OpenSSL™

Cryptography and SSL/TLS Toolkit

**OpenSSL FIPS**<br>
**Object Module**

**Version 2.0**

**By the**

**OpenSSL Software Foundation**

# OpenSSL FIPS 140-2 Security Policy

**Version 2.0**

**June 19, 2012**

# Copyright Notice

## Sponsored by:

Intersoft International, Inc.

opengear
Advanced Console Management

quintessencelabs

PKWARE®

CERBERUS

PSW GROUP

# Acknowledgments

The OpenSSL Software Foundation (OSF) serves as the "vendor" for this validation. Project management coordination for this 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.

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

Mark Minnoch
FIPS Program Manager, CISSP
InfoGard Laboratories
709 Fiero Lane, Suite 25
San Luis Obispo, CA 93401

805-783-0810 tel
805-783-0889 fax
mminnoch@infogard.com
http://www.infogard.com/

# Modification History

| | |
|---|---|
| 2012-06-19 | Updated URLs. |
| 2012-06-08 | Added note on re-distribution of AES-GCM keys |
| 2012-05-31 | Fixed errors in Table 2 for platforms 6, 7, 9 |
| 2012-05-02 | Added additional platform |
| 2012-04-30 | Added footnote on delivery of verification utility on physical media |
| 2012-04-13 | Added Appendix C |
| 2012-04-11 | Add discussion of FIPS validated mechanism for source distribution digest verification |
| 2012-04-06 | Added additional platforms |
| 2012-02-27 | Added additional platforms |
| 2011-12-23 | Final draft for submission |
| 2011-03-28 | Initial draft |

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 186-3]* | *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-* | *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher* |

| ***Reference*** | ***Full Specification Name*** |
|---|---|
| *67R1]* | |
| *[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 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).

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*

The Module's software version for this validation is 2.0, replacing the previous OpenSSL FIPS Object Module v1.2. The 2.0 Module incorporates changes from the v1.2 module to support the

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

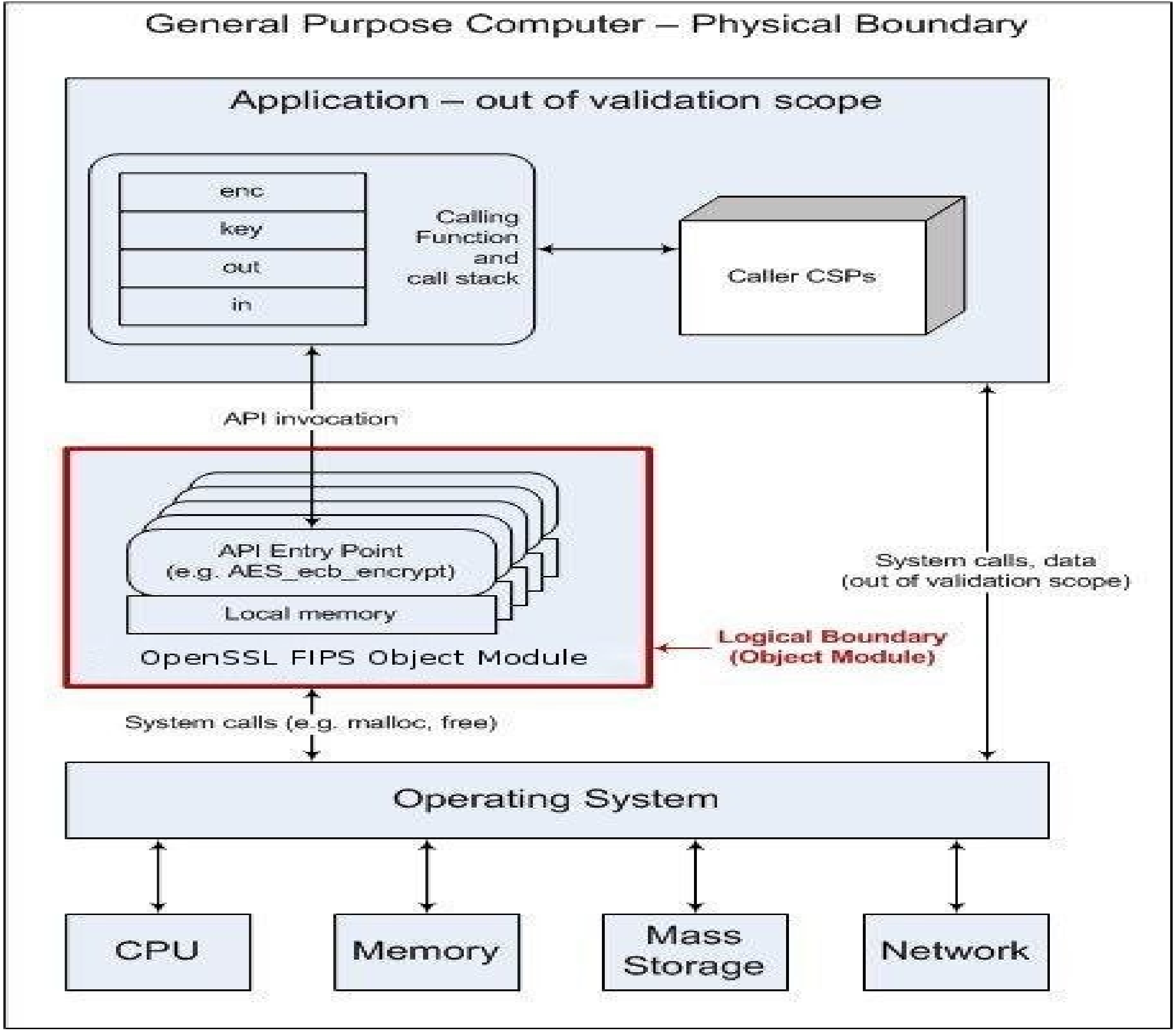
[2] UNIX is a registered trademark of The Open Group

