Apple Inc.

Apple CoreCrypto Cryptographic Module v9.0 for ARM
FIPS 140-2 Non-Proprietary Security Policy

March, 2019

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

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

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
2.2.1 Approved or Allowed Security Functions 10
2.2.2 Non-Approved Security Functions: 14
2.3 CRYPTOGRAPHIC MODULE BOUNDARY 16
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 Cryptographic Algorithm Tests 29
9.1.2 Software / Firmware Integrity Tests 29
9.1.3 Critical Function Tests 30
9.2 CONDITIONAL TESTS 30
9.2.1 Continuous Random Number Generator Test 30
9.2.2 Pair-wise Consistency Test 30
9.2.3 SP 800-90A Assurance Tests 30
9.2.4 Critical Function Test 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 Cryptographic Officer Guidance 31
10.4.2 User Guidance 31

11 MITIGATION OF OTHER ATTACKS 32

List of Tables

Table 1: Module Validation Level 8
Table 2: Tested Platforms 9
Table 3: Approved, Allowed and Vendor Affirmed Security Functions 13
Table 4: Non-Approved or Non-Compliant Security Functions........ 15
Table 5: Roles 18
Table 6: Approved and Allowed Services in Approved Mode 22
Table 7: Non-Approved Services in Non-Approved Mode........ 23
Table 8: Cryptographic Algorithm Tests........ 29

List of Figures

Figure 1: Logical Block Diagram........ 16

1 Introduction

1.1 Purpose

This document is a non-proprietary Security Policy for the Apple CoreCrypto Cryptographic Module v9.0 for ARM. The module's version is v9. It describes the module and the FIPS 140-2 cryptographic services it provides. This document also defines the FIPS 140-2 security rules for operating the module.
This document was prepared in fulfillment of the FIPS 140-2 requirements for cryptographic modules and is intended for security officers, developers, system administrators, and end-users.
FIPS 140-2 details the requirements of the Governments of the U.S. and Canada for cryptographic modules, aimed at the objective of protecting sensitive but unclassified information.
For more information on the FIPS 140-2 standard and validation program please refer to the NIST website at https://csrc.nist.gov/projects/cryptographic-module-validation-program.
Throughout the document "Apple CoreCrypto Cryptographic Module v9.0 for ARM." "cryptographic module", "CoreCrypto" or "the module" are used interchangeably to refer to the Apple CoreCrypto Cryptographic Module v9.0 for ARM. Throughout the document "OS" refers to "iOS", "tvOS", "watchOS" and "TxFW" unless specifically noted.

1.2 Document Organization / Copyright

This non-proprietary Security Policy document may be reproduced and distributed only in its original entirety without any revision, ©2019 Apple Inc.

1.3 External Resources / References

The Apple website (http://www.apple.com) contains information on the full line of products from Apple Inc. For a detailed overview of the operating system iOS and its security properties refer to [iOS] and [SEC]. For details on the OS 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 (https://csrc.nist.gov/projects/cryptographic-module-validation-program) contains links to the FIPS 140-2 certificate and Apple Inc. contact information.

1.3.1 Additional References

FIPS 140-2	Federal Information Processing Standards Publication, "FIPS PUB 140-2 Security Requirements for Cryptographic Modules," Issued May-25-2001, Effective 15-Nov-2001, Location: https://csrc.nist.gov/projects/cryptographic-module-validation-program/standards
FIPS 140-2 IG	NIST, "Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program," November, 2018 Location: https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Module-Validation-Program/documents/fips140-2/FIPS1402IG.pdf
FIPS 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
API	Application Programming Interface
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
KS	Key Size (Length)
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OFB	Output Feedback (mode of operation)
OS	Operating System
PBKDF	Password-based Key Derivation Function
PCT	Pair-wise Consistency Test
RNG	Random Number Generator
SHS	Secure Hash Standard
Triple-DES	Triple Data Encryption Standard
TLS	Transport Layer Security

