Cryptographic Module for Fognigma

FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.4

September 18, 2018

Prepared for:

Berryville Holdings, LLC
2465 Centerville Road
#J17-812
Herndon, VA 20171
bvhllc.com
+1 703.782.9840

Prepared by:

KeyPair Consulting Inc.
846 Higuera St.
Ste. 2
San Luis Obispo, CA 93401
keypair.us
+1 805.316.5024

References

Reference	*Full Specification Name*
[ANS X9.31]	*Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*
[FIPS 140-2]	*Security Requirements for Cryptographic Modules, May 25, 2001*
[FIPS 180-4]	*Secure Hash Standard (SHS)*
[FIPS 186-4]	*Digital Signature Standard (DSS)*
[FIPS 197]	*Advanced Encryption Standard (AES)*
[FIPS 198-1]	*The Keyed-Hash Message Authentication Code (HMAC)*
[IG]	*Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program*
[SP 800-38B]	*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 800-56A]	*Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*
[SP 800-56B]	*Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography*
[SP 800-57]	*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 800-90A]	*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*
[SP 800-131A]	*Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*

1. Introduction

This document is the non-proprietary security policy for the *Cryptographic Module for Fognigma*, 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* (Linux®[1]/Unix®[2] and VxWorks®[3]) or *fipscanister.lib* (Microsoft Windows®[4]). The Module performs no communications other than with the calling application (the process that invokes the Module services).

The current version of the *Cryptographic Module for Fognigma* is v2.0.16. This version is fully backwards compatible with all earlier revisions of the *Cryptographic Module for Fognigma*. The v2.0.16 Module incorporates support for new platforms without disturbing functionality for any previously tested platforms. The Module includes versions v2.0.9, v2.0.10, v2.0.11, v2.0.12, v2.0.13, v2.0.14, v2.0.15, and v2.0.16.

The *Cryptographic Module for Fognigma* is a general-purpose cryptographic module integrated in the Fognigma platform to provide FIPS 140-2 validated cryptography for the protection of sensitive information. This module provides the cryptographic services that are used by the Fognigma platform to generate its Virtual Private Network (VPN). For more information, please visit: https://www.dexteredward.com/

[1] Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.
[2] UNIX is a registered trademark of The Open Group.
[3] VxWorks is a registered trademark owned by Wind River Systems, Inc.
[4] Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

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

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	NA
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	NA

