR_DRBG	Entropy Input String, Seed, V value and Key	CO	G, R, E, Z	1
KBKDF	Input: key derivation key Output: derived key	Key Derivation	key derivation key, derived key	CO	G, R, E	1
ECDSA Key-pair Generation	Input: curve size Output: generated key pair	ECDSA KeyGen	ECDSA key pair	CO	G, R, E	1
RSA Key-pair Generation	Input: key size Output: generated key pair	RSA KeyGen	RSA key pair	CO	G, R, E	1

Service	Description and Input/ Output	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Release all resources of symmetric crypto function context	Input: handler of symmetric crypto function context Output: zeroised and released memory space	N/A	AES key	CO	Z	1
Release all resources of hash context	Input: handler of hash context Output: released memory space	N/A	HMAC key	CO	Z	1
Release of all resources of asymmetric crypto function context	Input: handler of asymmetric crypto function context Output: zeroised and released memory space	N/A	RSA/ ECDSA keys	CO	Z	1
Release of all resources of key derivation function context	Input: handler of key derivation function context Output: zeroised and released memory space	N/A	key derivation key	CO	Z	1
Self-test	Input: power Output: Pass/Fail status	AES-CCM, AES-GCM, AES-XTS, AES-CBC, AES-ECB, HMAC_DRBG CTR_DRBG, HMAC, , RSA Signature Generation, RSA Signature Verification, ECDSA Signature Generation, ECDSA Signature Verification, KBKDF	Software integrity key	CO	E	1
Show Status	Input: KPI invocation Output: Operational/Error status	N/A	None	CO	N/A	N/A
Show Module Info	Input: KPI invocation Output: Module Base Name + Module Version Number	N/A	None	CO	N/A	N/A

Table 7 - Approved Services

The table below lists all non-Approved services that can only be used in the non-Approved mode of operation.

Service	Description and Input/Output	Algorithms Accessed	Role	Indicator
Triple-DES encryption / decryption	Module does not meet FIPS 140-3 IG C.G because it does not have a control over the number of blocks to be encrypted under the same Triple-DES key. Input for Encryption: key, plain text data Output for Encryption: cipher text data Input for Decryption: key, cipher text data Output for Decryption: plain text data	Triple-DES	CO	0
RSA Key Wrapping	The CAST does not perform the full KTS, only the raw RSA encrypt/decrypt. Input (RSA encrypt): RSA public key, key to be wrapped Output (RSA encrypt): wrapped key Input (RSA decrypt): RSA private key, key to be unwrapped Output (RSA encrypt): plaintext key	RSA encrypt/decrypt	CO	0
RSA Key-pair Generation	ANSI X9.31 Key Pair Generation key size < 2048 Input for PKG: key size Output: generated key pair	RSA Key Gen	CO	0
RSA Signature Generation	PKCS#1 v1.5 and PSS Signature Generation Key Size < 2048 Input: RSA private key, message Output: signature	RSA SigGen	CO	0
RSA Signature Verification	PKCS#1 v1.5 and PSS Signature Verification Key Size < 1024 Input: RSA public key, signature Output: true or false	RSA SigVer	CO	0

Service	Description and Input/Output	Algorithms Accessed	Role	Indicator
ECDSA Key-pair Generation (PKG) and ECDSA Key Validation (PKV)	ECDSA PKG and PKV using curve P-192 Input for PKG: curve size (P-192) Output: generated (P-192) key pair Input for PKV: public key Output: true or false	ECDSA Key Generation, ECDSA Key Validation	CO	0
ECDSA Signature Generation	ECDSA Signature Generation using curve P-192 Input: (P-192) private key, message Output: signature	ECDSA Signature Generation	CO	0
ECDSA Signature Verification	ECDSA Signature Verification using curve P-192 Input: (P-192) public key, signature Output: true or false	ECDSA Signature Verification	CO	0
ECDSA	Key Pair Generation for compact point representation of points Input: key size Output: generated private and public key pair	ECDSA Key Generation	CO	0
Ed25519 Key Generation	Ed25519 Key Generation Input: none Output: generated Curve25519 key pair	Ed25519 Key Generation	CO	0
Ed25519 Signature Generation	EdDSA Signature Generation over Curve25519 Input: (Curve25519) private key, message Output: signature	Ed25519 Signature Generation	CO	0
Ed25519 Signature Verification	Ed25519 Signature Verification over Curve25519 Input: (Curve25519) public key, signature Output: true or false	Ed25519 Signature Verification	CO	0
Ed25519 Key Agreement	Key Agreement Input: peer public key, own private key Output: shared secret	Ed25519 Key Agreement	CO	0

