Nimble Storage OpenSSL FIPS Object Module

Version 2.0.9

By the

OpenSSL Software Foundation

and

Nimble Storage

Nimble Storage OpenSSL FIPS 140-2 Security Policy

Version 2.0.9

June 18, 2015

Copyright Notice

Sponsored by:

Intersoft International, Inc.

opengear
Advanced Console Management

PKWARE

PSW GROUP

CRYPTSOFT
sponsor of Beaglebone Black platforms

CITRIX®

Acknowledgments

The OpenSSL Software Foundation (OSF) serves as the "vendor" for the validation on which this validation is based. Project management coordination for that effort was provided by:

Steve Marquess The OpenSSL Software Foundation 1829 Mount Ephraim Road Adamstown, MD 21710 USA	+1 877-673-6775 marquess@opensslfoundation.com marquess@openssl.com

with technical work by:

Stephen Henson 4 Monaco Place, Westlands, Newcastle-under-Lyme Staffordshire. ST5 2QT. England, United Kingdom	 shenson@opensslfoundation.com shenson@drh-consultancy.co.uk http://www.drh-consultancy.co.uk/
Andy Polyakov Chalmers University of Technology SE-412 96 Gothenburg Sweden	 appro@openssl.org appro@fy.chalmers.se
Tim Hudson P.O. Box 6389 Fairfield Gardens 4103 Australia ACN 074 537 821	 tjh@opensslfoundation.com tjh@cryptsoft.com http://www.cryptsoft.com/

in coordination with the OpenSSL Team at www.openssl.org.

The original validation testing was performed by InfoGard Laboratories. For information on validation or revalidations of software contact:

Marc Ireland FIPS Program Manager, CISSP InfoGard Laboratories 709 Fiero Lane, Suite 25 San Luis Obispo, CA 93401	805-783-0810 tel 805-783-0889 fax mireland@infogard.com http://www.infogard.com/

Modification History

2015-07-07	Add OE entries for Linux 3.4 64-bit under Citrix XenServer without AES-NI.
2015-06-18	Change last line of Table 4a from “ECC CDH (KAS)” to “ECC CDH (CVL)”.
2015-04-20	Revise platform to specify Nimble Storage platform details only (by Nimble Storage).
2014-11-25	(2.0.9) Addition of new platforms #97, #98, VMware Horizon Workspace 2.1 x86 under vSphere Addition of new platform #99, QNX on ARMv4 Addition of new platforms #100, #101, Apple iOS 7.1 64-bit on ARMv8
2014-01-04	Addition of new platform #96, FreeBSD 8.4 on x86 without AES-NI
2014-07-30	Addition of two platforms #94, #95, FreeBSD 10.0 on x86, and re-removal of Dual EC DRBG
2014-07-28	Changed processor names for platforms #90, #91
2014-07-11	Added new platforms #88, #89, ArbOS 5.3 on x86 and #92, #93 FreeBSD 9.2 on x86
2014-06-12	Temporarily remove misplaced platform, move Dual EC DRBG to the Non-Approved Table 4c
2014-05-29	Added platforms #86, #87 FreeBSD 9.1 on x86, #90 Linux ORACLESP 2.6 on ASPEED AST-Series (ARMv5) , #91 ORACLESP 2.6 on Emulex PILOT 3 (ARMv5)
2014-05-12	Added platforms #81 Linux 2.6 on PPC, #82, #83 AcanOS 1.0 on x86, #84 AcanOS 1.0 on ARMv5, #85 FreeBSD 8.4 on x86 Multiple changes to separate the Approved services from those that are non-Approved per the SP 800-131A transition
2013-11-08	Added two platforms #79, #80 PexOS 1.0 under vSphere with/without AES-NI
2013-11-01	Added two platforms #77, #78 iOS 6.0 with/without NEON
2013-10-02	Added six platforms (Linux 3.4 x86 virtualized under XenSource/VMware/Hyper-V, with/without AES-NI) Updated URL in Appendix A footnote
2013-08-29	Added new sponsor acknowledgment
2013-08-14	Added two Ubuntu 13.04 on ARMv7 (Beaglebone Black) and one Linux 3.8 on ARMv5TEJ platforms
2013-07-24	Added two VMware Horizon Workspace platforms

