Cellcrypt Core V4 FIPS 140-2 Module

Software Version: CCoreV4

FIPS 140-2 Non-Proprietary Security Policy

Date: 20 January 2022

Copyright Notice

Acknowledgements

Cellcrypt acknowledges that this document was derived from the Oracle FIPS 140-2 Non-Proprietary Security Policy document from the CMVP FIPS validation certificate #3335.

TABLE OF CONTENTS

LIST OF TABLES

1 Introduction

This document is the non-proprietary security policy for the *Cellcrypt Core V4 FIPS 140-2 Module*, hereafter referred to as the Module.

The Module is a software library providing a C language application program interface (API) for use by other processes that require cryptographic functionality. The Module is classified by FIPS 140-2 as a software module, multi-chip standalone module embodiment. The physical cryptographic boundary is the general-purpose computer on which the module is installed. The logical cryptographic boundary of the Module is the fipscanister object module, a single object module file named fipscanister.o. The Module performs no communications other than with the calling application (the process that invokes the Module services).

The FIPS 140-2 security levels for the Module are as follows:

Table 1 Security Level of Security Requirements

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	2
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

The Module's software version for this validation is CCoreV4.

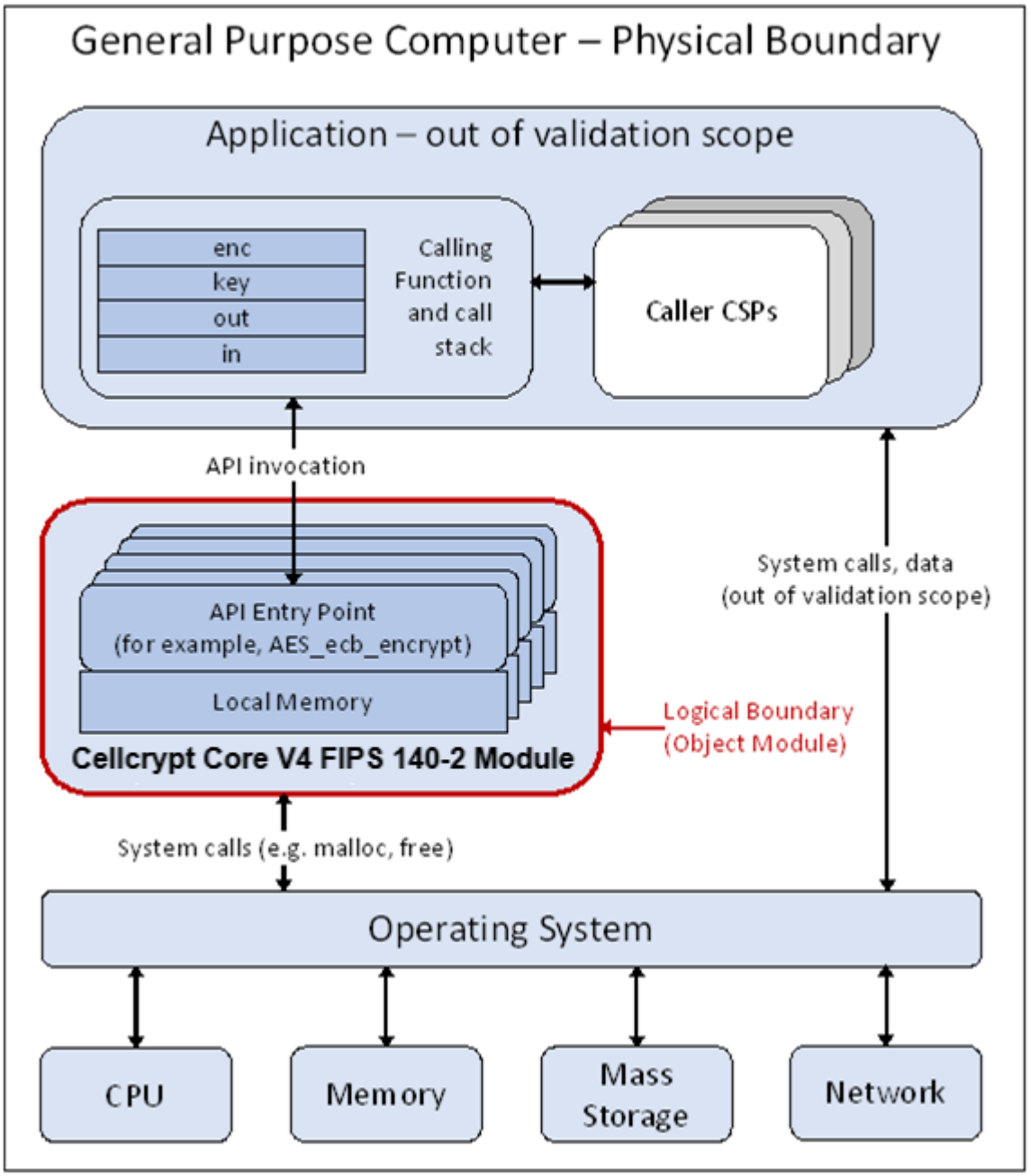

Figure 1 Module Block Diagram

2 Ports and Interfaces

The physical ports of the Module are the same as the computer system on which it is executing. The logical interface is a C language API.

Table 2 Logical Interfaces

Logical interface type	Description
Control input	API entry point and corresponding stack parameters
Data input	API entry point data input stack parameters
Status output	API entry point return values and status stack parameters
Data output	API entry point data output stack parameters

As a software module, control of the physical ports is outside module scope. However, when the module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The module is single-threaded and in error scenarios returns only an error value (no data output is returned).

3 Modes of Operation

The Module supports FIPS 140-2 Approved, Allowed and Non-Approved algorithms in a single mixed mode of operation.

3.1 Approved Mode

The Module supports the following services and algorithms in FIPS Approved Mode:

Table 3 FIPS Approved Cryptographic Functions

Function	Algorithm	Options	Cert #