Service	Description and Input/Output	Algorithms Accessed	Role	Indicator
ECIES	Integrated Encryption Scheme on elliptic curves Input for encryption: peer public key, plaintext Output for encryption: public key, ciphertext (with authentication tag) Input for decryption: authentication tag, ciphertext, own private key Output for decryption: plaintext message or error	ECIES Encrypt/Decrypt	CO	0
ANSI X9.63 Key Derivation	SHA-1 hash-based key derivation function Input: key derivation key Output: derived key	SHA-1	CO	0
SP800-56C Key Derivation (HKDF)	SHA-256 hash-based key derivation function Input: key derivation key Output: derived key	SHA-256	CO	0
RFC6637 Key Derivation	SHA hash based key derivation function Input: key derivation key Output: derived key	SHA-256, SHA-512, AES-128, AES-256	CO	0
OMAC Message Authentication Code Generation and Verification	One-Key CBC MAC using 128-bit key For Message Authentication Code Generation Input: message and key Output: message authentication code For Message Authentication Code Verification Input: message, key and message authentication code Output: True or False	OMAC	CO	0
Message digest generation.	Message digest generation using non-approved algorithms Input: message Output: message digest	MD2, MD4, MD5, RIPEMD	CO	0

Service	Description and Input/Output	Algorithms Accessed	Role	Indicator
(other) symmetric encryption / decryption	symmetric encryption / decryption using non-approved algorithms Input for Encryption: key and plain text data Output for Encryption: cipher text data Input for Decryption: key and cipher text data Output for Decryption: plain text data	Blowfish, CAST5, DES, RC2, RC4	CO	0

Table 8 - Non-Approved Services

5. Software/Firmware security

5.1. Integrity Techniques

The Apple corecrypto Module v11.1 [Intel, Kernel, Software], which is made up of a single component, is provided in the form of binary executable code. A software integrity test is performed on the runtime image of the module. The HMAC-SHA256 implemented in the module is used as an approved algorithm for the integrity test. If the test fails, the module enters an error state where no cryptographic services are provided, and data output is prohibited i.e., the module is not operational.

5.2. On-Demand Integrity Test

Integrity tests are performed as part of the Pre-Operational Self-Tests. It is automatically executed at power-on. It can also be invoked by self-test service or powering-off and reloading the module.

6. Operational Environment

The Apple corecrypto Module v11.1 [Intel, Kernel, Software] operates in a modifiable operational environment per FIPS 140-3 level 1 specifications. The module is supplied as part of macOS Big Sur 11.0.1, a commercially available general-purpose operating system executing on the computing platforms specified in section 2.

7. Physical Security

The FIPS 140-3 physical security requirements do not apply to the Apple corecrypto Module v11.1 [Intel, Kernel, Software], since it is a software module.

8. Non-invasive Security

Currently, the non-invasive security is not required by FIPS 140-3 (see NIST SP 800-140F). The requirements of this area are not applicable to the module.

