Apple Inc.

Apple corecrypto User Space Module for ARM (ccv10)
FIPS 140-2 Non-Proprietary Security Policy

Module Version 10.0

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

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

Trademarks

Apple’s trademarks applicable to this document are listed in https://www.apple.com/legal/intellectual-property/trademark/appletmlist.html. Other company, product, and service names may be trademarks or service marks of others.

Table of Contents

List of Tables

List of Figures

1 Introduction

2 Purpose

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

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

FIPS 140-2 details the security 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 Cryptographic Module Validation Program please refer to the NIST CMVP website [CMVP].

Throughout the document Apple corecrypto User Space Module for ARM (ccv10) is referred as : "cryptographic module", "corecrypto" or "the module" and "OS" refers to "iOS", "iPadOS", "tvOS", "watchOS" and "TxFW" unless specifically noted. "ccv10" is used to refer to the module version 10.0.

2.1 Document Organization / Copyright

This non-proprietary Security Policy document may be reproduced and distributed only in its original entirety without any revision, ©2022 Apple Inc.

2.2 External Resources / References

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

2.2.1 Additional References

CMVP Cryptographic Module Validation Program
https://csrc.nist.gov/projects/cryptographic-module-validation-program

CAVP Cryptographic Algorithm Validation Program

https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program

FIPS 140-2 Federal Information Processing Standards Publication, "FIPS PUB 140-2 Security Requirements for Cryptographic Modules," Issued May-25-2001, Effective 15-Nov-2001, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

FIPS 140-2 IG NIST, "Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program," August, 2020

https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/fips1402ig.pdf

FIPS 180-4 Federal Information Processing Standards Publication 180-4, Secure Hash Standard (SHS)

FIPS 186-4 Federal Information Processing Standards Publication 186-4, July 2013, Digital Signature Standard (DSS

FIPS 197 Federal Information Processing Standards Publication 197, November 26, 2001 Announcing the ADVANCED ENCRYPTION STANDARD (AES)

FIPS 198	Federal Information Processing Standards Publication 198, July, 2008 The Keyed-Hash Message Authentication Code (HMAC)
SP800-38 A	NIST Special Publication 800-38A, “Recommendation for Block Cipher Modes of Operation”, December 2001
SP800-38 C	NIST Special Publication 800-38C, “Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality”, May 2004
SP800-38 D	NIST Special Publication 800-38D, “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC”, November 2007
SP800-38 E	NIST Special Publication 800-38E, “Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices”, January 2010
SP800-38 F	NIST Special Publication 800-38F, “Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping”, December 2012
SP800-57P1	NIST Special Publication 800-57, “Recommendation for Key Management – Part 1: General),” July 2016
SP 800-90A	NIST Special Publication 800-90A, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators”
SP800-132	NIST Special Publication 800-132, “Recommendation for Password-Based Key Derivation”, December 2010
SEC	Security Overview https://developer.apple.com/security
OS	Technical Overview for all Apple Platforms https://developer.apple.com/
UGuide	User Guide https://support.apple.com/guide/ipad/welcome/ipados https://support.apple.com/guide/iphone/welcome/ios https://support.apple.com/guide/watch/welcome/watchos https://support.apple.com/guide/tv/welcome/tvos

2.3 Acronyms

AES	Advanced Encryption Standard
API	Application Programming Interface
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-Hellmann
DRBG	Deterministic Random Bit Generator
ECB	Electronic Codebook mode of operation
ECC	Elliptic Curve Cryptography
EC Diffie-Hellman	DH based on ECC
ECDSA	DSA based on ECC
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
FIPS PUB	FIPS Publication
GCM	Galois/Counter Mode
HMAC	Hash-Based Message Authentication Code
KAT	Known Answer Test
KDF	Key Derivation Function
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OFB	Output Feedback (mode of operation)
OS	Operating System
PBKDF	Password-based Key Derivation Function
PCT	Pair-wise Consistency Test
PRF	Pseudorandom Function
RNG	Random Number Generator
SHS	Secure Hash Standard
Triple-DES	Triple Data Encryption Standard
TLS	Transport Layer Security

3 Cryptographic Module Specification

3.1 Module Description

The Apple corecrypto User Space Module for ARM (ccv10) is a software cryptographic module version 10.0 running on a multi-chip standalone device. The cryptographic services provided by the module are:

- Data encryption and decryption
- Generation of hash values
- Key wrapping
- Message authentication
- Random number generation
- Key generation
- Digital signature generation and verification
- Key derivation

3.1.1 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

3.1.2 Module Components

There are no components excluded from the validation testing of the Apple corecrypto User Space Module for ARM (ccv10). corecrypto has an API layer that provides consistent interfaces to the supported algorithms. These implementations include proprietary optimization of algorithms that are fitted into the corecrypto framework.

