Apple CoreCrypto Kernel Module v9.0 for Intel
FIPS 140-2 Non-Proprietary Security Policy

March, 2019

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

©2019 Apple Inc.
This document may be reproduced and distributed only in its original entirety without revision
# Table of Contents

1  INTRODUCTION ............................................................................................................. 4
   1.1  PURPOSE .................................................................................................................. 4
   1.2  DOCUMENT ORGANIZATION / COPYRIGHT .................................................. 4
   1.3  EXTERNAL RESOURCES / REFERENCES ....................................................... 4
   1.3.1  Additional References .................................................................................... 4
   1.4  ACRONYMS ............................................................................................................ 6

2  CRYPTOGRAPHIC MODULE SPECIFICATION ............................................................... 7
   2.1  MODULE DESCRIPTION ...................................................................................... 7
       2.1.1  Module Validation Level ............................................................................. 7
       2.1.2  Module components ................................................................................... 7
       2.1.3  Tested Platforms ....................................................................................... 8
   2.2  MODES OF OPERATION ...................................................................................... 8
       2.2.1  Approved Security Functions ..................................................................... 8
       2.2.2  Non-Approved Security Functions: ......................................................... 12
   2.3  CRYPTOGRAPHIC MODULE BOUNDARY ......................................................... 14
   2.4  MODULE USAGE CONSIDERATIONS ................................................................. 14

3  CRYPTOGRAPHIC MODULE PORTS AND INTERFACES ............................................... 15

4  ROLES, SERVICES AND AUTHENTICATION ............................................................. 16
   4.1  ROLES .................................................................................................................. 16
   4.2  SERVICES ............................................................................................................ 16
   4.3  OPERATOR AUTHENTICATION ......................................................................... 20

5  PHYSICAL SECURITY ..................................................................................................... 21

6  OPERATIONAL ENVIRONMENT .................................................................................. 22
   6.1  APPLICABILITY .................................................................................................... 22
   6.2  POLICY ................................................................................................................. 22

7  CRYPTOGRAPHIC KEY MANAGEMENT ..................................................................... 23
   7.1  RANDOM NUMBER GENERATION .................................................................... 23
   7.2  KEY / CSP GENERATION .................................................................................. 23
   7.3  KEY / CSP ESTABLISHMENT ........................................................................... 23
   7.4  KEY / CSP ENTRY AND OUTPUT ...................................................................... 23
   7.5  KEY / CSP STORAGE ....................................................................................... 23
   7.6  KEY / CSP ZEROIZATION ................................................................................ 24

8  ELECTROMAGNETIC INTERFERENCE/ELECTROMAGNETIC COMPATIBILITY (EMI/EMC) .............................................................. 25

9  SELF-TESTS .................................................................................................................. 26
   9.1  POWER-UP TESTS ............................................................................................. 26
       9.1.1  Cryptographic Algorithm Tests ................................................................ 26
       9.1.2  Software / Firmware Integrity Tests ......................................................... 26
       9.1.3  Critical Function Tests ............................................................................. 26
   9.2  CONDITIONAL TESTS ...................................................................................... 26
       9.2.1  Continuous Random Number Generator Test ........................................ 26
       9.2.2  Pair-wise Consistency Test ..................................................................... 27
       9.2.3  SP 800-90A Assurance Tests .................................................................. 27
       9.2.4  Critical Function Test ............................................................................. 27

10 DESIGN ASSURANCE ................................................................................................. 28
    10.1 CONFIGURATION MANAGEMENT ................................................................. 28
    10.2 DELIVERY AND OPERATION ......................................................................... 28
    10.3 DEVELOPMENT ................................................................................................. 28
10.4 GUIDANCE ............................................................................................................................. 28
  10.4.1 Cryptographic Officer Guidance .................................................................................. 28
  10.4.2 User Guidance ............................................................................................................. 28

11 MITIGATION OF OTHER ATTACKS..................................................................................29

List of Tables
Table 1: Module Validation Level .......................................................................................... 7
Table 2: Tested Platforms ....................................................................................................... 8
Table 3: Approved, Allowed or Vendor Affirmed Security Functions ......................................... 12
Table 4: Non-Approved or Non-Compliant Security Functions .................................................. 13
Table 5: Roles .......................................................................................................................... 16
Table 6: Approved and Allowed Services in Approved Mode .................................................... 19
Table 7: Non-Approved Services in Non-Approved Mode ......................................................... 20
Table 8: Cryptographic Algorithm Tests .................................................................................. 26