9. Sensitive Security Parameter Management

The following table summarizes the keys and Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module:

Key/ SSP Name /Type	Strength	Security Function and Cert. #	Generation	Import /Export	Establishment	Storage	Zeroisation	Use and related keys
AES Key	128 to 256 bits	AES Encryption/ Decryption A945 (c_asm) A976 (c_aesni) A978 (asm_aesni) A977 (asm_x86) A943 (vng_asm) A973 (vng_aesni)	N/A	Import from calling application. No Export	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down	Symmetric Encryption and Decryption
AES Key-wrapping Key	128 to 256 bits	AES Key Wrapping A945 (c_asm) A976 (c_aesni)	N/A	Import from calling application. No Export	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down	Key Transport
HMAC Keys	Min: 112 bits	HMAC generation/ verification A979 (vng_Intel) A974 (c_avx2) A975(c_avx) A990 (c_ssse3)	N/A	Import from calling application. No Export	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down	Keyed Hash
ECDSA Key Pair (including intermediate keygen values)	112 to 256 bits	ECDSA signature generation and verification A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	The key pairs are generated conformant to SP800-133r2 (CKG) using FIPS186-4 Key Generation method, and the random value used in the key generation is generated using SP800-90A DRBG	Import and Export to calling application for key pair only. Intermediate keygen values are not output.	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down. Intermediate keygen values are zeroized before the module returns from the key generation function.	Digital Signature

RSA Key Pair (including intermediate keygen values)	112 to 150 bits	RSA signature generation and verification A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	The key pairs are generated conformant to SP800-133r2 (CKG) using FIPS186-4 Key Generation method, and the random value used in the key generation is generated using SP800-90A DRBG	Import and Export to calling application for key pair only. Intermediate keygen values are not output.	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down. Intermediate keygen values are zeroized before the module returns from the key generation function.	Digital Signature
Entropy Input String	256 bits	Random Number Generation ENT (P) and ENT (NP)	Obtained from two entropy sources	Import from OS. No Export	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down	Random Number Generation
DRBG Seed, Internal State V value and Key	256 bits	Random Number Generation A943 (vng_asm) A945 (c_asm) A976 (c-aesni) A973 (vng_aesni) A974 (c_avx2) A975 (c_avx) A990 (c_ssse3)	Derived from entropy input string as defined by SP800-90A	N/A	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down	Random Number Generation
KBKDF Key Derivation Key	Min: 112 bits	KBKDF A990 (c_ssse3)	N/A	imported from calling application, No Export	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down	Key Derivation
KBKDF Derived Key	Min: 112 bits		Internally generated via SP800-108 KBKDF key derivation algorithm	No Import. Export to calling application	N/A	N/A: The module does not provide persistent keys/ SSPs storage.	Automatic zeroisation when structure is deallocated or when the system is powered down	Key Derivation

Software integrity key	N/A	HMAC SHA-256 A990 (c_ssse3) A979 (vng_Intel) A974 (c_avx2) A975 (c_avx)	N/A	N/A	N/A	Stored in the module binary computed during build.	N/A	Self-Test

Table 9 - SSPs

9.1. Random Number Generation

A NIST approved deterministic random bit generator based on a block cipher as specified in NIST [SP 800-90A] is used. The default Approved DRBG used for random number generation is a CTR_DRBG using AES-256 with derivation function and without prediction resistance. The random numbers used for key generation are all generated by CTR_DRBG in this module. Per section 10.2.1.1 of [SP 800-90A], the internal state of CTR_DRBG is the value V, Key.

The module also employs a HMAC_DRBG for random number generation. The HMAC_DRBG is only used at the early boot time of macOS kernel for memory randomization. The output of HMAC_DRBG is not used for key generation. Per section 10.1.2.1 of [SP 800-90A], the internal state of HMAC_DRBG is the value V, Key.

The deterministic random bit generators are seeded by read_random. The read_random is the Kernel Space interface.Two entropy sources (one non-physical entropy source and one physical entropy source) residing within the TOEPP provide the random bits. The output of entropy pool provides 256-bits of entropy to seed and reseed SP800-90B DRBG during initialization (seed) and reseeding (reseed).

Entropy Source	Minimum number of bits of entropy	Details
NIST SP800-90B compliant ENT (P) and NIST SP800-90B compliant ENT (NP)	256	The seed is provided by post-processed entropy data from two entropy sources

Table 10 - Non-Deterministic Random Number Generation Specification

9.2. Key / SSP Generation

The module generates Keys and SSPs in accordance with FIPS 140-3 IG D.H. The cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per [SP800-133r2] sections 4 and 5.1 (vendor affirmed), compliant with [FIPS186-4], and using DRBG compliant with [SP800-90A]. A seed (i.e., the random value) used in asymmetric key generation is a direct output from [SP800-90A] DRBG. The key generation service for RSA, ECDSA, as well as the [SP 800-90A] DRBG have been ACVT tested with algorithm certificates found in Table 4.