3.1.3 Tested Platforms

The module has been tested with and without PAA on the following hardware platforms. PAA=NEON is present in Apple A, S and T series processors.

Manufacturer	Operating System	Processor (SoC)	Hardware Platform
Apple Inc.	iOS 13	Apple A9	iPhone 6S Plus
		Apple A10 Fusion	iPhone 7 Plus
		Apple A11 Bionic	iPhone 8 Plus
		Apple A12 Bionic	iPhone Xs Max

Manufacturer	Operating System	Processor (SoC)	Hardware Platform
		Apple A13 Bionic	iPhone 11 Pro Max
	iPadOS 13	Apple A8	iPad mini 4
		Apple A8X	iPad Air 2
		Apple A9	iPad (5th generation)
		Apple A9X	iPad Pro (9.7 inch)
		Apple A10 Fusion	iPad (6th generation)
		Apple A10X Fusion	iPad Pro (12.9-inch, 2nd generation)
		Apple A12 Bionic	iPad mini (5th generation)
		Apple A12X Bionic	iPad Pro (12.9-inch, 3rd generation)
	tvOS 13	Apple A10X Fusion	Apple TV 4K
	watchOS 6	Apple S1P	Apple Watch Series 1
		Apple S3	Apple Watch Series 3
		Apple S4	Apple Watch Series 4
		Apple S5	Apple Watch Series 5
	TxFW 10.15	Apple T2	Apple T2[1]

Table 2 Tested Platforms

In addition to the configurations tested by the laboratory, vendor-affirmed testing was performed on the following platforms:
for iOS13:

- iPhone 6s and iPhone SE with an Apple A9
- iPhone 7 with an Apple A10 Fusion
- iPhone 8 and iPhone X with an Apple A11 Bionic
- iPhone Xr and iPhone Xs with an Apple A12 Bionic
- iPhone 11 and iPhone 11 Pro with an Apple A13 Bionic

for iPadOS 13

- iPad Pro (12.9) with an Apple A9X
- iPad (7th generation) with an Apple A10 Fusion
- iPad Pro (10.5-inch) with an Apple A10X Fusion
- iPad Air (3rd generation) with an Apple A12 Bionic
- iPad Pro (11-inch) with an Apple A12X Bionic

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate (IG G.5).

3.2 Modes of Operation

The Apple corecrypto User Space Module for ARM (ccv10) has an Approved and non-Approved modes of operation. The Approved mode of operation with security functions listed in Table 3 is configured by default and cannot be changed. If the device starts up successfully then corecrypto framework has passed all self-tests and is operating in the Approved mode. Any calls to the non-Approved security functions listed in Table 4 will cause the module to assume the non-Approved mode of operation.

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

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

[1] The user for Apple T2 are iMac Pro, Mac Pro, Mac mini, MacBook Air and MacBook Pro

The module contains multiple implementations of the same cipher as listed Table 3. If multiple implementations of the same cipher are present, the module selects automatically which cipher is used based on internal heuristics. This includes the hardware-assisted AES (AES-NI) and SHA implementations.

3.2.1 Approved or Allowed Security Functions

The Algorithm Certificate Numbers (Table 3) are obtained from NIST for successful validation testing of the cryptographic algorithms implementations of the module that runs on the hardware platforms listed in Table 2

Please refer to [CAVP] website for the current standards, test requirements, and special abbreviations used in the following tables.

