Intel OpenSSL
FIPS Object Module
Version 2.0.5 and 2.0.8

Intel OpenSSL FIPS 140-2 Security Policy
Version 1.0
May 2, 2016
Copyright Notice

This document may be freely reproduced in whole or part without permission and without restriction.
Acknowledgments

This document was derived from the Security Policy associated with FIPS 140-2 validation certificate number 1747.
## Modification History

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-02-03</td>
<td>Initial update for 1B SUB</td>
</tr>
<tr>
<td>2016-04-13</td>
<td>Update due to CMVP Comments</td>
</tr>
<tr>
<td>2016-04-27</td>
<td>Update due to CMVP Comments</td>
</tr>
<tr>
<td>2016-05-02</td>
<td>Update due to CMVP Comments</td>
</tr>
</tbody>
</table>
## References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Full Specification Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ANS X9.31]</td>
<td>Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)</td>
</tr>
<tr>
<td>[FIPS 180-3]</td>
<td>Secure Hash Standard</td>
</tr>
<tr>
<td>[FIPS 186-4]</td>
<td>Digital Signature Standard</td>
</tr>
<tr>
<td>[FIPS 197]</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>[SP 800-38B]</td>
<td>Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication</td>
</tr>
<tr>
<td>[SP 800-38C]</td>
<td>Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality</td>
</tr>
<tr>
<td>[SP 800-38D]</td>
<td>Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC</td>
</tr>
<tr>
<td>[SP 800-56A]</td>
<td>Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography</td>
</tr>
<tr>
<td>[SP 800-67R1]</td>
<td>Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher</td>
</tr>
<tr>
<td>[SP 800-89]</td>
<td>Recommendation for Obtaining Assurances for Digital Signature Applications</td>
</tr>
<tr>
<td>[SP 800-90]</td>
<td>Recommendation for Random Number Generation Using Deterministic Random Bit Generators</td>
</tr>
<tr>
<td>[SP 800-131A]</td>
<td>Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction .................................................................................................................................. 6
2 Tested Configurations .................................................................................................................. 8
3 Ports and Interfaces .................................................................................................................... 8
4 Modes of Operation and Cryptographic Functionality ............................................................ 9
  4.1 Critical Security Parameters and Public Keys ................................................................ 13
5 Roles, Authentication and Services ....................................................................................... 16
6 Self-test ..................................................................................................................................... 18
7 Operational Environment ....................................................................................................... 20
8 Mitigation of other Attacks .................................................................................................... 21
Appendix A Installation and Usage Guidance ......................................................................... 22
Appendix B Controlled Distribution File Fingerprint ............................................................... 25
Appendix C Compilers ............................................................................................................. 27
1 Introduction
This document is the non-proprietary security policy for the Intel 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\(^1\)) 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:

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

\( Table 1 – Security Level of Security Requirements \)

The Module’s software version for this validation is 2.0.5 and 2.0.8.

The Dual EC DRBG algorithm shall not be used in the FIPS Approved mode of operation

\(^1\) Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.
Figure 1 – Module Block Diagram
2 Tested Configurations

<table>
<thead>
<tr>
<th>#</th>
<th>Operational Environment</th>
<th>Processor</th>
<th>Optimizations (Target)</th>
<th>EC</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linux 3.10 64bit</td>
<td>Intel Xeon (x86)</td>
<td>None</td>
<td>BKP</td>
<td>U1</td>
</tr>
<tr>
<td>2</td>
<td>Linux 3.10 64bit on VMware ESXi 6.00</td>
<td>Intel Xeon (x86)</td>
<td>None</td>
<td>BKP</td>
<td>U1</td>
</tr>
<tr>
<td>3</td>
<td>Linux 3.10 64bit</td>
<td>Intel Xeon (x86)</td>
<td>AES-NI</td>
<td>BKP</td>
<td>U2</td>
</tr>
<tr>
<td>4</td>
<td>Linux 3.10 64bit on VMware ESXi 6.00</td>
<td>Intel Xeon (x86)</td>
<td>AES-NI</td>
<td>BKP</td>
<td>U2</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Logical interface type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control input</td>
<td>API entry point and corresponding stack parameters</td>
</tr>
<tr>
<td>Data input</td>
<td>API entry point data input stack parameters</td>
</tr>
<tr>
<td>Status output</td>
<td>API entry point return values and status stack parameters</td>
</tr>
<tr>
<td>Data output</td>
<td>API entry point data output stack parameters</td>
</tr>
</tbody>
</table>

