FIPS 140-2 Non-Proprietary Security Policy

Oracle ILOM OpenSSL FIPS Object Module

FIPS 140-2 Level 1 Validation

Software Version: 2.0.10

Date: July 8, 2019

Title: Oracle ILOM OpenSSL FIPS Object Module Security Policy

Date: July 8, 2019

Author: Acumen Security

Contributing Authors:

Oracle Security Evaluations – Global Product Security

Oracle ILOM Engineering

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.
Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

TABLE OF CONTENTS

List of Tables

List of Figures

1. Introduction

Oracle's Integrated Lights Out Manager (ILOM) provides advanced service processor hardware and software that you can use to manage and monitor your Oracle Sun servers. Oracle ILOM's dedicated hardware and software is preinstalled on a variety of Oracle Sun server platforms, including x86-based Sun Fire servers, Sun Blade modular chassis systems, Sun Blade server modules, as well as on SPARC-based servers. Oracle ILOM is a vital management tool in the data center and can be used to integrate with other data center management tools already installed on your systems.

Oracle ILOM enables you to actively manage and monitor the server independently of the operating system state, providing you with a reliable lights out management (LOM) system. With Oracle ILOM, you can proactively:

- Learn about hardware errors and faults as they occur
- Remotely control the power state of your server
- View the graphical and non-graphical consoles for the host
- View the current status of sensors and indicators on the system
- Determine the hardware configuration of your system
- Receive generated alerts about system events in advance through IPMI PETs, SNMP traps, or email alerts.

You can access Oracle ILOM from the server 's host operating system (Solaris, Linux, and Windows). Using Oracle ILOM, you can remotely manage your server as if you were using a locally attached keyboard, monitor, and mouse. This document is the non-proprietary security policy for the Oracle ILOM OpenSSL FIPS Object 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 (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).

[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.

Note that the Oracle ILOM OpenSSL FIPS Object Module v2.0.10 is fully backwards compatible with all earlier revisions of the OpenSSL FIPS Object Module v2.0.10. The v2.0.10 Module incorporates support for new platforms without disturbing functionality for any previously tested platforms. The v2.0.10 Module can be used in any environment supported by the earlier revisions of the Module, and those earlier revisions remain valid.

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

Security Requirements Section	Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles and Services and Authentication	2
Finite State Machine 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

Table 1: Security Level of Security Requirements

The Module's software version for this validation is 2.0.10. The v2.0.10 Module incorporates changes from the v2.0 module to support additional platforms. The v2.0.10 Module can be used in all the environments supported by the earlier v2.0.9 revision of the Module.