Cryptographic Function	Standard and Algorithm	Modes and Options	Algorithm Certificate Number
Random Number Generation	[SP 800-90A] DRBG	CTR_DRBG Modes: AES-128 AES-256 Derivation Function Enabled Without Prediction Resistance	A7 (c_asm) A8 (c_ltc) A10 (vng_asm)
		HMAC_DRBG Modes: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 Without Prediction Resistance	A8 (c_ltc) A9 (vng_ltc)
Symmetric Encryption and Decryption	[FIPS 197] AES SP 800-38 A SP 800-38 D SP 800-38 E	Key Length: 128, 192, 256 Modes: ECB CFB128 OFB CBC CTR XTS (key length: 128 and 256-bits only) CCM GCM CFB8	A7 (c_asm) A8 (c_ltc)
		Key Length: 128, 192, 256 Mode: CBC	A11 (c_glad)
		Key Length: 128, 192, 256 Modes: ECB GCM CTR CCM	A10 (vng_asm)
		Key Length: 128, 192, 256 Modes ECB CFB128 XTS (key length: 128 and 256-bits only) CBC OFB	A6 (asm_arm)
	[SP 800-67] Triple-DES	Keying Option: 1; All Keys Independent Modes: ECB CFB64 CBC CTR CFB8 OFB	A8 (c_ltc)

Cryptographic Function	Standard and Algorithm	Modes and Options	Algorithm Certificate Number
Key Wrapping	SP 800-38 D	Key Length: 128, 192, 256 Modes: AES-GCM AES-CCM	A7 (c_asm) A8 (c_ltc) A10 (vng_asm)
	SP 800-38 F	Key Length: 128, 192, 256 Modes: AES-KW	A7 (c_asm) A8 (c_ltc)
Digital Signature and Asymmetric Key Generation	[FIPS186-4] RSA	Key Generation (ANSI X9.31), Modulus: 2048, 3072, 4096 Signature Generation (PKCS#1 v1.5 and PSS) Modulus: 2048, 3072, 4096 Signature Verification (PKCS#1 v1.5 and PSS) Modulus: 1024, 2048, 3072, 4096	A8 (c_ltc) A9 (vng_ltc)
	[FIPS 186-4] ECDSA ANSI X9.62	Key Pair Generation (PKG): P-224, P-256, P-384, P-521 Public Key Validation (PKV): P-224, P-256, P-384, P-521 Signature Generation: P-224, P-256, P-384, P-521 Signature Verification: P-224, P-256, P-384, P-521	A8 (c_ltc) A9 (vng_ltc)
Message Digest	[FIPS 180-4] SHS	Modes SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	A8 (c_ltc) A9 (vng_ltc)
		Modes SHA-256	A12[2] (vng_neon)
Keyed Hash	[FIPS 198] HMAC	Key size: 112 bits or greater Modes: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	A8 (c_ltc) A9 (vng_ltc)
		Key size: 112 bits or greater Modes: HMAC-SHA-256	A12[2] (vng_neon)
Key Derivation	[SP 800-132] PBKDF	Password Based Key Derivation using HMAC with SHA-1 or SHA-224, SHA-256, SHA-384, SHA-512 PRFs	Vendor Affirmed[3] A8 (c_ltc) A9 (vng_ltc)
RSA Key Wrapping	[SP800-56B]	KTS RSA-OAEP Modulus size: 2048, 3072 or 4096-bits	Vendor Affirmed

[2] The S1P and S3 from the armv7 processor family do not implement vng_neon and do not have the A12 ACVT certificate.

[3] PBKDF is claimed as vendor affirmed despite having been CAVP tested. Since it is claimed as vendor affirmed, in accordance with FIPS 140-2 IG D.6, comment 2, self-testing is neither required nor implemented for this algorithm.

Table 3 Approved and Vendor Affirmed Security Functions

Cryptographic Function	Standard and Algorithm	Modes and Options	Algorithm Certificate Number
RSA Key Wrapping	Non-[SP 800-56B], IG D.9, [SP800-131A]	PKCS#1 v1.5 Modulus size: 2048, 3072 or 4096-bits	Non-Approved, but Allowed
MD5 (used as part of the TLS key establishment scheme only)	Message Digest	Digest Size: 128-bit	Non-Approved, but Allowed
NDRNG	Random Number Generation	N/A	Non-Approved, but Allowed; provided by the underlying operational environment

