KeyPair FIPS Object Module for OpenSSL

FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.2

June 11, 2020

KeyPair Consulting Inc.
987 Osos Street
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[FIPS 186-2]	*Digital Signature Standard (DSS) [withdrawn]*
[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 800-38A]	*Recommendation for Block Cipher Modes of Operation: Methods and Techniques*
[SP 800-38B]	*Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
[SP 800-38C]	*Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*
[SP 800-38D]	*Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*
[SP 800-38E]	*Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices*
[SP 800-56A]	*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 R5]	*Recommendation for Key Management: Part 1 - General*
[SP 800-67 R2]	*Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*
[SP 800-89]	*Recommendation for Obtaining Assurances for Digital Signature Applications*
[SP 800-90A R1]	*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*
[SP 800-131A R2]	*Transitioning the Use of Cryptographic Algorithms and Key Lengths*
[SP 800-133 R2]	*Recommendation for Cryptographic Key Generation*

Table of Contents

References **2**
1 Introduction **4**
2 Ports and Interfaces **5**
3 Modes of Operation and Cryptographic Functionality **6**
3.1 Critical Security Parameters and Public Keys 8
4 Roles, Authentication and Services **10**
5 Self-Tests **12**
6 Operational Environment **13**
7 Mitigation of other Attacks **14**
Appendix A - Installation and Usage Guidance **15**
Appendix B - Controlled Distribution File Fingerprint **17**
Appendix C - Compilers **18**

1 Introduction

This document is the non-proprietary security policy for the *KeyPair FIPS Object Module for OpenSSL (FIPS 140-2 Cert. #3503)*, hereafter referred to as the Module.

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

The current version of the *KeyPair FIPS Object Module for OpenSSL* is 1.0.

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

Table 1 - Security Level of Security Requirements

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

This FIPS module is useful for applications using OpenSSL 1.0.2 that require FIPS 186-4 RSA key generation.

Please contact KeyPair Consulting to include your desired operating systems as *Tested Configurations* on a FIPS 140-2 certificate branded in your company's name.

