

Apple Inc.



Apple OS X CoreCrypto Kernel Module, v5.0
FIPS 140-2 Non-Proprietary Security Policy

Document Control Number
FIPS_CORECRYPTO_OSX_KS_SECPOL_01.02
Version 01.02
June, 2015

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

1	INTRODUCTION	4
1.1	PURPOSE	4
1.2	DOCUMENT ORGANIZATION / COPYRIGHT	4
1.3	EXTERNAL RESOURCES / REFERENCES	4
1.3.1	<i>Additional References</i>	4
1.4	ACRONYMS	5
2	CRYPTOGRAPHIC MODULE SPECIFICATION	7
2.1	MODULE DESCRIPTION	7
2.1.1	<i>Module Validation Level</i>	7
2.1.2	<i>Module components</i>	7
2.1.3	<i>Tested Platforms</i>	8
2.2	MODES OF OPERATION	8
2.3	CRYPTOGRAPHIC MODULE BOUNDARY	15
2.4	MODULE USAGE CONSIDERATIONS	16
3	CRYPTOGRAPHIC MODULE PORTS AND INTERFACES	17
4	ROLES, SERVICES AND AUTHENTICATION	18
4.1	ROLES	18
4.2	SERVICES	18
4.3	OPERATOR AUTHENTICATION	23
5	PHYSICAL SECURITY	24
6	OPERATIONAL ENVIRONMENT	25
6.1	APPLICABILITY	25
6.2	POLICY	25
7	CRYPTOGRAPHIC KEY MANAGEMENT	26
7.1	RANDOM NUMBER GENERATION	26
7.2	KEY / CSP GENERATION	26
7.3	KEY / CSP ESTABLISHMENT	26
7.4	KEY / CSP ENTRY AND OUTPUT	26
7.5	KEY / CSP STORAGE	26
7.6	KEY / CSP ZEROIZATION	27
8	ELECTROMAGNETIC INTERFERENCE/ELECTROMAGNETIC COMPATIBILITY (EMI/EMC)	28
9	SELF-TESTS	29
9.1	POWER-UP TESTS	29
9.1.1	<i>Cryptographic Algorithm Tests</i>	29
9.1.2	<i>Software / Firmware Integrity Tests</i>	29
9.1.3	<i>Critical Function Tests</i>	29
9.2	CONDITIONAL TESTS	29
9.2.1	<i>Continuous Random Number Generator Test</i>	30
9.2.2	<i>Pair-wise Consistency Test</i>	30
9.2.3	<i>SP 800-90A Assurance Tests</i>	30
9.2.4	<i>Critical Function Test</i>	30
10	DESIGN ASSURANCE	31
10.1	CONFIGURATION MANAGEMENT	31
10.2	DELIVERY AND OPERATION	31
10.3	DEVELOPMENT	31
10.4	GUIDANCE	31
10.4.1	<i>Cryptographic Officer Guidance</i>	31

10.4.2	User Guidance	31
11	MITIGATION OF OTHER ATTACKS.....	32

List of Tables

TABLE 1:	MODULE VALIDATION LEVEL	7
TABLE 2:	TESTED PLATFORMS	8
TABLE 3:	APPROVED SECURITY FUNCTIONS.....	9
TABLE 4:	NON-APPROVED FUNCTIONS	12
TABLE 5:	ROLES	18
TABLE 6:	APPROVED AND ALLOWED SERVICES IN APPROVED MODE.....	18
TABLE 6B:	NON-APPROVED SERVICES IN NON-APPROVED MODE	21
TABLE 7:	CRYPTOGRAPHIC ALGORITHM TESTS.....	29

List of Figures

FIGURE 1:	LOGICAL BLOCK DIAGRAM	15
-----------	-----------------------------	----

1 Introduction

1.1 Purpose

This document is a non-proprietary Security Policy for the Apple OS X CoreCrypto Kernel Module, v5.0. It describes the module and the FIPS 140-2 cryptographic services it provides. This document also defines the FIPS 140-2 security rules for operating the module.

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

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

For more information on the FIPS 140-2 standard and validation program please refer to the NIST website at <http://csrc.nist.gov/cryptval>.