Table 3a Non-Approved but Allowed Security Functions

3.2.2 Non-Approved Security Functions

Cryptographic Function	Usage / Description	Caveat
RSA Signature Generation / Signature Verification / Asymmetric Key Generation	ANSI X9.31 Key Pair Generation Signature Generation Key Size < 2048 Key sizes: 1024-4096 bits in multiple of 32 bits not listed in table 3 Signature Verification Key Size < 1024 Key sizes: 1024-4096 bits in multiple of 32 bits not listed in table 3	Non-Approved
	PKCS#1 v1.5 and PSS Signature Generation Key sizes: 1024-4096 bits in multiple of 32 bits not listed in table 3 Key Size < 2048 Signature Verification Key sizes: 1024-4096 bits in multiple of 32 bits not listed in table 3 Key Size < 1024	
RSA Key Wrapping	PKCS#1 v1.5 and KTS RSA-OAEP Key Size < 2048	Non-Approved
ECDSA Asymmetric Key Generation	Key Pair Generation for compact point representation of points	Non-Approved
ECDSA Signature Generation / Signature Verification / Asymmetric Key Generation	PKG: Curve P-192 PKV: Curve P-192 Signature Generation: Curve P-192 Signature Verification: Curve P-192	Non-Approved
Integrated Encryption Scheme on elliptic curves	Encryption / Decryption	Non-Approved
Diffie-Hellman Key Generation	For all key sizes	Non-Approved

Cryptographic Function	Usage / Description	Caveat
Diffie-Hellman Shared Secret Computation	For all key sizes	Not compliant to 56A rev3
Diffie-Hellman Key Agreement	Key agreement scheme	Non-Approved
EC Diffie-Hellman Key Generation	Key agreement scheme	Non-Approved
EC Diffie-Hellman Shared Secret Computation	For all key sizes	Not compliant to 56A rev3
EC Diffie-Hellman Key Agreement	Key agreement scheme	Non-Approved
Ed25519	Key Agreement Signature Generation Signature Verification	Non-Approved
ANSI X9.63 KDF	Hash based Key Derivation Function	Non-Approved
RFC6637 KDF	KDF based on RFC6637	Non-Approved
DES	Encryption / Decryption Key Size: 56-bits	Non-Approved
CAST5	Encryption / Decryption: Key Sizes: 40 to 128 bits in 8-bit increments	Non-Approved
RC4	Encryption / Decryption: Key Sizes: 8 to 4096-bits	Non-Approved
RC2	Encryption / Decryption: Key Sizes: 8 to 1024-bits	Non-Approved
MD2	Message Digest Digest Size: 128-bits	Non-Approved
MD4	Message Digest Digest Size: 128-bits	Non-Approved
RIPEMD	Message Digest Digest Sizes: 160-bits	Non-Approved
Blowfish	Encryption / Decryption	Non-Approved
OMAC (One-Key CBC MAC)	MAC generation	Non-Approved
[SP800-56C]	Key Derivation Function	Non-Compliant
[SP800-108] KBKDF	Modes: Counter (CMAC-AES128, CMAC-AES192, CMAC-AES256) Feedback (HMAC-SHA-1 or HMAC-SHA-2) Counter (HMAC-SHA-1 or HMAC-SHA-2)	Non-Compliant A8 (c_ltc)
	Modes: Feedback (HMAC-SHA-1 or HMAC-SHA-2) Counter (HMAC-SHA-1 or HMAC-SHA-2)	Non-Compliant A9 (vng_ltc)
Triple-DES	Encryption / Decryption Two Key Implementation	Non-Compliant
	Optimized Assembler (asm_arm) Implementation Encryption / Decryption Mode: CTR	
AES-CMAC	AES-128/192/256 MAC generation / verification	Non-Compliant

Table 4 Non-Approved or Non-Compliant Security Functions

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

3.3 Cryptographic Module Boundary

The physical boundary of the module is the physical boundary of the iPhone, iPad, Apple TV, Apple Watch or T2 running iOS, iPadOS, tvOS, watchOS or TxFW respectively. Consequently, the embodiment of the cryptographic module is a multi-chip standalone.

