Apple Inc.

Apple CoreCrypto Kernel Module v8.0 for Intel FIPS 140-2 Non-Proprietary Security Policy

March, 2018

Prepared for:
Apple Inc.
1 Infinite Loop
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 Module v8.0 for Intel. 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 validation program please refer to the NIST website at http://csrc.nist.gov/groups/STM/cmvp/.

Throughout the document "Apple CoreCrypto Kernel Module v8.0 for Intel", "cryptographic module", "CoreCrypto KEXT" or "the module" are used interchangeably to refer to the Apple CoreCrypto Kernel Module v8.0 for Intel.

1.2 Document Organization / Copyright

This non-proprietary Security Policy document may be reproduced and distributed only in its original entirety without any revision, ©2018 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 its 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 Apple Knowledge Base Article HT201159 - "Product security certifications, validations, and guidance for macOS" (https://support.apple.com/en-us/HT201159)

The Cryptographic Module Validation Program website (http://csrc.nist.gov/groups/STM/cmvp/index.html) contains links to the FIPS 140-2 certificate and Apple Inc. contact information.

1.3.1 Additional References

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," November 15, 2016 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 800-38E, “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),” March 2007
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 Location: https://developer.apple.com/library/mac/navigation/#section=Topics&topic=Security
MACOS	macOS Technical Overview Location: https://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/About.html
UG	User Guide Location: https://support.apple.com/en-us/HT201159

1.4 Acronyms

Acronyms found in this document are defined as follows:

AES	Advanced Encryption Standard
BS	Block Size
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
DS	Digest Size
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
KS	Key Size (Length)
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
RNG	Random Number Generator
SHS	Secure Hash Standard
Triple-DES	Triple Data Encryption Standard

2 Cryptographic Module Specification

2.1 Module Description

The Apple CoreCrypto Kernel Module v8.0 for Intel is a software cryptographic module 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 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 Module v8.0 for Intel 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 Core M CPU	macOS High Sierra 10.13[1]
Apple Inc.	MacBook Pro with Core i7 CPU	macOS High Sierra 10.13
Apple Inc.	Mac mini with i5 CPU	macOS High Sierra 10.13
Apple Inc.	iMac Pro with Xeon CPU	macOS High Sierra 10.13

Table 2: Tested Platforms

2.2 Modes of operation

The Apple CoreCrypto Kernel Module v8.0 for Intel 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 (CSP) 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 implementation (AES-NI).

Approved security functions are listed in Table 3. Column four (Algorithm Certificate Number) of Table 3 lists the validation numbers obtained from NIST based on the successful CAVP testing of the cryptographic algorithm implementations on the platforms referenced in Table 2

Refer to http://csrc.nist.gov/groups/STM/cavp/index.html for the current standards, test requirements, and special abbreviations used in the following table.

2.2.1 Approved Security Functions

Cryptographic Function	Algorithm	Options	Algorithm Certificate Number
Random Number Generation; Symmetric key generation	[SP 800-90] DRBG	CTR_DRBG Optimized Assembler Implementation	1804, 1805, 1806, 1807
		CTR_DRBG Optimized Assembler Implementation using VNG	1875, 1876, 1877, 1878
		CTR_DRBG AES-NI with Generic Software Implementation of block chaining modes	1882, 1883, 1888, 1889

[1] macOS High Sierra (version 10.13) is the fourteenth release of macOS (previously OS X). Throughout the document it is generically referred to as macOS High Sierra.

Cryptographic Function	Algorithm	Options	Algorithm Certificate Number
		HMAC_DRBG Generic Software Implementation: HMAC-SHA-384 HMAC-SHA-512	1872, 1873, 1874, 1885
		HMAC_DRBG Optimized Assembler Implementation using SSE3: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	1815, 1816, 1817, 1818
		HMAC_DRBG Optimized Assembler Implementation using AVX: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	1808, 1809, 1810, 1819
		HMAC_DRBG Optimized Assembler Implementation using AVX2: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	1811, 1812, 1813, 1814
Symmetric Encryption and Decryption	[FIPS 197] AES SP 800-38 A SP 800-38 C SP 800-38 E SP 800-38 F	Generic Software Implementation with Optimized Assembler Implementation of block chaining modes: ECB CFB128 OFB CBC CTR CFB8 KW	4985, 4986, 4987, 4988
		Optimized Assembler Implementation: Modes: ECB KW CBC XTS	4993, 4994, 4995, 4996
		Optimized Assembler Implementation using VNG: Modes: ECB GCM CCM KW	5054, 5055, 5056, 5057
		AES-NI with Optimized Assembler Implementation of block chaining modes: ECB KW CBC XTS	4989, 4990, 4991, 4992