9.3. Keys/SSPs Establishment

The module provides the following key/SSP establishment services in the Approved mode:

- AES-Key Wrapping

  The module implements a Key Transport Scheme (KTS) using AES-KW compliant to [SP800-38F]. The SSP establishment methodology provides between 128 and 256 bits of encryption strength.

- KBKDF Key Derivation

  The KBKDF is compliant to [SP800-108]. The module implements both Counter and Feedback modes with HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, or HMAC-SHA2-512 as the PRF.

9.4. Keys/SSPs Import/Export

All keys and SSPs that are entered from, or output to module, are entered from or output to the invoking application running on the same device. Keys/SSPs entered into the module are electronically entered in plain text form. Keys/SSPs are output from the module in plain text form if required by the calling application.

The module allows the output of plaintext CSPs (i.e., ECDSA/RSA Key Pairs). To prevent inadvertent output of sensitive information, the module performs the following two independent internal actions:

1. The module will internally request the random number generation service to obtain the random numbers and verify that the service completed without errors.

2. Once the keys are generated the module will perform the pairwise consistency test and verify that the test is completed without errors.

Only after successful completion of both actions, are the generated CSPs output via the KPI output parameter in plaintext.

9.5. Keys/SSPs Storage

The Apple corecrypto Module v11.1 [Intel, Kernel, Software] stores ephemeral keys/SSPs in memory only. They are received for use or generated by the module only at the command of the calling application. The module does not provide persistent keys/SSPs storage.

The module protects all keys/SSPs through the memory separation and protection mechanisms provided by the operating system. No process other than the module itself can access the keys/SSPs in its process' memory.

9.6. Keys/SSPs Zeroization

Keys and SSPs are zeroised when the appropriate context object is destroyed or when the system is powered down. Input and output interfaces are inhibited while zeroisation is performed.

10. Self-tests

This section specifies the pre-operational and conditional self-tests performed by the module. The pre-operational and conditional self-tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected. The module does not implement a bypass mode nor security functions critical to the secure operation of the cryptographic module and thus, does not implement neither a pre-operational bypass test nor pre-operational critical functions test. While the module is executing the self-tests, services are not available, and input and output are inhibited. If the test fails either pre-operational and conditional self-tests, the module reports an error message indicating the cause of the failure and enters the Error State (See section 10.3). The module permits operators to initiate the pre-operational or conditional self-tests on demand for periodic testing of the module by rebooting the system (i.e., power-cycling).

10.1. Pre-operational Software Integrity Test

The module performs a pre-operational software integrity automatically when the module is loaded into memory (i.e., at power on) before the module transitions to the operational state. A software integrity test is performed on the runtime image of the Apple corecrypto Module v11.1 [Intel, Kernel, Software] with HMAC-SHA256 used to perform the approved integrity technique. Prior to using HMAC-SHA-256, a Conditional Cryptographic Algorithm Self-Tests (CAST) is performed. If the CAST on the HMAC-SHA-256 is successful, the HMAC value of the runtime image is recalculated and compared with the stored HMAC value pre-computed at compilation time.

10.2. Conditional Self-Tests

Conditional self-tests are be performed by a cryptographic module when the conditions specified for the following tests occur: Cryptographic Algorithm Self-Test, Pair-Wise Consistency Test.

The module does not implement any functions requiring a Software/Firmware Load Test, Manual Entry Test, Conditional Bypass Test nor Conditional Critical Functions Test; therefore, these tests are not performed by the module.

The following sub-sections describe the conditional tests supported by the Apple corecrypto Module v11.1 [Intel, Kernel, Software].