Random Number Generation; symmetric key generation	[SP 800-90Ar1] DRBG[1] Prediction resistance supported for all variations	**Hash_Based DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled (SHA-1 , SHA-224 , SHA-256 , SHA-384 , SHA-512)] **HMAC_Based DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled (SHA-1 , SHA-224 , SHA-256 , SHA-384 , SHA-512)] **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; BlockCipher_Use_df: (AES-128 , AES-192 , AES-256)] BlockCipher_No_df: (AES-128 , AES-192 , AES-256)]	2129 C1651 A1999
Cryptographic Key Generation (CKG)	[SP 800-133r2] CKG		vendor affirmed
Encryption, Decryption, and CMAC	[SP 800-67r2] [SP 800-38A]	3-Key TDES TECB, TCBC, TCFB, TOFB; CMAC generate and verify	2735 C1651 A1999
	[FIPS 197] AES [SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS	128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CCM; GCM; CMAC generate and verify	5445 C1651 A1999
Message Digests	[FIPS 180-4]	SHA-1, SHA-2 (224, 256, 384, 512)	4364 C1651 A1999
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	3603 C1651 A1999
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	SigGen9.31, SigGenPKCS1.5, SigGenPSS (4096 with all SHA-2 sizes)[2] SigVer9.31, SigVerPKCS1.5, SigVerPSS (1024/1536/2048/3072/4096 with all SHA sizes)	2921 C1651 A1999
	[FIPS 186-4] RSA	Key Gen, SigGen9.31, SigGenPKCS1.5, SigGenPSS, (2048/3072/4096[3]) with all SHA2 sizes)	2921 C1651 A1999
	[FIPS 186-4] DSA	Key Pair Gen (2048/3072) PQG Gen, Sig Gen (2048/3072 with all SHA-2 sizes) PQG Ver, Sig Ver (1024/2048/3072 with all SHA sizes)	1400 C1651 A1999

[1] For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP800-90Ar1] and [SP800-57r5].
[2] CAVP Cert. #'s 2921 and C1651 only
[3] CAVP Cert. # A1999 only

Function	Algorithm	Options	Cert #
	[FIPS 186-4] ECDSA	Key Pair Gen: CURVES P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571 (ExtraRandomBits TestingCandidates) PKV: CURVES (ALL-P ALL-K ALL-B) SigGen: CURVES P-224: (SHA-224, 256, 384, 512) P-256: (SHA-224, 256, 384, 512) P-384: (SHA-224, 256, 384, 512) P-521: (SHA-224,256, 384, 512) K-233: (SHA-224, 256, 384, 512) K-283: (SHA-224, 256, 384, 512) K-409: (SHA-224, 256, 384, 512) K-571: (SHA-224, 256, 384, 512) B-233: (SHA-224, 256, 384, 512) B-283: (SHA-224, 256, 384, 512) B-409: (SHA-224, 256, 384, 512) B-571: (SHA-224, 256, 384, 512)) SigVer: CURVES P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P-256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256, 384, 512) K-163: (SHA-1, 224, 256, 384, 512) K-233: (SHA-1, 224, 256, 384, 512) K-283: (SHA-1, 224, 256, 384, 512) K-409: (SHA-1, 224, 256, 384, 512) K-571: (SHA-1, 224, 256, 384, 512) B-163: (SHA-1, 224, 256, 384, 512) B-233: (SHA-1, 224, 256, 384, 512) B-283: (SHA-1, 224, 256, 384, 512) B-409: (SHA-1, 224, 256,384, 512) B-571: (SHA-1, 224, 256, 384, 512)	1449 C1651 A1999
KAS-SSC [X1][4]	[SP 800-56Ar3]	Diffie-Hellman ≥ 2048 bits, ECDH B, K and P curves ≥ 256 bit curves	vendor affirmed

