Apple Inc.

Apple CoreCrypto 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 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 https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program.

Throughout the document "Apple CoreCrypto Module v8.0 for Intel", "cryptographic module", "CoreCrypto" or "the module" are used interchangeably to refer to the Apple CoreCrypto Module v8.0 for Intel. macOS High Sierra (version 10.13) is the thirteenth release of macOS (previously OS X). Throughout the document it is generically referred to as macOS High Sierra or macOS.

1.2 Document Organization / Copyright

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," May 2001

FIPS 140-2 IG NIST, "Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program," December 6, 2016

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)

FIPS 198 Federal Information Processing Standards Publication 198, July, 2008 The Keyed-Hash Message Authentication Code (HMAC)

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 D NIST Special Publication 800-38D, “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC”, November 2007

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),” July 2012

SP 800-90A NIST Special Publication 800-90, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised),” January 2012

SP800-132 NIST Special Publication 800-132, “Recommendation for Password-Based Key Derivation”, December 2010

MACOS macOS Technical Overview

https://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/About.html

SEC Security Overview

https://developer.apple.com/security/

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
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Codebook mode of operation
ECC	Elliptic Curve Cryptography
EC Diffie-Hellman	Diffie-Hellman based on ECC
ECDSA	DSA based on ECC
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
HMAC	Hash-Based Message Authentication Code
KAT	Known Answer Test
KDF	Key Derivation Function
KPI	Kernel Programming Interface
KS	Key Size (Length)
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OFB	Output Feedback (mode of operation)
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
TLS	Transport Layer Security

2 Cryptographic Module Specification

2.1 Module Description

The Apple CoreCrypto Module v8.0 for Intel is a software cryptographic module running on a multi-chip standalone general-purpose computing platform.

The cryptographic services provided by the module are:

- data encryption / decryption
- generation of hash values
- key wrapping
- message authentication
- random number generation
- key generation
- signature generation / verification
- key derivation

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 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 an API layer that provides consistent interfaces to the supported algorithms. These implementations include proprietary optimizations of algorithms that are fitted into the CoreCrypto framework.

2.1.2.2 Hardware components

AES-NI hardware acceleration is included 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 v10.13[1]
Apple Inc.	MacBook Pro with Core i7 CPU	macOS High Sierra v10.13
Apple Inc.	Mac mini with i5 CPU	macOS High Sierra v10.13
Apple Inc.	iMac Pro with Xeon CPU	macOS High Sierra v10.13

Table 2: Tested Platforms

2.2 Modes of operation

The Apple CoreCrypto Module v8.0 for Intel has an Approved and non-Approved mode of operation. The Approved mode of operation is configured by default and cannot be changed. If the device starts up successfully the CoreCrypto framework 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).

The Approved security functions are listed in Table 3. Column four (Algorithm Certificate) lists the validation numbers obtained from NIST for successful validation testing of the implementation of the cryptographic algorithms on the platforms as shown in Table 2 under CAVP.

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
Random Number Generation; Symmetric key generation	[SP 800-90] DRBG	CTR_DRBG Generic Software Implementation	1868, 1869, 1870, 1871
		CTR_DRBG AES Optimized Assembler Implementation	1793, 1794, 1795, 1796
		CTR_DRBG AES Optimized Assembler Implementation using VNG	1771, 1772, 1773, 1774

[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
		CTR_DRBG AES-NI Hardware Implementation	1789, 1790, 1791, 1792
		HMAC_DRBG Generic Software Implementation	1868, 1869, 1870, 1871
		HMAC_DRBG Generic Software Implementation using AVX	1779, 1780, 1781, 1782
		HMAC_DRBG Generic Software Implementation using AVX2	1783, 1784, 1785, 1786
		HMAC_DRBG Generic Software Implementation using SSE3	1775, 1776, 1777, 1778
Symmetric Encryption and Decryption	[FIPS 197] AES SP 800-38 A SP 800-38 C SP 800-38 D SP 800-38 E SP 800-38 F	Generic Software Implementation (Based on LibTomCrypt): Modes: ECB CTR CBC GCM CCM KW CFB8 OFB CFB128 XTS	5048, 5049, 5050, 5051
		Generic Software Implementation (Based on Gladman): Modes: CBC	4952, 4953, 4954, 4955
		Generic Software Implementation with Optimized Assembler Core: ECB CTR CBC GCM CCM KW CFB8 OFB CFB128 XTS	4973, 4974, 4975, 4976
		Optimized Assembler Implementation: Modes: ECB CBC KW XTS	4965, 4966, 4967, 4968
		Optimized Assembler Implementation using AES-NI: Modes: ECB KW CBC XTS	4961, 4962, 4963, 4964