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).
## 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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Algorithm</th>
<th>Options</th>
<th>Cert #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Number Generation;</td>
<td>[SP 800-90] DRBG(^2)</td>
<td>Prediction resistance supported for all variations</td>
<td>1092, 1093</td>
</tr>
<tr>
<td>Symmetric key Generation</td>
<td>[SP 800-90] DRBG(^2)</td>
<td>Hash DRBG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SP 800-90] DRBG(^2)</td>
<td>HMAC DRBG, no reseed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SP 800-90] DRBG(^2)</td>
<td>CTR DRBG (AES), no derivation function</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SP 800-57]</td>
<td>3-Key TDES TECB, TCBC, TCFB, TOFB; CMAC generate and verify</td>
<td>2119, 2120</td>
</tr>
<tr>
<td>Encryption, Decryption and</td>
<td>[FIPS 197] AES</td>
<td>3-Key TDES TECB, TCBC, TCFB, TOFB; CMAC generate and verify</td>
<td></td>
</tr>
<tr>
<td>CMAC</td>
<td>[SP 800-38B] CMAC</td>
<td>128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SP 800-38C] CCM</td>
<td>MAC; GCM; CMAC generate and verify</td>
<td>3848, 3849</td>
</tr>
<tr>
<td></td>
<td>[SP 800-38D] GCM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SP 800-38E] XTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message Digests</td>
<td>[FIPS 180-3]</td>
<td>SHA-1, SHA-2 (224, 256, 384, 512)</td>
<td>3170, 3171</td>
</tr>
<tr>
<td>Keyed Hash</td>
<td>[FIPS 198] HMAC</td>
<td>SHA-1, SHA-2 (224, 256, 384, 512)</td>
<td>2496, 2497</td>
</tr>
</tbody>
</table>

\(^2\) For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP800-90] and [SP800-57].
## Digital Signature and Asymmetric Key Generation

<table>
<thead>
<tr>
<th>[FIPS 186-2] RSA</th>
<th>GenKey9.31 (2048/3072/4096)</th>
<th>1966</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SigGen9.31, SigGenPSS (4096 with SHA-256, 384, 512)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigGenPKCS1.5 (4096 with SHA-224, 256, 384, 512)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigVer9.31 (1024/1536/2048/3072/4096 with SHA-1, 256, 384, 512)</td>
<td>1965, 1966</td>
</tr>
<tr>
<td></td>
<td>SigVerPKCS1.5 (1024/1536/2048/3072,4096 with SHA-1, 224, 256, 384, 512) note: 1024 on CAVP cert 1965 only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigVerPSS (1024/1536/2048/3072,4096 with SHA-1, 224, 256, 384, 512)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[FIPS 186-4] RSA</th>
<th>SigGen9.31 (2048/3072 with SHA-256, 384, 512)</th>
<th>1965</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SigGenPSS (2048/3072 with SHA-224, 256, 384, 512)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigGenPKCS1.5 (2048/3072 with SHA-224, 256, 384, 512)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigGen9.31 (2048/3072 with SHA-1, 256, 384, 512) note: SHA-1 affirmed for use with protocols only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigGenPSS (2048/3072 with SHA-1, 224, 256, 384, 512) note: SHA-1 affirmed for use with protocols only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigGenPKCS1.5 (2048/3072 with SHA-1, 224, 256, 384, 512) note: SHA-1 affirmed for use with protocols only</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[FIPS 186-4] DSA</th>
<th>PQG Gen (2048,224 with SHA-224, 256, 384, 512; 2048,256 with SHA-256, 384, 512; 3072,256 with SHA-256, 384, 512) *note: SHA-512 on CAVP cert 1052 only</th>
<th>1051, 1052</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PQG Ver (1024, 160 with SHA-1, 224, 256, 384, 512; 2048, 224 with SHA-224, 256, 384, 512; 2048, 256 with SHA-256, 384, 512; 3072,256 with SHA-256, 384, 512)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Key Pair Gen (2048,224; 2048,256; 3072,256)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sig Gen (2048,224 with SHA-1, 224, 256, 384, 512; 2048,256 with SHA-1, 224, 256, 384, 512; 3072,256 with SHA-1, 224, 256, 384, 512) note: SHA-1 affirmed for use with protocols only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SigVer (1024/2048/3072 with SHA-1, 224, 256, 384, 512)</td>
<td></td>
</tr>
</tbody>
</table>