[4] In the approved mode, KAS-SSC can only be used in conjunction with an Approved KDF from SP 800-56C or SP 800-135

3.2 Non-Approved but Allowed Services

The Module supports the following non-approved but allowed services.

Table 4 Non-FIPS Approved, but Allowed Cryptographic Functions

Category	Algorithm	Description
Key Encryption, Decryption	RSA	RSA may be used to perform key establishment with another module by securely exchanging symmetric encryption keys with another module.

The module supports the following non-FIPS 140-2 approved but allowed algorithms:

- RSA (key wrapping; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112 bits of encryption strength)

3.3 Non-Approved Services

The Module implements the following services which are Non-Approved per the SP 800131Ar1 transition:

Table 5 Non-FIPS Approved Cryptographic Functions

Function	Algorithm	Options
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS (1024/1536 with all SHA sizes, 2048/3072/4096 with SHA-1)
	[FIPS 186-2] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)
	[FIPS 186-4] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)
	[FIPS 186-2] ECDSA	PKG: CURVES (P-192 K-163 B-163) SIG(gen): CURVES(P-192 P-224 P-256 P-384 P-521 K-163 K--233 K-283 K-409 K-571 B-163 B-233 B-283 B-409 B-571)
	[FIPS 186-4] ECDSA	PKG: CURVES (P-192 K-163 B-163) SigGen: CURVES (P-192: (SHA-1, 224, 256, 384, 512) P224:(SHA-1) P-256:(SHA-1) P-384: (SHA-1) P-521:(SHA-1) K-163: (SHA-1, 224, 256, 384, 512) K-233:(SHA-1) K-283:(SHA-1) K-409:(SHA-1) K-571:(SHA-1) B-163: (SHA-1, 224, 256, 384, 512) B-233:(SHA-1) B-283: (SHA-1) B-409:(SHA-1) B-571:(SHA-1))
ECC CDH (KAS)	[SP 800-56Ar1] (§5.7.1.2)	B, K and P curves sizes 163 and 192

These algorithms shall not be used when operating in the FIPS Approved mode of operation. Use of the non-conformant algorithms listed in Table 5 will place the module in a non-approved mode of operation.

3.4 Critical Security Parameters and Public Keys

All CSPs used by the Module are described in this section. All access to these CSPs by Module services are described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

Table 6 Critical Security Parameters

CSP Name	Description
RSA SGK	RSA (2048 to 15360 bits) signature generation key
RSA KDK	RSA (2048 to 16384 bits) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (2048/3072) signature generation key
DH Private	Diffie-Hellman ≥ 2048 private key agreement key
ECDSA SGK	ECDSA (All NIST defined B, K, and P curves) signature generation key
EC DH Private	EC DH (All NIST defined B, K, and P curves) private key agreement key.
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM[5]	AES (128/192/256) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512) XTS encrypt / decrypt key
Triple-DES EDK	Triple-DES (3-Key) encrypt / decrypt key
Triple-DES CMAC	Triple-DES (3-Key) CMAC generate / verify key
HMAC Key	Keyed hash key (160/224/256/384/512)
Hash_DRBG CSPs	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
HMAC_DRBG CSPs	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
CTR_DRBG CSPs	V (128 bits) and Key (AES 128/192/256), entropy input (length dependent on security strength)
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

Authentication data is loaded into the module during the module build process, performed by an authorized operator (Crypto Officer), and otherwise cannot be accessed.

The module does not output intermediate key generation values.