Cryptographic Function	Algorithm	Options	Algorithm Certificate Number
		AES-NI with Generic Software Implementation of block chaining modes: ECB CFB128 OFB CBC CTR XTS CCM GCM CFB8 KW	5063, 5065, 5071, 5072
	[SP 800-67] Triple-DES	3-key Triple-DES (All keys independent) Modes: ECB CBC	2613, 2614, 2615, 2618
Digital Signature and Asymmetric Key Generation	[FIPS186-4] RSA PKCS #1.5	SigVerPKCS1.5 Key Sizes: 1024 2048 3072	2740, 2741, 2742, 2748
	[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	1308, 1309, 1310, 1313
Message Digest	[FIPS 180-4] SHS	Generic Software Implementation: SHA-384 SHA-512	4120, 4121, 4122, 4127
		Optimized Assembler Implementation using SSSE3: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4058, 4059, 4060, 4061
		Optimized Assembler Implementation using AVX: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4051, 4052, 4053, 4062
		Optimized Assembler Implementation using AVX2: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4054, 4055, 4056, 4057

Cryptographic Function	Algorithm	Options	Algorithm Certificate Number
Keyed Hash	[FIPS 198] HMAC	Generic Software Implementation: HMAC-SHA-384 HMAC-SHA-512	3375, 3376, 3377, 3382
		Optimized Assembler Implementation using SSSE3: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3315, 3316, 3317, 3318
		Optimized Assembler Implementation using AVX: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3308, 3309, 3310, 3319
		Optimized Assembler Implementation using AVX2: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3311, 3312, 3313, 3314
Key Derivation	[SP 800-132] PBKDF	Password based key derivation using HMAC with SHA-1 or SHA-2	Vendor Affirmed
MD5	Message Digest	Digest Size: 128-bit	Non-Approved, but Allowed[1]
NDRNG	Random Number Generation	N/A	Non-Approved, but Allowed[2]
RSA Key Wrapping	[FIPS 186-4] [SP800-56B]	PKCS#1 v1.5 PKCS#1 v2.1 OAEP	Non-Approved, but allowed[3]

Table 3: Approved, Allowed or Vendor Affirmed Security Functions

CAVEAT: The module generates cryptographic keys whose strengths are modified by available entropy – 160-bits. The encryption strength for the AES Key Wrapping using 192 and 256-bit keys is limited to 160 bits due to the entropy of the seed source.

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.

[1] MD5 is used as part of the TLS key establishment scheme only.

[2] NDRNG is provided by the underlying operational environment.

[3] RSA key wrapping is used for key establishment. Methodology provides 112 or 128 bits of encryption strength

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): curves P-192 Public Key Validation (PKV): curves P-192 Signature Generation: curves 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	RSA Key Wrapping using key sizes < 2048-bits	Non-Approved
RFC6637 KDF	KDF based on RFC 6637	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
	Optimized Assembler Implementation: Encryption / Decryption Mode: CTR	Non-Compliant
AES-CMAC	AES-128 MAC generation	Non-Compliant

Cryptographic Function	Usage / Description	Caveat
SP800-108	KBKDF Modes: CTR, Feedback	Non-Compliant
SP800-56C	Key Derivation Function	Non- Compliant
RSA FIPS186-4 Signature Verification	PKCS#1 v1.5 Signature Verification Key sizes (modulus): 1536 bits, 4096 Hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Non-Compliant

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

The encryption strengths included in Table 4 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].

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 not validated by the CAVP. Neither a Non-Compliant nor a Non-Approved function may be used in the Approved mode unless stated in the caveat found in Table 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.