Cryptographic Function	Algorithm	Options	Algorithm Certificate
		Generic Software Implementation using AES-NI: Modes: ECB CTR CBC GCM CCM KW CFB8 OFB CFB128 XTS	4969, 4970, 4971, 4972
		Generic Software Implementation using VNG: ECB KW CCM CTR	4948, 4949, 4950, 4951
		Optimized Assembler Implementation using VNG: ECB GCM CCM KW CTR	4944, 4945, 4946, 4947
	[SP 800-67] Triple-DES	3-Key Triple-DES (All Keys Independent) Modes: ECB CFB64 CBC OFB CFB8 CTR	2609, 2610, 2611, 2612
Digital Signature and Asymmetric Key Generation	[FIPS186-4] RSA PKCS #1.5	GenKey9.31 Signature Generation PKCS1.5 Key Sizes (Modulus): 2048 3072 Signature Verification PKCS1.5 Key Sizes (Modulus): 1024 2048 3072	2734, 2735, 2736, 2737
	[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	1304, 1305, 1306, 1307
		Signature Generation Component:	CVL: 1602, 1604, 1606, 1608

Cryptographic Function	Algorithm	Options	Algorithm Certificate
Message Digest	[FIPS 180-4] SHS	Generic Software Implementation SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4116, 4117, 4118, 4119
		Optimized Software Implementation using SSSE3: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4034, 4035, 4036, 4037
		Optimized Software Implementation using AVX: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4038, 4039, 4040, 4041
		Optimized Software Implementation using AVX2: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4042, 4043, 4044, 4045
Keyed Hash	[FIPS 198] HMAC	Generic Software Implementation HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3371, 3372, 3373, 3374
		Optimized Software Implementation using SSSE3 HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3292, 3293, 3294, 3295
		Optimized Software Implementation using AVX: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3296, 3297, 3298, 3299
		Optimized Software Implementation using AVX2: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3300, 3301, 3302, 3303
KAS	[SP800-56A] EC Diffie-Hellman Implementation follows SP800-56A for primitive only.	6.2.2.2 One-Pass Diffie-Hellman, C(1e, 1s, ECC CDH) Curves: P-224, P-256, P-384, P-521	CVL: 1603, 1605, 1607, 1609

Cryptographic Function	Algorithm	Options	Algorithm Certificate
Key Derivation	[SP 800-132] PBKDF	Password Based Key Derivation using HMAC with SHA-1 or SHA-2	Vendor Affirmed
RSA Key Wrapping	[SP 800-56B]	KTS-OAEP	Vendor Affirmed
	[FIPS 186-4]	PKCS#1 v1.5 PKCS#1 v2.1	Non-Approved, but allowed[1]
Key Agreement	ANSI X9.63 SP 800-56A EC Diffie-Hellman	ECC curves P-256, P-384	Non-Approved, but allowed[2]
	[ANSI X9.42] [SP 800-56A] Diffie-Hellman	Key sizes: 2048 bits 3072 bits	Non-Approved, but allowed[3]
MD5	Message Digest	Digest Size: 128-bit	Non-Approved, but Allowed[4]
NDRNG	Random Number Generation	N/A	Non-Approved, but Allowed[5]

Table 3: Approved, Allowed and 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 7.3.

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

2 EC Diffie-Hellman is used for key establishment. Methodology provides 128 bits or 160 bits of encryption strength - the strength for P-384 is limited by the entropy of the seed source as specified in the caveat.

3 Diffie-Hellman is used for key establishment; Methodology provides 112 or 128 bits of encryption strength.

4 MD5 is used as part of the TLS key establishment scheme only.

5 NDRNG is provided by the underlying operational environment.

2.2.2 Non-Approved Security Functions:

Cryptographic Function	Usage / Description	Caveat
RSA Signature Generation / Signature Verification / Asymmetric Key Generation	ANSI X9.31 Signature Generation Key Pair Generation Key Size < 2048 Signature Verification Key Size < 1024	Non-Approved
	PKCS#1 v1.5 Signature Generation Key Size < 2048 Signature Verification Key Size < 1024	
RSA Key Wrapping	PCKS#1 v1.5, PKCS#1 v2.1 Key Size < 2048	Non-Approved
Diffie-Hellman	Key Agreement ANSI X9.42, SP 800-56A Key sizes < 2048 bits	Non-Approved
Ed25519	Key Agreement Sig(gen) Sig(ver)	Non-Approved
ANSI X9.63	Hash based Key Derivation Function	Non-Approved
RFC6637	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
RIPEMD	Message Digest Digest size 128, 160, 256, 320 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
ECDSA	Key Pair Generation for compact point representation of points	Non-Approved

Cryptographic Function	Usage / Description	Caveat
Integrated Encryption Scheme on elliptic curves	Encryption / Decryption	Non-Approved
Blowfish	Encryption / Decryption	Non-Approved
OMAC (One-Key CBC MAC)	MAC generation	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
SP800-108 KBKDF	Key Based Key Derivation Function Modes: CTR and Feedback	Non-Compliant
SP800-56C	Key Derivation Function	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 otherwise 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.