	Fixed typo in Table 4.1a, Hash DRBGs 888 bits not 880
2013-06-09	Added QNX, iOS 6.1, eCos for revision 2.0.5
2013-05-01	Added OpenWRT 2.6 for revision 2.0.4
2013-03-01	Added VMware Horizon Mobile 1.3, Apple OS X 10.7 , Apple iOS 5.0
2013-02-23	Added WinEC7 and Android 4.0 for revision 2.0.3
2013-02-14	Table 5: Removed references to non-existent Table 9
	Table 4a: added certs
	Table 4.1a: Added AES GCM
2013-01-28	Added four platforms: Android 4.1 and Android 4.2 with and without NEON
2013-01-08	Reworded section 8
2013-01-03	Added Win2008, RHEL 32/64 bit under vSphere and Win7 with AES-NI.
2012-12-08	Note EC DH Key Agreement and RSA Key Wrapping strength.
2012-10-10	Added NetBSD 5.1 on PowerPC-e500, NetBSD 5.1 on Intel Xeon 5500 (x86-64) for revision 2.0.2
2011-07-02	Added DSP Media Framework, Linux 2.6/Freescale PowerPC-e500, Android 4.0
2011-06-15	Added iOS, WinCE 5, WinCE 6 OEs

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 180-3]	*Secure Hash Standard*
[FIPS 186-4]	*Digital Signature Standard*
[FIPS 197]	*Advanced Encryption Standard*
[FIPS 198-1]	*The Keyed-Hash Message Authentication Code (HMAC)*
[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*

Reference	***Full Specification Name***
	GMAC
[SP 800-56A]	*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*
[SP 800-67R1]	*Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*
[SP 800-89]	*Recommendation for Obtaining Assurances for Digital Signature Applications*
[SP 800-90]	*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*

Table of Contents

1 Introduction

This document is the non-proprietary security policy for the Nimble Storage OpenSSL FIPS Object Module, hereafter referred to as the Module, which is build from the OpenSSL FIPS Object Module source code according to the the instructions in Appendix A.

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* compiled on Linux®[1] . 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:

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

1 Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

The Module's software version for this validation is 2.0.9. It is build from the 2.0.9 version of the OpenSSL FIPS Object Module source code.