10.2.1. Conditional Cryptographic Algorithm Self-Tests

In addition to the pre-operational software integrity test described in Section 10.1, the Apple corecrypto Module v11.1 [Intel, Kernel, Software] also runs the Conditional Cryptographic Algorithm Self-Tests (CAST) for all cryptographic functions of each approved cryptographic algorithm implemented by the module during power-up as well. All CASTs are performed prior to the first operational use of the cryptographic algorithm. These tests are detailed in Table 11 - Pre-Operational Cryptographic Algorithm Self-Tests below.

Cryptographic Algorithm	Notes
HMAC-SHA256	Used for module integrity test
AES implementations selected by the module for the corresponding environment AES-CCM, AES-GCM, AES-XTS, AES-CBC, AES-ECB using 128-bit key	Separate encryption / decryption operations are performed
CTR_DRBG and HMAC_DRBG	Each DRBG mode tested separately
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	KAT
SHA-1, SHA-256, SHA-512	Covered by high level HMAC self-test
RSA, 2048-bit modulus with SHA-256	Separate Signature generation/ verification KAT are performed
ECDSA, P-256 curve with SHA-256	Signature generation/verification KAT are performed
KBKDF (counter and feedback modes)	KAT

Table 11 - Pre-Operational Cryptographic Algorithm Self-Tests

10.2.2.Conditional Pairwise Consistency Test

The Apple corecrypto Module v11.1 [Intel, Kernel, Software] does generate RSA and ECDSA asymmetric keys and performs the required RSA and ECDSA pair-wise consistency tests on the newly generated key pairs.

10.3.Error Handling

If any of the above-mentioned self-tests described in Sections 10.1, 10.2.1 or 10.2.2 fail, the module reports the cause of the error and enters an error state. In the Error State, no cryptographic services are provided, and data output is prohibited. The only method to recover from the error state is to power cycle the device which results in the module being reloaded into memory and reperforming the pre-operational software integrity test and the Conditional CASTs. The module will only enter into the operational state after successfully passing the pre-operational software integrity test and the Conditional CASTs. The table below shows the different causes that lead to the Error State and the status indicators reported.

Cause of Error	Error Indicator
Failed Pre-operational Software Integrity Test	print statement "*FAILED: fipspost_post_integrity*" to stdout
Failed Conditional CAST	print statement "*FAILED:<event>*" to stdout (<event> refers to any of the cryptographic functions listed in Table 11.)
Failed Conditional PCT	Error code "*CCEC_GENERATE_KEY_CONSISTENCY*" returned for ECDSA Error code "*CCRSA_GENERATE_KEY_CONSISTENCY*" returned for RSA

Table 12 Error Indicators

11. Life-cycle assurance

11.1. Delivery and Operation

The module is built into macOS Big Sur 11.0.1 and delivered with macOS. There is no standalone delivery of the module as a software library.

The vendor's internal development process guarantees that the correct version of module goes with its intended macOS version. For additional assurance, the module is digitally signed by vendor, and it is verified during the integration into macOS.

This digital signature-based integrity protection during the delivery/integration process is not to be confused with the HMAC-256 based integrity check performed by the module itself as part of its pre-operational self-tests.

11.2. Crypto Officer Guidance

The Approved mode of operation is configured in the system by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved services listed in Table 8 - Non-Approved Services. If the device starts up successfully, then the module has passed all self-tests and is operating in the Approved mode.

A Crypto Officer Role Guide is provided by Apple which offers IT System Administrators with the necessary technical information to ensure FIPS 140-3 Compliance of macOS Big Sur 11.0.1 systems. This guide walks the reader through the system's assertion of cryptographic module integrity and the steps necessary if module integrity requires remediation. A link to the Guide can be found on the Product security, validations, and guidance page found in [macOS].

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

- AES-GCM IV is constructed in compliance with IG C.H scenario 1b (IPsec-v3).

  The GCM IV generation follows RFC 4106 and shall only be used for the IPsec-v3 protocol version 3. The counter portion of the IV is set by the module within its cryptographic boundary. The module does not implement the IPsec protocol. The module's implementation of AES-GCM is used together with an application that runs outside the module's cryptographic boundary. The design of the IPsec protocol implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all of its possible values.