*note: SHA-512 on CAVP cert 1052 only*
| PKV: CURVES( ALL-P ALL-K ALL-B ) | |
| SigGen: CURVES( 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-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-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) ) SIG(gen) with SHA-1 affirmed for use with protocols only. | 832 |
| 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) ) | 832 |
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").

<table>
<thead>
<tr>
<th>Category</th>
<th>Algorithm</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Agreement</td>
<td>EC DH</td>
<td>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.</td>
</tr>
<tr>
<td>Key Encryption, Decryption</td>
<td>RSA</td>
<td>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</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Algorithm</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Number Generation;</td>
<td>[SP 800-90] DRBG</td>
<td>Dual EC DRBG</td>
</tr>
<tr>
<td>Symmetric key generation</td>
<td>[ANS X9.31] RNG</td>
<td>AES 128/192/256</td>
</tr>
<tr>
<td>Digital Signature and</td>
<td>[FIPS 186-2] RSA</td>
<td>GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS (1024/1536 with all SHA sizes, 2048/3072/4096 with SHA-1)</td>
</tr>
<tr>
<td>Asymmetric Key Generation</td>
<td>[FIPS 186-2] DSA</td>
<td>PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)</td>
</tr>
<tr>
<td></td>
<td>[FIPS 186-4] DSA</td>
<td>PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)</td>
</tr>
</tbody>
</table>
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(), 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.

<table>
<thead>
<tr>
<th>CSP Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA SGK</td>
<td>RSA (1024 to 16384 bits) signature generation key</td>
</tr>
<tr>
<td>RSA KDK</td>
<td>RSA (1024 to 16384 bits) key decryption (private key transport) key</td>
</tr>
<tr>
<td>DSA SGK</td>
<td>[FIPS 186-4] DSA (1024/2048/3072) signature generation key or [FIPS 186-2] DSA (1024) signature generation key</td>
</tr>
<tr>
<td>ECDSA SGK</td>
<td>ECDSA (All NIST defined B, K, and P curves) signature generation key</td>
</tr>
<tr>
<td>EC DH Private</td>
<td>EC DH (All NIST defined B, K, and P curves) private key agreement key.</td>
</tr>
</tbody>
</table>

---

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

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 4a. The calling application is responsible for storage of generated keys returned by the module.
**Entry:** All CSPs enter the Module’s logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.

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

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

Private and secret keys as well as seeds and entropy input are provided to the Module by the calling application, and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a nonoverlapping 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] Table 2 (Hash_DRBG, HMAC_DRBG), Table 3 (CTR_DRBG). This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.
5 Roles, Authentication and Services

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

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

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/2^{16}$, 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.