Table 1 - Security Level of Security Requirements

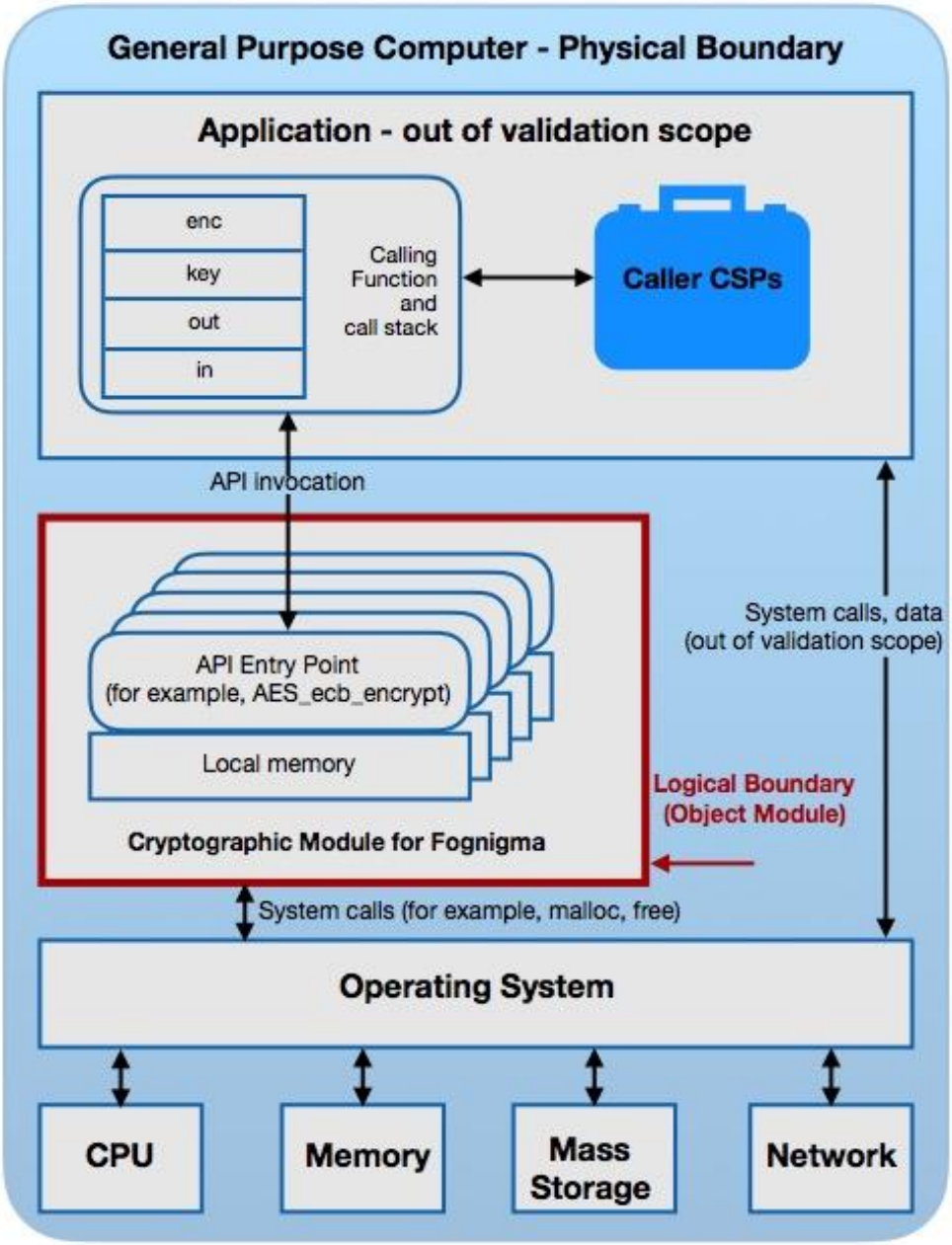

Figure 1 - Module Block Diagram

2. Tested Configurations

#	Operational Environment	Processor	Optimizations (Target)	EC	B
1	iOS 8.1 64-bit	Apple A7 (ARMv8)	None	BKP	U2
2	iOS 8.1 64-bit	Apple A7 (ARMv8)	NEON and Crypto Extensions	BKP	U2
3	iOS 8.1 32-bit	Apple A7 (ARMv8)	None	BKP	U2
4	iOS 8.1 32-bit	Apple A7 (ARMv8)	NEON	BKP	U2
5	Android 5.0	Qualcomm APQ8084 (ARMv7)	None	BKP	U2
6	Android 5.0	Qualcomm APQ8084 (ARMv7)	NEON	BKP	U2
7	Android 5.0 64-bit	SAMSUNG Exynos7420 (ARMv8)	None	BKP	U2
8	Android 5.0 64-bit	SAMSUNG Exynos7420 (ARMv8)	NEON and Crypto Extensions	BKP	U2
9	Linux 3.10	Intel Atom E3845 (x86)	None	BKP	U2
10	Linux 3.10	Intel Atom E3845 (x86)	AES-NI	BKP	U2
11	Debian 9	Intel Atom E3845 (x86)	None	BKP	U2
12	Debian 9	Intel Atom E3845 (x86)	AES-NI	BKP	U2
13	Linux 3.12	NXP T2080 (PPC)	None	BKP	U2
14	Raspbian 9 (Stretch)	ARMv7	None	BKP	U2
15	Raspbian 9 (Stretch)	ARMv7	NEON	BKP	U2
16	Ubuntu 16.04 LTS (Xenial)	Intel® Xeon® E5 (family)	None	BKP	U2
17	Ubuntu 16.04 LTS (Xenial)	Intel® Xeon® E5 (family)	AES-NI	BKP	U2

Table 2 - Tested Configurations

(B = Build Method; EC = Elliptic Curve Support). The EC column indicates support for prime curve only (P), or all NIST defined B, K, and P curves (BKP).

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.

As allowed by Implementation Guidance for FIPS 140-2 [IG] G.5, *Maintaining validation compliance of software or firmware cryptographic modules*, the validation status of the Module is maintained when operated in the following additional operating environments:

- Android 7
- Android 8

The CMVP makes no statement as to the correct operation of the Module or the security strengths of the generated keys when the specific operational environment is not listed on the validation certificate.