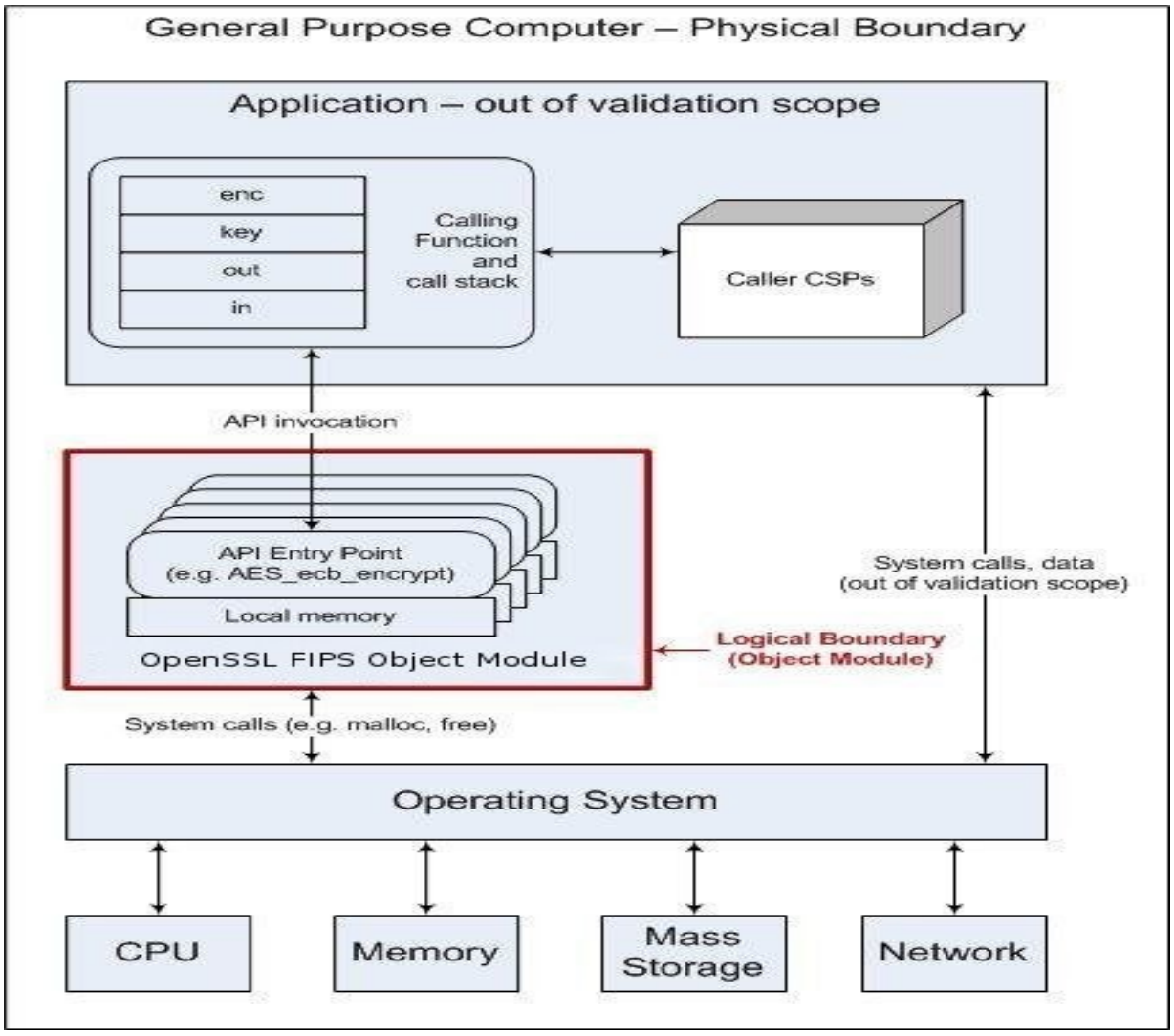

Figure 1 - Module Block Diagram

2 Tested Configurations

#	Operational Environment	Processor	Optimiz-ations (Target)	EC	B
1	Linux 2.6	Intel ES-2403V2 (x86_64)	AES-NI	BKP	U2
2	Linux 2.6	Intel ES-2450V2 (x86_64)	AES-NI	BKP	U2
3	Linux 2.6	Intel ES-2470V2 (x86_64)	AES-NI	BKP	U2
4	Linux 3.4 64-bit under Citrix XenServer	Intel Xeon E5-2430L (x86)	None	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.

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. Tables 4a and 4b list the Approved and Non-approved but Allowed algorithms, respectively.

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[ANS X9.31] RNG	AES 128/192/256	1202, 1363
	[SP 800-90A] DRBG[2] Prediction resistance supported for all variations	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function	342, 784
Encryption, Decryption and CMAC	[SP 800-67]	3-Key Triple-DES TECB, TCBC, TCFB, TOFB; CMAC generate and verify	1522, 1912
	[FIPS 197] AES	128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CCM; GCM; CMAC generate and verify	2484, 3351
	[SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS		
Message Digests	[FIPS 180-3]	SHA-1, SHA-2 (224, 256, 384, 512)	2102, 2778
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	1526, 2134
Digital Signature and Asymmetric Key Generation	[FIPS 186-3] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS, SigVer9.31, SigVerPKCS1.5, SigVerPSS (2048/3072/4096 with all SHA-2 sizes)	1273, 1718
	[FIPS 186-4] DSA	PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024/2048/3072 with all SHA-2 sizes)	764, 950