Throughout the document “Apple OS X CoreCrypto Kernel Module, v5.0.”, “cryptographic module”, “CoreCrypto KEXT” or “the module” are used interchangeably to refer to the Apple OS X CoreCrypto Kernel Module, v5.0.

1.2 Document Organization / Copyright

This non-proprietary Security Policy document may be reproduced and distributed only in its original entirety without any revision, ©2015 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 OS X and its security properties refer to [OS X] and [SEC]. For details on OS X releases with their corresponding validated modules and Crypto Officer Role Guides refer to the Apple Knowledge Base Article HT201159 - “OS X: Security certifications and validations” (<https://support.apple.com/en-us/HT201159>)

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

1.3.1 Additional References

- FIPS 140-2 Federal Information Processing Standards Publication, “FIPS PUB 140-2 Security Requirements for Cryptographic Modules,” Issued May-25-2001, Effective 15-Nov-2001, Location: <http://csrc.nist.gov/groups/STM/cmvp/standards.html>
- FIPS 180-4 Federal Information Processing Standards Publication 180-4, March 2012, Secure Hash Standard (SHS)
- FIPS 197 Federal Information Processing Standards Publication 197, November 26, 2001 Announcing the ADVANCED ENCRYPTION STANDARD (AES)
- PKCS7 RSA Laboratories, “PKCS#7 v1.5: Cryptographic Message Syntax Standard,” 1993. Location: <http://www.rsa.com/rsalabs/node.asp?id=2129>
- PKCS3 RSA Laboratories, “PKCS#3 v1.4: Diffie-Hellman Key Agreement Standard,” 1993. Location: <http://www.rsa.com/rsalabs/node.asp?id=2126>
- IG NIST, “Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program,” March 2, 2015
Location: <http://csrc.nist.gov/groups/STM/cmvp/standards.html>

OS X	OS X Technical Overview Location: https://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/About.html
SEC	Security Overview Location: https://developer.apple.com/library/mac/navigation/#section=Topics&topic=Security
SP800-57P1	NIST Special Publication 800-57, "Recommendation for Key Management – Part 1: General (Revised)," March 2007
SP 800-90A	NIST Special Publication 800-90A, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)," January 2012
UG	User Guide Location: https://developer.apple.com/library/mac/navigation/

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
DMA	Direct Memory Access
DRBG	Deterministic Random Bit Generator
DS	Digest Size
ECB	Electronic Codebook mode of operation
ECC	Elliptic Curve Cryptography
EC Diffie-Hellman	Diffie-Hellman based on ECC
ECDSA	DSA based on ECC
E/D	Encrypt/Decrypt
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard

FIPS PUB	FIPS Publication
GCM	Galois/Counter Mode
HMAC	Hash-Based Message Authentication Code
HW	Hardware
IPCU	iPhone Configuration Utility
KAT	Known Answer Test
KEK	Key Encryption Key
KEXT	Kernel extension
KDF	Key Derivation Function
KO 1	Triple-DES Keying Option 1: All three keys are independent
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
PWCT	Pair Wise Consistency Test
RNG	Random Number Generator
SHS	Secure Hash Standard
SW	Software
Triple-DES	Triple Data Encryption Standard
TLS	Transport Layer Security

2 Cryptographic Module Specification

2.1 Module Description

The Apple OS X CoreCrypto Kernel Module, v5.0 is a software cryptographic module running on a multi-chip standalone general-purpose computer.

The cryptographic services provided by the module are:

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

2.1.1 Module Validation Level

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

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

Table 1: Module Validation Level

2.1.2 Module components

In the following sections the components of the Apple OS X CoreCrypto Kernel Module, v5.0 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 OS X kernel.