3. 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 application program interface (API).

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*

Table 3 - Logical interfaces

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).

4. Modes of Operation and Cryptographic Functionality

The Module supports only a FIPS 140-2 Approved mode. Table 4 and Table 5 list the Approved and Non-approved but Allowed algorithms, respectively.

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[SP 800-90A] DRBG[5] Prediction resistance supported for all variations	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function	723 845 1027 1451 2301
Encryption, Decryption and CMAC	[SP 800-67]	3-Key TDES TECB, TCBC, TCFB, TOFB; CMAC generate and verify	1853 1942 2086 2399 2850
	[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	3264 3451 3751 4469 5687
Message Digests	[FIPS 180-4]	SHA-1, SHA-2 (224, 256, 384, 512)	2702 2847 3121 3681 4559
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	2063 2197 2452 2966 3788
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS, SigVer9.31, SigVerPKCS1.5, SigVerPSS (2048/3072/4096 with all SHA-2 sizes)	1664 1766 1928 2444 3060
	[FIPS 186-4] DSA	PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024/2048/3072 with all SHA-2 sizes)	933 970 1040 1195 1463
	[FIPS 186-2] ECDSA	PKG: CURVES(P-224 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571) PKV: CURVES(P-192 P-224 P-256 P-384 P-521 K-163	620 698 801

[5] For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP 800-90A] and [SP 800-57].

Function	Algorithm	Options	Cert #
		K-233 K-283 K-409 K-571 B-163 B-233 B-283 B-409 B-571)	1091 1541
		PKG: CURVES(P-224 P-384 P-521) PKV: CURVES(P-192 P-224 P-256 P-384 P-521)	
	[FIPS 186-4] ECDSA	PKG: CURVES(P-224 P-256 P-384 P-521 K-224 K-256 K-384 K-521 B-224 B-256 B-384 B-521 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))	620 698 801 1091 1541
		PKG: CURVES(P-224 P-256 P-384 P-521) PKV: CURVES(ALL-P) 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)) 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))	

Function	Algorithm	Options	Cert #
ECC CDH (KAS) (CVL)	[SP 800-56A] (§5.7.1.2)	All NIST defined B, K and P curves except sizes 163 and 192	472 534 699 1181 2078
		All NIST defined P curves	

Table 4 - FIPS Approved Cryptographic Functions

The Module supports only NIST defined curves for use with ECDSA and ECC CDH. The Module supports two operational environment configurations for elliptic curve; NIST prime curve only (listed in Table 2 with the EC column marked "P") and all NIST defined curves (listed in Table 2 with the EC column marked "BKP").

Category	Algorithm	Description
Key Agreement	EC DH	Non-compliant (untested) DH scheme using elliptic curve, supporting all NIST defined B, K and P curves. Key agreement is a service provided for calling process use, but is not used to establish keys into the Module.
Key Encryption, Decryption	RSA	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services.

Table 5 - Non-FIPS Approved but Allowed Cryptographic Functions

The Module implements the following services which are Non-Approved per the SP 800-131A transition:

Function	Algorithm	Options
Random Number Generation; Symmetric key generation	[ANS X9.31] RNG	AES 128/192/256
Random Number Generation; Symmetric key generation	[SP 800-90A] DRBG	Dual EC DRBG (note the Dual EC DRBG algorithm shall not be used in the FIPS Approved mode of operation)
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)

Function	Algorithm	Options
	[FIPS 186-4] ECDSA	PKG: CURVES(P-192 K-163 B-163) SigGen: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224:(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 (CVL)	[SP 800-56A] (§5.7.1.2)	All NIST Recommended B, K and P curves sizes 163 and 192

Table 6 - FIPS Non-Approved Cryptographic Functions

X9.31 RNG is Non-Approved effective December 31, 2015, per the CMVP Notice "X9.31 RNG transition, December 31, 2015".

These algorithms shall not be used when operating in the FIPS Approved mode of operation.

EC DH Key Agreement provides a maximum of 256 bits of security strength. RSA Key Wrapping provides a maximum of 256 bits of security strength. For RSA Key Wrapping, modulus sizes between 2048 and 15360 are supported by the Module ($2048 \leq k \leq 15360$).

