Apple Inc.

Apple corecrypto Kernel Space Module for Intel (ccv10)

FIPS 140-2 Non-Proprietary Security Policy

Module Version 10.0

Prepared for:
Apple Inc.
One Apple Park Way
Cupertino, CA 95014
www.apple.com

Prepared by:
atsec information security Corp.
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This document is a non-proprietary Security Policy for the Apple corecrypto Kernel Space Module for Intel (ccv10). It describes the module and the FIPS 140-2 cryptographic services it provides. This document also defines the FIPS 140-2 security rules for operating the module.

This document was prepared in fulfillment of the FIPS 140-2 requirements for cryptographic modules and is intended for security officers, developers, system administrators, and end-users.

FIPS 140-2 details the requirements of the Governments of the U.S. and Canada for cryptographic modules, aimed at the objective of protecting sensitive but unclassified information.

For more information on the FIPS 140-2 standard and the Cryptographic Module Validation Program please refer to the NIST [CMVP] website.

Throughout the document "Apple corecrypto Kernel Space Module for Intel (ccv10)", "cryptographic module", "corecrypto KEXT" or "the module" are used interchangeably to refer to the Apple corecrypto Kernel Space Module for Intel (ccv10). "ccv10" is used to refer to the module version 10.0.

1.2 Document Organization / Copyright

This non-proprietary Security Policy document may be reproduced and distributed only in its original entirety without any revision, ©2021 Apple Inc.

1.3 External Resources / References

The Apple website (http://www.apple.com) contains information on the full line of products from Apple Inc. For a detailed overview of the operating system macOS and the associated security properties refer to [MACOS] and [SEC]. For details on macOS releases with their corresponding validated modules and Crypto Officer Role Guides refer to the "Product security certifications, validations, and guidance for macOS" [UGuide].

Additional references are provided in the list below:

CMVP	Cryptographic Module Validation Program https://csrc.nist.gov/projects/cryptographic-module-validation-program
CAVP	Cryptographic Algorithm Validation Program https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program
FIPS 140-2	Federal Information Processing Standards Publication, "FIPS PUB 140-2 Security Requirements for Cryptographic Modules," Issued May-25-2001, Effective 15-Nov-2001, Location: http://csrc.nist.gov/groups/STM/cmvp/standards.html
FIPS 140-2 IG	NIST, "Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program," August 2020 Location: http://csrc.nist.gov/groups/STM/cmvp/standards.html
FIPS 180-4	Federal Information Processing Standards Publication 180-4, March 2012, Secure Hash Standard (SHS)
FIPS 186-4	Federal Information Processing Standards Publication 186-4, July 2013, Digital Signature Standard (DSS)
FIPS 197	Federal Information Processing Standards Publication 197, November 26, 2001 Advanced Encryption Standard(AES)
SP800-38 A	NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation", December 2001

SP800-38 C	NIST Special Publication 800-38C, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality", May 2004
SP800-38 E	NIST Special Publication 800-38E, "Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices", January 2010
SP800-38 F	NIST Special Publication 800-38F, "Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping", December 2012
SP800-57P1	NIST Special Publication 800-57, "Recommendation for Key Management – Part 1: General (Revised)," July 2016
SP 800-90A	NIST Special Publication 800-90A, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", January 2012
SP800-132	NIST Special Publication 800-132, "Recommendation for Password-Based Key Derivation", December 2010
SEC	Security Overview
	https://developer.apple.com/security/
MACOS	macOS Technical Overview
	https://developer.apple.com/macos/
UGuide	User Guide
	https://support.apple.com/HT201159

1.4 Acronyms

AES	Advanced Encryption Standard
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining mode of operation
CFB	Cipher Feedback mode of operation
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter mode of operation
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
ECB	Electronic Codebook mode of operation
ECC	Elliptic Curve Cryptography
ECDSA	DSA based on ECC
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
FIPS PUB	FIPS Publication
GCM	Galois/Counter Mode
HMAC	Hash-Based Message Authentication Code

KAT	Known Answer Test
KEXT	Kernel extension
KDF	Key Derivation Function
KPI	Kernel Programming Interface
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OS	Operating System
PBKDF	Password-based Key Derivation Function
PCT	Pair-wise Consistency Test
PRF	PseudoRandom Functions
RNG	Random Number Generator
SHS	Secure Hash Standard
Triple-DES	Triple Data Encryption Standard
TLS	Transport Layer Security

2 Cryptographic Module Specification

2.1 Module Description

The Apple corecrypto Kernel Space Module for Intel (ccv10) is a software cryptographic module version 10.0 running on a multi-chip standalone general-purpose computer.