List of Figures
Figure 1: Logical Block Diagram ......................................................................................... 14
1 Introduction

1.1 Purpose

This document is a non-proprietary Security Policy for the Apple CoreCrypto Kernel Module v9.0 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 Kernel Module v9.0 for Intel”, “cryptographic module”, “CoreCrypto KEXT” or “the module” are used interchangeably to refer to the Apple CoreCrypto Kernel Module v9.0 for Intel. macOS 10.14 Mojave is the fifteenth release of macOS (previously OS X). Throughout the document it is generically referred to as macOS Mojave or macOS.

1.2 Document Organization / Copyright

This non-proprietary Security Policy document may be reproduced and distributed only in its original entirety without any revision, ©2019 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 (https://csrc.nist.gov/projects/cryptographic-module-validation-program) contains links to the FIPS 140-2 certificate and Apple Inc. contact information.

1.3.1 Additional References


FIPS 140-2 NIST, “Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program,” November, 2018

Location: https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402IG.pdf


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


SP800-132 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 Standards
- 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 v9.0 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.

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

Table 1: Module Validation Level

2.1.2 Module components

In the following sections the components of the Apple CoreCrypto Kernel Module v9.0 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:

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Inc.</td>
<td>Mac mini with Intel i5 CPU</td>
<td>macOS Mojave 10.14</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>MacBook Pro with Intel i7 CPU</td>
<td>macOS Mojave 10.14</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>MacBook Pro with Intel i9 CPU</td>
<td>macOS Mojave 10.14</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>iMac Pro with Intel Xeon CPU</td>
<td>macOS Mojave 10.14</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>MacBook with Intel Core M CPU</td>
<td>macOS Mojave 10.14</td>
</tr>
</tbody>
</table>

Table 2: Tested Platforms

2.2 Modes of operation

The Apple CoreCrypto Kernel Module v9.0 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 https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program for the current standards, test requirements, and special abbreviations used in the following table.

2.2.1 Approved Security Functions

<table>
<thead>
<tr>
<th>Cryptographic Function</th>
<th>Algorithm</th>
<th>Options</th>
<th>Algorithm Certificate Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Number Generation; Symmetric key generation</td>
<td>SP 800-90] DRBG</td>
<td>Generic Software (C) Implementation using Assembler Implementation of ECB Modes: CTR_DRBG AES-128, AES-256 Derivation Function Enabled</td>
<td>2403, 2404, 2405, 2406, 2428</td>
</tr>
</tbody>
</table>

