Apple Inc.

Apple iOS CoreCrypto Module, v7.0
FIPS 140-2 Non-Proprietary Security Policy

Document Control Number
FIPS_CORECRYPTO_IOS_US_SECPOL_3.2

January, 2017

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 5

1.1 PURPOSE 5
1.2 DOCUMENT ORGANIZATION / COPYRIGHT 5
1.3 EXTERNAL RESOURCES / REFERENCES 5
1.3.1 Additional References 5
1.4 ACRONYMS 7

2 CRYPTOGRAPHIC MODULE SPECIFICATION 8

2.1 MODULE DESCRIPTION 8
2.1.1 Module Validation Level 8
2.1.2 Module Components 8
2.1.3 Tested Platforms 9
2.2 MODES OF OPERATION 9
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 22

5 PHYSICAL SECURITY 23

6 OPERATIONAL ENVIRONMENT 24

6.1 APPLICABILITY 24
6.2 POLICY 24

7 CRYPTOGRAPHIC KEY MANAGEMENT 25

7.1 RANDOM NUMBER GENERATION 25
7.2 KEY / CSP GENERATION 25
7.3 KEY / CSP ESTABLISHMENT 25
7.4 KEY / CSP ENTRY AND OUTPUT 25
7.5 KEY / CSP STORAGE 25
7.6 KEY / CSP ZEROIZATION 26

8 ELECTROMAGNETIC INTERFERENCE/ELECTROMAGNETIC COMPATIBILITY (EMI/EMC) 27

9 SELF-TESTS 28

9.1 POWER-UP TESTS 28
9.1.1 Cryptographic Algorithm Tests 28
9.1.2 Software / Firmware Integrity Tests 29
9.1.3 Critical Function Tests 29
9.2 CONDITIONAL TESTS 29
9.2.1 Continuous Random Number Generator Test 29
9.2.2 Pair-wise Consistency Test 29
9.2.3 SP 800-90A Assurance Tests 29
9.2.4 Critical Function Test 29

10 DESIGN ASSURANCE 30

10.1 CONFIGURATION MANAGEMENT 30
10.2 DELIVERY AND OPERATION 30
10.3 DEVELOPMENT 30
10.4 GUIDANCE 30
10.4.1 Cryptographic Officer Guidance 30
10.4.2 User Guidance 30
11 MITIGATION OF OTHER ATTACKS 31

List of Tables

Table 1: Module Validation Level 8
Table 2: Tested Platforms 9
Table 3: Approved or Allowed Security Functions 11
Table 4: Non-Approved Functions 13
Table 5: Roles 16
Table 6: Approved and Allowed Services in Approved Mode 20
Table 6b – Non-Approved Services in Non-Approved Mode 22
Table 7: Cryptographic Algorithm Tests 28

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 iOS CoreCrypto Module, v7.0. 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 iOS CoreCrypto Module, v7.0.” “cryptographic module”, “CoreCrypto” or “the module” are used interchangeably to refer to the Apple iOS CoreCrypto Module, v7.0.

1.2 Document Organization / Copyright

This non-proprietary Security Policy document may be reproduced and distributed only in its original entirety without any revision, ©2017 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 iOS and its security properties refer to [iOS] and [SEC]. For details on iOS releases with their corresponding validated modules and Crypto Officer Role Guides refer to the Apple Knowledge Base Article HT202739 - “ Product security certifications, validations, and guidance for iOS” (https://support.apple.com/en-us/HT202739)

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

1.3.1 Additional References

FIPS 140-2	Federal Information Processing Standards Publication, “FIPS PUB 140-2 Security Requirements for Cryptographic Modules,” Issued May-25-2001, Effective 15-Nov-2001, Location: http://csrc.nist.gov/groups/STM/cmvp/standards.html
FIPS 140-2 IG	NIST, “Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program,” November 15, 2016 Location: http://csrc.nist.gov/groups/STM/cmvp/standards.html
FIPS 180-4	Federal Information Processing Standards Publication 180-4, March 2012, Secure Hash Standard (SHS)
FIPS 186-4	Federal Information Processing Standards Publication 186-4, July 2013, Digital Signature Standard (DSS)
FIPS 197	Federal Information Processing Standards Publication 197, November 26, 2001 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 800-38E, “Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping”, December 2012