The cryptographic services provided by the module are:

- Data encryption / decryption
- Generation of hash values
- Message authentication
- Signature generation / verification
- Random number generation
- Key derivation
- Key generation

2.1.1 Module Validation Level

The module is intended to meet requirements of FIPS 140-2 security level 1 overall. The following Table 1 shows the security level for each of the eleven requirement areas of the validation.

FIPS 140-2 Security Requirement Area	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	1

Table 1: Module Validation Level

2.1.2 Module components

In the following sections the components of the Apple corecrypto Kernel Space Module for Intel (ccv10) are listed in detail. There are no components excluded from the validation testing.

2.1.2.1 Software components

corecrypto has a KPI layer that provides consistent interfaces to the supported algorithms. These implementations include proprietary optimizations of algorithms that are fitted into the corecrypto framework.

The corecrypto KEXT is linked dynamically into the macOS kernel.

2.1.2.2 Hardware components

There is hardware acceleration for AES-NI within the cryptographic module boundary.

2.1.3 Tested Platforms

The module has been tested on the following platforms with and without AES-NI:

Manufacturer	Model	Operating System
Apple Inc.	MacBook with Intel Core M	macOS Catalina 10.15
Apple Inc.	MacBook Pro with Intel Core i9	macOS Catalina 10.15
Apple Inc.	MacBook Pro with Intel Core i7	macOS Catalina 10.15
Apple Inc.	Mac mini with Intel Core i5	macOS Catalina 10.15
Apple Inc.	iMac Pro with Intel Xeon W	macOS Catalina 10.15

Table 2: Tested Platforms

In addition to the configurations tested by the laboratory, vendor-affirmed testing was performed on the following platforms for macOS 10.15 Catalina:

- MacBook, MacBook Air, MacBook Pro and iMac with an Intel i5
- Mac mini, MacBook Air, MacBook and iMac with an Intel i7
- iMac with an Intel i9

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 (IG G.5).

2.2 Modes of operation

The Apple corecrypto Kernel Space Module for Intel (ccv10) has an Approved and Non-Approved Mode of operation. The Approved Mode of operation is configured in the system by default and cannot be changed. If the device starts up successfully then corecrypto KEXT has passed all self-tests and is operating in the Approved Mode. Any calls to the Non-Approved security functions listed in Table 4 will cause the module to assume the Non-Approved Mode of operation.

The module transitions back into FIPS mode immediately when invoking one of the approved ciphers as all keys and Critical Security Parameters (CSPs) handled by the module are ephemeral and there are no keys and CSPs shared between any functions. A re-invocation of the self-tests or integrity tests is not required.

Even when using this FIPS 140-2 non-approved mode, the module configuration ensures that the self-tests are always performed during initialization time of the module.

The module contains multiple implementations of the same cipher as listed below. If multiple implementations of the same cipher are present, the module automatically selects which cipher is used based on internal heuristics. This includes the hardware-assisted AES and SHA implementations (AES-NI).

The Approved security functions are listed in Table 3. The Algorithm Certificate Numbers (Table 3) obtained from NIST are based on the successful ACVT testing of the cryptographic algorithm implementations of the module that runs on the hardware platforms referenced in Table 2.

Refer to [CAVP] for the current standards, algorithm test requirements, and special abbreviations used in the following Table.

2.2.1 Approved Security Functions

Cryptographic Function	Standard and Algorithm	Modes and Options	Algorithm Certificate Number
Random Number Generation	[SP 800-90] DRBG	CTR_DRBG Modes: AES-128, AES-256 Derivation Function Enabled Without Prediction Resistance	A13 (vng_asm) A15 (c_asm) A23 (c_aesni) A28 (vng_aesni)

Cryptographic Function	Standard and Algorithm	Modes and Options	Algorithm Certificate Number
		HMAC_DRBG Modes: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	A26 (c_avx2) A30 (c_avx) A34 (c_ssse3)
Symmetric Encryption and Decryption	[FIPS 197] AES [SP 800-38 A] [SP 800-38 C] [SP 800-38 E]	Key Length: 128, 192, 256 Modes: ECB CFB128 CBC CTR CFB8 OFB	A15 (c_asm) A23 (c_aesni)
		Key Length: 128, 192, 256 Modes: ECB GCM CCM CTR	A13 (vng_asm) A28 (vng_aesni)
		Key Length: 128, 192, 256 Modes: ECB XTS (key length: 128 and 256-bits only) CBC	A20 (asm_aesni) A24 (asm_x86)
	[SP 800-67] Triple-DES	(All keys 3-key independent) Modes: ECB	A16 (c_ltc)
Key Wrapping	[SP 800-38 D]	Key Length: 128, 192, 256 Modes: AES-CCM AES-GCM	A13 (vng_asm) A28 (vng_aesni)
	[SP 800-38 F]	Key Length: 128, 192, 256 Modes: AES-KW	A15 (c_asm) A23 (c_aesni)
Digital Signature and Asymmetric Key Generation	[FIPS186-4] RSA	Key Generation (ANSI X9.31), Modulus: 2048, 3072, 4096 Signature Generation (PKCS#1 v1.5) and (PKCS PSS) Modulus: 2048, 3072, 4096 Signature Verification (PKCS#1 v1.5) and (PKCS PSS) Modulus: 1024, 2048, 3072, 4096	A26 (c_avx2) A30 (c_avx) A34 (c_ssse3)