2.1.2.2 Hardware components

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

2.1.3 Tested Platforms

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

Manufacturer	Model	Operating System
Apple Inc.	Mac mini with i5 CPU	OS X 10.10
Apple Inc.	iMac with i7 CPU	OS X 10.10
Apple Inc.	MacPro with Xeon CPU	OS X 10.10
Apple Inc.	MacBook with Core M	OS X 10.10

OS X 10.10 refers to OS X Yosemite v10.10

Table 2: Tested Platforms

2.2 Modes of operation

The Apple OS X CoreCrypto Kernel Module, v5.0 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 (“Val. No.”) of Table 3 lists the validation numbers obtained from NIST based on the successful CAVP testing of the cryptographic algorithm implementations on the platforms referenced in Table 2

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

Approved Security Functions

Cryptographic Function	Standards	Usage / Description	Val. No.			
			i5 CPU	i7 CPU	Xeon CPU	Core M CPU
Triple-DES	ANSIX9.52-1998 FIPS 46-3 SP 800-67 SP 800-38A Appendix E	Encryption / decryption with all keys independent Block chaining modes: ECB, CBC	1895	1896	1897	1921
AES	FIPS 197 SP 800-38A SP 800-38 E	Generic-software implementation (non-optimized using LibTomCrypt): Encryption / decryption Key sizes: 128 bits, 192 bits, 256 bits for block chaining modes ECB, CBC	3323	3324	3325	3385
		Optimized-software implementation: Encryption / decryption Key sizes: 128 bits, 192 bits, 256 bits for block chaining modes ECB, CBC	3066	3067	3102	3382
		AES-NI hardware implementation with optimized software implementation of block chaining modes: Key sizes: 128 bits, 192 bits, 256 bits for block chaining modes ECB, CBC	3068	3069	3070	3384

Cryptographic Function	Standards	Usage / Description	Val. No.			
			i5 CPU	i7 CPU	Xeon CPU	Core M CPU
		AES-NI hardware implementation with generic software implementation (non-optimized) of block chaining modes: Key sizes: 128 bits, 192 bits, 256 bits for block chaining mode CBC	3071	3072	3073	3383
SHS	FIPS 180-4	Generic-software implementation (non-optimized): SHA-1 (BYTE-only) SHA-224 (BYTE-only) SHA-256 (BYTE-only) SHA-384 (BYTE-only) SHA-512 (BYTE-only)	2755	2756	2757	2804
		Optimized-software implementation using SSE: SHA-1 (BYTE-only) SHA-224 (BYTE-only) SHA-256 (BYTE-only)	2586	2545	2546	2803
		Optimized-software implementation not using SSE: SHA-1 (BYTE-only) SHA-224 (BYTE-only) SHA-256 (BYTE-only)	2585	2543	2544	2802
		Optimized-software implementation using AVX1: SHA-256(BYTE-only) 64 bit word size	2579	2580	2581	2800

Cryptographic Function	Standards	Usage / Description	Val. No.			
			i5 CPU	i7 CPU	Xeon CPU	Core M CPU
		Optimized-software implementation using AVX2: SHA-256(BYTE-only) 64 bit word size	2582	2583	2584	2801
RSA	PKCS1.5	SIG(ver) Key sizes (modulus): 1024 bits, 2048 bits, 3072 bits Hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	1704	1705	1706	1737
ECDSA	FIPS 186-2 ANSI X9.62	PKG: curves P-256, P-384 PKV: curves P-256, P-384 SIG(gen): curves P-256, P-384 SIG(ver): curves P-256, P-384	652	653	654	673
HMAC	FIPS 198	Generic-software implementation (non-optimized): KS<BS, KS=BS, KS>BS HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512 Key Size: at least 112 bits	2114	2115	2116	2159
		Optimized-software implementation using SSE: KS<BS, KS=BS, KS>BS HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 Key Size: at least 112 bits	1967	1929	1930	2158

Cryptographic Function	Standards	Usage / Description	Val. No.			
			i5 CPU	i7 CPU	Xeon CPU	Core M CPU
		Optimized-software implementation not using SSE: KS<BS, KS=BS, KS>BS HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 Key Size: 112 bits	1966	1927	1928	2157
		Optimized-software implementation using AVX1: KS<BS, KS=BS, KS>BS HMAC-SHA-256 64 bit word size	1960	1961	1962	2155
		Optimized-software implementation using AVX2: KS<BS, KS=BS, KS>BS HMAC-SHA-256 64 bit word size	1963	1964	1965	2156
CTR DRBG	SP 800-90A	Generic-software implementation of AES (non-optimized) AES with 128 bit key size	769	770	771	806
		Optimized-software implementation of AES: AES with 128 bit key size	598	599	609	816
		AES-NI hardware implementation AES with 128 bit key size	600	601	602	805
PBKDF	SP 800-132	Password based key derivation according using HMAC with SHA-1 or SHA-2 as pseudorandom function	Vendor Affirmed			

Table 3: Approved Security Functions

CAVEAT: The module generates cryptographic keys whose strengths are modified by available entropy – 160-bits.