Table 7 Public Keys

CSP Name	Description
RSA SVK	RSA (1024 to 16384 bits) signature verification public key
RSA KEK	RSA (2048 to 16384 bits) key encryption (public key transport) key
DSA SVK	[FIPS 186-4] DSA (2048/3072) signature verification key
ECDSA SVK	ECDSA (All NIST defined B, K and P curves) signature verification key
DH Public	Diffie-Hellman public key agreement key
EC DH Public	EC DH (All NIST defined B, K and P curves) public key agreement key

[5] The module's IV is generated internally by the module's Approved DRBG. The DRBG seed is generated inside the module's physical boundary. The IV is 96-bits in length per NIST SP 800-38D, Section 8.2.2 and FIPS 140-2 IG A.5 scenario 2. The selection of the IV construction method is the responsibility of the user of this cryptographic module. In approved mode, users of the module must not utilize GCM with an externally generated IV. The only approved use of GCM is with TLS and with a randomly generated IV.

For all CSPs and Public Keys:

Storage: RAM, associated to entities by memory location. The Module stores DRBG state values for the lifetime of the DRBG instance. The module uses CSPs passed in by the calling application on the stack. The Module does not store any CSP persistently (beyond the lifetime of an API call), with the exception of DRBG state values used for the Modules' default key generation service.

Generation: The Module implements SP 800-90A compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys as shown in Table 3. The calling application is responsible for storage of generated keys returned by the module. For operation in the Approved mode, Module users (the calling applications) shall use entropy sources that contain at least 112 bits of entropy. To ensure full DRBG strength, the entropy sources must meet or exceed the security strengths shown in the table below:

Table 8 DRBG Entropy Requirements

DRBG Type	Underlying Algorithm	Minimum Seed Entropy
Hash_DRBG or HMAC_DRBG	SHA-1	128
	SHA-224	192
	SHA-256	256
	SHA-384	256
	SHA-512	256
CTR_DRBG	AES-128	128
	AES-192	192
	AES-256	256

Entry: All CSPs enter the Module's logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.

Output: The Module does not output CSPs, other than as explicit results of key generation services. However, none cross the physical boundary.

Destruction: Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module.

Private and secret keys as well as seeds and entropy input are provided to the Module by the calling application and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute

concurrently. An authorized application as user (Crypto Officer and User) has access to all key data generated during the operation of the Module.

Use: In the case of AES-GCM, the IV generation method is user selectable and the value can be computed in more than one manner.

Following RFC 5288 for TLS, the module ensures that it's strictly increasing and thus cannot repeat. When the IV exhausts the maximum number of possible values for a given session key, the first party, client or server, to encounter this condition may either trigger a handshake to establish a new encryption key in accordance with RFC 5246, or fail. In either case, the module prevents and IV duplication and thus enforces the security property.

In the event Module power is lost and restored the calling application must ensure that any AES-GCM keys used for encryption or decryption are redistributed.

The calling application shall ensure that the same Triple-DES key is not used to encrypt more than 2^{16} 64-bit blocks of data.

4 Roles, Authentication and Services

The Module implements the required User and Crypto Officer roles and requires authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators. The User or Crypto Officer role is assumed by passing the appropriate password to the FIPS_module_mode_set() function. The password values may be specified at build time and must have a minimum length of 16 characters. Any attempt to authenticate with an invalid password will result in an immediate and permanent failure condition rendering the Module unable to enter the FIPS mode of operation, even with subsequent use of a correct password.

Authentication data is loaded into the Module during the Module build process, performed by the Crypto Officer, and otherwise cannot be accessed.

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/256^{16}$, or less than $1/10^{38}$. The Module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

Both roles have access to all of the services provided by the Module:

- User Role (User): Loading the Module and calling any of the API functions.
- Crypto Officer Role (CO): Installation of the Module on the host computer system and calling of any API functions.

All services implemented by the Module are listed below, along with a description of service CSP access. The access types are determined as follows:

- Generate (G): Generates the Critical Security Parameter (CSP_ using an approved Random Bit Generator
- Read (R): Export the CSP
- Write (W): Enter/establish and store a CSP
- Destroy (D): Overwrite the CSP
- Execute (E): Employ the CSP
- None: No access to CSP's

Table 9 Services and CSP Access

Service	Role	Description	Access type
Initialize	User, CO	Module initialization. Does not access CSPs. CO-AD-Digest, User-AD-Digest	E
Self-test	User, CO	Perform self-tests (FIPS_selftest).	None
Show status	User, CO	Functions that provide module status information: • Version (as unsigned long or const char *) • FIPS Mode (Boolean)	None
Zeroize	User, CO	Functions that destroy CSPs: fips_drbg_uninstantiate:	D