[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.

cross-compiled uClinux[5] platform and a a number of bug fixes.  The 2.0 Module can be used in all the  environments supported by the earlier v1.2 Module.

*Figure 1 - Module Block Diagram*


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

[5] uClinux is the registered trademark of Arcturus Networks Inc.

# 2 Tested Configurations

| # | Operational Environment | Processor | Optimiz-ations (Target) | EC | B |
|---|---|---|---|---|---|
| 1 | Android 2.2 (HTC Desire) | Qualcomm QSD 8250 (ARMv7) | NEON | P | U2 |
| 2 | Android 2.2 (Dell Streak) | Qualcomm QSD 8250 (ARMv7) | None | P | U2 |
| 3 | Microsoft Windows 7 32 bit | Intel Celeron (x86) | None | BKP | W2 |
| 4 | uClinux 0.9.29 | ARM 922T (ARMv4) | None | BKP | U2 |
| 5 | Fedora 14 | Intel Core i5 (x86) | AES-NI | BKP | U2 |
| 6 | HP-UX 11i ( hpux-ia64-cc, 32 bit mode) | Intel Itanium 2 (IA64) | None | BKP | U1 |
| 7 | HP-UX 11i ( hpux64-ia64-cc, 64 bit mode) | Intel Itanium 2 (IA64) | None | BKP | U1 |
| 8 | Ubuntu 10.04 | Intel Pentium T4200 (x86) | None | BKP | U2 |
| 9 | Android 3.0 | NVIDIA Tegra 250 T20 (ARMv7) | None | P | U2 |
| 10 | Linux 2.6.27 | PowerPC e300c3 (PPC) | None | BKP | U2 |
| 11 | Microsoft Windows 7 64 bit | Intel Pentium 4 (x86) | None | BKP | W2 |
| 12 | Ubuntu 10.04 32 bit | Intel Core i5 (x86) | AES-NI | BKP | U2 |
| 13 | Linux 2.6.33 | PowerPC32 e300 (PPC) | None | BKP | U2 |
| 16 | Android 2.2 | OMAP 3530 (ARMv7) | NEON | BKP | U2 |
| 19 | VxWorks 6.8 | TI TNETV1050 (MIPS) | None | BKP | U2 |
| 20 | Linux 2.6 | Broadcom BCM11107 (ARMv6) | None | BKP | U2 |
| 21 | Linux 2.6 | TI TMS320DM6446 (ARMv4) | None | BKP | U2 |
| 22 | Linux 2.6.32 | TI AM3703CBP (ARMv7) | None | BKP | U2 |
| 23 | Solaris 10 32bit | SPARC-T3 (SPARCv9) | None | BKP | U2 |
| 24 | Solaris 10 64bit | SPARC-T3 (SPARCv9) | None | BKP | U2 |
| 25 | Solaris 11 32bit | Intel Xeon 5675 (x86) | None | BKP | U2 |
| 26 | Solaris 11 64bit | Intel Xeon 5675 (x86) | None | BKP | U2 |
| 27 | Solaris 11 32bit | Intel Xeon 5675 (x86) | AES-NI | BKP | U2 |
| 28 | Solaris 11 64bit | Intel Xeon 5675 (x86) | AES-NI | BKP | U2 |