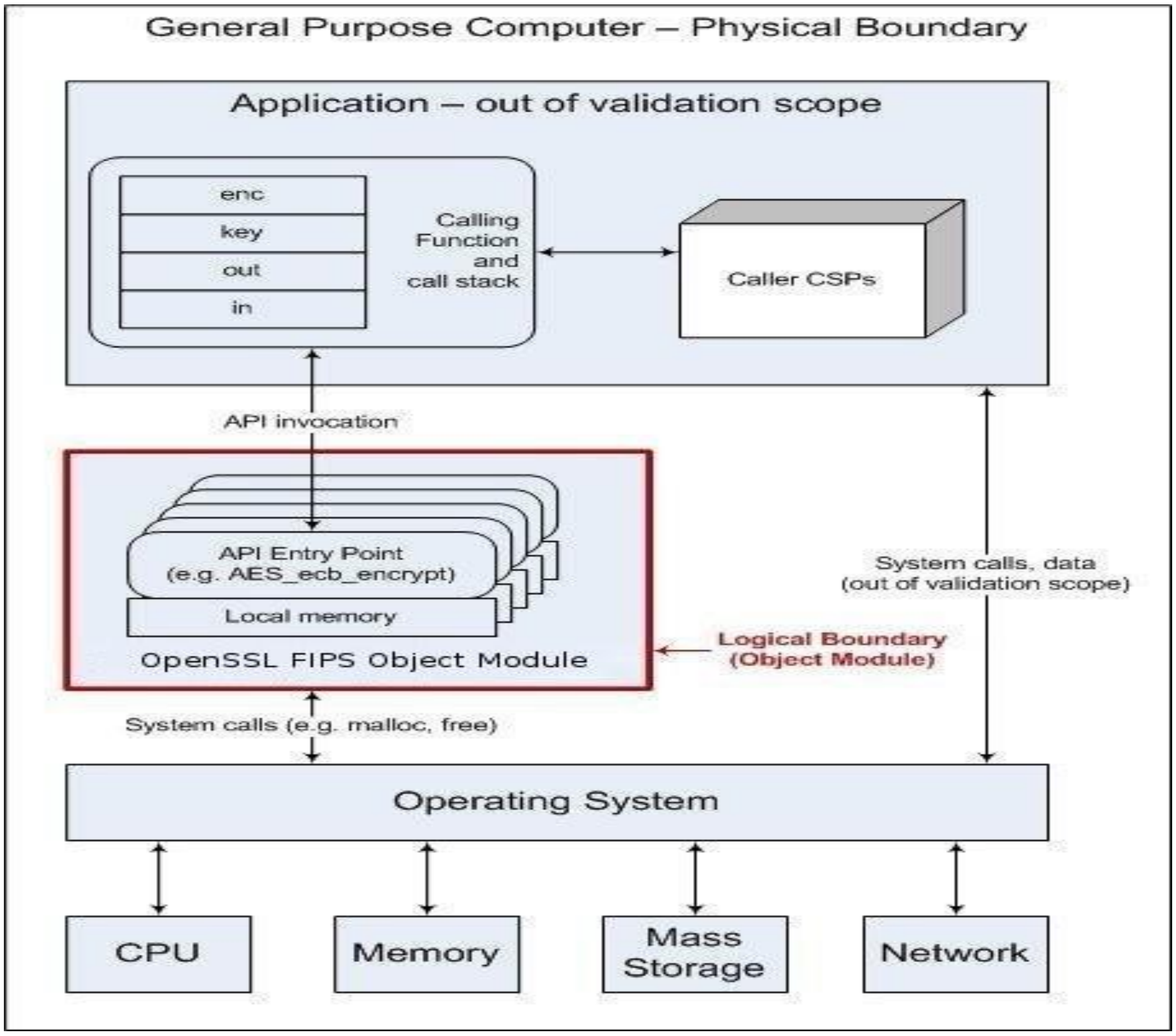
General Purpose Computer – Physical Boundary
Application – out of validation scope
enc
key
out
in
Calling Function and call stack
Caller CSPs
API invocation
API Entry Point (e.g. AES_ecb_encrypt)
Local memory
OpenSSL FIPS Object Module
Logical Boundary (Object Module)
System calls, data (out of validation scope)
System calls (e.g. malloc, free)
Operating System
CPU
Memory
Mass Storage
Network

Figure 1: Module Block Diagram

2. Tested Configurations

#	Operational Environment	Processor	Optimizations Target	EC	B
1	Oracle ILOM OS v3.0	Oracle ILOM SP v3 (ARM 7)	NEON	BKP	U2
2	Oracle ILOM OS v3.0	Oracle ILOM SP v3 (ARM 7)	None	BKP	U2
3	Oracle ILOM OS v3.0	Oracle ILOM SP v4 (ARM 5)	None	BKP	U2
4	Oracle ILOM OS v4.0	Oracle ILOM SP v5 (ARM 11)	None	BKP	U2

Table 2: Tested Configurations

(B = Build Method: EC = Elliptic Curve Support). The EC column indicates support for 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