Service	Role	Description	Access type
		DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.	
Random number generation	User, CO	Used for random number and symmetric key generation. • Seed or reseed a DRBG instance • Determine security strength of a DRBG instance • Obtain random data Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.	E
Asymmetric key generation	User, CO	Used to generate DSA, ECDSA and RSA keys: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK	G
Symmetric Encrypt/Decrypt	User, CO	Used to encrypt or decrypt data. AES EDK, Triple-DES EDK, AES-GCM, AES XTS (passed in by the calling process).	E
Symmetric Digest	User, CO	Used to generate or verify data integrity with CMAC. AES CMAC, Triple-DES CMAC (passed in by the calling process).	E
Message Digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest.	None
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. HMAC Key (passed in by the calling process).	E
Key Transport[6]	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). RSA KDK, RSA KEK (passed in by the calling process).	E
Key Agreement	User, CO	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). Diffie-Hellman/EC Diffie-Hellman Private, Diffie-Hellman/EC Diffie-Hellman Public (passed in by the calling process).	E

[6] "Key transport" can refer to a) moving keys in and out of the module or b) the use of keys by an external application. The latter definition is the one that applies to the *Cellcrypt Core V4 FIPS Module*.

Service	Role	Description	Access type
Digital Signature	User, CO	Used to generate or verify RSA, DSA or ECDSA digital signatures. RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process).	E
Utility	User, CO	Miscellaneous helper functions.	None

5 Self-Tests

The Module performs the self-tests listed below on invocation of “initialize” or “self-test”.

Table 10 Power-On Self-Tests

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA-1
HMAC	KAT	One KAT per SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 Per IG 9.3, this testing covers SHA POST requirements.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128-bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192 key length
AES GCM	KAT	Separate encrypt and decrypt, 256 key length
XTS-AES	KAT	128, 256-bit key sizes to support either the 256-bit key size (for XTS-AES-128) or the 512-bit key size (for XTS-AES-256)
AES CMAC	KAT	Sign and verify CBC mode, 128, 192, 256 key lengths
Triple-DES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key
Triple-DES CMAC	KAT	CMAC generate and verify, CBC mode, 3-Key
RSA	KAT	Sign and verify using 2048 bit key, SHA-256, PKCS#1
DSA	PCT	Sign and verify using 2048 bit key, SHA-384
DRBG	KAT	CTR_DRBG: AES, 256 bit with and without derivation function HASH_DRBG: SHA256 HMAC_DRBG: SHA256
ECDSA	PCT	Key gen, sign, verify using P-224, K-233 and SHA-512.
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6

The Module is installed using one of the set of instructions in Appendix A, as appropriate for the target system. The HMAC-SHA-1 of the Module distribution file as tested by the CMT Laboratory and listed in Appendix A is verified during installation of the Module file as described in Appendix A.

Per IG 9.10, the Module implements a default entry point and automatically runs the FIPS self-tests upon start-up.

The module has a function called FIPS_module_mode_set() within the init code that is automatically set to enable "FIPS Mode" by default. When the *Cellcrypt Core V4 FIPS 140-2 Module* is initialized, it will always run its power-on self-tests meeting the IG 9.10 requirement.

The module also has a Boolean check value to verify whether the module has run its power-on self-tests upon subsequent instantiations. If the module is determined to have already run its power-on self-tests, future instantiations will only run the power-up integrity test and not the full set of POST's. If power is lost to the module, the Boolean check value "1" is zeroized and the module will run its power-up self-tests again to verify the correctness of the module operation. Upon successful completion of the POST's, the Boolean check value is restored. This is consistent with the requirement described in IG 9.11.

The Module also implements the following conditional tests:

Table 11 Conditional Tests

Algorithm	Test
DRBG	Tested as required by [SP800-90A] Section 11
DRBG	FIPS 140-2 continuous test for stuck fault
NDRNG	FIPS 140-2 continuous test for NDRNG
DSA	Pairwise consistency test on each generation of a key pair
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair

In the event of a DRBG self-test failure the calling application must un-instantiate and re-instantiate the DRBG per the requirements of [SP 800-90A]; this is not something the Module can do itself.

Pairwise consistency tests are performed for both possible modes of use, e.g. Sign/Verify and Encrypt/Decrypt.

6 Operational Environment

The tested operating systems segregate user processes into separate process spaces. Each process space is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the calling application, and implicitly satisfies the FIPS 140-2 requirement for a single user mode of operation.