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

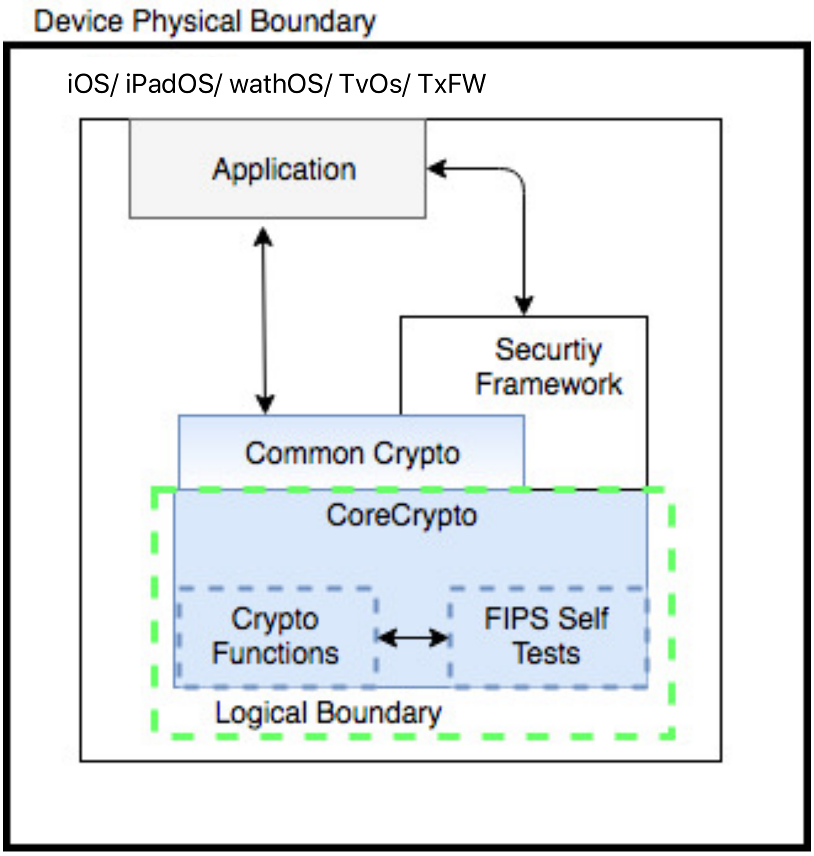

Figure 1: Logical Block Diagram

3.4 Module Usage Considerations

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

- AES-GCM IV is constructed in accordance with [SP800-38D] section 8.2.1.in compliance with IG A.5 scenario 1. The GCM IV generation follows RFC 5288 and shall only be used for the TLS protocol version 1.2. Users should consult [SP 800-38D], especially section 8, for all of the details and requirements of using AES-GCM mode. In case the module's power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed.
- AES-XTS mode is only approved for hardware storage applications. The length of the AES-XTS data unit does not exceed 2^{20} blocks
- 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 user 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.

4 Cryptographic Module Ports and Interfaces

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

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

The module is optimized for library use within the OS user space and does not contain any terminating assertions or exceptions. It is implemented as an OS dynamically loadable library. The dynamically loadable library is loaded into the OS application and its cryptographic functions are made available. Any internal error detected by the module is reflected back to the caller with an appropriate return code. The calling OS application must examine the return code and act accordingly. There is one notable exception: ECDSA and RSA do not return a key if the pair-wise consistency test fails.

The function executing FIPS 140-2 module self-tests does not return an error code but causes the system to crash if any self-test fails – see Section 10.

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

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

Cryptographic bypass capability is not supported by the module.

5 Roles, Services and Authentication

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

5.1 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
User	Utilization of services (section 4.2) of the module tested on hardware platforms section 2.1.
Crypto Officer (CO)	Utilization of services (section 4.2) of the module tested on hardware platforms section 2.1.

Table 5 Roles

5.2 Services

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

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

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

The access types are denoted as follows:

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

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

Service	Roles		CSPs and crypto keys	Access Type
	User	CO		
AES Key Wrapping Encryption Input: plaintext, key Output: ciphertext Decryption Input: ciphertext, key Output: plaintext	X	X	AES key	R
RSA Key Wrapping using RSA-OAEP Encryption Input: plaintext, the modulus n, the public key e Output: ciphertext Decryption Input: ciphertext, the modulus n, the private key d Output: plaintext	X	X	RSA key pair	R
RSA Key Wrapping Using PKCS#1 v1.5 (non-approved but allowed) Encryption Input: plaintext, the modulus n, the public key e Output: ciphertext Decryption Input: ciphertext, the modulus n, the private key d Output: plaintext	X	X	RSA key pair	R
Secure Hash Generation using SHA1, SHA-224, SHA-256, SHA-384, or SHA-512 Input: message Output: message digest	X	X	none	N/A
Secure Hash Generation using MD5 (non-approved but allowed)	X	X	none	N/A
HMAC generation using HMAC-SHA1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-512 Input: HMAC key, message Output: HMAC value of message	X	X	HMAC key	R
RSA signature generation and verification Signature generation Input: modulus n, private key d, SHA algorithm (SHA-224/ SHA-256/ SHA-384/SHA-512), a message m to be signed Output: the signature s of the message Signature verification Input: the modulus n, the public key e, SHA algorithm (SHA-1/ SHA-224/ SHA-256/ SHA-384/ SHA-512), a message m, a signature for the message Output: pass if the signature is valid, fail if the signature is invalid	X	X	RSA key pair	R

Service	Roles		CSPs and crypto keys	Access Type
	User	CO		
ECDSA signature generation and verification Signature generation Input: message m, q, a, b, X_G, Y_G, n, SHA algorithm (SHA-224/ SHA-256/ SHA-384/ SHA-512), sender's private key d Output: signature of m as a pair of r and s Signature verification Input: received message m', signature in form on r' and s' pair, q, a, b, X_G, Y_G, n, sender's public key Q, the SHA algorithm (SHA-1/ SH-224/ SHA-256/ SHA-384/SHA-512) Output: pass if the signature is valid, fail if the signature is invalid	X	X	ECDSA key pair	R
ECDSA key pair generation Input: q, FR, a, b, domain_parameter_seed, G, n, h. Output: private key d, public key Q	X	X	ECDSA private key	W
Random number generation Input: Entropy Input, Nonce, Personalization String Output: Returned Bits	X	X	Entropy input string, Nonce, V and Key	R W Z
PBKDF Password-based key derivation Input: salt, password, Iteration count, key length. Output: derived key	X	X	PBKDF derived key, PBKDF password	R W Z
RSA key pair generation Input: modulus size, the public key, random numbers: X_{p1}, X_{p2}, X_{q1} and X_{q2} Output: the private prime factor p, the private prime factor q, the value of the modulus n, the value of the private signature, exponent d	X	X	RSA key pair	W
Release all resources of symmetric crypto function context (i.e., Symmetric Key Zeroization) Input: context Output: N/A	X	X	AES/Triple-DES key	Z
Release all resources of hash context (i.e., MAC Key Zeroization) Input: context Output: N/A	X	X	HMAC key	Z
Release of all resources of asymmetric crypto function context (Asymmetric Key Zeroization) Input: context Output: N/A	X	X	RSA key pairs	Z
Reboot Input: N/A Output: N/A	X	X	N/A	N/A
Self-test Input: N/A Output: pass if the Self-test is successful, fail if the Self-test is unsuccessful	X	X	Software integrity key	R
Show Status Input: N/A Output: Status of module	X	X	None	N/A