The Module requires an initialization sequence (see [IG 9.5]): the calling application invokes `FIPS_mode_set()`[6], which returns a “1” for success and “0” for failure. If `FIPS_mode_set()` fails then all cryptographic services fail from then on. The application can test to see if FIPS mode has been successfully performed.

The Module is a cryptographic engine library, which can be used only in conjunction with additional software. Aside from the use of the NIST defined elliptic curves as trusted third-party domain parameters, all other FIPS 186-4 assurances are outside the scope of the Module, and are the responsibility of the calling process.

[6] The function call in the Module is `FIPS_module_mode_set()` which is typically used by an application via the `FIPS_mode_set()` wrapper function.

4.1. 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.

CSP Name	Description
RSA SGK	RSA (1024 to 16384 bits) signature generation key
RSA KDK	RSA (1024 to 16384 bits) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (1024/2048/3072) signature generation key or [FIPS 186-2] DSA (1024) signature generation 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	AES (128/192/256) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512) XTS encrypt / decrypt key
TDES EDK	TDES (3-Key) encrypt / decrypt key
TDES CMAC	TDES (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

Table 7 - Critical Security Parameters

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.

Public Key Name	Description
RSA SVK	RSA (1024 to 16384 bits) signature verification public key
RSA KEK	RSA (1024 to 16384 bits) key encryption (public key transport) key
DSA SVK	[FIPS 186-4] DSA (1024/2048/3072) signature verification key or [FIPS 186-2] DSA (1024) signature verification key
ECDSA SVK	ECDSA (All NIST defined B, K and P curves) signature verification key
EC DH Public	EC DH (All NIST defined B, K and P curves) public key agreement key.

Table 8 - Public Keys

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 4. The calling application is responsible for storage of generated keys returned by the module.

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.

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 re-distributed.

Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism as shown in [SP 800-90A] Table 2 (Hash_DRBG, HMAC_DRBG) and Table 3 (CTR_DRBG). This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

5. 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.

Service	Role	Description
Initialize	User, CO	Module initialization. Does not access CSPs.
Self-test	User, CO	Perform self-tests (FIPS_selftest). Does not access CSPs.
Show status	User, CO	Functions that provide module status information: • Version (as unsigned long or const char *) • FIPS Mode (Boolean) Does not access CSPs.
Zeroize	User, CO	Functions that destroy CSPs: • fips_drbg_uninstantiate: for a given DRBG context, overwrites 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 Uses and updates Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.

Service	Role	Description
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 There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in [SP 800-90A]
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. Executes using AES EDK, TDES EDK (passed in by the calling process).
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, TDES, CMAC (passed in by the calling process).
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs.
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. Executes using HMAC Key (passed in by the calling process).
Key transport[7]	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). Executes using RSA KDK, RSA KEK (passed in by the calling process).
Key agreement	User, CO	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). Executes using EC DH Private, EC DH Public (passed in by the calling process).
Digital signature	User, CO	Used to generate or verify RSA, DSA or ECDSA digital signatures. Executes using RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process).
Utility	User, CO	Miscellaneous helper functions. Does not access CSPs.

Table 9 - Services and CSP Access

[7] "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 *Module*.

6. Self-Tests

The Module performs the self-tests listed below on invocation of Initialize or Self-test.

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512 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
TDES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key
TDES 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 bits with and without derivation function HASH_DRBG: SHA256 HMAC_DRBG: SHA256 Dual_EC_DRBG: P-256 and SHA256
ECDSA	PCT	Keygen, sign, verify using P-224, K-233 and SHA512. The K-233 self-test is not performed for operational environments that support prime curve only (see Table 2).
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, [IG 9.6]

Table 10 - Power-On Self-Tests (KAT = Known answer test; PCT = Pairwise consistency test)

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.

The `FIPS_mode_set()`[8] function performs all power-up self-tests listed above with no operator intervention required, returning a "1" if all power-up self-tests succeed, and a "0" otherwise. If any component of the power-up self-test fails an internal flag is set to prevent subsequent invocation of any cryptographic function calls. The module will only enter the FIPS Approved mode if the module is reloaded and the call to `FIPS_mode_set()`[8] succeeds.