Cryptographic Function	Standard and Algorithm	Modes and Options	Algorithm Certificate Number
	[FIPS 186-4] ECDSA ANSI X9.62	Key Pair Generation (PKG): curves P-224, P-256, P-384, P-521 Public Key Validation (PKV): curves P-224, P-256, P-384, P-521 Signature Generation: curves P-224, P-256, P-384, P-521 Signature Verification: curves P-224, P-256, P-384, P-521	A26 (c_avx2) A30 (c_avx) A34 (c_ssse3)
Message Digest	[FIPS 180-4] SHS	Modes: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	A26 (c_avx2) A30 (c_avx) A32 (vng_intel) A34 (c_ssse3)
Keyed Hash	[FIPS 198] HMAC	Key size: 112 bits or greater Modes: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	A26 (c_avx2) A30 (c_avx) A32 (vng_intel) A34 (c_ssse3)
Key Derivation	[SP 800-132] PBKDF	Password based key derivation using HMAC with SHA-1 or SHA-224, SHA-256, SHA-384, SHA-512PRFs	Vendor Affirmed
CKG	[SP800-133]	RSA Key Generation (ANSI X9.31), Modulus: 2048, 3072, 4096 ECDSA Key Pair Generation: curves P-224, P-256, P-384, P-521	Vendor Affirmed

Table 3: Approved or Vendor Affirmed Security Functions

Cryptographic Function	Standard and Algorithm	Modes and Options	Algorithm Certificate Number
MD5 (used as part of the TLS key establishment scheme only)	Message Digest	Digest Size: 128-bit	Non-Approved, but Allowed
NDRNG (is provided by the underlying operational environment)	Random Number Generation	N/A	Non-Approved, but Allowed
RSA Key Wrapping	Non [SP800-56B], IG D.9	PKCS#1 v1.5 and PSS Modulus size: 2048-bits, 3072 or 4096-bits	Non-Approved, but allowed

Table 3a: Non-Approved but Allowed Security Functions

Note: PBKDFv2 is implemented to support all options specified in section 5.4 of [SP800-132]. The password consists of at least 6 alphanumeric characters from the ninety-six (96) printable and human-readable characters. The probability that a random attempt at guessing the password will succeed or a false acceptance will occur is equal to 1/96^6. The derived keys may only be used in storage applications. Additional guidance to appropriate usage is specified in section 7.

2.2.2 Non-Approved Security Functions:

Cryptographic Function	**Usage / Description**	**Caveat**
ANSI X9.63	Hash Based KDF	Non-Approved
Blowfish	Encryption / Decryption	Non-Approved
CAST5	Encryption / Decryption Key Sizes: 40 to 128 bits in 8-bit increments	Non-Approved
DES	Encryption / Decryption Key Size 56 bits	Non-Approved
ECDSA	Key Pair Generation (PKG): curve P-192 Public Key Validation (PKV): curve P-192 Signature Generation: curve P-192 Signature Verification: curve P-192	Non-Approved
	Key generation for compact point representation of points	Non-Approved
Ed25519	Key Agreement Sig(gen) Sig(ver)	Non-Approved
Integrated Encryption Scheme on elliptic curves	Encryption / Decryption	Non-Approved
MD2	Message Digest Digest size 128 bit	Non-Approved
MD4	Message Digest Digest size 128 bit	Non-Approved
OMAC (One-Key CBC MAC)	MAC generation	Non-Approved
RSA Key Wrapping	Non-[SP800-56B] IG D.9 RSA PCKS#1 v1.5 and PSS using key sizes < 2048-bits	Non-Approved
	[SP800-56B] KTS RSA-OAEP Modulus size: 2048, 3072 or 4096 bits	Non-Compliant
RFC6637	KDF	Non-Approved
RIPEMD	Message Digest Digest size 128, 160, 256, 320 bits	Non-Approved
RC2	Encryption / Decryption Key sizes: 8 to 1024 bits	Non-Approved
RC4	Encryption / Decryption Key sizes: 8 to 1024 bits	Non-Approved
Triple-DES	Encryption / Decryption: Two Key Implementation	Non-Approved
	asm_x86 (Optimized Assembler)Implementation: Encryption / Decryption Mode: CTR	Non-Compliant