Table 6 Approved and Allowed Services in Approved Mode

Service	Roles: User	Roles: CO
Integrated Encryption Scheme on elliptic curves encryption / decryption	X	X
DES Encryption / Decryption	X	X
Triple-DES Encryption / Decryption: asm_arm implementation CTR mode (non-compliant) Triple-DES Encryption / Decryption: Two-Key Triple-DES (non-approved)	X	X
CAST5 Encryption/ Decryption	X	X
Blowfish Encryption / Decryption	X	X
RC2 Encryption / Decryption	X	X
RC4 Encryption /Decryption	X	X
MD2 Message Digest Generation	X	X
MD4 Message Digest Generation	X	X
RIPEMD Message Digest Generation	X	X
RSA PKCS#1 (v1.5 and PSS) Signature Generation / Verification Key sizes: 1024-4096 bits in multiple of 32 bits not listed in table 3	X	X
RSA PKCS#1 (v1.5 and PSS) and ANSI X9.31 Signature Generation, Key Size < 2048 Signature Verification, Key Size < 1024	X	X
RSA ANSI X9.31 Key Pair Generation, Signature Generation, Signature Verification Key sizes (modulus): 1024-4096 bits in multiple of 32 bits not listed in table 3 Public key exponent values: 65537 or larger	X	X
RSA PKCS#1 v1.5 and KST RSA-OAEP Key Wrapping Key sizes < 2048	X	X
ECDSA Key Pair Generation for compact point representation of points	X	X
Diffie-Hellman Key Generation	X	X
Diffie-Hellman Shared Secret Computation	X	X
Diffie-Hellman Key Agreement	X	X
EC Diffie-Hellman Key Generation	X	X
EC Diffie-Hellman Shared Secret Computation	X	X
EC Diffie-Hellman Key Agreement	X	X
ECDSA Key Generation: curve P-192 Public Key Verification: curve P-192 Signature Generation / Verification: curve P-192	X	X
Ed25519 Key agreement/ Signature Generation/ Signature Verification	X	X
[SP800-56C] Key Derivation	X	X
ANSI X9.63 Hash Based Key Derivation	X	X
RFC6637 Key Derivation	X	X
[SP800-108] Key Derivation using HMAC-SHA1 or HMAC-SHA-224 or HMAC-SHA-256 or HMAC-SHA-384 or HMAC-SHA-512 and AES-CMAC Based Pseudo Random Functions Modes: Feedback, Counter	X	X
AES-CMAC MAC Generation/ Verification	X	X
OMAC MAC Generation	X	X

Table 7 Non-Approved Services in Non-Approved Mode

5.3 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.

6 Physical Security

The FIPS 140-2 physical security requirements do not apply to the Apple corecrypto User Space Module for ARM (ccv10) since it is a software module.

7 Operational Environment

7.1 Applicability

The Apple corecrypto User Space Module for ARM (ccv10) operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. It is part of a commercially available general-purpose operating system executing on the hardware specified in section 3.1.3.

7.2 Policy

The operating system is restricted to a single operator (single-user mode; i.e. concurrent operators are explicitly excluded).

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

8 Cryptographic Key Management

The Table 8 summarizes the cryptographic keys and CSPs used in the Apple corecrypto User Space Module for ARM (ccv10), with the ley lengths supported, the available methods for key generation, key entry and key output, and zeroization.

Name	Key / CSP Size	Generation	Entry / Output	Zeroization
AES Keys	128, 192, 256 bits	N/A. The key is entered via API parameter	Entry : calling application (see 7.4) Output: N/A	automatic zeroization when structure is deallocated or when the system is powered down (see 7.6).
HMAC Keys	min 112- bits			
Triple-DES Keys	192 bits			
ECDSA key pair	P-224, P-256, P-384, P-521 curves	The private keys are generated using FIPS186-4 Key Generation method, and the random value used in the key generation is generated using SP800-90A DRBG	Entry : calling application (see 7.4) Output: calling application (see 7.4)	
RSA key pair	2048, 3072, 4096			
Entropy Input string		Obtained from the NDRNG.	Entry: OS Output: N/A	
DRBG nonce		Obtained from the NDRNG.		
DRBG V, Key		Derived from entropy input string as defined by SP800-90A	Entry: N/A Output: N/A	
PBKDF Keys	min: 112 bits	Internally generated via SP800-132 PBKDF key derivation algorithm	Entry: N/A Output: calling application (see 7.4)	
PBKDF Password		N/A. The password is provided by calling application	Entry : calling application (see 7.4) Output: N/A	

Table 8 Module Cryptographic key and CSPs

8.1 Random Number Generation

A FIPS 140-2 approved deterministic random bit generator based on a block cipher as specified in NIST [SP 800-90A] is used. The default Approved DRBG used for random number generation is a CTR_DRBG using AES-256 with derivation function and without prediction resistance. The module also employs a HMAC-DRBG for random number generation. The deterministic random bit generators are seeded by the /dev/random interface. The /dev/random is the User Space interface that receives random bits from an entropy source composed by a Fortuna PRNG and the and the NDRNG from the ARM-based processor. The entropy source provides 256-bits of security strength in seeding and reseeding the module approved DRBGs.

8.2 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 [FIPS 186-4], and using DRBG compliant with [SP800-90A]. A seed (i.e. the random value) used in asymmetric key generation is obtained from [SP800-90A] DRBG. The generated seed is an unmodified output from the DRBG. The key generation service for RSA and ECDSA as well as the [SP 800-90A] DRBG have been ACVT tested with algorithm certificates found in Table 3.