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

Function	Algorithm	Options	Cert #
Random Number Generation; symmetric key generation	[SP 800-90] DRBG[5] Prediction resistance supported for all variations	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function	1557
Encryption, Decryption, and CMAC	[SP 800-67]	3-Key TDES TECB, TCBC, TCFB, TOFB; CMAC generate and verify	2462
	[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	4629
Message Digests	[FIPS 180-4]	SHA-1, SHA-2 (224, 256, 384, 512)	3793
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	3064
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	GenKey9.31 (2048/3072/4096) SigGen9.31, SigGenPKCS1.5, SigGenPSS (4096 with all SHA-2 sizes) SigVer9.31, SigVerPKCS1.5, SigVerPSS (1024/1536/2048/3072/4096 with all SHA sizes)	2527
	[FIPS 186-4] RSA	SigGen9.31, SigGenPKCS1.5, SigGenPSS (2048/3072 with all SHA-2 sizes)	2527
	[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)	1224
	[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 K-233 K-283 K-409 K-571 B-163 B-233 B-283 B-409 B-571)	1138
	[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)	1138

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

Function	Algorithm	Options	Cert #
		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)	
ECC CDH (KAS)	[SP 800-56A] (§5.7.1.2)	All NIST defined B, K and P curves except sizes 163 and 192	1289

Table 4: FIPS Approved Cryptographic Functions

The Module supports only NIST defined curves for use with ECDSA and ECC CDH. The Module supports all NIST defined curves on the platforms listed in Table 2 with the EC column marked "BKP".

Category	Algorithm	Description
Key Agreement	ECDH	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	[ANSI X9.31] RNG	AES 128/192/256
Random Number Generation; Symmetric Key Generation	[SP 800-90] 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)
	[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 (KAS)	[SP 800-56A] (§5.7.1.2)	All NIST Recommended B, K and P curves sizes 163 and 192

Table 6: Non-FIPS 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.

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 186-3 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
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.

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 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 800-90 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 800-90] for DRBG in Table 4 (Hash_DRBG, HMAC_DRBG, CTR_DRBG), and Table 6 (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 1/25616, or less than 1/1038. 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.
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, TDES EDK (passed in by the calling process).
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC.

Service	Role	Description
		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 *OpenSSL FIPS Object Module RE*.

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-SHA-1
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 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.
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6

Table 10: Power On Self-Test (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.

The Module also implements the following conditional tests:

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

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

Table 11: Conditional Self-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 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-2.0.10.tar.gz* (all NIST defined curves as listed in Table 2 with the EC column marked "BKP"). 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 OVS[9].
3. Unpack the distribution
   ```
   gunzip -c openssl-fips-2.0.10.tar.gz | tar xf - cd openssl-fips-2.0.10
   ```
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()[10] function.

[9] For some prospective users the acquisition, installation, and configuration of a suitable FIPS 140-2 validated product may not be convenient. OVS 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://openssl.com/fips/verify.html.

[10] FIPS_mode_set()calls the Module function FIPS_module_mode_set()

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() 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()[10] 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 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: Control Distribution File Fingerprint

The *OpenSSL FIPS Object Module RE* v2.0.10 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-2.0.10.tar.gz* with HMAC-SHA-1 digest of

```
af8bda4bb9739e35b4ef00a9bc40d21a6a97a780
```

located at http://www.openssl.org/source/openssl-fips-2.0.10.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-2.0.10.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 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.

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	Oracle ILOM OS v3.0	gcc Compiler Version 4.9
2	Oracle ILOM OS v3.0	gcc Compiler Version 4.9
3	Oracle ILOM OS v3.0	gcc Compiler Version 4.9
4	Oracle ILOM OS v4.0	gcc Compiler Version 4.9

Table 12: Compilers

References

The FIPS 140-2 standard, and information on the CMVP, can be found at http://csrc.nist.gov/groups/STM/cmvp/index.html. More information describing the module can be found on the Oracle web site at www.oracle.com .

This Security Policy contains non-proprietary information. All other documentation submitted for FIPS 140-2 conformance testing and validation is "Oracle - Proprietary" and is releasable only under appropriate non-disclosure agreements.

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 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 13: References