6.1 Tested Configurations

The module was tested in the following configurations:

Table 12 Tested Configurations

Operating System	Hardware Platform and Processor	Optimizations (Target)
Android 11	Samsung Galaxy S20 with Qualcomm Snapdragon 865	NEON & None
Android 11	Samsung Galaxy Tab Active3 with Samsung Exynos 9810	NEON & None
iOS 15	Apple iPhone 12 with Apple A14 Bionic	NEON & None
Windows 10	Microsoft Surface Pro 7 with Intel Core i5-1035G4	AES-NI & None
RHEL 7.6	HPE ProLiant DL360 Gen9 Server with Intel Xeon E5-2680 V4	AES-NI & None
Raspbian 11	Raspberry Pi 4 with Broadcom BCM2711 Cortex-A72	AES & None
Oracle Linux 7.6 64-bit	Oracle X7-2 Server with AMD® EPYC® 7551	AES-NI & None
Oracle Linux 7.6 64-bit	Oracle X7-2 Server with Intel® Xeon® Silver 4114	AES-NI & None
Oracle ILOM OS v4.0	AST2400 Server Management Processor with Oracle ILOM SP v4 (ARM v9)	None
Solaris 11.4	Oracle X5-2 with Intel Xeon E5-2600 v3	AES-NI & None
Oracle ZFS Storage OS 8.8	Oracle ZFS Storage ZS5-2 with Intel Xeon E5	AES-NI & None
Oracle ZFS	Oracle ZFS Storage ZS5-4 with Intel Xeon E7	AES-NI & None

Storage OS 8.8		
Solaris 11.4	Oracle S7-2L with Oracle SPARC S7	SPARC & None
Oracle ILOM OS v3.0	Oracle X5-2 server with Oracle ILOM SP v3 (ARM v7)	NEON & None
Oracle ILOM OS v3.0	Emulex Pilot-4 Orion mainboard with Oracle ILOM SP v2 (ARM v5)	None

See Appendix A for additional information on build method and optimizations. See Appendix C for a list of the specific compilers used to generate the Module for the respective operational environments.

7 Mitigation of other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.

Appendix A: Installation and Usage Guidance

The test platforms represent different combinations of installation instructions. For each platform that was tested, there is a build system, the host providing the build environment in which the installation instructions are executed, and a target system on which the generated object code is executed. The build and target systems may be the same type of system or even the same device or may be different systems – the Module supports cross--compilation environments.

The command set is relative to the top of the directory containing the uncompressed and expanded contents of the distribution files *OpenSSL_2.0.13_OracleFIPS_1.0*

Installation and Configuration Instructions for Solaris 11.4, Oracle ZFS Storage OS 8.8, Oracle Linux and ARM

As FIPS mode is enabled by default, the administrator can verify FIPS mode is set by calling the FIPS_module_mode(). The module can be downloaded from the Solaris Git Repository. The link to the Module code is here:
https://github.com/oracle/solaris-openssl-fips/releases/download/v1.0/OpenSSL_2.0.13_OracleFIPS_1.0.tar.gz

If one wishes to download and build the Module to the exact instructions for which the module was validated, they can follow the following steps:

1. Download the Module from the link above.
2. Verify the HMAC-SHA-1 digest of the distribution file; see Appendix B. An independently acquired FIPS 140-2 validated implementation of HMAC-SHA-1 must be used for this digest verification.

   ** Note that this verification can be performed on any convenient system and not necessarily on the specific build or target system.
3. Unpack the distribution

   $ tar -zxf OpenSSL_2.0.13_OracleFIPS_1.0.tar.gz
4. Run the command set

SPARC

$ ./Configure fipscanisterbuild solaris64-sparcv9-cc
$ make
$ make install

X86

$ ./Configure fipscanisterbuild solaris64-x86_64-cc
$ make
$ make install

ARM64 arch:

$ ./Configure linux-aarch64

$ make
$ make install

5. The resulting fipscanister.o file is now available for linking into the latest OpenSSL 1.0.2 distribution.

Installation and Configuration Instructions for Oracle ILOM OS v3.0 and Oracle ILOM OS v4.0

The administrator sets FIPS mode by setting "state=enabled" under /SP/services/fips and then rebooting. The Oracle ILOM OS 3.0 was tested using the following process:

1. Download the Module.
2. Verify the HMAC-SHA-1 digest of the distribution file; see Appendix B. An independently acquired FIPS 140-2 validated implementation of HMAC-SHA-1 must be used for this digest verification. Note that this verification can be performed on any convenient system and not necessarily on the specific build or target system.
3. Unpack the distribution