<table>
<thead>
<tr>
<th>Service</th>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>User, CO</td>
<td>Module initialization. Does not access CSPs.</td>
</tr>
<tr>
<td>Self-test</td>
<td>User, CO</td>
<td>Perform self tests (FIPS_selftest). Does not access CSPs.</td>
</tr>
<tr>
<td>Show status</td>
<td>User, CO</td>
<td>Functions that provide module status information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Version (as unsigned long or const char *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FIPS Mode (Boolean)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Does not access CSPs.</td>
</tr>
<tr>
<td>Zeroize</td>
<td>User, CO</td>
<td>Functions that destroy CSPs:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• fips_drbg_uninstantiate: for a given DRBG context, overwrites DRBG CSPs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRGB CSPs.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All other services automatically overwrite CSPs stored in allocated memory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stack cleanup is the responsibility of the calling application.</td>
</tr>
<tr>
<td>Random number generation</td>
<td>User, CO</td>
<td>Used for random number and symmetric key generation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Seed or reseed a DRBG instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Determine security strength of a DRBG instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Obtain random data</td>
</tr>
<tr>
<td>Service Type</td>
<td>User Access</td>
<td>CSP Access</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>Asymmetric key generation</td>
<td>User, CO</td>
<td>Uses and updates Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetric encrypt/decrypt</td>
<td>User, CO</td>
<td>Uses to encrypt or decrypt data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetric digest</td>
<td>User, CO</td>
<td>Uses to generate or verify data integrity with CMAC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message digest</td>
<td>User, CO</td>
<td>Uses to generate a SHA-1 or SHA-2 message digest.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyed Hash</td>
<td>User, CO</td>
<td>Uses to generate or verify data integrity with HMAC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key transports</td>
<td>User, CO</td>
<td>Uses to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key agreement</td>
<td>User, CO</td>
<td>Uses to perform key agreement primitives on behalf of the calling process (does not establish keys into the module).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital signature</td>
<td>User, CO</td>
<td>Uses to generate or verify RSA, DSA or ECDSA digital signatures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utility</td>
<td>User, CO</td>
<td>Miscellaneous helper functions. Does not access CSPs.</td>
</tr>
</tbody>
</table>

*Table 5 – Services and CSP Access*
6 Self-test

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Type</th>
<th>Test Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software  integrity</td>
<td>KAT</td>
<td>HMAC-SHA1</td>
</tr>
<tr>
<td>HMAC</td>
<td>KAT</td>
<td>One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512 Per IG 9.3, this testing covers SHA POST requirements.</td>
</tr>
<tr>
<td>AES</td>
<td>KAT</td>
<td>Separate encrypt and decrypt, ECB mode, 128 bit key length</td>
</tr>
<tr>
<td>AES GCM</td>
<td>KAT</td>
<td>Separate encrypt and decrypt, 192 key length</td>
</tr>
<tr>
<td>XTS-AES</td>
<td>KAT</td>
<td>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)</td>
</tr>
<tr>
<td>AES CMAC</td>
<td>KAT</td>
<td>Sign and verify CBC mode, 128, 192, 256 key lengths</td>
</tr>
<tr>
<td>TDES</td>
<td>KAT</td>
<td>Separate encrypt and decrypt, ECB mode, 3-Key</td>
</tr>
<tr>
<td>TDES CMAC</td>
<td>KAT</td>
<td>CMAC generate and verify, CBC mode, 3-Key</td>
</tr>
<tr>
<td>RSA</td>
<td>KAT</td>
<td>Sign and verify using 2048 bit key, SHA-256, PKCS#1</td>
</tr>
<tr>
<td>DSA</td>
<td>PCT</td>
<td>Sign and verify using 2048 bit key, SHA-384</td>
</tr>
<tr>
<td>DRBG</td>
<td>KAT</td>
<td>CTR_DRBG: AES, 256 bit with and without derivation function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HASH_DRBG: SHA256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HMAC_DRBG: SHA256</td>
</tr>
<tr>
<td>ECDSA</td>
<td>PCT</td>
<td>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).</td>
</tr>
<tr>
<td>ECC CDH</td>
<td>KAT</td>
<td>Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6</td>
</tr>
</tbody>
</table>

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() 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() 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.