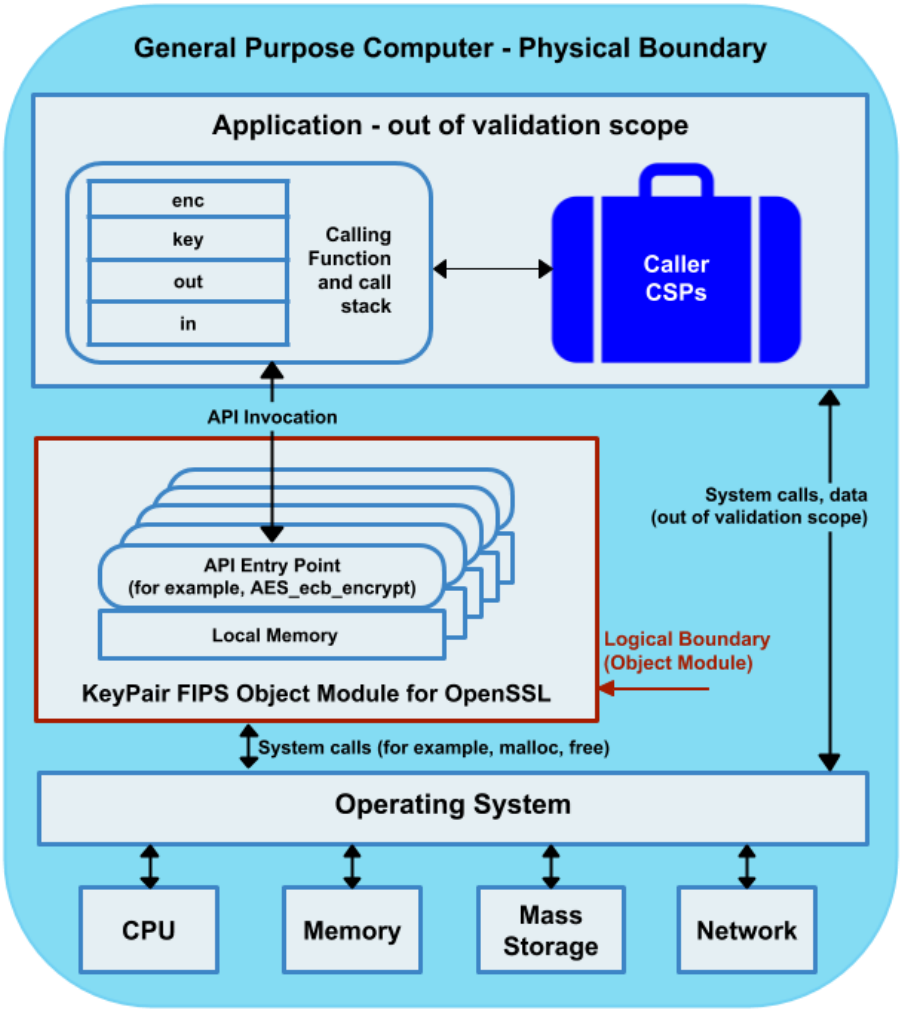

Figure 1 - Module Block Diagram

2 Ports and Interfaces

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

Table 2 - Logical interfaces

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

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

3 Modes of Operation and Cryptographic Functionality

The Module supports a FIPS 140-2 Approved mode and a non-Approved Mode. Table 3 and Table 4 list the Approved and non-Approved but Allowed algorithms, respectively. Table 5 lists the non-Approved algorithms that shall not be used in the FIPS Approved mode of operation. All services available in the Approved mode of operation are available in the non-Approved mode with the addition of the non-Approved services.

Table 3 - FIPS Approved Cryptographic Functions

Function	Algorithm	Options	Cert. #
Random Number Generation; Symmetric Key Generation	[SP 800-90A R1] DRBG[1]	Hash_Based DRBG: All SHA sizes HMAC_Based DRBG: All SHA sizes CTR_DRBG: AES-128, AES-192, AES-256 (with and without derivation function) Prediction resistance supported for all variations	C904 C1318 C1795
Cryptographic Key Generation (CKG)	[SP 800-133 R2] CKG		Vendor affirmed
Encryption, Decryption and CMAC	[SP 800-67 R2] TDES [SP 800-38B] CMAC	TECB, TCBC, TCFB, TOFB: 3-Key CMAC generate and verify: 3-Key	C904 C1318 C1795
	[FIPS 197] AES [SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS	ECB, CBC, OFB, CFB, CTR: 128/192/256 CMAC generate and verify: 128/192/256 CCM: 128/192/256 GCM: 128/192/256 XTS: 128/256	C904 C1318 C1795
Message Digests	[FIPS 180-4] SHA	SHA-1, SHA-2 (224, 256, 384, 512)	C904 C1318 C1795
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	C904 C1318 C1795
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	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	C904 C1318 C1795
	[FIPS 186-4] RSA	KeyGen: 2048/3072 SigGen9.31, SigGenPKCS1.5, SigGenPSS: 2048/3072 with all SHA2 sizes	C904 C1318 C1795
	[FIPS 186-4] DSA	KeyPairGen: 2048/3072 PQGGen, SigGen: 2048/3072 with all SHA-2 sizes PQGVer, SigVer: 1024/2048/3072 with all SHA sizes	C904 C1318 C1795
	[FIPS 186-4] ECDSA	PKG: 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: All NIST defined B, K and P curves SigGen: P-224, P-256, P-384, P-521, K-224, K-256, K-384, K-521, B-224, B-256, B-384, B-521; all SHA2 sizes SigVer: All NIST defined B, K and P curves; all SHA sizes	C904 C1318 C1795

[1] For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP 800-90A R1] and [SP 800-57 R5].