2 Cryptographic Module Specification

2.1 Module Description

The Apple CoreCrypto Cryptographic Module v9.0 for ARM is a software cryptographic module 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

2.1.1 Module Validation Level

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

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

Table 1: Module Validation Level

2.1.2 Module Components

In the following sections the components of the Apple CoreCrypto Cryptographic Module v9.0 for ARM 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 with and without PAA[1] on the following platforms :

Manufacturer	Model	Operating System
Apple Inc.	iPhone 5S with Apple A7 CPU	iOS 12
Apple Inc.	iPhone 6 with Apple A8 CPU (iPhone 6 and iPhone 6 Plus)	iOS 12
Apple Inc.	iPhone 6S with Apple A9 CPU (iPhone 6S and iPhone 6S Plus)	iOS 12
Apple Inc.	iPhone 7 with Apple A10[2] Fusion CPU (iPhone 7 and iPhone 7 Plus)	iOS 12
Apple Inc.	iPhone 8 and iPhone X with Apple A11[3] Bionic CPU (iPhone 8, iPhone 8 Plus, iPhone X)	iOS 12
Apple Inc.	iPhone XS (XR / iPhone XS / iPhone XS Max) with Apple A12 Bionic[3] CPU	iOS 12
Apple Inc.	iPad Air 2 with Apple A8X CPU	iOS 12
Apple Inc.	iPad Pro with Apple A9X CPU	iOS 12
Apple Inc.	iPad Pro with Apple A10X[2] Fusion CPU	iOS 12
Apple Inc.	iPad Pro with Apple A12X[3] Bionic CPU	iOS 12
Apple Inc.	Apple TV 4K with Apple A10X[2] Fusion CPU	tvOS 12
Apple Inc.	Apple Watch Series 1 with Apple S1P CPU	watchOS 5
Apple Inc.	Apple Watch Series 3 with Apple S3 CPU	watchOS 5
Apple Inc.	Apple Watch Series 4 with Apple S4 CPU	watchOS 5
Apple Inc.	iMac Pro with Apple T2	TxFW 16P374
Apple Inc.	MacBook Pro (13-inch and 15-inch) with Apple T2	TxFW 16P374

Table 2: Tested Platforms

2.2 Modes of Operation

The Apple CoreCrypto Cryptographic Module v9.0 for ARM 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.

[1] PAA provided here is the ARM NEON present in Apple A and S series processors.
[2] Apple A10 and A10X are also known as Apple A10 Fusion and Apple A10X Fusion.
[3] Apple A11, A12, A12X and are also known as Apple A11 Bionic and Apple A12 Bionic.

Refer to https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program for the current standards, test requirements, and special abbreviations used in the following table.

2.2.1 Approved or Allowed Security Functions

Cryptographic Function	Algorithm	Modes/Options	Algorithm Certificate Number
Random Number Generation; Symmetric Key Generation	[SP 800-90] DRBG	Generic Software (C) Implementation Modes: CTR_DRBG AES-128, AES-256 Derivation Function Enabled HMAC_DRBG HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 Without Prediction Resistance	2449, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
		Generic Software (C) Implementation using Assembler Implementation of ECB Modes: CTR_DRBG AES-128, AES-256 Derivation Function Enabled	2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2429, 2432, 2443, C106, C145, C240, C241
		VNG Implementation using (C) Implementation of ECB: Modes: HMAC_DRBG HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 Without Prediction Resistance	2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2431, 2434, 2445, C13, C16, C29, C30
		VNG Implementation using Assembler Implementation of ECB Modes: CTR_DRBG AES-128, AES-256 Derivation Function Enabled	2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2430, 2433, 2444, C105, C146, C246, C247
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 (C) Implementation (Based on LibTomCrypt): Modes: ECB CFB128 OFB CBC CTR XTS[4] CCM GCM CFB8 KW	5886, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245

[4] XTS approved with 128-bit and 256-bit key size only;