| | | | | | |
|---|---|---|---|---|---|
| 29 | Oracle Linux 5 64bit | Intel Xeon 5675 (x86) | None | BKP | U2 |
| 30 | CascadeOS 6.1 32bit | Intel Pentium T4200 (x86) | None | BKP | U2 |
| 31 | CascadeOS 6.1 64bit | Intel Pentium T4200 (x86) | None | BKP | U2 |
| 32 | Ubuntu 10.04 32bit | Intel Pentium T4200 (x86) | None | BKP | U1 |
| 33 | Ubuntu 10.04 64bit | Intel Pentium T4200 (x86) | None | BKP | U1 |
| 34 | Oracle Linux 5 | Intel Xeon 5675 (x86) | AES-NI | BKP | U2 |
| 35 | Oracle Linux 6 | Intel Xeon 5675 (x86) | None | BKP | U2 |
| 36 | Oracle Linux 6 | Intel Xeon 5675 (x86) | AES-NI | BKP | U2 |
| 37 | Solaris 11 32bit | SPARC-T3 (SPARCv9) | None | BKP | U2 |
| 38 | Solaris 11 64bit | SPARC-T3 (SPARCv9) | None | BKP | U2 |
| 39 | Android 4.0 | NVIDIA Tegra 250 T20 ARMv7) | None | P | 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 | 985 |
| | [SP 800-90] DRBG[6]<br>Prediction resistance supported for all variations | Hash DRBG<br>HMAC DRBG, no reseed<br>CTR DRBG (AES), no derivation function<br>Dual EC DRBG: P-256, P-384, P-521 | 157 |
| Encryption, Decryption and CMAC | [SP 800-67] | 3-Key TDES TECB, TCBC, TCFB, TOFB; CMAC generate and verify | 1223 |
| | [FIPS 197] AES | 128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CCM; GCM; CMAC generate and verify | 1884 |
| | [SP 800-38B] CMAC<br>[SP 800-38C] CCM<br>[SP 800-38D] GCM<br>[SP 800-38E] XTS | | |
| Message Digests | [FIPS 180-3] | SHA-1, SHA-2 (224, 256, 384, 512) | 1655 |
| Keyed Hash | [FIPS 198] HMAC | SHA-1, SHA-2 (224, 256, 384, 512) | 1126 |
| Digital Signature and Asymmetric Key Generation | [FIPS 186-2] RSA | GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS, SigVer9.31, SigVerPKCS1.5, SigVerPSS (1024/1536/2048/3072/4096 with all SHA sizes) | 960 |
| | [FIPS 186-2] DSA | PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024 with SHA-1 only) | 589 |
| | [FIPS 186-3] DSA | PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024/2048/3072 with all SHA sizes) | 589 |
| | [FIPS 186-2] ECDSA | Key Pair, PKV, SigGen, SigVer (all NIST defined B, K, and P curves with SHA-1 only) | 270 |
| | | Key Pair, PKV, SigGen, SigVer (all NIST defined P curves with SHA-1 only) | 264 |
| | [FIPS 186-3] ECDSA | Key Pair, PKV, SigGen, SigVer (all NIST defined B, K and P curves with all SHA sizes) | 270 |
| | | Key Pair, PKV, SigGen, SigVer (all NIST defined P curves with all SHA sizes) | 264 |
| ECC CDH (KAS) | [SP 800-56A] (§5.7.1.2) | All NIST defined B, K and P curves | 12 |
| | | All NIST defined P curves | 10 |

*Table 4a – FIPS Approved Cryptographic Functions*

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

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*

The Module supports only a FIPS 140-2 Approved mode. The Module requires an initialization sequence (see IG 9.5): the calling application invokes `FIPS_mode_set()`[7], 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 186-3] 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 |

[7] 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.

| | |
|---|---|
| AES XTS | AES (256/512) XTS cipher 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) |
| RNG CSPs | Seed (128 bit), AES 128/192/256 seed key and associated state variables for ANS X9.31 AES based RNG[8] |
| Hash_DRBG CSPs | V (440/880 bits) and C (440/880 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) |
| 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 186-3] 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

[8] There is an explicit test for equality of the seed and seed key inputs

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, inclusive of all Table 9 tests (FIPS_module_mode_set). Does not access CSPs. |
| Self-test | User, CO | Perform all Table 9 tests (FIPS_selftest). Does not access CSPs. |
| Show status | User, CO | Functions that provide module status information:<br>• Version (as unsigned long or const char *)<br>• FIPS Mode (Boolean)<br>Does not access CSPs. |
| Zeroize | User, CO | Functions that destroy CSPs:<br>• fips_rand_prng_reset: destroys RNG CSPs.<br>• fips_drbg_uninstantiate: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_DRBG CSPs.) |

| Service | Role | Description |
|---|---|---|
| | | 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.<br>• Seed or reseed an RNG or DRBG instance<br>• Determine security strength of an RNG or DRBG instance<br>• Obtain random data<br>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:<br>RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK<br>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.<br>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.<br>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.<br>Does not access CSPs. |
| Keyed Hash | User, CO | Used to generate or verify data integrity with HMAC.<br>Executes using HMAC Key (passed in by the calling process). |
| Key transport[9] | User, CO | Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module).<br>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).<br>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.<br>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*

[9] "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<br>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<br>HASH_DRBG: SHA256<br>HMAC_DRBG: SHA256<br>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()`[10] 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

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

mode if the module is reloaded and the call to `FIPS_mode_set()`[10] 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").