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	C904 C1318 C1795

The Module supports the following non-Approved but allowed functions:

Table 4 - Non-FIPS Approved but Allowed Cryptographic Functions

Category	Algorithm	Description
Key Agreement	DH	Key agreement is a service provided by the module to establish session keys for secure communication with another module using the Diffie-Hellman (DH) algorithm.
Key Agreement	EC DH	Key agreement is a service provided by the module to establish session keys for secure communication with another module using the EC Diffie-Hellman (EC DH) algorithm.
Key Encryption, Decryption	RSA	RSA may be used to perform key establishment with another module by securely exchanging symmetric encryption keys with another module.
Entropy source	NDRNG	Used only to seed the Approved DRBG

The module supports the following non-Approved but allowed algorithms:

- RSA (key wrapping; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112 bits of encryption strength)
- Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112 bits of encryption strength)
- EC Diffie-Hellman (CVL Certs. #C904, #C1318 and #C1795, key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)

The Module implements the following services which are non-Approved per the [SP 800-131A R2] transition:

Table 5 - FIPS Non-Approved Cryptographic Functions

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

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

3.1 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 is described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

Table 6 - Critical Security Parameters

CSP Name	Description
RSA SGK	RSA (2048 to 15360 bits) signature generation key
RSA KDK	RSA (1024 to 16384 bits) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (2048/3072) signature generation key
DH Private	Diffie-Hellman ≥ 2048 private key agreement key
ECDSA SGK	ECDSA (All NIST defined B, K, and P curves except sizes 163 and 192) signature generation key
EC DH Private	EC DH (All NIST defined B, K, and P curves except sizes 163 and 192) 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)
Hash_DRBG CSPs	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
HMAC_DRBG CSPs	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
CTR_DRBG CSPs	V (128 bits) and Key (AES 128/192/256), entropy input (length dependent on security strength)
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

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

The module does not output intermediate key generation values.

Table 7 - Public Keys

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

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 Module's default key generation service.

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

Table 8 - DRBG Entropy Requirements

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

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

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

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

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

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

Following RFC 5288 for TLS, the module ensures that it's strictly increasing and thus cannot repeat. When the IV exhausts the maximum number of possible values for a given session key, the first party, client or server, to encounter this condition may either trigger a handshake to establish a new encryption key in accordance with RFC 5246, or fail. In either case, the module prevents and IV duplication and thus enforces the security property. The Module's IV is generated internally by the Module's Approved DRBG. The DRBG seed is generated inside the Module's physical boundary. The IV is 96 bits in length per NIST SP 800-38D, Section 8.2.2 and FIPS 140-2 [IG] A.5 scenario 2.

The selection of the IV construction method is the responsibility of the user of this Module. In the approved mode, users of the Module must not utilize GCM with an externally generated IV.

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

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

4 Roles, Authentication and Services

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

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

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

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

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

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

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

Table 9 - Services and CSP Access

Service	Role	Description	Access Type
Initialize	User, CO	Module initialization. Does not access CSPs. CO-AD-Digest, User-AD-Digest	E
Self-test	User, CO	Perform self-tests (FIPS_selftest).	None
Show status	User, CO	Functions that provide module status information: ● Version (as unsigned long or const char *) ● FIPS Mode (Boolean)	None
Zeroize	User, CO	Functions that destroy CSPs: ● fips_drbg_uninstantiate 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.	D
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 DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs)	E
Asymmetric key generation	User, CO	Used to generate DSA, ECDSA and RSA keys: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK	G
Symmetric encrypt/ decrypt	User, CO	Used to encrypt or decrypt data. AES EDK, Triple-DES EDK, AES GCM, AES XTS (passed in by the calling process)	E
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. AES CMAC, Triple-DES CMAC (passed in by the calling process)	E
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest.	None
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. HMAC Key (passed in by the calling process)	E
Key transport[2]	User, CO	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). RSA KDK, RSA KEK (passed in by the calling process)	E
Key agreement	User, CO	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). DH/EC DH Private, DH/EC DH Public (passed in by the calling process)	E
Digital signature	User, CO	Used to generate or verify RSA, DSA or ECDSA digital signatures. RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process)	E
Utility	User, CO	Miscellaneous helper functions.	None

[2] "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*.