SP800-57P1 NIST Special Publication 800-57, “Recommendation for Key Management – Part 1: General (Revised),” July 2012

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

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

SEC Security Overview

Location: http://developer.apple.com/library/ios/#documentation/Security/Conceptual/Security_Overview/Introduction/Introduction.html

iOS iOS Technical Overview

Location: http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898

UG User Guide

Location: https://support.apple.com/en-us/HT202739

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-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
API	Kernel Programming Interface
KS	Key Size (Length)
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OFB	Output Feedback (mode of operation)
OS	Operating System
PBKDF	Password-based Key Derivation Function
PCT	Pair-wise Consistency Test
RNG	Random Number Generator
SHS	Secure Hash Standard
Triple-DES	Triple Data Encryption Standard
TLS	Transport Layer Security

2 Cryptographic Module Specification

2.1 Module Description

The Apple iOS CoreCrypto Module, v7.0 is a software cryptographic module running on a multi-chip standalone mobile device.

The cryptographic services provided by the module are:

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

2.1.1 Module Validation Level

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

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

Table 1: Module Validation Level

2.1.2 Module Components

In the following sections the components of the Apple iOS CoreCrypto Module, v7.0 are listed in detail. There are no components excluded from the validation testing.

2.1.2.1 Software components

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

2.1.3 Tested Platforms

The module has been tested on the following platforms:

Manufacturer	Model	Operating System
Apple Inc.	iPhone5S with Apple A7 CPU	iOS 10.2[1]
Apple Inc.	iPhone6 with Apple A8 CPU (iPhone6 and iPhone6 Plus)	iOS 10.2
Apple Inc.	iPhone6S with Apple A9 CPU (iPhone6S and iPhone6S Plus)	iOS 10.2
Apple Inc.	iPhone7 with Apple A10[2] CPU (iPhone7 and iPhone7 Plus)	iOS 10.2
Apple Inc.	iPad Air 2 with Apple A8X CPU	iOS 10.2
Apple Inc.	iPad Pro with Apple A9X CPU	iOS 10.2

Table 2: Tested Platforms

2.2 Modes of Operation

The Apple iOS CoreCrypto Module, v7.0 has an Approved and non-Approved modes of operation. The Approved mode of operation 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 (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 selects automatically which cipher is used based on internal heuristics.

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

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

[1] Throughout the document iOS 10.2 is generically referred to as iOS 10.

[2] Apple A10 is also known as Apple A10 Fusion.

Approved or Allowed Security Functions

Cryptographic Function	Algorithm	Modes/Options	Algorithm Certificate Number
Random Number Generation; Symmetric Key Generation	[SP 800-90] DRBG	CTR_DRBG Generic Software Implementation Key Size: 128 bit key size	1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1339
		CTR_DRBG (AES Optimized Assembler Implementation) Key Size: 128 bit key size	1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275
		HMAC_DRBG Generic Software Implementation SHA-1, SHA-2 (224, 256, 384, 512)	1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1339
Symmetric Encryption and Decryption	[FIPS 197] AES SP 800-38 A SP 800-38 D SP 800-38 E SP 800-38 F	Generic Software Implementation (Based on LibTomCrypt): Key Sizes: 128/192/256 bits Modes: CFB8, CFB128, CTR, CCM, ECB, GCM, OFB, XTS, KW	4180, 4181, 4182, 4183, 4184, 4185, 4186, 4187, 4188, 4189, 4190, 4269
		Generic Software Implementation (Based on Gladman): Key Sizes: 128/192/256 bits Modes: CBC	4165, 4167, 4168, 4171, 4172, 4173, 4174, 4175, 4176, 4177, 4178, 4179
		Optimized Assembler Implementation: Key Sizes: 128/192/256 bits Modes: CBC, CFB8, CFB128, CCM, CTR, ECB, GCM, KW, OFB, XTS	4156, 4157, 4158, 4159, 4160, 4161, 4162, 4163, 4164, 4166, 4169, 4170
	[SP 800-67] Triple-DES ANSIX9.52-1 998 FIPS 46-3 SP 800-38A Appendix E	Triple-DES (3 independent keys) Modes: ECB, CBC, CFB8, CFB64, OFB, CTR	2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2308
Digital Signature and Asymmetric Key Generation	[FIPS186-4] RSA PKCS #1.5	GenKey9.31 (2048/3072) SigGenPKCS1.5 (2048/3072) SigVerPKCS1.5 (1024/2048/3072)	2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2299
	[FIPS 186-4] ECDSA ANSI X9.62	PKG: curves P-256, P-384 PKV: curves P-256, P-384 SIG(gen): curves P-256, P-384 using (SHA-224, SHA-256, SHA384, SHA512) SIG(ver): curves P-256, P-384 using (SHA-1, SHA-224, SHA-256, SHA384, SHA512)	957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 997
Message Digest	[FIPS 180-4] SHS	Generic Software Implementation: SHA-1, SHA-2 (224, 256, 384, 512)	3421, 3422, 3423, 3424, 3425, 3426, 3427, 3428, 3429, 3430, , 3443, 3514