   ```
   $ tar -zxf  OpenSSL_2.0.13_OracleFIPS_1.0.tar.gz
   ```

4. Set up the following compiler environment to build the module:

   ```
   PATH=${PATH}:/opt/ilomtools/crosscompiler/gcc-4.9__150325/arm/bin
   export OPENSSL_ia32cap=~0x200000200000000
   export ARCH=arm
   export MACHINE=ARM3        # for "generic32"
   export HOSTCC=gcc
   export CROSS_COMPILE="arm-linux-gnueabi-"
   export CC=gcc
   export FIPS_CLOSED_SYSTEM=yes
   export INSTALL_PREFIX=$PWD/_install.
   ```

5. Run the command set

   ```
   $ cd OracleFIPS_1.0$
   export FIPS_SIG=$PWD/util/incore
   $ ./config
   $ make
   $ make install
   ```

6. The resulting fipscanister.o or fipscanister.lib file is now available for linking into the latest OpenSSL 1.0.2 distribution.

Note that failure to use one of the specified commands sets exactly as shown will result in a module that cannot be considered compliant with FIPS 140-2.

Installation and Configuration Instructions for RHEL 7.6, Raspbian 11, Windows 10, Android 11 and iOS 15

As FIPS mode is enabled by default, the administrator can verify FIPS mode is set by calling the FIPS_module_mode(). The module can be downloaded from the Solaris Git Repository. The link to the Module code is here:

https://github.com/oracle/solaris-openssl-fips/releases/download/v1.0/OpenSSL_2.0.13_OracleFIPS_1.0.tar.gz

If one wishes to download and build the Module to the exact instructions for which the module was validated, they can follow the following steps:

6. Download the Module from the link above.
7. Verify the HMAC-SHA-1 digest of the distribution file; see Appendix B. An independently acquired FIPS 140-2 validated implementation of HMAC-SHA-1 must be used for this digest verification.
   ** Note that this verification can be performed on any convenient system and not necessarily on the specific build or target system.
8. Unpack the distribution

   $ tar -zxf OpenSSL_2.0.13_OracleFIPS_1.0.tar.gz
9. Run the command set

X86 RHEL

```
$ ./config --prefix="$FIPSDIR"
$ make
$ make install
$ cp -p util/incore $FIPSDIR/bin
```

ARMv7 Raspbian

```
$ ./config --prefix="$FIPSDIR"
$ make
$ make install
$ cp -p util/incore $FIPSDIR/bin
```

X86 Windows

```
set PROCESSOR_ARCHITECTURE=x86
set INCLUDE=%CD%\windows_fips_headers
mkdir tmp32dll
set "VSCMD_START_DIR=%CD%"
call "%VS170COMNTOOLS%..\..\VC\Auxiliary\Build\vcvarsall.bat" amd64_x86
call ms\do_fips.bat %NO_ASM_FLAG% <nul
```

ARMv8 Android

```
$ perl ./Configure android64-aarch64 --prefix="$FIPSDIR"
```

```
$ make
$ make install
$ cp -p util/incore $FIPSDIR/bin
```

ARMv8 iOS

```
$ perl Configure "ios64-cross" --prefix=$FIPSDIR
$ make
$ make install
$ cp util/incore $FIPSDIR/bin
```

10. The resulting fipscanister.o file is now available for linking into the latest OpenSSL 1.0.2 distribution.

Linking the Runtime Executable Application

Note that applications interfacing with the FIPS Object Module are outside of the cryptographic boundary. When linking the application with the FIPS Object Module two steps are necessary:

1. The HMAC-SHA-1 digest of the FIPS Object Module file must be calculated and verified against the installed digest to ensure the integrity of the FIPS object module.
2. A HMAC-SHA-1 digest of the FIPS Object Module must be generated and embedded in the FIPS Object Module for use by the FIPS_module_mode_set() function at runtime initialization.

The fips_standalone_sha1 command can be used to perform the verification of the FIPS Object Module and to generate the new HMAC-SHA-1 digest for the runtime executable application. Failure to embed the digest in the executable object will prevent initialization of FIPS mode.

At runtime the FIPS_module_mode_set() function compares the embedded HMAC-SHA-1 digest with a digest generated from the FIPS Object Module object code. This digest is the final link in the chain of validation from the original source to the runtime executable application file.

Optimization

The “asm” designation means that assembler language optimizations were enabled when the binary code was built, “no-asm” means that only C language code was compiled.