Cryptographic Function	Algorithm	Modes/Options	Algorithm Certificate Number
		Generic Software (C) Implementation using Assembler Implementation of ECB: Modes: ECB CFB128 OFB CBC CTR XTS[4] CCM GCM CFB8 KW	5701, 5702, 5703, 5704, 5705, 5706, 5707, 5716, 5836, 5841, 5879, C106, C145, C240, C241
		Generic Software (C) Implementation (Based on Gladman): Modes: CBC	5708, 5709, 5710, 5711, 5712, 5713, 5714, 5715, 5838, 5842, C10, C11, C14, C25, C26
		VNG Implementation using (C) Implementation of ECB: Modes: ECB CTR CCM	5733, 5734, 5735, 5736, 5737, 5738, 5739, 5740, 5840, 5845, 5882, C13, C16, C29, C30
		VNG Implementation using Assembler Implementation of ECB: Modes: ECB GCM CTR CCM	5725, 5726, 5727, 5728, 5729, 5730, 5731, 5732, 5837, 5844, 5880, C105, C146, C246, C247
		Assembler Implementation with ARM PAA: Modes: ECB OFB CBC XTS[4] CFB128	5717, 5718, 5719, 5720, 5721, 5722, 5723, 5724, 5839, 5843, 5881, C12, C15, C27, C28
	[SP 800-67] Triple-DES	Triple-DES (Keying Option: 1; All Keys Independent) Modes: ECB CFB64 CBC CTR CFB8 OFB	2866, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245

Cryptographic Function	Algorithm	Modes/Options	Algorithm Certificate Number
Digital Signature and Asymmetric Key Generation	[FIPS186-4] RSA PKCS#1 v1.5	Key Generation (ANSI X9.31), Signature Generation (PKCS#1 v1.5) Key Sizes: 2048 3072 Signature Verification (PKCS#1 v1.5) Key Sizes: 1024 2048 3072	3084, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
	[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	1567, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
		ECDSA Signature Generation Component:	CVL: 2180, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
	[FIPS 186-4] DSA[5] used for Diffie-Hellman key generation only	Asymmetric Key Generation Key Sizes: L=2048, N=256	1481, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
Message Digest	[FIPS 180-4] SHS	Generic Software (C) Implementation: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4638, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
		VNG Implementation using (C) Implementation of ECB: SHA-1 SHA-384 SHA-224 SHA-512 SHA-256	4571, 4572, 4573, 4574, 4575, 4576, 4577, 4578, 4631, 4632, 4636, C13, C16, C29, C30
Keyed Hash	[FIPS 198] HMAC	Generic Software (C) Implementation: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3863, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245

[5] The DSA key pair generation is not available as an explicit service. It is used to create Diffie-Hellman Keys in the SP800-56A Key Establishment.

Cryptographic Function	Algorithm	Modes/Options	Algorithm Certificate Number
		VNG Implementation using (C) Implementation of ECB: HMAC-SHA-1 HMAC-SHA-384 HMAC-SHA-224 HMAC-SHA-512 HMAC-SHA-256	3798, 3799, 3800, 3801, 3802, 3803, 3804, 3805, 3856, 3857, 3861, C13, C16, C29, C30
Key Agreement and Establishment	[SP800-56A] DLC Primitive Diffie-Hellman	Public key size: 2048-bits or larger and Private key size: 256-bits	CVL: 2180, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
	[SP800-56A] DLC Primitive EC Diffie-Hellman	NIST Curves: P-256, P-384	CVL: 2115, C95, C96, C97, C98, C99, C100, C101, C102, C107, C148, C239, C242, C243, C245
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
	[FIPS 186-4]	PKCS#1 v1.5, PKCS#1 v2.1 Modulus size: 2048-bits or 3072-bits	Non-Approved, but Allowed[6]
Key Agreement	ANSI X9.63 SP 800-56A EC Diffie-Hellman	ECC curves P-256, P-384	Non-Approved, but Allowed[7]
	ANSI X9.42 SP 800-56A Diffie-Hellman	Key sizes: 2048-bits 3072-bits	Non-Approved, but allowed[8]
MD5	Message Digest	Digest Size: 128-bit	Non-Approved, but Allowed[9]
NDRNG	Random Number Generation	N/A	Non-Approved, but Allowed: provided by the underlying operational environment