Non-Approved Security Functions:

Cryptographic Function	Usage / Description	Caveat
AES	Optimized-assembler implementation using AVX: Encryption / Decryption Block Chaining Mode: GCM 64 bit word	Non-compliant
AES	Optimized-assembler implementation using SSE3: Encryption / Decryption Block Chaining Mode: GCM, CTR 64 bit word	Non-compliant
DES	Encryption and decryption: key size 56 bit; Used for NFS support in the raccoon IPSec cipher suite as a last resort when AES and Triple-DES ciphers are not supported by the remote end.	Non-Approved
Triple-DES	Optimized-assembler implementation: Encryption / Decryption Block Chaining Mode: CTR	Non-compliant
Triple-DES	Encryption and Decryption: One key and two key implementation	Non-compliant
ANSI X9.63	Hash Based KDF	Non-Approved
RFC6637	KDF	Non-compliant
SP800-108	KDF Modes: Counter, Feedback	Non-Approved
SP800-56C	KDF	Non-Approved
MD2	Hashing Digest size 128 bit	Non-Approved

Cryptographic Function	Usage / Description	Caveat
MD4	Hashing Digest size 128 bit	Non-Approved
MD5	Hashing Digest size 128 bit	Non-Approved
RIPEMD	Hashing Digest size 128, 160, 256, 320 bits	Non-Approved
Ed25519	Key Agreement Sig(gen) Sig(ver)	Non-Approved
ECDSA	PKG: curves P-192, P-224, P-521 PKV: curves P-192, P-224, P-521 SIG(gen): curves P-192, P-224, P-521 SIG(ver): curves P-192, P-224, P-521	Non-Approved
ECDSA	key generation for compact point representation of points	Non-Approved
Integrated Encryption Scheme on elliptic curves	Encryption/Decryption	Non-Approved
RSA FIPS186-2	PKCS#1 v1.5 SIG(ver) Key sizes (modulus): 1536 bits, 4092 bits Hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Non-compliant
RSA OAEP	Encryption and Decryption	Non-Approved, but allowed: RSA (key wrapping; key establishment methodology provides between 112 and 150 bits of encryption strength; non-compliant less than 112 bits of encryption strength)
AES KWP SP800-38F	Key Wrapping with 80 to 150 bits of security strength	Non-Approved AES (key wrapping; key establishment methodology provides between 128 and 160 bits of encryption strength).
CAST5	Encryption and decryption: key sizes 40 to 128 bits in 8-bit increments	Non-Approved

Cryptographic Function	Usage / Description	Caveat
Blowfish	Encryption and decryption	Non-Approved
RC2	Encryption and decryption: key size 8 to 1024 bits	Non-Approved
RC4	Encryption and decryption: key size 8 to 1024 bits	Non-Approved
Hash-DRBG	Hash-DRBG using SHA1 derived from the SP800-90A without derivation function, without prediction resistance	Non-compliant
HMAC-DRBG	HMAC based RNG derived from SP800-90A without prediction resistance	Non-compliant
CMAC AES128	MAC generation	Non-Approved
OMAC (One-Key CBC MAC)	MAC generation	Non-Approved

Table 4: Non-Approved Function

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].

2.3 Cryptographic Module Boundary

The physical boundary of the module is the physical boundary of the OS X 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.