The power-up self-tests may also be performed on-demand by calling `FIPS_selftest()`, which returns a "1" for success and "0" for failure. Interpretation of this return code is the responsibility of the calling application.

[8] `FIPS_mode_set()` calls Module function `FIPS_module_mode_set()`

The Module also implements the following conditional tests:

Algorithm	Test
DRBG	Tested as required by [SP 800-90A] Section 11
DRBG	FIPS 140-2 continuous test for stuck fault
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

Table 11 - Conditional Tests

In the event of a DRBG self-test failure the calling application must uninstantiate 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.

The Module supports two operational environment configurations for elliptic curve: NIST prime curves only (listed in Table 2 with the EC column marked "P") and all NIST defined curves (listed in Table 2 with the EC column marked "BKP").

7. 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.

8. 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, 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.

Each of these command sets are relative to the top of the directory containing the uncompressed and expanded contents of the distribution files *openssl-fips-2.0.16.tar.gz* (all NIST defined curves as listed in Table 2 with the EC column marked "BKP") or *openssl-fips-ecp-2.0.16.tar.gz* (NIST prime curves only as listed in Table 2 with the EC column marked "P"). The command sets are:

```
U1:
        ./config no-asm
        make
        make install

U2:
        ./config
        make
        make install

W1:
        ms\do_fips no-asm

W2:
        ms\do_fips

C1:
        c6x/do_fips
```

Installation instructions

1. Download and copy the distribution file to the build system.

   These files can be downloaded from https://www.openssl.org/source/.

2. Verify the HMAC-SHA-1 digest of the distribution file; see Appendix B. An independently acquired FIPS 140-2 validated implementation of SHA-1 HMAC 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

```
gunzip -c openssl-fips-2.0.16.tar.gz | tar xf -
cd openssl-fips-2.0.16
```

   or

```
gunzip -c openssl-fips-ecp-2.0.16.tar.gz | tar xf -
```

```
cd openssl-fips-ecp-2.0.16
```

4. Execute one of the installation command sets U1, W1, U2, W2 as shown above. No other command sets shall be used.
5. The resulting *fipscanister.o* or *fipscanister.lib* file is now available for use.
6. The calling application enables FIPS mode by calling the `FIPS_mode_set()`[9] function.

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.

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-SHA1 digest of the FIPS Object Module must be generated and embedded in the FIPS Object Module for use by the `FIPS_mode_set()`[9] 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_mode_set()`[9] 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. SSE2 optimization
3. AES-NI+PCLMULQDQ+SSSE3 optimization

Other theoretically possible combinations (e.g. AES-NI only, or SSE3 only) are not addressed individually, so that a processor which does not support all three of AES-NI, PCLMULQDQ, and SSSE3 will fall back to SSE2 optimization.

[9] `FIPS_mode_set()` calls the Module function `FIPS_module_mode_set()`

For more information, see:

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

For OpenSSL with ARM there are two possible optimization levels:

1. Without NEON
2. With NEON (ARM7 only)

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

Appendix B Controlled Distribution File Fingerprint

The *Cryptographic Module for Fognigma* v2.0.16 consists of the OpenSSL FIPS Object Module v2.0.16 (the *fipscanister.o* or *fipscanister.lib* contiguous unit of binary object code) generated from the specific source files.

For all NIST defined curves (listed in Table 2 with the EC column marked "BKP") the source files are in the specific special OpenSSL distribution *openssl-fips-2.0.16.tar.gz* with HMAC-SHA-1 digest of

```
e8dbfa6cb9e22a049ec625ffb7ccaf33e6116598
```

located at http://www.openssl.org/source/openssl-fips-2.0.16.tar.gz.

The openssl command from a version of OpenSSL that incorporates a previously validated version of the module may be used:

```
openssl sha1 -hmac etaonrishdlcupfm openssl-fips-2.0.16.tar.gz
```

For NIST prime curves only (listed in Table 2 with the EC column marked "P") the source files are in the specific special OpenSSL distribution *openssl-fips-ecp-2.0.16.tar.gz* with HMAC-SHA-1 digest of

```
205ce773d4d6a7abbcee2cf6ca9e923a0d120f1a
```

located at http://www.openssl.org/source/openssl-fips-ecp-2.0.16.tar.gz. Note this is from the previous revision of the Cryptographic Module for Fognigma as no modifications relevant to NIST prime curves only were introduced in revision 2.0.16.