Table 3: Approved, Allowed and Vendor Affirmed Security Functions

CAVEAT: The module generates cryptographic keys whose strengths are modified by available entropy – 128-bits. The encryption strength for the AES Key Wrapping using 192 and 256-bit keys is limited to 128 bits due to the entropy of the seed source.

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

[7] EC Diffie-Hellman is used for key establishment. Methodology provides 128 bits of encryption strength - the encryption strength is limited by the entropy of the seed source as specified in the caveat.

[8] Diffie-Hellman is used for key establishment. Methodology provides 112 or 128 bits of encryption strength.

[9] MD5 Used as part of the TLS key establishment scheme only.

The encryption strengths included in Table 3 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].

2.2.2 Non-Approved Security Functions:

Cryptographic Function	Usage / Description	Caveat
RSA Signature Generation / Signature Verification / Asymmetric Key Generation	ANSI X9.31 Signature Generation Key Pair Generation Key Size < 2048 Signature Verification PKCS#1 v1.5 Signature Generation Key Size < 2048 Signature Verification Key Size < 1024	Non-Approved
RSA Key Wrapping	PKCS#1 v1.5 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	Non-Approved
Integrated Encryption Scheme on elliptic curves	Encryption / Decryption	Non-Approved
Diffie-Hellman Key Agreement	Key agreement scheme using key sizes < 2048-bits	Non-Approved
Ed25519	Key Agreement Signature Generation Signature Verification	Non-Approved
ANSI X9.63 KDF	ANSI X9.63 Hash based KDF	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

Cryptographic Function	Usage / Description	Caveat
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: 128, 160, 256, 320 -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	Key Based Key Derivation Function Mode(s): CTR, Feedback	Non-Compliant
Triple-DES	Encryption / Decryption Two Key Implementation	Non-Compliant
	Optimized Assembler Implementation Encryption / Decryption Mode: CTR	
AES-CMAC	AES-128 MAC generation	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 not validated by the CAVP.

2.3 Cryptographic Module Boundary

The physical boundary of the module is the physical boundary of the iOS, tvOS, watchOS or TxFW device (iPhone, iPad, Apple TV, Apple Watch or iMac Pro) 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.

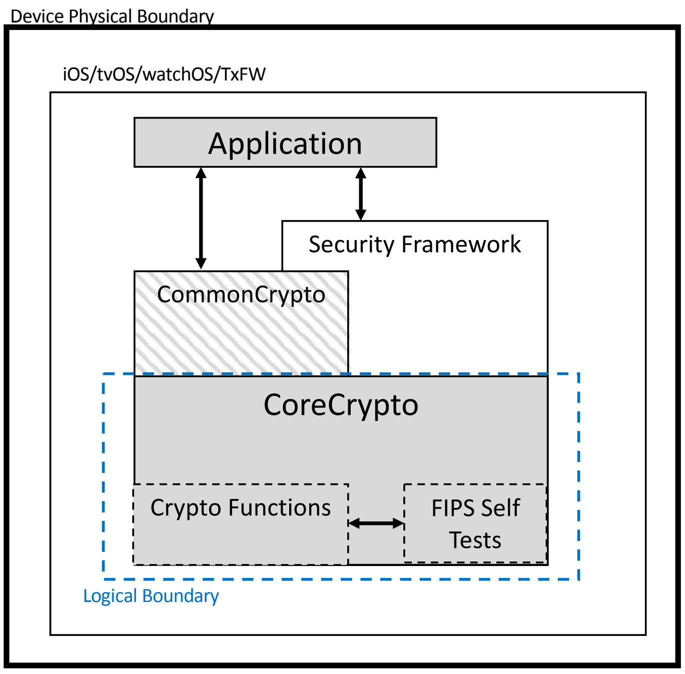

Figure 1: Logical Block Diagram

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.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.