---

4 FIPS_mode_set() calls Module function FIPS_module_mode_set()
The Module also implements the following conditional tests:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRBG</td>
<td>Tested as required by [SP800-90] Section 11</td>
</tr>
<tr>
<td>DRBG</td>
<td>FIPS 140-2 continuous test for stuck fault</td>
</tr>
<tr>
<td>DSA</td>
<td>Pairwise consistency test on each generation of a key pair</td>
</tr>
<tr>
<td>ECDSA</td>
<td>Pairwise consistency test on each generation of a key pair</td>
</tr>
<tr>
<td>RSA</td>
<td>Pairwise consistency test on each generation of a key pair</td>
</tr>
</tbody>
</table>

*Table 6b - Conditional Tests*

In the event of a DRBG self-test failure the calling application must uninitialize and reinstantiate 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 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-2.0.8.tar.gz (all NIST defined curves as listed in Table 2 with the EC column marked "BKP") or openssl-fips-ecp-2.0.8.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
```

Installation instructions

1. Download and copy the distribution file to the build system. These files can be downloaded from [http://www.openssl.org/source/](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 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. Alternatively, a copy of the distribution on physical media can be obtained from OSF\(^5\).

3. Unpack the distribution
   ```
   gunzip -c openssl-fips-2.0.8.tar.gz | tar xf
   cd openssl-fips-2.0.8 or
   ```

---

\(^5\) For some prospective users the acquisition, installation, and configuration of a suitable FIPS 140-2 validated product may not be convenient. OSF will on request mail a CD containing the source code distribution, via USPS or international post. A distribution file received by that means need not be verified by a FIPS 140-2 validated implementation of HMAC-SHA-1. For instructions on requesting this CD see [http://opensslfoundation.com/fips/verify.html](http://opensslfoundation.com/fips/verify.html).
gunzip -c openssl-fips-ecp-2.0.8.tar.gz | tar xf cd openssl-fips-ecp-2.0.8

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() 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() 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 this module there are two possible optimization levels:

1. No optimization (plain C)
2. AES-NI+PCLMULQDQ+SSSE3 optimization

For more information, see:

6 FIPS_mode_set() calls the Module function FIPS_module_mode_set()
• http://www.intel.com/supportprocessors/sb/CS-030123.htm?wapkw=sse2
Appendix B Controlled Distribution File Fingerprint

The Intel OpenSSL FIPS Object Module v2.0.8 consists of the Intel 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-2.0.8.tar.gz with HMAC-SHA-1 digest of

7f486fbb598f3247ab9db10c1308f1c19f384671


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-2.0.8.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-2.0.8.tar.gz with HMAC-SHA-1 digest of

7a5f40ef8ebe959372d16e26391fcd23689209b

located at http://www.openssl.org/source/openssl-fips-ecp-2.0.8.tar.gz. Note this is from the previous revision of the FIPS Object Module as no modifications relevant to NIST prime curves only were introduced in revision 2.0.8.

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

The arbitrary 16 byte key of:

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

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

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

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

openssl-fips-ecp-2.0.5.tar.gz
URL: http://www.openssl.org/source/openssl-fips-ecp-2.0.5.tar.gz
Digest: 148e4e127ffe1df80c0ed61bae35b07ec7b7b36
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>
<thead>
<tr>
<th>#</th>
<th>Operational Environment</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Linux 3.10 64bit</td>
<td>gcc 4.8.3</td>
</tr>
<tr>
<td>2</td>
<td>Linux 3.10 64bit on VMware ESXi 6.00</td>
<td>gcc 4.8.3</td>
</tr>
<tr>
<td>3</td>
<td>Linux 3.10 64bit</td>
<td>gcc 4.8.3</td>
</tr>
<tr>
<td>4</td>
<td>Linux 3.10 64bit on VMware ESXi 6.00</td>
<td>gcc 4.8.3</td>
</tr>
</tbody>
</table>