Cryptographic Function	Algorithm	Modes/Options	Algorithm Certificate Number
		Optimized Assembler Implementation using VNG: SHA-1, SHA-2 (224, 256, 384, 512)	3431, 3432, 3433, 3434, 3435, 3436, 3437, 3438, 3439, 3440, 3441, 3442
Keyed Hash	[FIPS 198] HMAC	Generic Software Implementation: SHA-1, SHA-2 (224, 256, 384, 512) Key Size: at least 112 bits	2723, 2724, 2725, 2726, 2727, 2728, 2729, 2730, 2731, 2732, 2745, 2813
		Optimized Assembler Implementation using VNG: SHA-1, SHA-2 (224, 256, 384, 512) Key Size: at least 112 bits	2733, 2734, 2735, 2736, 2737, 2738, 2739, 2740, 2741, 2742, 2743, 2744
KAS	[SP800-56A] EC Diffie-Hellman Implementation follows SP800-56A for primitive only	6.2.2.2 One-Pass Diffie-Hellman, C(1e, 1s, ECC CDH) Curves:P-256, P384	CVL: 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 1010
Key Derivation	[SP 800-132] PBKDF	Password Based Key Derivation using HMAC with SHA-1 or SHA-2	Vendor Affirmed
RSA Key Wrapping	SP800-56B	KTS-OAEP	Vendor Affirmed

Table 3: Approved or Allowed Security Functions

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

Note: PBKDFv2 is implemented to support all options specified in Section 5.4 of SP800-132. The password consists of at least 6 alphanumeric characters from the ninety-six (96) printable and human-readable characters. The probability that a random attempt at guessing the password will succeed or a false acceptance will occur is equal to 1/96^6. The derived keys may only be used in storage applications. Additional guidance to appropriate usage is specified in section 7.3.

Non-Approved Security Functions:

Cryptographic Function	Usage / Description	Caveat
RSA Key Wrapping	Encryption / Decryption RSAES-PKCS1-v1_5 PKCS#1 v2.1	Non-Approved, but allowed: RSA (key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength; non-compliant less than 112 bits of encryption strength)