For OpenSSL with x86 there are three possible optimization levels:

1. No optimization (plain C)
2. SPARC optimization (Solaris)
3. AESNI+PCLMULQDQ+SSSE3 optimization

For more information on enabling AES-NI on Intel processors, see:

- http://www.intel.com/support/processors/sb/CS¬030123.htm?wapkw=sse2
- https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni

For more information on enabling AES hardware acceleration on Exynos and Bionic Armv8 processors, see:

- https://gcc.gnu.org/onlinedocs/gcc/AArch64-Options.html

For more information on setting FIPS Mode on Solaris, see:

- https://docs.oracle.com/cd/E37838_01/html/E61028/index.html

For OpenSSL with ARM there are two possible optimization levels:

1. Without NEON
2. With NEON (ARM7 and ARMv8)

For more information, see http://www.arm.com/products/processors/technologies/neon.php

Appendix B: Control Distribution File Fingerprint

The *Cellcrypt Core V4 FIPS 140-2 Module* consists of the FIPS Object Module (the fipscanister.o contiguous unit of binary object code) generated from the specific source files.

The source files are in the specific Oracle OpenSSL distribution OpenSSL_2.0.13_OracleFIPS_1.0.tar.gz with HMAC-SHA-1 digest of:

ef8f7a91979cad14d033d8803a89fdf925102a30

located at:

https://github.com/oracle/solaris-openssl-fips/releases/download/v1.0/OpenSSL_2.0.13_OracleFIPS_1.0.tar.gz

The set of files specified in this tar file constitutes the complete set of source files of this module. There shall be no additions, deletions, or alterations of this set as used during module build. The *Cellcrypt Core V4 FIPS 140-2 Module* distribution tar file shall be verified using the above HMAC-SHA-1 digest.

The arbitrary 16-byte key of:

65 74 61 6f 6e 72 69 73 68 64 6c 63 75 70 66 6d

(equivalent to the ASCII string "etaonrishdlcupfm") is used to generate the HMAC-SHA-1 value for the FIPS Object Module integrity check.

Appendix C: Compilers

This appendix lists the specific compilers used to generate the Module for the respective Operational Environments. Note this list does not imply that use of the Module is restricted to only the listed compiler versions, only that the use of other versions has not been confirmed to produce a correct result.

Table 13 Compilers

#	Operational Environment	Compiler
1	Android 11	gcc Compiler Version 4.9
2	Android 11	gcc Compiler Version 4.9
3	iOS 15	Clang 13.0.0
4	Windows 10	C/C++ Optimizing Compiler Version 19.30
5	RHEL 7.6	gcc Compiler Version 4.8.5
6	Raspbian 11	gcc Compiler Version 10.2.1
7	Oracle Linux 7.6 64-bit	gcc Compiler Version 4.9
8	Oracle Linux 7.6 64-bit	gcc Compiler Version 4.9
9	Oracle ILOM OS v4.0	gcc Compiler Version 4.9
10	Solaris 11.4	Studio Compiler 12.4
11	Oracle ZFS Storage OS 8.8	Studio Compiler 12.4
12	Oracle ZFS Storage OS 8.8	Studio Compiler 12.4
13	Solaris 11.4	Studio Compiler 12.4
14	Oracle ILOM OS v3.0	gcc Compiler Version 4.9
15	Oracle ILOM OS v3.0	gcc Compiler Version 4.9

References

The FIPS 140-2 standard, and information on the CMVP, can be found at https://csrc.nist.gov/projects/cryptographic-module-validation-program.

Table 14 References

Reference	Full Specification Name
[FIPS 140-2]	Security Requirements for Cryptographic Modules
[FIPS 180-4]	Secure Hash Standard
[FIPS 186-2]	Digital Signature Standard (DSS) [withdrawn]
[FIPS 186-4]	Digital Signature Standard
[FIPS 197]	Advanced Encryption Standard
[FIPS 198-1]	The Keyed Hash Message Authentication Code (HMAC)
IG	Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program
[SP 800-38A]	Recommendation for Block Cipher Modes of Operation: Methods and Techniques
[SP 800-38B]	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
[SP 800-38C]	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality
[SP 800-38D]	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
[SP 800-38E]	Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices
[SP 800-56Ar1]	Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
[SP 800-56Ar3]	Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography
[SP 800-57r5]	Recommendation for Key Management: Part 1 – General
[SP 800-67r2]	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
[SP 800-89]	Recommendation for Obtaining Assurances for Digital Signature Applications
[SP 800-90Ar1]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
[SP 800-131Ar2]	Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
[SP 800-133r2]	Recommendation for Cryptographic Key Generation