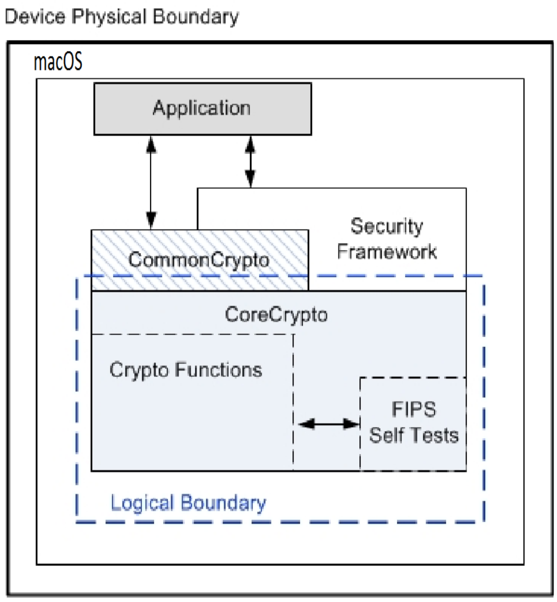

Figure 1: Logical Block Diagram

2.4 Module Usage Considerations

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

- AES-GCM IV is constructed in accordance with SP800-38D section 8.2.1. 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.

- 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.

- 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.

3 Cryptographic Module Ports and Interfaces

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

- Data input and data output are provided in the variables passed in the API and callable service invocations, generally through caller-supplied buffers. Hereafter, APIs and callable services will be referred to as “API.”
- Control inputs which control the mode of the module are provided through dedicated parameters, as well as mach-o header holding the HMAC check file.
- Status output is provided in return codes and through messages. Documentation for each API lists possible return codes. A complete list of all return codes returned by the C language APIs within the module is provided in the header files and the API documentation. Messages are documented also in the API documentation.

The module is optimized for library use within the macOS user space and does not contain any terminating assertions or exceptions. It is implemented as an macOS dynamically loadable library. The dynamically loadable library is loaded into the macOS application and its cryptographic functions are made available. Any internal error detected by the module is reflected back to the caller with an appropriate return code. The calling macOS application must examine the return code and act accordingly. There are two notable exceptions: (i) ECDSA and RSA do 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 crash if any self-test fails – see Section 9.

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 140-2 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 section 2.1 and 4.2.
Crypto Officer (CO)	Utilization of services of the module listed in section 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 (for example, secret and private cryptographic keys) 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 / Decryption	X	X	secret key	R
AES Encryption / Decryption	X	X	secret key	R
AES Key Wrapping	X	X	secret key	R
RSA Key Wrapping	x	x	RSA Private Key	R
Secure Hash Generation	X	X	none	N/A
HMAC generation	X	X	secret HMAC key	R
RSA signature generation and verification	X	X	RSA key pair	R W
ECDSA signature generation and verification	X	X	ECDSA key pair	R W
Random number generation	X	X	Entropy input string, Nonce, V and K	R W Z

Service	Roles		CSPs & Crypto Keys	Access Type
	USER	CO		
PBKDF Password-based key derivation	X	X	secret key, password	R W Z
RSA (key pair generation)	X	X	Asymmetric key pair	R W
ECDSA (key pair generation)	X	X	Asymmetric key pair	R W
Diffie-Hellman Key agreement	X	X	Asymmetric keys (RSA/ECDSA key) and secret session key (AES/Triple-DES key)	R W
EC Diffie-Hellman Key agreement	X	X	Asymmetric keys (RSA/ECDSA key) and secret session key (AES/Triple-DES key)	R W
Release all resources of symmetric crypto function context	X	X	AES/Triple-DES key	Z
Release all resources of hash context	X	X	HMAC key	Z
Release of all resources of Diffie-Hellman context for Diffie-Hellman and EC Diffie-Hellman	X	X	Asymmetric keys (RSA/ECDSA) and secret session key (AES/Triple-DES)	Z
Release of all resources of asymmetric crypto function context	X	X	RSA/ECDSA keys	Z
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 / Decryption	X	X	R
DES Encryption / Decryption	X	X	R
Triple-DES (Optimized Assembler Implementation) Encryption / Decryption Mode: CTR	X	X	R
Triple-DES (Two-Key implementation) Encryption / Decryption	X	X	R