Cryptographic Function	Usage / Description	Caveat
RSA Signature Generation / Signature Verification / Asymmetric Key Generation	ANSI X9.31 Signature Generation Signature Verification Key Pair Generation Key Size: 1024-4096 bits in multiple of 32 bits not listed in Table 3	Non-Compliant
	PKCS1-v1_5 Signature Generation Signature Verification Key Size: 1024-4096 bits in multiple of 32 bits not listed in Table 3	
ECDSA Asymmetric Key Generation	Key Pair Generation for compact point representation of points	Non-Approved
ECDSA Signature Generation / Signature Verification / Asymmetric Key Generation	PKG: curves P-224, P-521 PKV: curves P-224, P-521 SIG(gen): curves P-224, P-521 SIG(ver): curves P-192, P-224, P-521	Non-Compliant
	PKG: curves P-192 PKV: curves P-192 SIG(gen): curves P-192	Non-Approved
Integrated Encryption Scheme on elliptic curves	Encryption / Decryption	Non-Approved
Diffie-Hellman	Key Agreement ANSI X9.42, SP 800-56A Key sizes: Min 2048 bits, Max 3072 bits	Non-Approved, but allowed: Diffie-Hellman (key agreement; key establishment methodology provides 112 or 128 bits of encryption strength).
	Key sizes < 2048 bits	Non-Approved
EC Diffie-Hellman	Key Agreement ANSI X9.63, SP 800-56A Bit length of ECC subgroup order P-256, P-384	Non-Approved, but allowed: EC Diffie-Hellman (key agreement; key establishment methodology provides 128 bits or 160 bits of encryption strength - the strength for P-384 is limited by the entropy of the seed source as specified in the caveat).
Ed25519	Key Agreement Sig(gen) Sig(ver)	Non-Approved
SP800-108 KBKDF	Key Based Key Derivation Function Modes: CTR, Feedback	Non-Compliant
SP800-56C	Key Derivation Function	Non- Compliant

Cryptographic Function	Usage / Description	Caveat
ANSI X9.63 KDF	Hash based KDF based on ANSI X9.63	Non-Approved
RFC6637 KDF	KDF based on RFC6637	Non-Approved
DES Symmetric Encryption and Decryption	Encryption / Decryption Key Size: 56 bits	Non-Approved
Triple-DES Symmetric Encryption and Decryption	Optimized Assembler Implementation: Encryption / Decryption Block Chaining Mode: CTR	Non-Compliant
	Two-Key Triple-DES Encryption / Decryption	
CAST5	Encryption / Decryption: Key Sizes: 40 to 128 bits in 8-bit increments	Non-Approved
RC4	Encryption / Decryption: Key Sizes: 8 to 4096 bits	Non-Approved
RC2	Encryption / Decryption: Key Sizes: 8 to 1024 bits	Non-Approved
MD2	Message Digest Digest Size: 128 bit	Non-Approved
MD4	Message Digest Digest Size: 128 bit	Non-Approved
MD5	Message Digest Digest Size: 128 bit	Non-Approved, but allowed: Used as part of the TLS key establishment scheme only
RIPEMD	Message Digest Digest Sizes: 128, 160, 256, 320 bits	Non-Approved
Blowfish	Encryption / Decryption	Non-Approved
AES-CMAC	AES-128 MAC generation	Non-Compliant
OMAC (One-Key CBC MAC)	MAC generation	Non-Approved
Hash_DRBG	Hash_DRBG using SHA1 derived from the SP800-90A without derivation function, without prediction resistance	Non-Compliant

Table 4: Non-Approved Functions

The encryption strengths included in Table 4 for the key establishment methods are determined in accordance with FIPS 140-2 Implementation Guidance [IG] section 7.5 and NIST Special Publication 800-57 (Part1) [SP800-57P1].

Note: A Non-Approved function in Table 4 is that the function implements a non-Approved algorithm, while a Non-Compliant function is that the function implements an Approved algorithm but the implementation is not validated by the CAVP. Neither a Non-Compliant nor a Non-Approved function may be used in the Approved mode unless stated in the caveat found in Table 4.

2.3 Cryptographic Module Boundary

The physical boundary of the module is the physical boundary of the iOS device (iPhone or iPad) that contains the module. Consequently, the embodiment of the module is a multi-chip standalone cryptographic module.