Cryptographic Function	Usage / Description	Caveat
AES-CMAC	AES-128/192/256 MAC generation/ verification	Non-Approved
[SP800-108] KBKDF	HMAC-SHA1 or HMAC-SHA-224 or HMAC-SHA-256 or HMAC-SHA-384 or HMAC-SHA-512 based PRFs Modes: CTR, Feedback	Non-Compliant A34 (c_ssse3)
[SP800-56C]	Key Derivation Function	Non- Compliant
RSA Asymmetric Key Generation Signature Generation Signature Verification	ANSI X9.31 Key Generation with modulus not listed in Table 3	Non-Approved
	Signature Generation PKCS#1 v1.5 and PSS Key Size < 2048 Signature Verification PKCS#1 v1.5 and PSS Key sizes (modulus): 1536 bits Hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	

Table 4: Non-Approved or Non-Compliant Security Functions

Note: A Non-Approved function in Table 4 is that the function implements a non-Approved algorithm, while a Non-Compliant function is that the function implements an Approved algorithm but the implementation is either not validated by the CAVP or/and the self-tests are not implemented (IG 9.4) .

2.3 Cryptographic Module Boundary

The physical boundary of the module is the physical boundary of the macOS device that contains the module. Consequently, the embodiment of the module is a multi-chip standalone cryptographic module.

The logical module boundary is depicted in the logical block diagram given in Figure 1.

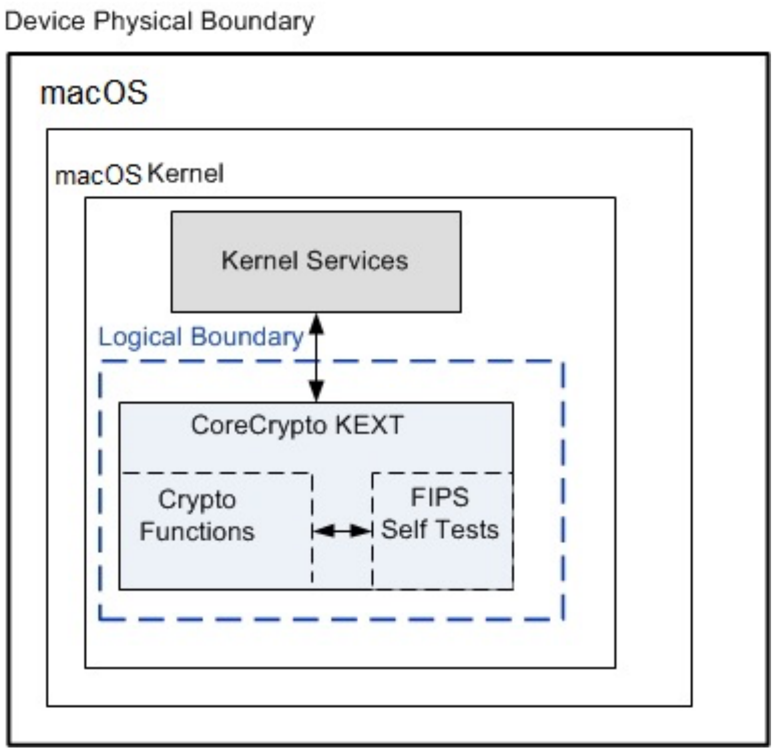

Figure 1: Logical Block Diagram

2.4 Module Usage Considerations

A user of the module must consider the following requirements and restrictions when using the module:

- AES-GCM IV is constructed in accordance with [SP800-38D] in compliance with IG A.5 scenario 1. The GCM IV generation follows RFC 5288 and shall only be used for the TLS protocol version 1.2. Users should consult [SP 800-38D], especially section 8, for all of the details and requirements of using AES-GCM mode. In case the module's power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed.
- AES-XTS mode is only approved for hardware storage applications. The length of the XTS-AES data unit does not exceed 2^{20} blocks.
- In order to meet the IG A.13 requirement, the same Triple-DES key shall not be used to encrypt more than 2^{16} 64-bit blocks of data.
- When using AES, the caller must obtain a reference to the cipher implementation via the functions of ccaes_[cbc|ecb|...]_[encrypt|decrypt]_mode.
- When using SHA, the caller must obtain a reference to the cipher implementation via the functions ccsha[1|224|256|384|512]_di.