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 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: (i) 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 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[10]': 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	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 W

[10] The R access type refers to Reading of the CSP. This access type can be thought as synonymous with Execute CSP/key

Service	**Roles**		**CSPs & 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 W
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	RSA private key	R W
RSA Key Wrapping Using PKCS#1 v1.5, PKCS#1 v2.1 (non-approved but allowed) 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	RSA private key	R W
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

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	RSA private 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	ECDSA private key	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	ECDSA private 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	DRBG Entropy string, Nonce, V and Key	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	PBKDF 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	RSA private key	R W
Diffie-Hellman key pair generation *Input:* p, q, g *Output:* private key, public key	X	X	Diffie-Hellman 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 private 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 private key) and secret session key (AES/Triple-DES key)	R 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 Diffie-Hellman context for Diffie-Hellman and EC Diffie-Hellman (Symmetric, Asymmetric and MAC Key Zeroization) *Input:* context Output: N/A	X	X	Asymmetric keys (RSA/ECDSA private key) and secret session key (AES/Triple-DES key)	Z

Service	**Roles**		**CSPs & crypto keys**	**Access Type**
	USER	CO		
Release of all resources of asymmetric crypto function context (Asymmetric Key Zeroization) *Input:* context Output: N/A	X	X	RSA/ECDSA private keys	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	None	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**		**Access Type**
		USER	CO	
Integrated Encryption Scheme on elliptic curves encryption		X	X	R W
Integrated Encryption Scheme on elliptic curves decryption				
DES Encryption		X	X	R W
DES Decryption				
Triple-DES Encryption	CTR mode (non-compliant)	X	X	R W
	Two-Key Triple-DES (non-approved)			
Triple-DES Decryption	CTR mode (non-compliant)	X	X	R
	Two-Key Triple-DES (non-approved)			
CAST5 Encryption		X	X	R W
CAST5 Decryption				
Blowfish Encryption		X	X	R W
Blowfish Decryption				
RC2 Encryption		X	X	R W
RC2 Decryption				
RC4 Encryption		X	X	RW
RC4 Decryption		X	X	RW
MD2 Message Digest Generation		X	X	R W
MD4 Message Digest Generation		X	X	R W

Service	**Roles**		**Access Type**
	USER	CO	
RIPEMD Message Digest Generation	X	X	R W
RSA ANSI X9.31 Signature Generation	X	X	R W
RSA ANSI X9.31 Signature Verification			
RSA PKCS#1 v1.5 Signature Generation Key sizes: 1024-4096 bits in multiple of 32 bits not listed in table 3	X	X	R W
RSA PKCS#1 v1.5 Signature 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
RSA PKCS#1 v1.5 Key Wrapping Key sizes < 2048	X	X	R W
ECDSA Key Pair Generation for compact point representation of points	X	X	R W
Diffie-Hellman Key Agreement Key Sizes < 2048 bits	X	X	R W
ECDSA Key Generation: curve P-192	X	X	R W
ECDSA Public Key Verification: curve P-192			
ECDSA Signature Generation: curve P-192			
ECDSA Signature Verification: curve P-192			
Ed25519 Key agreement	X	X	R W
Ed25519 Signature Generation			
Ed25519 Signature Verification			
SP800-56C Key Derivation	X	X	R W
ANSI X9.63 Key Derivation	X	X	R W
RFC6637 Key Derivation	X	X	R W
SP800-108 Key Derivation	X	X	R W
AES-CMAC MAC Generation	X	X	R W
OMAC MAC Generation	X	X	R W

Table 7: Non-Approved Services in Non-Approved Mode

4.3 Operator authentication

Within the constraints of FIPS 140-2 level 1, the module does not implement an authentication mechanism for operator authentication. The assumption of a role is implicit in the action taken. The module relies upon the operating system for any operator authentication.