Service	Roles		Access Type
	USER	CO	
CAST5 Encryption / Decryption	X	X	R
Blowfish Encryption / Decryption	X	X	R
RC4 Encryption / Decryption	X	X	R W
RC2 Encryption / Decryption	X	X	R W
MD2 Hash	X	X	R W
MD4 Hash	X	X	R W
RIPEMD Hash	X	X	R W
RSA Key Wrapping using Key Size < 2048	X	X	R
RSA ANSI X9.31 Signature Generation and Verification	X	X	R W
RSA PKCS#1 v1.5 Signature Generation and Verification Key Sizes: 1024-4096-bits in multiple of 32 bits not listed in Table 3	X	X	R W
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	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
Diffie-Hellman Key Agreement Key Sizes < 2048 bits	X	X	R W
Ed25519 Key agreement, Signature Generation, Signature Verification	X	X	R W
SP800-56C Key Derivation Function	X	X	R W
ANSI X9.63 Hash based Key Derivation Function using	X	X	R W
SP800-108 Key Derivation Function Modes: Feedback, Counter	X	X	R W
RFC6637 Key Derivation Function	X	X	R W
AES-CMAC AES-128 MAC Generation	X	X	R W

Service	Roles		Access Type
	USER	CO	
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 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 Module v8.0 for Intel.

6.1 Applicability

The Apple CoreCrypto Module v8.0 for Intel operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. It is part of 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 operator (single-user mode; i.e. concurrent operators are explicitly excluded).

When the operating system loads the module into memory, it invokes the FIPS Self-Test functionality, which in turn runs the mandatory FIPS 140-2 tests.

7 Cryptographic Key Management

The following section defines the key management features available through the Apple CoreCrypto Module v8.0 for Intel.

7.1 Random Number Generation

A FIPS 140-2 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-128 with derivation function and without prediction resistance. The module also employs a HMAC-DRBG for random number generation. The deterministic random bit generators are seeded by /dev/random. The /dev/random generator is a true random number generator that 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 and RSA algorithm.

The module does not output any information or intermediate results during the key generation process. The RNG 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, as modified by the available entropy, is limited to 160-bits.

7.3 Key / CSP Establishment

The module provides AES key wrapping, RSA key wrapping, Diffie-Hellman- and EC Diffie-Hellman-based key establishment services.

In addition, the module also implements vendor affirmed PBKDFv2 algorithm. 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 application 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 application. The same holds for the CSPs.

7.5 Key / CSP Storage

The Apple CoreCrypto 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 application. The same holds for CSPs.

The module protects all keys, secret or private, and CSPs through the memory protection mechanisms provided by the operating system. No process can read the memory of another process.

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 Apple CoreCrypto Module v8.0 for Intel 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 perform 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 application runs all required module self-tests. This application is invoked by the macOS startup process upon device initialization.

The execution of an independent application for invoking the self-tests in the libcorecrypto.dylib makes use of features of the macOS architecture: the module, implemented in libcorecrypto.dylib, is linked by libcommoncrypto.dylib which is linked by libSystem.dylib. The libSystem.dylib is a library that must be loaded into every application for operation. The operating system ensures that there is a strict CSP separation between the instances used by each application.

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 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 device fails to startup. To invoke 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	CBC, ECB, GCM, 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
RSA	Signature Generation, Signature Verification Encrypt / Decrypt	KAT, pair-wise consistency test Separate encryption /decryption operations are performed
ECDSA	Signature Generation, Signature Verification	pair-wise consistency test
Diffie-Hellman “Z” computation	N/A	KAT
EC Diffie-Hellman “Z” computation	N/A	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 Module v8.0 for Intel. The CoreCrypto’s HMAC-SHA256 is used as an approved algorithm for the integrity test. If the test fails, then the device powers itself off.

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 Module v8.0 for Intel.

9.2.1 Continuous Random Number Generator Test

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

9.2.3 SP 800-90A Assurance Tests

The Apple CoreCrypto 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."

The 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_US_SECPOL_4.0

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 is built into macOS High Sierra. For additional assurance, it is digitally signed.

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 has passed all self-tests and is operating in the Approved mode.

10.4.2 User Guidance

As above, 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 has passed all self-tests and is operating in the Approved mode.

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}