3 Cryptographic Module Ports and Interfaces

The underlying logical interfaces of the module are the C language Kernel Programming Interfaces (KPIs). In detail these interfaces are the following:

- Data input and data output are provided in the variables passed in the KPI and callable service invocations, generally through caller-supplied buffers. Hereafter, KPIs and callable services will be referred to as "KPI."
- 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.
- 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 documented also in the KPI documentation.

The module is optimized for library use within the macOS kernel and does not contain any terminating assertions or exceptions. It is implemented as an macOS kernel extension. 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. There is one notable exception: RSA and ECDSA do not return a key if the pair-wise consistency test fails.

The function executing FIPS 140-2 module self-tests does not return an error code but causes the system to panic if any self-test fails – see section 9.

The module communicates error status synchronously through the use of documented return codes indicating the module's status. It is the responsibility of the caller to handle exceptional conditions in a FIPS 140-2 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

This section defines the roles, services and authentication mechanisms and methods with respect to the applicable FIPS 140-2 requirements.

4.1 Roles

The module supports a single instance of the two authorized roles: the Crypto Officer and the User. No support is provided for multiple concurrent operators or a maintenance operator.

Role	General Responsibilities and Services (details see below)
User	Utilization of services of the module listed in sections 2.1 and 4.2
Crypto Officer (CO)	Utilization of services of the module listed in sections 2.1 and 4.2.

Table 5: Roles

4.2 Services

The module provides services to authorized operators of either the User or Crypto Officer roles according to the applicable FIPS 140-2 security requirements.

Table 6 contains the cryptographic functions employed by the module in the Approved Mode. For each available service it lists, the associated role, the Critical Security Parameters (CSPs) and cryptographic keys involved, and the type(s) of access to the CSPs and cryptographic keys.

CSPs contain security-related information (secret and private cryptographic keys, for example) whose disclosure or modification can compromise the main security objective of the module, namely the protection of sensitive information.

The access types are denoted as follows:

- R: the item is read/execute or referenced by the service
- W: the item is written or updated by the service
- Z: the persistent item is zeroized by the service

Service	Roles		CSPs & crypto keys	Access Type
	User	CO		
Triple-DES Encryption *Input:* plaintext, IV, key *Output:* ciphertext Decryption *Input:* ciphertext, IV, key *Output:* plaintext	X	X	Triple-DES key	R
AES Encryption / Decryption Encryption *Input:* plaintext, IV, key *Output:* ciphertext Decryption *Input:* ciphertext, IV, key *Output:* plaintext	X	X	AES key	R

Service	Roles		CSPs & crypto keys	Access Type
	User	CO		
AES Key Wrapping Encryption *Input:* plaintext, key *Output:* ciphertext Decryption *Input:* ciphertext, key *Output:* plaintext	X	X	AES key	R
RSA Key Wrapping Encryption *Input:* plaintext, RSA public key, SHA algorithm *Output:* ciphertext Decryption *Input:* ciphertext, RSA private key, SHA algorithm *Output:* plaintext	X	X	RSA key pair	R
Secure Hash Generation *Input:* message *Output:* message digest	X	X	None	N/A
HMAC generation *Input:* HMAC key, message *Output:* HMAC value of message	X	X	Secret HMAC key	R
RSA signature generation and verification Signature generation *Input:* the modulus n, the private key d, the SHA algorithm (SHA -224/ SHA-256/ SHA-384 /SHA-512), a message m to be signed *Output:* the signature s of the message Signature verification *Input:* the modulus n, the public key e, SHA algorithm (SHA-1/SHA -224/ SHA-256/ SHA-384/ SHA-512), a message m, a signature for the message *Output:* pass if the signature is valid, fail if the signature is invalid	X	X	RSA key pair	R
ECDSA signature generation and verification Signature generation *Input:* message m,q, a, b, X_G, Y_G, n, SHA algorithm (SHA-224/ SHA-256/ SHA-384/ SHA-512), sender's private key d *Output:* signature of m as a pair of r and s Signature verification *Input:* received message m', signature in form on r' and s',pair, q, a, b, X_G, Y_G, n, sender's public key Q, SHA algorithm (SHA-1/ SHA-224 / SHA-256/ SHA-384/ SHA-512) *Output:* pass if the signature is valid, fail if the signature is invalid	X	X	ECDSA key pair	R
Random number generation *Input:* Entropy Input, Nonce, Personalization String *Output:* Returned Bits	X	X	Entropy input string, Nonce, V and K	R W Z