2 For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP800-90] and [SP800-57].

	[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))	413, 664
ECC CDH (CVL)	[SP 800-56A] (§5.7.1.2)	All NIST defined B, K and P curves except sizes 163 and 192	85, 496

Table 4a – 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 4b – Non-FIPS Approved But Allowed Cryptographic Functions

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[SP 800-90] DRBG	Dual EC DRBG	342
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)	764
	[FIPS 186-2] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)	764
	[FIPS 186-4] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)	764
	[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)	413
	[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))	413
ECC CDH (CVL)	[SP 800-56A] (§5.7.1.2)	All NIST Recommended B, K and P curves sizes 163 and 192	85

Table 4c – FIPS Non-Approved Cryptographic Functions

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.

The Module requires an initialization sequence (see IG 9.5): the calling application invokes

`FIPS_mode_set()`[3], 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 186-3 assurances are outside the scope of the Module, and are the responsibility of the calling process.

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
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)
RNG CSPs	Seed (128 bit), AES 128/192/256 seed key and associated state variables for ANS X9.31 AES based RNG[4]
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

3 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 There is an explicit test for equality of the seed and seed key inputs

	strength)
Dual_EC_DRBG CSPs	S (P-256, P-384, P-521), 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 4.1a – 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.

CSP 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 4.1b – Public Keys

For all CSPs and Public Keys:

Storage: RAM, associated to entities by memory location. The Module stores RNG and DRBG state values for the lifetime of the RNG or 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 RNG and DRBG state values used for the Modules' default key generation service.

Generation: The Module implements ANSI X9.31 compliant RNG and SP 800-90 compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys as shown in Table 4a. 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: 128 bits for the [ANS X9.31] RNG mechanism, and as shown in [SP 800-90] Table 2 (Hash_DRBG, HMAC_DRBG), Table 3 (CTR_DRBG) and Table 4 (Dual_EC_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_rand_prng_reset: destroys RNG CSPs.

Service	Role	Description
		• fips_drbg_uninstantiate: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_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 an RNG or DRBG instance • Determine security strength of an RNG or DRBG instance • Obtain random data Uses and updates RNG CSPs, Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_DRBG CSPs.
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 SP800-90
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. Executes using AES EDK, Triple-DES EDK (passed in by the calling process).
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, Triple-DES, 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[5]	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 5 - Services and CSP Access

5 "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 OpenSSL FIPS Object Module.

6 Self-test

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
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 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
X9.31 RNG	KAT	128, 192, 256 bit AES keys

Table 6a - 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()`[6] 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()`[6] 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.

The Module also implements the following conditional tests:

Algorithm	Test
DRBG	Tested as required by [SP800-90] 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
ANSI X9.31 RNG	Continuous test for stuck fault

Table 6b - 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 800-90]; 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").

6 `FIPS_mode_set()` calls Module function `FIPS_module_mode_set()`

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-2.0.9.tar.gz* (all NIST defined curves as listed in Table 2 with the EC column marked "BKP") or *openssl-fips-ecp-2.0.9.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
```

Installation instructions

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

   These files can be downloaded from *http://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 implemention 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.

Alternatively, a copy of the distribution on physical media can be obtained from OSF[7].

3. Unpack the distribution

```
gunzip -c openssl-fips-2.0.9.tar.gz | tar xf -
cd openssl-fips-2.0.9
```

or

```
gunzip -c openssl-fips-ecp-2.0.9.tar.gz | tar xf -
cd openssl-fips-ecp-2.0.9
```

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()`[8] 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()`[8] 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

---

7 For some prospective users the acquisition, installation, and configuration of a suitable FIPS 140-2 validated product may not be convenient. OSF will on request mail a CD containing the source code distribution, via USPS or international post. A distribution file received by that means need not be verified by a FIPS 140-2 validated implementation of HMAC-SHA-1. For instructions on requesting this CD see http://opensslfoundation.com/fips/verify.html.

8 `FIPS_mode_set()` calls the Module function `FIPS_module_mode_set()`

FIPS mode.

At runtime the `FIPS_mode_set()`[8] 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.

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 *OpenSSL FIPS Object Module v2.0.9* consists of the FIPS Object Module (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-2.0.9.tar.gz* with HMAC-SHA-1 digest of