  In case the module's power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed. This condition is not enforced by the module; however, it is met implicitly. The module does not retain any state when power is lost. As indicated in Table 10, column Storage, the module exclusively uses volatile storage. This means that AES-GCM key/IVs are not persistently stored during power off: therefore, there is no re-connection possible when the power is back on with re-

  generation of the key used for GCM. After restoration of the power, the user of the module (e.g., IKE) along with User application that implements the protocol, must perform a complete new key establishment operation using new random numbers (Entropy input string, DRBG seed, DRBG internal state V and Key, shared secret values that are not retained during power cycle, see table 10) with subsequent KDF operations to establish a new GCM key/IV pair on either side of the network communication channel

- AES-XTS mode is only approved for hardware storage applications. The length of the AES-XTS data unit does not exceed 2^{20} blocks. The module checks explicitly that Key_1 ≠ Key_2 before using the keys in the XTS-Algorithm to process data with them compliant with IG C.I.

12. Mitigation of other attacks

The module does not claim mitigation of other attacks.

A. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CAST	Cryptographic Algorithm Self-Test
CAST5	A symmetric-key 64-bit block cipher with 128-bit key
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ENT	NIST SP 800-90B Compliant Entropy Source
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
KBKDF	Key Based Key Derivation Function
KDF	Key Derivation Function
KEXT	Kernel Extension
KW	AES Key Wrap
MAC	Message Authentication Code
KPI	Kernel Programming Interface
NIST	National Institute of Science and Technology
OAEP	Optimal Asymmetric Encryption Padding
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PKG	Key-Pair Generation
PKV	Public Key Validation
PRF	Pseudo-Random Function
PSS	Probabilistic Signature Scheme
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
TEOPP	Tested Operational Environment's Physical Perimeter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

B. References

FIPS140-3 **FIPS PUB 140-3 - Security Requirements for Cryptographic Modules**
March 2019
https://doi.org/10.6028/NIST.FIPS.140-3

SP 800-140x **CMVP FIPS 140-3 Related Reference**
https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-standards

FIPS140-3_IG **Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program**
September 2020
https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements

FIPS140-3_MM **CMVP FIPS 140-3 Draft Management Manual**
https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/Draft%20FIPS-140-3-CMVP%20Management%20Manual%2009-18-2020.pdf

SP 800-140 **FIPS 140-3 Derived Test Requirements (DTR)**
https://csrc.nist.gov/publications/detail/sp/800-140/final

SP 800-140A **CMVP Documentation Requirements**
https://csrc.nist.gov/publications/detail/sp/800-140a/final

SP 800-140B **CMVP Security Policy Requirements**
https://csrc.nist.gov/publications/detail/sp/800-140b/final

SP 800-140C **CMVP Approved Security Functions**
https://csrc.nist.gov/publications/detail/sp/800-140c/final

SP 800-140D **CMVP Approved Sensitive Security Parameter Generation and Establishment Methods**
https://csrc.nist.gov/publications/detail/sp/800-140d/final

SP 800-140E **CMVP Approved Authentication Mechanisms** https://csrc.nist.gov/publications/detail/sp/800-140e/final

SP 800-140F **CMVP Approved Non-Invasive Attack Mitigation Test Metrics** https://csrc.nist.gov/publications/detail/sp/800-140f/final

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

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

RFC3394 **Advanced Encryption Standard (AES) Key Wrap Algorithm**
September 2002
http://www.ietf.org/rfc/rfc3394.txt

RFC5649 **Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm**
September 2009
http://www.ietf.org/rfc/rfc5649.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
SP800-38C	**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
SP800-56Cr2	**Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP800-57	**NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management Part 1: General** May 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.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
SP800-90Ar1	**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
SP800-90B	**NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP800-108	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions (Revised)** October 2009 http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
SP800-131Ar2	**Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP800-133r2	**Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf

SP800-135	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
MACOS	**macOS Technical Overview** https://developer.apple.com/macos/
SEC	**Apple Platform Security** (Spring 2020) https://support.apple.com/guide/security/welcome/web
macOS	**Product security certifications for macOS** https://support.apple.com/HT201159

B.