5 Self-Tests

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

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

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT per SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 Per [IG] 9.3, this testing covers SHA POST requirements.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128-bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192-bit key length
AES GCM	KAT	Separate encrypt and decrypt, 256-bit 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	Generate and verify CBC mode, 128, 192, 256-bit 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: SHA-256 HMAC_DRBG: SHA-256
ECDSA	PCT	Key gen, sign, verify using P-224, K-233 and SHA-512
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, [IG] 9.6

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

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

The module has a function called `FIPS_module_mode_set()` within the init code that is automatically set to enable “FIPS Mode” by default. When the Module is initialized, it will always run its power-on self-tests, meeting the [IG] 9.10 requirement.

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

The Module also implements the following conditional tests:

Table 11 - Conditional Tests

Algorithm	Test
DRBG	Tested as required by [SP 800-90A R1] Section 11
DRBG	FIPS 140-2 continuous test for stuck fault
NDRNG	FIPS 140-2 continuous test for NDRNG
DSA	Pairwise consistency test on each generation of a key pair
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair

In the event of a DRBG self-test failure, the calling application must uninstantiate and re-instantiate the DRBG per the requirements of [SP 800-90A R1]; this is not something the Module can do itself.

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

6 Operational Environment

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

The module was tested in the following configurations.

Table 12 - Tested Configurations

#	Operating System	Processor	Optimizations (Target)	Platform
1	Ubuntu 18.04 LTS	Intel Xeon E5-2609	PAA	HPE ProLiant DL60 Gen9
2	Ubuntu 18.04 LTS	Intel Xeon E5-2609	None	HPE ProLiant DL60 Gen9
3	CentOS 6	Intel Xeon E5-2609	PAA	HPE ProLiant DL60 Gen9
4	CentOS 6	Intel Xeon E5-2609	None	HPE ProLiant DL60 Gen9
5	CentOS 7	Intel Xeon E5-2609	PAA	HPE ProLiant DL60 Gen9
6	CentOS 7	Intel Xeon E5-2609	None	HPE ProLiant DL60 Gen9
7	Fedora Linux 24	ARM Cortex-A53	PAA	Samsung ARTIK 710 SOM
8	Fedora Linux 24	ARM Cortex-A53	None	Samsung ARTIK 710 SOM
9	Windows Server 2019	Intel Xeon E5-2609	PAA	HPE ProLiant DL60 Gen9
10	Windows Server 2019	Intel Xeon E5-2609	None	HPE ProLiant DL60 Gen9

As described in [IG] 1.21, Processor Algorithm Acceleration (PAA) describes mathematical constructs and not the complete cryptographic algorithm (as defined in the NIST standards). Examples of PAA supported by the Module include AES-NI and NEON.

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

7 Mitigation of other Attacks

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

Appendix A - Installation and Usage Guidance

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

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

Installation and Configuration Instructions

The module can be downloaded from the Solaris Git Repository. The link to the Module code is here:

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

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

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

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

4. Run the command set

   ```
   $ ./config
   $ make
   $ make install
   ```

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

Note that failure to use one of the specified command 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. An HMAC-SHA-1 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()` function compares the embedded HMAC-SHA-1 digest with a digest generated from the FIPS Object Module object code. This digest is the final link in the chain of validation from the original source to the runtime executable application file.

Optimization

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

For OpenSSL with x86, there are three possible optimization levels:

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

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

- http://www.intel.com/support/processors/sb/CS-030123.htm?wapkw=sse2
- 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 *KeyPair FIPS Object Module for OpenSSL* consists of the FIPS Object Module (the fipscanister.o contiguous unit of binary object code) generated from the specific source files.

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

```
ef8f7a91979cad14d033d8803a89fdf925102a30
```

located at

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

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

The arbitrary 16-byte key of:

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

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

Appendix C - Compilers

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

Table 13 - Compilers

#	Operational Environment	Compiler
1	Ubuntu 18.04 LTS	gcc 7.4.0
2	Ubuntu 18.04 LTS	gcc 7.4.0
3	CentOS 6	gcc 4.4.7
4	CentOS 6	gcc 4.4.7
5	CentOS 7	gcc 4.8.5
6	CentOS 7	gcc 4.8.5
7	Fedora Linux 24	gcc 6.2.1
8	Fedora Linux 24	gcc 6.2.1
9	Windows Server 2019	gcc 9.3.0
10	Windows Server 2019	gcc 9.3.0