# 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 does not claim any attack mitigation beyond [FIPS 140-2] Level 1 requirements.

# 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.tar.gz* (all NIST defined curves as listed in Table 2 with the EC column marked "BKP") or *openssl-fips-ecp-2.0.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[11].

[11] 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://openssl.com/fips/verify.html.

3. Unpack the distribution

   ```
   gunzip -c openssl-fips-2.0.tar.gz | tar xf -
   cd openssl-fips-2.0
   ```

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()`[12] 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()`[12] 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()`[12] 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). Currently no platforms are at this level
2. SSE2 optimization
3. AES-NI+PCLMULQDQ+SSSE3 optimization

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

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 2.0* 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.tar.gz* with HMAC-SHA-1 digest of

```
2cdd29913c6523df8ad38da11c342b80ed3f1dae
```

located at http://www.openssl.org/source/openssl-fips-2.0.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.tar.gz* with HMAC-SHA-1 digest of

```
e8d5ee306425b278bf6c8b077dae8e4a542e8215
```

located at http://www.openssl.org/source/openssl-fips-ecp-2.0.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.tar.gz
openssl sha1 -hmac etaonrishdlcupfm openssl-fips-ecp-2.0.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 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 | Android 2.2 (HTC Desire) | gcc 4.4.0 |
| 2 | Android 2.2 (Dell Streak) | gcc 4.4.0 |
| 3 | Microsoft Windows 7 32 bit | Microsoft 32-bit C/C++ Optimizing Compiler Version 16.00 |
| 4 | uClinux 0.9.29 | gcc 4.2.1 |
| 5 | Fedora 14 | gcc 4.5.1 |
| 6 | HP-UX 11i ( hpux-ia64-cc, 32 bit mode) | HP C/aC++ B3910B |
| 7 | HP-UX 11i ( hpux64-ia64-cc, 64 bit mode) | HP C/aC++ B3910B |
| 8 | Ubuntu 10.04 | gcc 4.1.3 |
| 9 | Android 3.0 | gcc 4.4.0 |
| 10 | Linux 2.6.27 | gcc 4.2.4 |
| 11 | Microsoft Windows 7 64 bit | Microsoft C/C++ Optimizing Compiler Version 16.00 for x64 |
| 12 | Ubuntu 10.04 32 bit | gcc 4.1.3 |
| 13 | Linux 2.6.33 | gcc 4.1.0 |
| 16 | Android 2.2 | gcc 4.1.0 |
| 19 | VxWorks 6.8 | gcc 4.1.2 |
| 20 | Linux 2.6 | gcc 4.3.2 |
| 21 | Linux 2.6 | gcc 4.3.2 |
| 22 | Linux 2.6.32 | gcc 4.3.2 |
| 23 | Solaris 10 32bit | gcc 3.4.3 |
| 24 | Solaris 10 64bit | gcc 3.4.3 |
| 25 | Solaris 11 32bit | gcc 4.5.2 |
| 26 | Solaris 11 64bit | gcc 4.5.2 |
| 27 | Solaris 11 32bit | gcc 4.5.2 |
| 28 | Solaris 11 64bit | gcc 4.5.2 |
| 29 | Oracle Linux 5 64bit | gcc 4.1.2 |
| 30 | CascadeOS 6.1 32bit | gcc 4.4.5 |
| 31 | CascadeOS 6.1 64bit | gcc 4.4.5 |

| | | |
|---|---|---|
| 32 | Ubuntu 10.04 32bit | gcc 4.1.3 |
| 33 | Ubuntu 10.04 64bit | gcc 4.1.3 |
| 34 | Oracle Linux 5 | gcc 4.1.2 |
| 35 | Oracle Linux 6 | gcc 4.4.6 |
| 36 | Oracle Linux 6 | gcc 4.4.6 |
| 37 | Solaris 11 32bit | Sun C 5.12 |
| 38 | Solaris 11 64bit | Sun C 5.12 |
| 39 | Android 4.0 | gcc 4.4.3 |

*Table C - Compilers*