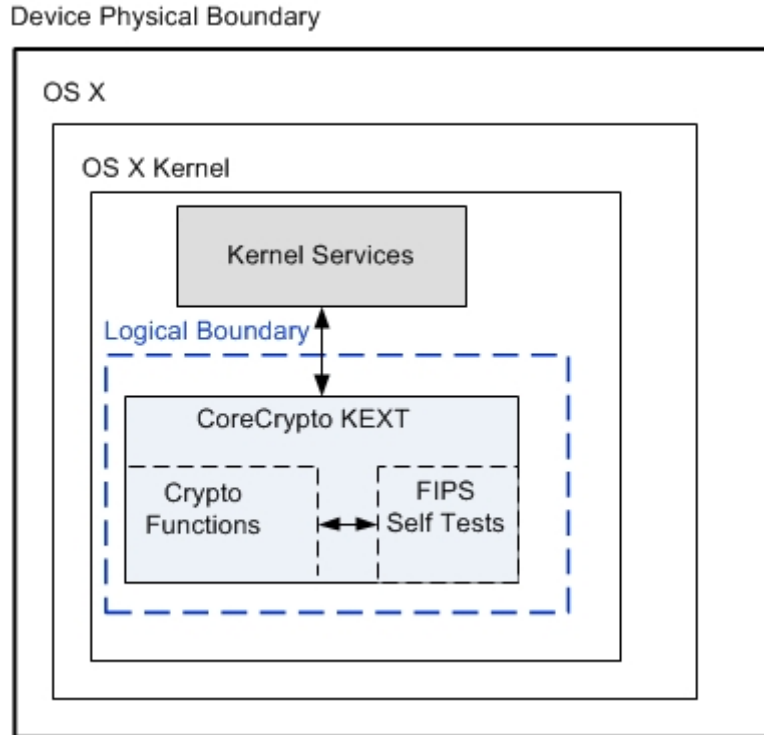


Figure 1: Logical Block Diagram

2.4 Module Usage Considerations

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

- When using AES-GCM, the caller must use the module's DRBG to generate at least 96 bits of random data that is used for the IV of AES-GCM. The caller is permitted to add additional deterministic data to that IV value in accordance with SP800-38D section 8.2.2. Users should consult SP 800-38D, especially section 8, for all of the details and requirements of using AES-GCM mode.
- 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 case the module's power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed

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 OS X kernel and does not contain any terminating assertions or exceptions. It is implemented as an OS X kernel extension. The dynamically loadable library is loaded into the OS X kernel and its cryptographic functions are made available to OS X Kernel services only. Any internal error detected by the module is reflected back to the caller with an appropriate return code. The calling OS X Kernel service must examine the return code and act accordingly. There are two notable exceptions: (i) ECDSA does not return a key if the pair-wise consistency test fails; (ii) the DRBG algorithm loops a few iterations internally if the continuous test fails, eventually recovering from the error or causing a shutdown if the problem persists.

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

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

Caller-induced or internal errors do not reveal any sensitive material to callers.

Cryptographic bypass capability is not supported by the module.

4 Roles, Services and Authentication

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

4.1 Roles

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

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

Table 5: Roles

4.2 Services

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

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

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

The access types are denoted as follows:

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

Service	Roles		CSPs & crypto keys	Access Type
	U S E R	C O		
Triple-DES Encryption <i>Input:</i> plaintext, IV, key <i>Output:</i> ciphertext Decryption <i>Input:</i> ciphertext, IV, key <i>Output:</i> plaintext	X	X	Secret key	R

Service	Roles		CSPs & crypto keys	Access Type
	U S E R	C O		
AES Encryption <i>Input:</i> plaintext, IV, key <i>Output:</i> ciphertext Decryption <i>Input:</i> ciphertext, IV, key <i>Output:</i> plaintext	X	X	Secret key	R
Secure Hash Generation <i>Input:</i> message <i>Output:</i> message digest	X	X	None	N/A
HMAC generation <i>Input:</i> HMAC key, message <i>Output:</i> HMAC value of message	X	X	Secret HMAC key	R
RSA signature verification <i>Input:</i> the module n, the public key e, the SHA algorithm (SHA-1/SHA-224/SHA-256/SHA-384/SHA-512), a message m, a signature for the message <i>Output:</i> pass if the signature is valid, fail if the signature is invalid	X	X	RSA key pair	R W