Service	Roles		CSPs & crypto keys	Access Type
	User	CO		
RSA key pair generation Input: private key d, public key Q, modulus with size 2048-bits, 3072 or 4096-bits Output: modulus n, public exponent e, private signature exponent d	X	X	RSA key pair	W
ECDSA key pair generation Input: q, FR, a, b, domain_parameter_seed, G, n, h. Output: private key d, public key Q	X	X	ECDSA key pair	W
PBKDF Password-based key derivation *Input:* salt, password, Iteration count, key length. *Output:* derived key	X	X	key derivation, password	R W Z
Release all resources of symmetric crypto function context *Input:* context *Output:* N/A	X	X	AES / Triple-DES key	Z
Release all resources of hash context *Input:* context *Output:* N/A	X	X	HMAC key	Z
Release all resources of asymmetric crypto function context *Input:* context *Output:* N/A	X	X	RSA/ ECDSA keys	Z
Reboot	X	X	N/A	N/A
Self-test	X	X	Software integrity key	R
Show Status	X	X	None	N/A

Table 6: Approved and Allowed Services in Approved Mode

Service	Roles	
	User	CO
Integrated Encryption Scheme on elliptic curves encryption and decryption	X	X
DES Encryption / Decryption	X	X
Triple-DES (Optimized Assembler: asm_x86 Implementation) Encryption / Decryption Mode: CTR	X	X
Triple-DES Encryption / Decryption with Two-Key implementation	X	X
CAST5 Encryption / Decryption	X	X
Blowfish Encryption / Decryption	X	X
RC2 Encryption / Decryption	X	X
RC4 Encryption / Decryption	X	X
MD2 Hash	X	X
MD4 Hash	X	X
RIPEMD Hash	X	X

Service	Roles	
	User	CO
RSA Key Wrapping with RSA-KTS-OAEP-with all key sizes- , with RSA PKCS#1 v1.5 and PSS using key sizes < 2048	X	X
RSA ANSI X9.31 Key Pair Generation Key sizes (modulus): 1024-4096 bits in multiple of 32 bits not listed in Table 3 Public key exponent values: 65537 or larger	X	X
RSA Signature Generation with PKCS#1 v1.5 and PSS Key Sizes: 1024-4096-bits in multiple of 32 bits not listed in Table 3	X	X
RSA Signature Verification with PKCS#1 v1.5 and PSS Key sizes: 1536 bits,	X	X
ECDSA Key Pair Generation for compact point representation of points	X	X
ECDSA PKG: curves P-192 PKV: curves P-192 SIG(gen): curves P-192 SIG(ver): curves P-192	X	X
Ed25519 Key agreement, Signature Generation, Signature Verification	X	X
[SP800-56C] Key Derivation Function	X	X
Hash based Key Derivation Function using ANSI X9.63	X	X
[SP800-108] Key Derivation Function using HMAC-SHA1 or HMAC-SHA-224 or HMAC-SHA-256 or HMAC-SHA-384 or HMAC-SHA-512 Based Pseudo Random Functions Modes: Feedback, CTR	X	X
RFC6637 Key Derivation Function	X	X
AES-CMAC (AES-128/192/256) MAC Generation/Verification	X	X
OMAC MAC Generation	X	X

Table 7: Non-Approved Services in Non-Approved Mode

4.3 Operator authentication

Within the constraints of FIPS 140-2 level 1, the module does not implement an authentication mechanism for operator authentication. The assumption of a role is implicit in the action taken.

The module relies upon the operating system for any operator authentication.

5 Physical Security

The Apple corecrypto Kernel Space Module for Intel (ccv10) is a software cryptographic module running on a multi-chip standalone general-purpose computer. This module being a software, it is not subject to the FIPS 140-2 physical security requirements.

6 Operational Environment

6.1 Applicability

The Apple corecrypto Kernel Space Module for Intel (ccv10) operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module is included in macOS Catalina 10.15 a commercially available general-purpose operating system executing on the hardware specified in section 2.1.3

6.2 Policy

The operating system is restricted to a single-user mode of operation of the module (single-user mode; concurrent operators are explicitly excluded).

FIPS Self-Test functionality is invoked along with mandatory FIPS 140-2 tests when the module is loaded into memory by the operating system.

7 Cryptographic Key Management

The following Table summarizes the cryptographic keys and CSPs used in the Apple corecrypto Kernel Space Module for Intel (ccv10) with the ley lengths supported, the available methods for key generation, key entry and key output and zeroization.