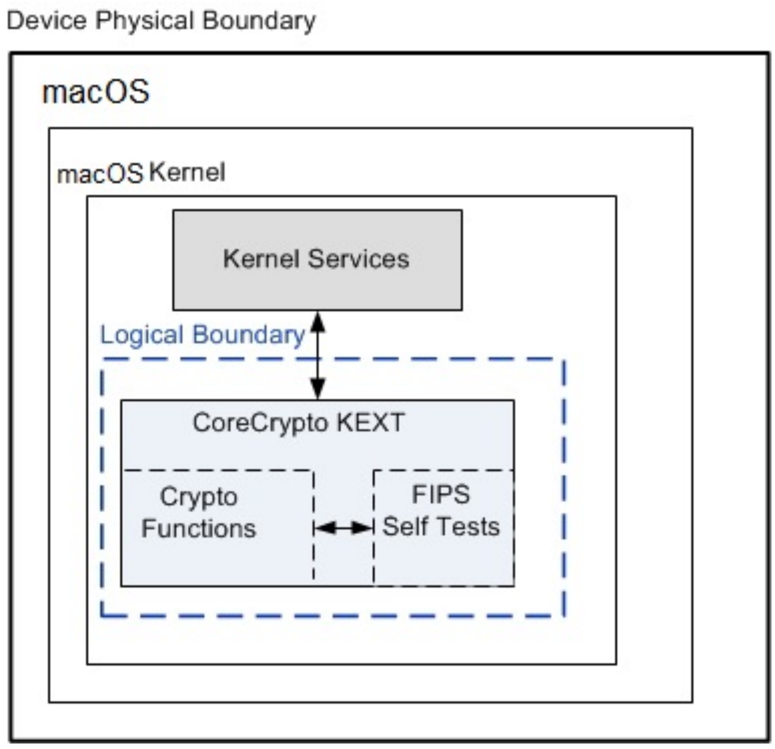

Figure 1: Logical Block Diagram

2.4 Module Usage Considerations

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

- In order to meet the IG A.13 requirement, the same Triple-DES key shall not be used to encrypt more than 2^28 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 are two notable exceptions: (i) ECDSA does not return a key if the pair-wise consistency test fails; (ii) the DRBG algorithm loops a few iterations internally if the continuous test fails, eventually recovering from the error or causing a shutdown if the problem persists.

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 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	Secret key	R

Service	Roles		CSPs & crypto keys	Access Type
	USER	CO		
AES Encryption / Decryption Encryption *Input:* plaintext, IV, key *Output:* ciphertext Decryption *Input:* ciphertext, IV, key *Output:* plaintext	X	X	Secret key	R W
AES Key Wrapping Encryption *Input:* plaintext, key *Output:* ciphertext Decryption *Input:* ciphertext, key *Output:* plaintext	X	X	secret key	R W
RSA Key Wrapping Encryption *Input:* plaintext, RSA public key, the SHA algorithm *Output:* ciphertext Decryption *Input:* ciphertext, RSA private key, the SHA algorithm *Output:* plaintext	X	X	RSA key pair	R W
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 W
RSA signature verification *Input:* the module n, the public key e, the 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 W

Service	**Roles**		**CSPs & crypto keys**	**Access Type**
	USER	CO		
ECDSA Signature generation *Input:* message m, q, a, b, X_G, Y_G, n, the 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, the 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 W
Random number generation *Input:* Entropy Input, Nonce, Personalization String *Output:* Returned Bits	X	X	Entropy input string, Nonce, V and K	R W Z
ECDSA (key pair generation) Input: q, FR, a, b, domain_parameter_seed, G, n, h. Output: private key d, public key Q	X	X	Asymmetric key pair	R W Z
PBKDF Password-based key derivation *Input:* encrypted key and password *Output:* plaintext key or *Input:* plaintext key and password *Output:* encrypted data	X	X	Secret key, 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