Service	Roles		CSPs & crypto keys	Access Type
	U S E R	C O		
<p>ECDSA</p> <p>Signature generation <i>Input:</i> message m, q, a, b, X_G, Y_G, n, the SHA algorithm (SHA-224/SHA-256/SHA-384/SHA-512) sender's private key d <i>Output:</i> signature of m as a pair of r and s</p> <p>Signature verification <i>Input:</i> received message m', signature in form on r' and s' pair, q, a, b, X_G, Y_G, n, sender's public key Q, the SHA algorithm (SHA-1/SHA-224/SHA-256/SHA-384/SHA-512) <i>Output:</i> pass if the signature is valid, fail if the signature is invalid</p>	X	X	ECDSA key pair	R W
<p>Random number generation <i>Input:</i> Entropy Input, Nonce, Personalization String <i>Output:</i> Returned Bits</p>	X	X	Entropy input string, Nonce, V and K	R W Z
<p>PBKDF Password-based key derivation <i>Input:</i> encrypted key and password <i>Output:</i> plaintext key or <i>Input:</i> plaintext key and password <i>Output:</i> encrypted data</p>	X	X	Secret key, password	R W Z
<p>AES key import <i>Input:</i> key <i>Output:</i> N/A</p>	X	X	Secret key	R

Service	Roles		CSPs & crypto keys	Access Type
	USER	CO		
Triple-DES key import <i>Input:</i> key <i>Output:</i> N/A	X	X	Secret key	R
HMAC key import <i>Input:</i> key <i>Output:</i> N/A	X	X	HMAC key	R
Release all resources of symmetric crypto function context <i>Input:</i> context <i>Output:</i> N/A	X	X	AES/Triple-DES key	Z
Release all resources of hash context <i>Input:</i> context <i>Output:</i> N/A	X	X	HMAC key	Z
Release all resources of asymmetric crypto function context <i>Input:</i> context <i>Output:</i> N/A	X	X	Asymmetric keys (ECDSA)	Z
Reboot	X	X	N/A	N/A
Self-test	X	X	Software integrity key	R
Show Status	X	X	None	N/A

Table 6: Approved and Allowed Services in Approved Mode

Service	Roles		Access Type
	USER	CO	
AES encryption and decryption Modes: CCM, GCM, CTR, CBC	X	X	R
Integrated Encryption Scheme on elliptic curves encryption and decryption	X	X	R
DES encryption and decryption	X	X	R

Service	Roles		Access Type
	USER	CO	
TDES encryption and decryption Mode: CTR	X	X	R
TDES encryption and decryption with One-Key and Two-Key implementations	X	X	R
CAST5 encryption and decryption	X	X	R
Blowfish encryption and decryption	X	X	R
RC4 encryption and decryption	X	X	R
RC2 encryption and decryption	X	X	R
MD2 Hash	X	X	R W
MD4 Hash	X	X	R W
MD5 Hash	X	X	R W
RIPEDM Hash	X	X	R W
SP800-38F Key Wrapping with AES Core	X	X	R
RSA Key Wrapping with RSAES-OAEP	X	X	R
RSA PKCS1-v1_5 Signature Verification Key sizes: 1536 bits, 4096 bits	X	X	R W
ECDSA Key Pair Generation for compact point representation of points	X	X	R W
ECDSA PKG: curves P-192, P-224, P-521 PKV: curves P-192, P-224, P-521 SIG(gen): curves P-192,P-224, P-521 SIG(ver): curves P-192,P-224 P-521	X	X	R W

Service	Roles		Access Type
	USER	CO	
Ed 25519 Key agreement, Signature Generation, Signature Verification	X	X	R W
SP800-56C Key Derivation Function	X	X	R W
Hash based Key Derivation Function using ANSI X9.63	X	X	R W
SP800-108 Key Derivation Function Modes: Feedback, Counter	X	X	R W
Key Derivation Function based on RFC6637	X	X	R W
CMAC AES 128 MAC Generation	X	X	R W
OMAC MAC Generation	X	X	R W
HMAC-DRBG Random Number Generation	X	X	R W
Hash-DRBG Random Number Generation	X	X	R W

Table 6b: 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 OS X CoreCrypto Kernel Module, v5.0 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 OS X CoreCrypto Kernel Module, v5.0.