Name	Key / CSP Size	Generation	Entry / Output	Zeroization
AES Keys	128, 192, 256 bits	N/A. Supplied by the caller	Entry : calling application (see 7.4) Output: calling application (see 7.4)	automatic zeroization when structure is deallocated or when the system is powered down (see 7.6).
Triple-DES Keys	192 bits			
ECDSA key pair	P-224, P-256, P-384, P-521 curves	The private keys are generated using FIPS186-4 Key Generation method, and the random value used in the key generation is generated using SP800-90A DRBG.		
RSA key pair	2048-4096 bits			
Entropy input string		Obtained from the NDRNG	Entry: OS Output: N/A	
DRBG nonce		Obtained from the DRBG		
DRBG V, Key		Derived from DRBG input string as defined by SP800-90A	Entry: N/A Output: N/A	
HMAC Keys	min 112 bits	N/A. Supplied by the caller	Entry : calling application (see 7.4) Output: calling application (see 7.4)	
PBKDF Keys	min 112 bits	Internally generated via SP800-132 PBKDF key derivation algorithm	Entry: N/A Output: calling application (see 7.4)	
PBKDF Password		N/A. Provided by calling user	Entry: calling application (see 7.4) Output: N/A	

Table 8: Module Cryptographic key and CSPs

7.1 Random Number Generation

The module uses a FIPS 140-2 approved deterministic random bit generator (DRBG) based on a block cipher as specified in NIST [SP 800-90A]. The default Approved DRBG used for random number generation (i.e., random padding, nonce/salt generation, etc.) is a CTR_DRBG using AES-256 with derivation function and without prediction resistance. The module also provides the caller with additional random number generation functionality through a HMAC-DRBG which can be configure by the caller. Seeding is obtained by the seed source interface read_random (a general purpose kernel-internal function) that extracts random bits from the entropy pool. The NDRNG feeds entropy from the pool into the DRBG on demand. The NDRNG provides 256-bits of entropy.

7.2 Key / CSP Generation

The following approved key generation methods are used by the module:

- The module does not implement symmetric key generation.
- In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per [SP800-133] (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 obtained from [SP800-90A] DRBG. The generated seed is an unmodified output from the DRBG. The key generation service for RSA and ECDSA as well as the [SP 800-90A] DRBG have been ACVT tested with algorithm certificates found in Table 3

The module does not output any information or intermediate results during the key generation process. The DRBG itself is single-threaded.

7.3 Key / CSP Establishment

The module provides the following key establishment services in the Approved Mode:

- AES key wrapping using KW, CCM and GCM modes,
- RSA key wrapping, using PKCS#1 v1.5 and PSS modes, non-approved but allowed per IG D.9.
- PBKDFv2 [SP800-132]. The [SP800-132] PBKDFv2 function is provided as a service and returns the key derived from the provided password to the caller. The caller shall observe all requirements and should consider all recommendations specified in [SP800-132] with respect to the strength of the generated key, including the quality of the salt as well as the number of iterations. The implementation of the PBKDFv2 function requires the user to provide this information.

The encryption strengths for the key establishment methods are determined in accordance with FIPS 140-2 Implementation Guidance [IG] section 7.5 and NIST Special Publication 800-57 (Part1) [SP800-57P1].

- AES key wrapping is used for key establishment. Methodology provides between 128 and 256 bits of encryption strength.
- RSA key wrapping is used for key establishment. Methodology provides between 112 and 152 bits of encryption strength.

7.4 Key / CSP Entry and Output

All keys are entered from, or output to, the invoking kernel service running on the same device. All keys entered into the module are electronically entered in plain text form. Keys are output from the module in plain text form if required by the calling kernel service. The same holds for the CSPs.

7.5 Key / CSP Storage

The Apple corecrypto Kernel Space Module for Intel (ccv10) considers all keys in memory to be ephemeral. They are received for use or generated by the module only at the command of the calling kernel service. The same holds for CSPs.

The module protects all keys, secret or private, and CSPs through the memory protection mechanisms provided by macOS, including the separation between the kernel and user-space. No process can read the memory of another process. No user-space application can read the kernel memory.

7.6 Key / CSP Zeroization

Keys and CSPs are zeroized when the appropriate context object is destroyed or when the system is powered down.

8 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The EMI/EMC properties of the corecrypto KEXT are not meaningful for the software library. The devices containing the software components of the module have their own overall EMI/EMC rating. The validation test environments have FCC, part 15, Class B rating.

9 Self-Tests

FIPS 140-2 requires that the module performs self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition, the DRBG requires continuous verification. The FIPS Self-Tests functionality runs all required module self-tests. This functionality is invoked by the macOS Kernel startup process upon device initialization. If the self-tests succeed, the corecrypto KEXT instance is maintained in the memory of the macOS Kernel on the device and made available to each calling kernel service without reloading. All self-tests performed by the module are listed and described in this section.