```
54552e9a3ed8d1561341e8945fcdec55af961322
```

located at http://www.openssl.org/source/openssl-fips-2.0.9.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-2.0.9.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-2.0.9.tar.gz* with HMAC-SHA-1 digest of

```
91d267688713c920f85bc5e69c8b5d34e1112672
```

located at http://www.openssl.org/source/openssl-fips-ecp-2.0.9.tar.gz. Note this is from the previous revision of the FIPS Object Module as no modifications relevant to NIST prime curves only were introduced in revision 2.0.9.

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 FIPS Object Module integrity check.

The functionality of all earlier revisions of the FIPS Object Module 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-2.0.8.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.8.tar.gz
Digest: `7f486fbb598f3247ab9db10c1308f1c19f384671`

openssl-fips-ecp-2.0.8.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.8.tar.gz
Digest: `7a5f40ef8cebe959372d16e26391fcf23689209b`

openssl-fips-2.0.7.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.7.tar.gz
Digest: `295064925a6d95271e2fa2920181ec060f95c7ab`

openssl-fips-ecp-2.0.7.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.7.tar.gz
Digest: `dddfdc78c7e827c61fe92bd4817a7f2c3e67153`

openssl-fips-2.0.6.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.6.tar.gz
Digest: `2b8d831df22d4dfe6169aa2a8e74c35484c26c21`

openssl-fips-ecp-2.0.6.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.6.tar.gz
Digest: `852f43cd9ae1bd2eba60e4f9f1f266d3c16c0319`

openssl-fips-2.0.5.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.5.tar.gz
Digest: `8b44f2a43d098f6858eb1ebe77b73f8f027a9c29`

openssl-fips-ecp-2.0.5.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.5.tar.gz
Digest: `148e4e127ffef1df80c0ed61bae35b07ec7b7b36`

openssl-fips-2.0.4.tar.gz

URL: http://www.openssl.org/source/openssl-fips-2.0.4.tar.gz
Digest: `eaa5f86dab2c5da7086aec4786bce27d3b3c1b8a`

openssl-fips-ecp-2.0.4.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.4.tar.gz
Digest: `13302f75c82c8b482c9ac96828984a270a45c284`

openssl-fips-2.0.3.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.3.tar.gz
Digest: `5dfe03bc3f57c2862ea97823ea3111d7faf711b2`

openssl-fips-ecp-2.0.3.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.3.tar.gz
Digest: `9d6b21218d7d5480aa0add68e682d321e3ffbfa7`

openssl-fips-2.0.2.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.2.tar.gz
Digest: `e099d5096eb69c2dd8591379f38b985801188663`

openssl-fips-ecp-2.0.2.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.2.tar.gz
Digest: `887fa6802c253c32e6c4c83b7a091118fa8c6217`

openssl-fips-2.0.1.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.1.tar.gz.
Digest: `1e05b021fdcd6e77c6155512bbce2d0cbc725aec`

openssl-fips-ecp-2.0.1.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.1.tar.gz.
Digest: `af82c8ebb9d3276be11feffd35e6b55bd0d1839f`

openssl-fips-2.0.tar.gz
URL: http://www.openssl.org/source/openssl-fips-2.0.tar.gz.
Digest: `2cdd29913c6523df8ad38da11c342b80ed3f1dae`

openssl-fips-ecp-2.0.tar.gz

URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.tar.gz.
Digest: `e8d5ee306425b278bf6c8b077dae8e4a542e8215`

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	Linux 2.6	gcc 4.1.2
2	Linux 2.6	gcc 4.1.2
3	Linux 2.6	gcc 4.1.2
4	Linux 3.4	gcc 4.8.0

Table C - Compilers