Service	Roles		CSPs & crypto keys	Access Type
	USER	CO		
Release all resources of asymmetric crypto function context *Input:* context *Output:* N/A	X	X	Asymmetric keys (ECDSA)	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		Access Type
	USER	CO	
Integrated Encryption Scheme on elliptic curves encryption and decryption	X	X	R
DES Encryption / Decryption	X	X	R
Triple-DES Encryption / Decryption Mode: CTR	X	X	R
Triple-DES Encryption / Decryption with Two-Key implementation	X	X	R
CAST5 Encryption / Decryption	X	X	R
Blowfish Encryption / Decryption	X	X	R
RC2 Encryption / Decryption	X	X	R
RC4 Encryption / Decryption	X	X	R
MD2 Hash	X	X	R W
MD4 Hash	X	X	R W
RIPEMD Hash	X	X	R W

Service	Roles		Access Type
	USER	CO	
RSA Key Wrapping with RSA-OAEP, PKCS#1 v1.5, PKCS#1 v2.1 using key sizes < 2048	X	X	R
RSA Signature Verification with PKCS#1 v1.5 Key sizes: 1536 bits, 4096 bits	X	X	R W
ECDSA Key Pair Generation for compact point representation of points	X	X	R W
ECDSA PKG: curves P-192 PKV: curves P-192 SIG(gen): curves P-192 SIG(ver): curves P-192	X	X	R W
Ed25519 Key agreement, Signature Generation, Signature Verification	X	X	R W
SP800-56C Key Derivation Function	X	X	R W
Hash based Key Derivation Function using ANSI X9.63	X	X	R W
SP800-108 Key Derivation Function Modes: Feedback, CTR	X	X	R W
RFC6637 Key Derivation Function	X	X	R W
AES-CMAC MAC Generation	X	X	R W
OMAC MAC Generation	X	X	R W

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 Module v8.0 for Intel is intended to operate on a multi-chip standalone platform. The device is comprised of production grade components and a production grade enclosure.

6 Operational Environment

The following sections describe the operational environment of the Apple CoreCrypto Kernel Module v8.0 for Intel.

6.1 Applicability

The Apple CoreCrypto Kernel Module v8.0 for Intel operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module is included in macOS High Sierra, 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 section defines the key management features available through the Apple CoreCrypto Kernel Module v8.0 for Intel.

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 is a CTR_DRBG with derivation function and without prediction resistance. The module also employs a HMAC-DRBG for random number generation. Seeding is obtained by read_random (a true random number generator). read_random obtains entropy from interrupts generated by the devices and sensors attached to the system and maintains an entropy pool. The TRNG feeds entropy from the pool into the DRBG on demand. The TRNG provides 160-bits of entropy.

7.2 Key / CSP Generation

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

- The default Approved DRBG specified in section 7.1 is used to generate asymmetric key pairs for the ECDSA algorithm.

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

The cryptographic strength of the 192 and 256-bit AES keys as well as the ECDSA keys for the curve P-384 and P-521, as modified by the available entropy, is limited to 160-bits.

7.3 Key / CSP Establishment

The module provides key establishment services in the Approved Mode through the AES key wrapping and PBKDFv2 algorithm. The RSA key wrapping is non-approved but allowed. The 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.

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 Module v8.0 for Intel 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 Module v8.0 for Intel 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	KAT Separate encryption / decryption operations are performed
DRBG (CTR_DRBG and HMAC_DRBG; tested separately)	N/A	KAT
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT
ECDSA	Signature Generation, Signature Verification	pair-wise consistency test
RSA	Signature Verification	KAT

Table 8: 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 Module v8.0 for Intel. 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 Module v8.0 for Intel.

9.2.1 Continuous Random Number Generator Test

The Apple CoreCrypto Kernel Module v8.0 for Intel performs a continuous random number generator test, whenever the DRBG is invoked.

In addition, the seed source implemented in the operating system kernel also performs a continuous self-test.

9.2.2 Pair-wise Consistency Test

The Apple CoreCrypto Kernel Module v8.0 for Intel generates asymmetric keys and performs all required pair-wise consistency tests (signature generation and verification) with the newly generated key pairs.

9.2.3 SP 800-90A Assurance Tests

The Apple CoreCrypto Kernel Module v8.0 for Intel performs a subset of the assurance tests as specified in section 11 of SP 800-90A, in particular it complies with the mandatory documentation requirements and performs know-answer tests.

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_3.0

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 High Sierra. 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 High Sierra v10.13 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 [UG].

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 API shall not attempt to invoke any API 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}.
```