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

8.3 Key / CSP Establishment

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

- AES key wrapping using KW with certs #A7 and #A8 and AES in GCM and CCM modes with CAVP certificates #A7, #A8 and #A10. The key establishment methodology provides between 128 and 256 bits of encryption strength.
- RSA key wrapping. RSA key wrapping encompasses:
    - RSA key wrapping using OAEP mode compliant to [SP 800-56B] (vendor affirmed)
    - RSA key wrapping using PKCS#1 v1.5, non-approved but allowed per IG D.9
- [SP 800-132] PBKDFv2 algorithm.

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.

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.

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

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

8.4 Key / CSP Entry and Output

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

8.5 Key / CSP Storage

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

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

8.6 Key / CSP Zeroization

Keys and CSPs are zeroized when the appropriate context object is destroyed or when the device is powered down. Additionally, the user can zeroize the entire device directly (locally) or remotely, returning it to the original factory settings.

9 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The EMI/EMC properties of the corecrypto 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.

10 Self-Tests

FIPS 140-2 requires that the module perform self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition, the random bit generator requires continuous verification. The FIPS Self Tests application runs all required module self-tests. This application is invoked by the OS startup process upon device power on.

The execution of an independent application for invoking the self-tests in the corecrypto.dylib makes use of features of the OS architecture: the module, implemented in corecrypto.dylib, is linked by libcommoncrypto.dylib which is linked by libSystem.dylib. The libSystem.dylib is a library that must be loaded into every application for operation. The library is stored in the kernel cache and therefore is not available on the disk as directly visible files. The OS ensures that there is only one physical instance of the library and maps it to all application linking to that library. In this way the module always stays in memory. Therefore, the self-test during startup time is sufficient as it tests the module instance loaded in memory which is subsequently used by every application on the OS.

All self-tests performed by the module are listed and described in this section.

10.1Power-Up Tests

The following tests are performed each time the Apple corecrypto User Space Module for ARM (ccv10) starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the following tests fails the device powers itself off. To rerun the self-tests on demand, the user must reboot the device.

10.1.1 Cryptographic Algorithm Tests[4]

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

Table 9 Cryptographic Algorithm Tests

[4] The module also includes KATs for DH and ECDH shared secret computation but they are a non-approved algorithms hence are not listed in this table.

10.1.2 Software / Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple corecrypto User Space Module for ARM (ccv10). The corecrypto’s HMAC-SHA-256 is used as an Approved algorithm for the integrity test. If the test fails, then the device powers itself off.

10.1.3 Critical Function Tests

No other critical function test is performed on power up.

10.2 Conditional Tests

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

10.2.1 Continuous Random Number Generator Test

The Apple corecrypto User Space Module for ARM (ccv10) performs a continuous random number generator test on the noise source (i.e. NDRNG), whenever it is invoked to seed the SP800-90A DRBG.

10.2.2 Pair-wise Consistency Test

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

10.2.3 SP 800-90A Health Tests

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

10.2.4 Critical Function Test

No other critical function test is performed conditionally.

11 Design Assurance

11.1 Configuration Management

Apple manages and records source code and associated documentation files by using the revision control system called "Git".

The Apple module hardware data, which includes descriptions, parts data, part types, bills of materials, manufacturers, changes, history, and documentation are managed and recorded. Additionally, configuration management is provided for the module's FIPS documentation.

The following naming/numbering convention for documentation is applied.

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

Example: FIPS_CORECRYPTO_IOS_tvOS_US_SECPOL_4.0

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

11.2 Delivery and Operation

The corecrypto is built into the OS. 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.

11.3 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.

11.4 Guidance

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

11.4.1 Cryptographic Officer Guidance

The Approved mode of operation is configured in the system by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved algorithms listed in Table 4. If the device starts up successfully then corecrypto has passed all self-tests and is operating in the Approved mode.

11.4.2 User Guidance

As above, the Approved mode of operation is configured in the system by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved algorithms listed in Table 4. If the device starts up successfully then corecrypto has passed all self-tests and is operating in the Approved mode.

12 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}.