5 Physical Security

The FIPS 140-2 physical security requirements do not apply to the Apple CoreCrypto Cryptographic Module v9.0 for ARM since it is a software module.

6 Operational Environment

The following sections describe the ioperational environment of the Apple CoreCrypto Cryptographic Module v9.0 for ARM.

6.1 Applicability

The Apple CoreCrypto Cryptographic Module v9.0 for ARM 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 2.1.3.

6.2 Policy

The operating system is restricted to a single operator (single-user mode; i.e. concurrent operators are explicitly excluded).
When the operating system loads the module into memory, it invokes the FIPS Self-Test functionality, which in turn runs the mandatory FIPS 140-2 tests.

7 Cryptographic Key Management

The following section defines the key management features available through the Apple CoreCrypto Cryptographic Module v9.0 for ARM.

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-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 /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 NDRNG feeds entropy from the pool into the DRBG on demand. The NDRNG provides 128-bits of entropy.

7.2 Key / CSP Generation

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

- The module does not implement symmetric key generation. In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per SP800-133 (vendor affirmed), compliant with [FIPS186-4], and using DRBG compliant with [SP800-90A]. A seed (i.e. the random value) used in asymmetric key generation is obtained from [SP800-90A] DRBG. The generated seed is an unmodified output from the DRBG. The key generation service for RSA, ECDSA and Diffie-Hellman as well as the SP 800-90A DRBG have been CAVS tested with algorithm certificates found in Table 3.

It is not possible for the module to output information during the key generating process. The cryptographic strength of keys generated by the module are modified by the available entropy, which is limited to 128-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 also provides key establishment services in the Approved mode through the SP 800-132 PBKDFv2 algorithm. The PBKDFv2 function returns the key derived from the provided password to the caller. The keys derived from SP 800-132 map to section 4.1 of SP 800-133 as indirect generation from DRBG. The caller shall observe all requirements and should consider all recommendations specified in SP800-132 with respect to the strength of the generated key, including the quality of the salt as well as the number of iterations. The implementation of the PBKDFv2 function requires the user to provide this information.

7.4 Key / CSP Entry and Output

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

7.5 Key / CSP Storage

The Apple CoreCrypto Cryptographic Module v9.0 for ARM 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.

7.6 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.

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 OS 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 OS 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. iThe 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.

9.1 Power-Up Tests

The following tests are performed each time the Apple CoreCrypto Cryptographic Module v9.0 for ARM 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	Signature Generation, Signature Verification	PCT
	Encryption / Decryption	KAT. Separate encryption /decryption operations are performed
ECDSA	Signature Generation, Signature Verification	PCT
Diffie-Hellman “Z” computation	N/A	KAT
EC Diffie-Hellman “Z” computation	N/A	KAT

Table 8: Cryptographic Algorithm Tests

9.1.2 Software / Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple CoreCrypto Cryptographic Module v9.0 for ARM. 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 CoreCrypto Cryptographic Module v9.0 for ARM.

9.2.1 Continuous Random Number Generator Test

The Apple CoreCrypto Cryptographic Module v9.0 for ARM performs a continuous random number generator test on the noise source (i.e. NDRNG), whenever it is invoked to seed the SP800-90A DRBG.

9.2.2 Pair-wise Consistency Test

The Apple CoreCrypto Cryptographic Module v9.0 for ARM 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 Health Tests

The Apple CoreCrypto Cryptographic Module v9.0 for ARM performs the health tests as specified in section 11.3 of SP 800-90A.

9.2.4 Critical Function Test

No other critical function test is performed conditionally.

10 Design Assurance

10.1 Configuration Management

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

10.2 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.

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}.