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

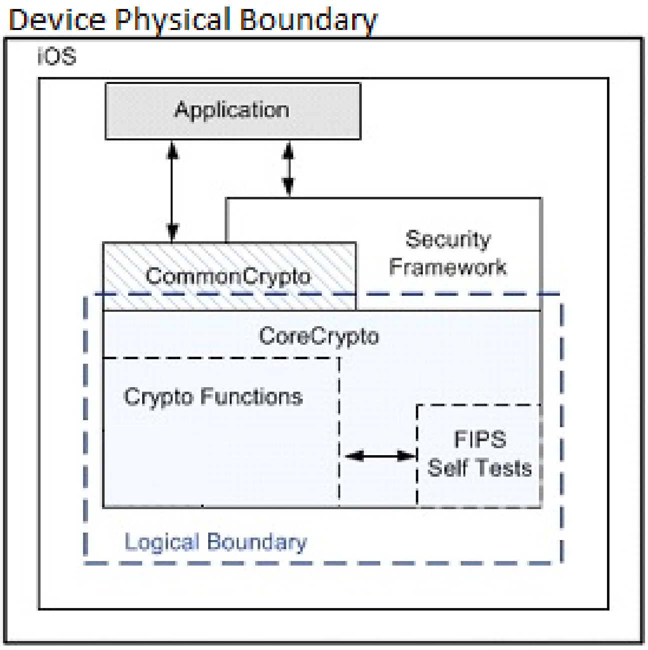

Figure 1: Logical Block Diagram

2.4 Module Usage Considerations

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

- AES-GCM IV is constructed in accordance with SP800-38D section 8.2.1. Users should consult SP 800-38D, especially section 8, for all of the details and requirements of using AES-GCM mode.
- In case the module's power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed.
- When using AES, the caller must obtain a reference to the cipher implementation via the functions of ccaes_[cbc|ecb|…]_[encrypt|decrypt]_mode.
- When using SHA, must obtain a reference to the cipher implementation via the functions ccsha[1|224|256|384|512]_di.

3 Cryptographic Module Ports and Interfaces

The underlying logical interfaces of the module are the C language 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 /var/db/FIPS/fips_data 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 iOS user space and does not contain any terminating assertions or exceptions. It is implemented as an iOS dynamically loadable library. The dynamically loadable library is loaded into the iOS 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 iOS application must examine the return code and act accordingly. There are two notable exceptions: (i) ECDSA and RSA do not return a key if the pair-wise consistency test fails; (ii) the DRBG algorithm loops a few iterations internally if the continuous test fails, eventually recovering from the error or causing a shutdown if the problem persists.

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

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

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

Cryptographic bypass capability is not supported by the module.

4 Roles, Services and Authentication

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

4.1 Roles

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

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

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 and non-Approved mode. For each available service it lists, the associated role, the Critical Security Parameters (CSPs) and cryptographic keys involved, and the type(s) of access to the CSPs and cryptographic keys.

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

The access types are denoted as follows:

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

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

Service	Roles		CSPs & crypto keys	Access Type
	USER	CO		
AES encryption and decryption Encryption *Input:* plaintext, IV, key *Output:* ciphertext Decryption *Input:* ciphertext, IV, key *Output:* plaintext	X	X	secret key	R
AES Key Wrapping Encryption *Input:* plaintext, key *Output:* ciphertext Decryption *Input:* ciphertext, key *Output:* plaintext	X	X	secret key	R
RSA Key Wrapping Encryption *Input:* plaintext, the modulus n, the public key e, the SHA algorithm (SHA-224/SHA-256/SHA-384/SHA-512) *Output:* ciphertext Decryption *Input:* ciphertext, the modulus n, the private key d, the SHA algorithm (SHA-224/SHA-256/SHA-384/SHA-512) *Output:* plaintext	X	X	secret key	R
Secure Hash Generation *Input:* message *Output:* message digest	X	X	none	N/A
HMAC generation *Input:* HMAC key, message *Output:* HMAC value of message	X	X	secret HMAC key	R

Service	Roles		CSPs & crypto keys	Access Type
	USER	CO		
RSA signature generation and verification Signature generation *Input:* the modulus n, the private key d, the 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, the SHA algorithm (SHA-1/SHA -224/SHA-256/SHA-384/SHA- 512), a message m, a signature for the message Output: pass if the signature is valid, fail if the signature is invalid	X	X	secret key	R W
ECDSA signature generation and verification Signature generation *Input:* message m, q, a, b, X_G, Y_G, n, the SHA algorithm (SHA -224/SHA-256/SHA-384/SHA- 512), sender's private key d *Output:* signature of m as a pair of r and s Signature verification *Input:* received message m', signature in form on r' and s' pair, q, a, b, X_G, Y_G, n, sender's public key Q, the SHA algorithm (SHA-1/SHA -224/SHA-256/SHA-384/SHA- 512) *Output:* pass if the signature is valid, fail if the signature is invalid	X	X	secret key	R W