9.1 Power-Up Tests

The following tests are performed each time the Apple corecrypto Kernel Space Module for Intel (ccv10) starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the following tests fails the system shuts down automatically. To run the self-tests on demand, the user may reboot the system.

9.1.1 Cryptographic Algorithm Tests

Algorithm	Modes	Test
Triple-DES	CBC	KAT (Known Answer Test) Separate encryption / decryption operations are performed
AES implementations selected by the module for the corresponding environment AES-128	ECB, CBC, XTS, GCM, CCM	KAT Separate encryption / decryption operations are performed
DRBG (CTR_DRBG and HMAC_DRBG; tested separately)	N/A	KAT
HMAC implementations selected by the module for the corresponding environment HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT
ECDSA	Signature Generation, Signature Verification	pair-wise consistency test
RSA	Signature Generation Signature Verification	KAT

Table 9: Cryptographic Algorithm Tests

9.1.2 Software / Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple corecrypto Kernel Space Module for Intel (ccv10). The corecrypto's HMAC-SHA-256 is used as an approved algorithm for the integrity test. If the test fails, then the system shuts down automatically.

9.1.3 Critical Function Tests

No other critical function test is performed on power up.

9.2 Conditional Tests

The following sections describe the conditional tests supported by the Apple corecrypto Kernel Space Module for Intel (ccv10).

9.2.1 Continuous Random Number Generator Test

The Apple corecrypto Kernel Space Module for Intel (ccv10) performs a continuous random number generator test, whenever it is invoked to seed the [SP800-90A] DRBG

.

9.2.2 Pair-wise Consistency Test

The Apple corecrypto Kernel Space Module for Intel (ccv10) generates RSA and ECDSA asymmetric keys and performs all required pair-wise consistency tests with the newly generated key pairs.

9.2.3 SP 800-90A Assurance Tests

The Apple corecrypto Kernel Space Module for Intel (ccv10) performs the health tests as specified in section 11.3 of [SP800-90A]

9.2.4 Critical Function Test

No other critical function test is performed conditionally.

10 Design Assurance

10.1 Configuration Management

Apple manages and records source code and associated documentation files by using the revision control system called “Git”.
Apple module hardware data, which includes descriptions, parts data, part types, bills of materials, manufacturers, changes, history, and documentation are managed and recorded. Additionally, configuration management is provided for the module’s FIPS documentation.

The following naming/numbering convention for documentation is applied.

<evaluation>_<module>_<os>_<mode>_<doc name>_<doc version (##.##)>

Example: FIPS_CORECRYPTO_MACOS_KS_SECPOL_5.0

Document management utilities provide access control, versioning, and logging. Access to the Git repository (source tree) is granted or denied by the server administrator in accordance with company and team policy.

10.2 Delivery and Operation

The corecrypto KEXT is built into macOS Catalina For additional assurance, it is digitally signed. The Approved Mode is configured by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved algorithms listed in Table 4.

10.3 Development

The Apple crypto module (like any other Apple software) undergoes frequent builds utilizing a “train” philosophy. Source code is submitted to the Build and Integration group (B & I). B & I builds, integrates and does basic sanity checking on the operating systems and apps that they produce. Copies of older versions are archived offsite in underground granite vaults.

10.4 Guidance

The following guidance items are to be used for assistance in maintaining the module’s validated status while in use.

10.4.1 Cryptographic 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 algorithms listed in Table 4. If the device starts up successfully then corecrypto KEXT 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 140-2 Compliance of macOS Catalina v10.15 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 [UGuide].

10.4.2 User 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 algorithms listed in Table 4. If the device starts up successfully then corecrypto KEXT has passed all self-tests and is operating in the Approved Mode. Kernel programmers that use the module KPI shall not attempt to invoke any KPI call directly and only adhere to defined interfaces through the kernel framework.

11 Mitigation of Other Attacks

The module protects against the utilization of known Triple-DES weak keys. The following keys are not permitted:

{0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},

{0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE},

{0x1F,0x1F,0x1F,0x1F,0x0E,0x0E,0x0E,0x0E},

{0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},

{0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},

{0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},

{0x1F,0xE0,0x1F,0xE0,0x0E,0xF1,0x0E,0xF1},

{0xE0,0x1F,0xE0,0x1F,0xF1,0x0E,0xF1,0x0E},

{0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},

{0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01},

{0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E,0xFE},

{0xFE,0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E},

{0x01,0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E},

{0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E,0x01},

{0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1,0xFE},

{0xFE,0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1}.