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 OpenSSL distribution tar file (and patch file if used) shall be verified using the above HMAC-SHA-1 digest(s).

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 Cryptographic Module for Fognigma integrity check.

The functionality of all earlier revisions of the Cryptographic Module for Fognigma are subsumed by this latest revision, so there is no reason to use older revisions for any new deployments. However, older revisions remain valid. The source distribution files and corresponding HMAC-SHA-1 digests are listed below:

openssl-fips-2.0.15.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-2.0.15.tar.gz
Digest: `b4967baaff637dda636224fdc732992dc5a5dd77`

openssl-fips-ecp-2.0.15.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-ecp-2.0.15.tar.gz
Digest: `79038b02ac9ea08a63ce3fc60e493e3a0a728ebd`

openssl-fips-2.0.14.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-2.0.14.tar.gz
Digest: `fa4eea2c159885c896f4760dccb73efe59244f1d`

openssl-fips-ecp-2.0.14.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-ecp-2.0.14.tar.gz
Digest: `bf676ee9f52b30e9f7348d19d5e6bf5b244ccf52`

openssl-fips-2.0.13.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-2.0.13.tar.gz
Digest: `26f923491458df77a1f4c6ce39fef2f5bea88cd5`

openssl-fips-ecp-2.0.13.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-ecp-2.0.13.tar.gz
Digest: `bd935902af260cceb29d30708c7ed5461698280d`

openssl-fips-2.0.12.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-2.0.12.tar.gz
Digest: `86ec30179f1bfb2edde4ababf0fb519ba7380b69`

openssl-fips-ecp-2.0.12.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-ecp-2.0.12.tar.gz
Digest: `3da3e6d610378ad4b6ee2638a141c17cb3a2aabf`

openssl-fips-2.0.11.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-2.0.11.tar.gz
Digest: `b9d2a466c66841fcbf65a3cbe21abf81fe140bcf`

openssl-fips-ecp-2.0.11.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-ecp-2.0.11.tar.gz
Digest: `571662bb0e413bd42f612c695c0b76deb2e9b33e`

openssl-fips-2.0.10.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-2.0.10.tar.gz
Digest: `af8bda4bb9739e35b4ef00a9bc40d21a6a97a780`

openssl-fips-ecp-2.0.10.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-ecp-2.0.10.tar.gz
Digest: `02cc9ddfffb2e917d1cdc9ebc97a9731c40f6394`

openssl-fips-2.0.9.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-2.0.9.tar.gz
Digest: `54552e9a3ed8d1561341e8945fcdec55af961322`

openssl-fips-ecp-2.0.9.tar.gz
URL: http://www.openssl.org/source/old/fips/openssl-fips-ecp-2.0.9.tar.gz
Digest: `91d267688713c920f85bc5e69c8b5d34e1112672`

Note that older versions of the FIPS module are migrated from http://www.openssl.org/source/ to http://www.openssl.org/source/old/fips/, so depending on the time at which this document is referenced the URLs above may need to be adjusted accordingly.

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.

#	Operational Environment	Compiler
1	iOS 8.1 64-bit	clang-600.0.56
2	iOS 8.1 64-bit	clang-600.0.56
3	iOS 8.1 32-bit	clang-600.0.56
4	iOS 8.1 32-bit	clang-600.0.56
5	Android 5.0	gcc 4.9
6	Android 5.0	gcc 4.9
7	Android 5.0 64-bit	gcc 4.9
8	Android 5.0 64-bit	gcc 4.9
9	Linux 3.10	gcc 4.8.1
10	Linux 3.10	gcc 4.8.1
11	Debian 9	gcc 6.3.0
12	Debian 9	gcc 6.3.0
13	Linux 3.12	gcc 4.9.2
14	Raspbian 9 (Stretch)	gcc 6.3.0
15	Raspbian 9 (Stretch)	gcc 6.3.0
16	Ubuntu 16.04 LTS (Xenial)	gcc 5.4.0
17	Ubuntu 16.04 LTS (Xenial)	gcc 5.4.0

Table 12 - Compilers