Service	Roles		CSPs & crypto keys	Access Type
	USER	CO		
Random number generation *Input:* Entropy Input, Nonce, Personalization String *Output:* Returned Bits	X	X	Entropy input string, Nonce, V and K	R W Z
PBKDF Password-based key derivation *Input:* encrypted key and password *Output:* plaintext key or *Input:* plaintext key and password *Output:* encrypted data	X	X	Secret key, password	R W Z
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	Asymmetric key pair	R W
ECDSA (key pair generation) *Input: q, FR, a, b, domain_parameter_seed, G, n, h.* *Output: private key d, public key Q*	X	X	Asymmetric key pair	R W
Diffie-Hellman Key agreement *Input:* prime number (p), base (g), secret integers(a,b) *Output:* shared secret	X	X	Asymmetric keys (RSA/ECDSA key) and secret session key (AES/Triple-DES key)	R W
EC Diffie-Hellman Key agreement *Input:* domain parameter (p,a,b,G,n,h), key pair (d, Q) *Output:* shared secret	X	X	Asymmetric keys (RSA/ECDSA key) and secret session key (AES/Triple-DES key)	R W
Release all resources of symmetric crypto function context *Input:* context Output: N/A	X	X	AES/Triple-DES key	Z
Release all resources of hash context *Input:* context Output: N/A	X	X	HMAC key	Z

Service	Roles		CSPs & crypto keys	Access Type
	USER	CO		
Release of all resources of Diffie-Hellman context for Diffie-Hellman and EC Diffie-Hellman *Input:* context Output: N/A	X	X	Asymmetric keys (RSA/ECDSA key) and secret session key (AES/Triple-DES key)	Z
Release of all resources of asymmetric crypto function context *Input:* context Output: N/A	X	X	RSA/ECDSA keys	Z
Reboot	X	X	N/A	N/A
Self-test	X	X	Software integrity key	R
Show Status	X	X	None	N/A

Table 6: Approved and Allowed Services in Approved Mode

Service	Roles		Access Type
	USER	CO	
Integrated Encryption Scheme on elliptic curves encryption and decryption	X	X	R
DES Encryption and Decryption	X	X	R
Triple-DES (Optimized assembler Implementation) Encryption and Decryption Mode: CTR	X	X	R
Triple-DES (Two-key implementation) Encryption and Decryption	X	X	R
CAST5 Encryption and Decryption	X	X	R
Blowfish Encryption and Decryption	X	X	R
RC4 Encryption and Decryption	X	X	R W
RC2 Encryption and Decryption	X	X	R W
MD2 Hash	X	X	R W
MD4 Hash	X	X	R W

Service	Roles		Access Type
	USER	CO	
MD5 Hash	X	X	R W
RIPEMD Hash	X	X	R W
RSA Key Wrapping with RSA-PKCS1-v1_5 PKCS#1 v2.1	X	X	R
RSA ANSI X9.31 Signature Generation and Verification	X	X	R W
RSA PKCS1-v1_5 Signature Generation and Verification Key sizes: 1024-4096 bits in multiple of 32 bits not listed in table 3	X	X	R W
RSA ANSI X9.31 Key Pair Generation Key sizes (modulus): 1024-4096 bits in multiple of 32 bits not listed in table 3 Public key exponent values: 65537 or larger	X	X	R W
ECDSA Key Pair Generation for compact point representation of points	X	X	R W
ECDSA PKG: curves P-192, 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
Diffie-Hellman Key Agreement Key Sizes < 2048 bits	X	X	R W
Ed 25519 Key agreement, Signature Generation, Signature Verification	X	X	R W
SP800-56C Key Derivation Function	X	X	R W
ANSI X9.63 Key Derivation Function	X	X	R W
RFC6637 Key Derivation Function	X	X	R W
AES-CMAC AES-128 MAC Generation	X	X	R W
OMAC MAC Generation	X	X	R W

Service	Roles		Access Type
	USER	CO	
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 iOS CoreCrypto Module, v7.0 is intended to operate on a multi-chip standalone platform used as a mobile device. The mobile 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 iOS CoreCrypto Module, v7.0.