¹ macOS Mojave 10.14 is the fifteenth release of macOS (previously OS X). Throughout the document it is generically referred to as macOS Mojave or simply macOS.
<table>
<thead>
<tr>
<th>Cryptographic Function</th>
<th>Algorithm</th>
<th>Options</th>
<th>Algorithm Certificate Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNG Implementation using Assembler Implementation of ECB</td>
<td></td>
<td>CTR_DRBG, AES-128, AES-256, Derivation Function Enabled</td>
<td>2418, 2419, 2420, 2421, 2422</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generic Software (C) Implementation with AES-NI</td>
<td>C171, C172, C173, C174, C175</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generic Software (C) Implementation</td>
<td>C176, C177, C236, C237, C238</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generic Software (C) Implementation with SSE3</td>
<td>2413, 2414, 2415, 2416, 2417</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generic Software (C) Implementation with AVX</td>
<td>2423, 2424, 2425, 2426, 2427</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generic Software (C) Implementation with AVX2:</td>
<td>2408, 2409, 2410, 2411, 2412</td>
</tr>
<tr>
<td>Cryptographic Function</td>
<td>Algorithm</td>
<td>Options</td>
<td>Algorithm Certificate Number</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Symmetric Encryption and Decryption</td>
<td>[FIPS 197] AES</td>
<td>Generic Software (C) Implementation using Assembler Implementation of ECB</td>
<td>5824, 5826, 5827, 5828, 5835</td>
</tr>
<tr>
<td></td>
<td>SP 800-38 A</td>
<td>ECB, KW, CBC, CTR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SP 800-38 C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SP 800-38 E</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SP 800-38 F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optimized Assembler Implementation</td>
<td>5820, 5821, 5822, 5823, 5825</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modes: ECB, XTS, CBC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VNG Implementation using Assembler Implementation of ECB</td>
<td>5830, 5831, 5832, 5833, 5834</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modes: ECB, GCM, CCM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optimized Assembler Implementation using AES-NI</td>
<td>5815, 5816, 5817, 5818, 5819</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modes: ECB, XTS, CBC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generic Software (C) Implementation with AES-NI</td>
<td>C171, C172, C173, C174, C175</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modes: ECB, CFB128, OFB, CBC, CTR, XTS, GCM, CFB8, KW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SP 800-67] Triple-DES</td>
<td>Triple-DES (K1, K2, K3 independent)</td>
<td>C176, C177, C236, C237, C238</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modes: ECB, CBC</td>
<td></td>
</tr>
<tr>
<td>Digital Signature and Asymmetric Key Generation</td>
<td>[FIPS186-4] RSA</td>
<td>Signature Verification</td>
<td>C176, C177, C236, C237, C238</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Key Sizes: 1024, 2048, 3072</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[FIPS 186-4] ECDSA</td>
<td>Key Pair Generation (PKG): P-224, P-256, P-384, P-521</td>
<td>C176, C177, C236, C237, C238</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Public Key Validation (PKV): P-224, P-256, P-384, P-521</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signature Generation: P-224, P-256, P-384, P-521</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signature Verification: P-224, P-256, P-384, P-521</td>
<td></td>
</tr>
<tr>
<td>Cryptographic Function</td>
<td>Algorithm Options</td>
<td>Algorithm Certificate Number</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Message Digest</strong></td>
<td>[FIPS 180-4] SHS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation</td>
<td>C176, C177, C236, C237, C238</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-384</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-512</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-1</td>
<td>SHA-384</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-224</td>
<td>SHA-512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation with SSE3</td>
<td>4621, 4622, 4623, 4624, 4625</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-1</td>
<td>SHA-384</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-224</td>
<td>SHA-512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation with AVX</td>
<td>4626, 4627, 4628, 4629, 4630</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-1</td>
<td>SHA-384</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-224</td>
<td>SHA-512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation with AVX2</td>
<td>4616, 4617, 4618, 4619, 4620</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-1</td>
<td>SHA-384</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-224</td>
<td>SHA-512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHA-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Keyed Hash</strong></td>
<td>[FIPS 198] HMAC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation</td>
<td>C176, C177, C236, C237, C238</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-384</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-512</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation with SSE3</td>
<td>3846, 3847, 3848, 3849, 3850</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-1</td>
<td>HMAC-SHA-384</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-224</td>
<td>HMAC-SHA-512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation with AVX</td>
<td>3851, 3852, 3853, 3854, 3855</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-1</td>
<td>HMAC-SHA-384</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-224</td>
<td>HMAC-SHA-512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic Software (C) Implementation with AVX2</td>
<td>3841, 3842, 3843, 3844, 3845</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-1</td>
<td>HMAC-SHA-384</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-224</td>
<td>HMAC-SHA-512</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC-SHA-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Key Derivation</strong></td>
<td>[SP 800-132] PBKDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Password based key derivation using HMAC with SHA-1 or SHA-2</td>
<td>Vendor Affirmed</td>
<td></td>
</tr>
<tr>
<td><strong>MD5</strong></td>
<td>Message Digest</td>
<td>Digest Size: 128-bit</td>
<td>Non-Approved, but Allowed¹</td>
</tr>
<tr>
<td><strong>NDRNG</strong></td>
<td>Random Number Generation</td>
<td>N/A</td>
<td>Non-Approved, but Allowed²</td>
</tr>
</tbody>
</table>

¹ MD5 is used as part of the TLS key establishment scheme only.
² NDRNG is provided by the underlying operational environment.
<table>
<thead>
<tr>
<th>Cryptographic Function</th>
<th>Algorithm</th>
<th>Options</th>
<th>Algorithm Certificate Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA Key Wrapping</td>
<td>[FIPS 186-4]</td>
<td>PKCS#1 v1.5 Modulus size: 2048-bits or 3072-bits</td>
<td>Non-Approved, but allowed³</td>
</tr>
<tr>
<td></td>
<td>[SP800-56B]</td>
<td>KTS-OAEP</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Approved, Allowed or Vendor Affirmed 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:

<table>
<thead>
<tr>
<th>Cryptographic Function</th>
<th>Usage / Description</th>
<th>Caveat</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI X9.63</td>
<td>Hash Based KDF</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>Blowfish</td>
<td>Encryption / Decryption</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>CAST5</td>
<td>Encryption / Decryption Key Sizes: 40 to 128 bits in 8-bit increments</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>DES</td>
<td>Encryption / Decryption Key Size 56 bits</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>ECDSA</td>
<td>Key Pair Generation (PKG): curves P-192 Public Key Validation (PKV): curves P-192 Signature Generation: curves P-192 Key generation for compact point representation of points</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>Ed25519</td>
<td>Key Agreement Signature Generation Signature Verification</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>Integrated Encryption Scheme on elliptic curves</td>
<td>Encryption / Decryption</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>MD2</td>
<td>Message Digest Digest size 128 bit</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>MD4</td>
<td>Message Digest Digest size 128 bit</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>OMAC (One-Key CBC MAC)</td>
<td>MAC generation</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>RSA Key Wrapping</td>
<td>RSA Key Wrapping using key sizes &lt; 2048-bits</td>
<td>Non-Approved</td>
</tr>
</tbody>
</table>

³ RSA key wrapping is used for key establishment. Methodology provides 112 or 128 bits of encryption strength
<table>
<thead>
<tr>
<th>Cryptographic Function</th>
<th>Usage / Description</th>
<th>Caveat</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC6637 KDF</td>
<td>KDF based on RFC 6637</td>
<td>Non-Approved</td>
</tr>
<tr>
<td>RIPEMD</td>
<td>Message Digest</td>
<td>Non-Approved</td>
</tr>
<tr>
<td></td>
<td>Digest size 128, 160, 256, 320 bits</td>
<td></td>
</tr>
<tr>
<td>RC2</td>
<td>Encryption / Decryption</td>
<td>Non-Approved</td>
</tr>
<tr>
<td></td>
<td>Key sizes: 8 to 1024 bits</td>
<td></td>
</tr>
<tr>
<td>RC4</td>
<td>Encryption / Decryption</td>
<td>Non-Approved</td>
</tr>
<tr>
<td></td>
<td>Key sizes: 8 to 1024 bits</td>
<td></td>
</tr>
<tr>
<td>Triple-DES</td>
<td>Encryption / Decryption: Two Key Implementation</td>
<td>Non-Approved</td>
</tr>
<tr>
<td></td>
<td>Optimized Assembler Implementation:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Encryption / Decryption Mode: CTR</td>
<td></td>
</tr>
<tr>
<td>AES-CMAC</td>
<td>AES-128 MAC generation</td>
<td>Non-Compliant</td>
</tr>
<tr>
<td>SP800-108</td>
<td>KBKDF</td>
<td>Non-Compliant</td>
</tr>
<tr>
<td></td>
<td>Modes: CTR, Feedback</td>
<td></td>
</tr>
<tr>
<td>SP800-56C</td>
<td>Key Derivation Function</td>
<td>Non-Compliant</td>
</tr>
<tr>
<td>RSA</td>
<td>PKCS#1 v1.5</td>
<td>Non-Compliant</td>
</tr>
<tr>
<td>FIPS186-4</td>
<td>Signature Verification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Key sizes (modulus): 1536 bits, 4096</td>
<td></td>
</tr>
</tbody>
</table>

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.
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](image)

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 redistributed.

- AES-XTS mode is only approved for hardware storage applications. The length of the XTS-AES data unit does not exceed $2^{20}$ blocks.

- When using AES, the caller must obtain a reference to the cipher implementation via the functions `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 $2^{16}$ 64-bit blocks of data.
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: ECDSA does 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.