6.1 Applicability

The Apple OS X CoreCrypto Kernel Module, v5.0 operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module is included in OS X 10.10, 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 OS X CoreCrypto Kernel Module, v5.0.

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

7.2 Key / CSP Generation

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

- The Approved DRBG specified in section 7.1 is used to generate cryptographic secret keys for symmetric key algorithms (AES, Triple-DES) and message authentication (HMAC).
- The Approved DRBG specified in section 7.1 is used to generate asymmetric key pairs for the ECDSA algorithm.

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

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

7.3 Key / CSP Establishment

The module provides key establishment services in the Approved Mode through the 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 password, 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 imported 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 OS X CoreCrypto Kernel Module, v5.0 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 OS X, including the separation between the kernel and user-space. No process can read the memory of another process. No user-space application can read the kernel memory.

7.6 Key / CSP Zeroization

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

8 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

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

9 Self-Tests

FIPS 140-2 requires that the module performs self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition, the DRBG requires continuous verification. The FIPS Self-Tests functionality runs all required module self-tests. This functionality is invoked by the OS X Kernel startup process upon device initialization. If the self-tests succeed, the CoreCrypto KEXT instance is maintained in the memory of the OS X 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 OS X CoreCrypto Kernel Module, v5.0 starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the following tests fails the system shuts down automatically. To run the self-tests on demand, the user may reboot the system.

9.1.1 Cryptographic Algorithm Tests

Algorithm	Modes	Test
Triple-DES	CBC	KAT (Known Answer Test) Separate encryption / decryption operations are performed
AES implementations selected by the module for the corresponding environment AES-128, AES-192, AES-256	ECB, CBC	KAT Separate encryption / decryption operations are performed
DRBG	N/A	KAT
SHA implementations selected by the module for the corresponding environment SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	KAT
HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	N/A	KAT
ECDSA	SIG(ver), SIG(gen)	KAT, pair-wise consistency test
RSA	SIG(ver)	KAT

Table 7: Cryptographic Algorithm Tests

9.1.2 Software / Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple OS X CoreCrypto Kernel Module, v5.0. 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 OS X CoreCrypto Kernel Module, v5.0.

9.2.1 Continuous Random Number Generator Test

The Apple OS X CoreCrypto Kernel Module, v5.0 performs a continuous random number generator test, whenever CTR_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 OS X CoreCrypto Kernel Module, v5.0 generates asymmetric keys and performs all required pair-wise consistency tests (signature generation and verification) with the newly generated key pairs.

9.2.3 SP 800-90A Assurance Tests

The Apple OS X CoreCrypto Kernel Module, v5.0 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 and prediction resistance.

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_OSX_KS_SECPOL_01.01

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 OS X 10.10. For additional assurance, it is digitally signed. The Approved Mode is configured by default and cannot be changed by a user.

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 cannot be changed. If the device starts up successfully then CoreCrypto KEXT has passed all self-tests and is operating in the Approved Mode.

10.4.2 User Guidance

The Approved Mode of operation is configured in the system by default. If the device starts up successfully then CoreCrypto KEXT has passed all self-tests and is operating in the Approved Mode.

Kernel programmers that use the module API shall not attempt to invoke any API call directly and only adhere to defined interfaces through the kernel framework.

11 Mitigation of Other Attacks

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

{0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},
{0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE},
{0x1F,0x1F,0x1F,0x1F,0x0E,0x0E,0x0E,0x0E},
{0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},
{0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},
{0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},
{0x1F,0xE0,0x1F,0xE0,0x0E,0xF1,0x0E,0xF1},
{0xE0,0x1F,0xE0,0x1F,0xF1,0x0E,0xF1,0x0E},
{0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},
{0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01},
{0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E,0xFE},
{0xFE,0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E},
{0x01,0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E},
{0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E,0x01},
{0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1,0xFE},
{0xFE,0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1}.