6.1 Applicability

The Apple iOS CoreCrypto Module, v7.0 operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. It is part of iOS 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 operator (single-user mode; i.e. concurrent operators are explicitly excluded).

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

7 Cryptographic Key Management

The following section defines the key management features available through the Apple iOS CoreCrypto Module, v7.0.

7.1 Random Number Generation

A FIPS 140-2 approved deterministic random bit generator based on a block cipher as specified in NIST SP 800-90A is used. The default Approved DRBG used for random number generation is a CTR_DRBG using AES-128 with derivation function and without prediction resistance. The module also employs a HMAC-DRBG for random number generation. The deterministic random bit generators are seeded by /dev/random. The /dev/random generator is a true random number generator that obtains entropy from interrupts generated by the devices and sensors attached to the system and maintains an entropy pool. The TRNG feeds entropy from the pool into the DRBG on demand. The TRNG provides 160-bits of entropy.

7.2 Key / CSP Generation

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

- The default Approved DRBG specified in section 7.1 is used to generate secret asymmetric keys for the ECDSA and RSA algorithm.

It is not possible for the module to output information during the key generating 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 AES Key wrapping, RSA key wrapping, Diffie-Hellman- and EC Diffie-Hellman-based key establishment services.

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 salt as well as the number of iterations. The implementation of the PBKDFv2 function requires the user to provide this information.

7.4 Key / CSP Entry and Output

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

7.5 Key / CSP Storage

The Apple iOS CoreCrypto Module, v7.0 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 iOS. No process can read the memory of another process.

7.6 Key / CSP Zeroization

Keys and CSPs are zerorized 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.

8 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.

9 Self-Tests

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

The execution of an independent application for invoking the self-tests in the libcorecrypto.dylib makes use of features of the iOS architecture: the module, implemented in libcorecrypto.dylib, is linked by libcommoncrypto.dylib which is linked by libSystem.dylib. The libSystem.dylib is a library that must be loaded into every application for operation. The library is stored in the kernel cache and therefore is not available on the disk as directly visible files. iOS 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 iOS.

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 iOS CoreCrypto Module, v7.0 starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the following tests fails the device powers itself off. To rerun the self-tests on demand, the user must reboot the device.

9.1.1 Cryptographic Algorithm Tests

Algorithm	Modes	Test
Triple-DES	CBC	KAT (Known Answer Test) Separate encryption / decryption operations are performed
AES implementations selected by the module for the corresponding environment AES-128	ECB, CBC, GCM, XTS	KAT Separate encryption / decryption operations are performed
DRBG (CTR_DRBG and HMAC_DRBG; tested separately)	N/A	KAT
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT
RSA	SIG(ver), SIG(gen)	pair-wise consistency checks
	Encrypt / Decrypt	KAT. Separate encryption /decryption operations are performed
ECDSA	SIG(ver), SIG(gen)	pair-wise consistency checks
Diffie-Hellman “Z” computation	N/A	KAT
EC Diffie-Hellman “Z” computation	N/A	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 iOS CoreCrypto Module, v7.0. 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.

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 iOS CoreCrypto Module, v7.0.

9.2.1 Continuous Random Number Generator Test

The Apple iOS CoreCrypto Module, v7.0 performs a continuous random number generator test, whenever the DRBG is invoked.

9.2.2 Pair-wise Consistency Test

The Apple iOS CoreCrypto Module, v7.0 does generate asymmetric keys and performs all required pair-wise consistency tests, the encryption/decryption as well as signature verification tests, with the newly generated key pairs.

9.2.3 SP 800-90A Assurance Tests

The Apple iOS CoreCrypto Module, v7.0 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.

9.2.4 Critical Function Test

No other critical function test is performed conditionally.

10 Design Assurance

10.1 Configuration Management

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

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

The following naming/numbering convention for documentation is applied.

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

Example: FIPS_CORECRYPTO_IOS_US_SECPOL_3.0

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

10.2 Delivery and Operation

The CoreCrypto is built into iOS. 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 has passed all self-tests and is operating in the Approved mode.

10.4.2 User Guidance

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

11 Mitigation of Other Attacks

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

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