<table>
<thead>
<tr>
<th>Role</th>
<th>General Responsibilities and Services (details see below)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Utilization of services of the module listed in sections 2.1 and 4.2</td>
</tr>
<tr>
<td>Crypto Officer (CO)</td>
<td>Utilization of services of the module listed in sections 2.1 and 4.2.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Service</th>
<th>Roles</th>
<th>CSPs &amp; crypto keys</th>
<th>Access Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple-DES Encryption / Decryption</td>
<td>USER</td>
<td>Secret key</td>
<td>R</td>
</tr>
<tr>
<td>Encryption</td>
<td>CO</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Input</em>: plaintext, IV, key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Output</em>: ciphertext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decryption</td>
<td>USER</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Input</em>: ciphertext, IV, key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Output</em>: plaintext</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 The access type of R refers to reading of the CSP. This can be thought as synonymous with execute CSP/key.
<table>
<thead>
<tr>
<th>Service</th>
<th>Roles</th>
<th>CSPs &amp; crypto keys</th>
<th>Access Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>USER</td>
<td>CO</td>
<td></td>
</tr>
<tr>
<td>AES Encryption / Decryption</td>
<td>X</td>
<td>X</td>
<td>Secret key</td>
</tr>
<tr>
<td>Encryption</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>Input: plaintext, IV, key</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>Output: ciphertext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decryption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input: ciphertext, IV, key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: plaintext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AES Key Wrapping</td>
<td>X</td>
<td>X</td>
<td>secret key</td>
</tr>
<tr>
<td>Encryption</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>Input: plaintext, key</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>Output: ciphertext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decryption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input: ciphertext, key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: plaintext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSA Key Wrapping</td>
<td>X</td>
<td>X</td>
<td>RSA key pair</td>
</tr>
<tr>
<td>Encryption</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>Input: plaintext, RSA public key, the SHA algorithm</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>Output: ciphertext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decryption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input: ciphertext, RSA private key, the SHA algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: plaintext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Hash Generation using SHA1, SHA-256, SHA-384, or SHA-512</td>
<td>X</td>
<td>X</td>
<td>None</td>
</tr>
<tr>
<td>Input: message</td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Output: message digest</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMAC generation using HMAC-SHA1, HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-512</td>
<td>X</td>
<td>X</td>
<td>Secret HMAC key</td>
</tr>
<tr>
<td>Input: HMAC key, message</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>Output: HMAC value of message</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>RSA signature verification</td>
<td>X</td>
<td>X</td>
<td>RSA key pair</td>
</tr>
<tr>
<td>Input: the module n, the public key e,</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>the SHA algorithm (SHA-1/SHA -224/SHA-256/SHA-384/SHA-512), a message m,</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>a signature for the message</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: pass if the signature is valid,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fail if the signature is invalid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>Roles</td>
<td>CSPs &amp; crypto keys</td>
<td>Access Type</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-------</td>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ECDSA Signature generation / Signature verification</td>
<td>USER</td>
<td>CO</td>
<td></td>
</tr>
<tr>
<td>Signature generation</td>
<td>X</td>
<td>X</td>
<td>ECDSA key pair</td>
</tr>
</tbody>
</table>
| **Input:** message m, q, a, b, X<sub>G</sub>, Y<sub>G</sub>, 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<sub>G</sub>, Y<sub>G</sub>, 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 |       |                    |             |
| ECDSA Key Pair Generation                         | USER  | X                  | ECDSA key pair | R, W |
| **Input:** q, FR, a, b, domain_parameter_seed, G, n, h.  
**Output:** private key d, public key Q | X     | X                  |             |
| Random number generation                          | USER  | X                  | Entropy input string, Nonce, V and Key | R, W |
| **Input:** Entropy Input, Nonce, Personalization String  
**Output:** Returned Bits | X     | X                  |             |
| PBKDF Password-based key derivation               | USER  | X                  | Secret key, password | R, W |
| **Input:** encrypted key and password  
**Output:** plaintext key or  
**Input:** plaintext key and password  
**Output:** encrypted data | X     | X                  |             |
| Release all resources of symmetric crypto function context (Symmetric Key Zeroization) | USER  | X                  | AES/Triple-DES key | Z |
| **Input:** context  
**Output:** N/A | X     | X                  |             |
| Release all resources of hash context (MAC Key Zeroization) | USER  | X                  | HMAC key | Z |
| **Input:** context  
**Output:** N/A | X     | X                  |             |
<table>
<thead>
<tr>
<th>Service</th>
<th>Roles</th>
<th>CSPs &amp; crypto keys</th>
<th>Access Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release all resources of asymmetric crypto function context (Asymmetric Key Zeroization)</td>
<td>X</td>
<td>Asymmetric keys (ECDSA)</td>
<td>Z</td>
</tr>
<tr>
<td>Input: context</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reboot</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Input: N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self-test</td>
<td>X</td>
<td>Software integrity key</td>
<td>R</td>
</tr>
<tr>
<td>Input: N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: pass if the Self-test is successful, fail if the Self-test is unsuccessful</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show Status</td>
<td>X</td>
<td>None</td>
<td>N/A</td>
</tr>
<tr>
<td>Input: N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output: Status of module</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Approved and Allowed Services in Approved Mode

<table>
<thead>
<tr>
<th>Service</th>
<th>Roles</th>
<th>Access Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Encryption Scheme on elliptic curves encryption and decryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>DES Encryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>DES Decryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>Triple-DES Encryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>CTR mode (non-compliant)</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>Two-Key Triple-DES (non-approved)</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>Triple-DES Decryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>CTR mode (non-compliant)</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>Two-Key Triple-DES (non-approved)</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>CAST5 Encryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>CAST5 Decryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>Blowfish Encryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>Blowfish Decryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>RC2 Encryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>RC2 Decryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>RC4 Encryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>RC4 Decryption</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>MD2 Hash Generation</td>
<td>X</td>
<td>R</td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>Roles</td>
<td>Access Type</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>USER</td>
<td>CO</td>
</tr>
<tr>
<td>MD4 Hash Generation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RIPEMD Hash Generation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RSA Key Wrapping</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RSA Signature Verification with PKCS#1 v1.5</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Key sizes: 1536 bits, 4096 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDSA Key Pair Generation for compact point</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>representation of points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDSA Key Generation: curve P-192</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ECDSA Public Key Verification: curve P-192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDSA Signature Generation: curve P-192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDSA Signature Verification: curve P-192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ed25519 Key agreement</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Ed25519 Signature Generation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ed25519 Signature Verification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP800-56C Key Derivation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ANSI X9.63 Hash based Key Derivation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SP800-108 Key Derivation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Modes: Feedback, CTR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFC6637 Key Derivation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AES-CMAC MAC Generation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OMAC MAC Generation</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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 v9.0 for Intel is a software cryptographic module running on a multi-chip standalone general-purpose computer. The FIPS 140-2 physical security requirements do not apply to this module since it is a software module.
6 Operational Environment

The following sections describe the macOS Mojave 10.14, operational environment of the Apple CoreCrypto Kernel Module v9.0 for Intel.

6.1 Applicability

The Apple CoreCrypto Kernel Module v9.0 for Intel operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module is included in macOS Mojave, 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 v9.0 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 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 ECDSA as well as the SP 800-90A DRBG have been CAVS tested with algorithm certificates found in Table 3.

It is not possible for the module to output information during the key generating process.

7.3 Key / CSP Establishment
The module provides AES key wrapping, and RSA key wrapping -based key establishment services. The RSA key wrapping service is non-approved but allowed.

The module also provides key establishment services in the Approved mode through the SP 800-132 PBKDFv2 algorithm. The PBKDFv2 function returns the key derived from the provided password to the caller. The keys derived from SP 800-132 map to section 4.1 of SP 800-133 as indirect generation from DRBG. 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 v9.0 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 applicable 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 v9.0 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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Modes</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple-DES</td>
<td>CBC</td>
<td>KAT (Known Answer Test)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separate encryption / decryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operations are performed</td>
</tr>
<tr>
<td>AES</td>
<td>ECB, CBC, XTS</td>
<td>KAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separate encryption / decryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operations are performed</td>
</tr>
<tr>
<td>DRBG</td>
<td>N/A</td>
<td>KAT</td>
</tr>
<tr>
<td>HMAC-SHA-1</td>
<td>N/A</td>
<td>KAT</td>
</tr>
<tr>
<td>HMAC-SHA-256, HMAC-SHA-512</td>
<td>N/A</td>
<td>KAT</td>
</tr>
<tr>
<td>ECDSA</td>
<td>Signature Generation,</td>
<td>pair-wise consistency test</td>
</tr>
<tr>
<td></td>
<td>Signature Verification</td>
<td></td>
</tr>
<tr>
<td>RSA</td>
<td>Signature Verification</td>
<td>KAT</td>
</tr>
</tbody>
</table>

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 v9.0 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 v9.0 for Intel.

9.2.1 Continuous Random Number Generator Test

The Apple CoreCrypto Kernel Module v9.0 for Intel performs a continuous random number generator test on the noise source (i.e. NDRNG), whenever it is invoked to seed the SP800-90A DRBG.
9.2.2 Pair-wise Consistency Test
The Apple CoreCrypto Kernel Module v9.0 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 Health Tests
The Apple CoreCrypto Kernel Module v9.0 for Intel performs the health tests as specified in section 11.3 of SP 800-90A.

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_SECPO_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 Mojave. 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 Mojave 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 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,0xFE,0xFE,0xFE,0xF1,0xFE,0xF1,0xFE}. 
