OpenSSL FIPS
Object Module
Version 1.1.2
By the
Open Source Software Institute
http://www.oss-institute.org/

OpenSSL FIPS 140-2 Security Policy

Version 1.1.2

January 29, 2008

Copyright Notice

Sponsored by:

The Defense Medical Logistics Program

The Hewlett-Packard Company

Acknowledgments

The principal author of this document is:

Steve Marquess	301-619-3933
Consultant	Steve.Marquess@det.amedd.army.mil
DMLSS Program	marquess@veridicalsystems.com
JMLFDC	
623 Porter Street	
Ft. Detrick, MD 21702	

working under contract to the Defense Medical Logistics Standard Support program. Other significant contributors to this document are, in alphabetical order:

Gary Gross	HP Corporation
Ben Laurie	OpenSSL
Peter Sargent	PreVal Specialist, Inc
John Weathersby	Open Source Software Institute

Thanks also to Andy Polyakov, Joel I. Kirch and Rick Pearce for their reviews of earlier versions of this document.

The OpenSSL FIPS 140-2 validation effort was sponsored by the Defense Medical Logistics Standard Support program, part of the TRICARE Management Activity, Office of the Assistant Secretary of Defense for Health Affairs, and co-sponsored by the Hewlett-Packard Company. For further information contact:

Debra Bonner	707-575-9771
Director of Operations	Debra.Bonner@tma.osd.mil
DMLSS Program Management Office	http://ww.tricare.osd.mil/dmlss/
5109 Leesburg Pike, Suite 908	
Falls Church, VA 22041	
Gary Gross	408-447-6966 office
Security Evaluations Program Manager	650-380-1327 cell
Hewlett-Packard	gary.gross@hp.com
1501 Page Mill Road	http://hp.com/
Palo Alto, CA 94304	

The Open Source Software Institute (OSSI) serves as the "vendor" for this validation. Project management coordination for this effort was provided by the OSSI:

John Weathersby	601-427-0152 office
Executive Director	601-818-7161 cell
Open Source Software Institute	601-427-0156 fax
Administrative Office	jmw@oss-institute.org
P.O. Box 547	http://oss-institute.org/
Oxford, MS 38655	

The initial OpenSSL FIPS 140-2 software development work was performed by:

Ben Laurie	+44 (20) 8735 0686 office
17 Perryn Road	+44 (20) 8735 0689 fax
London	ben@algroup.co.uk
W3 7LR	
UK	

with subsequent significant contributions by:

Stephen Henson	
4 Monaco Place,	shenson@drh-consultancy.co.uk
Westlands, Newcastle-under-Lyme	
Staffordshire. ST5 2QT.	

England, United Kingdom http://www.drh-consultancy.co.uk/

Andy Polyakov
Chalmers University of Technology appro@fy.chalmers.se
SE-412 96 Gothenburg
Sweden

in coordination with the OpenSSL Team at www.openssl.org, in particular:

Richard Levitte levitte@stacken.kth.se

Validation testing was performed by The DOMUS IT Security Laboratory. For information on validation or revalidations of software contact:

Christian Brych 613-726-5091 office
Lab Director 613-867-1241 cell
DOMUS IT Security Laboratory cbrych@nuvo.com
2650 Queensview Drive http://www.domusitsl.com/
Suite 100
Ottawa, Ontario
K2B 8H6

Assistance in preparation of the initial FIPS 140 documentation was provided by:

Peter C. Sargent 443-742-4430
PreVal Specialist, Inc. preval@att.net
214 Kennedy Dr.
Severna Park, MD 21146

Revision History

Date	Description
2008-01-29	Demoted DSA to non-approved.
2007-12-04	Added new algorithm certificate numbers.
2007-11-14	Updated to v1.1.2 from v1.1.1 to reflect incorporation of a patch for addressing the CVS-2007-5502 PRNG seeding vulnerability and continuous PRNG self-test bug.
2007-01-29	Table 2.3: update RSA certificate number; Appendix B: update distribution file HMAC-SHA-1 digest.
2007-01-24	4.5.2: Reword “Pair-wise Consistency Test” paragraph.
2007—01-12	Change every occurrence of “1.1” to “1.1.1” (thus changing the name of the source distribution file, although the contents of that file are unchanged).
2007-01-10	1.1,1.3: change module name; 2.3: clarify example use of external keys; Table 2.3: remove superfluous line; Table 3.3: add missing Key Establishment algorithms; 4: replace “same as” with “bitwise identical”; 4.1: clarify input/output of keys; 4.5.2: elaborate on pair-wise consistency test, remove continuous random number generator test.
2006-09-07	2: added commentary to DES footnote; 2.5: deprecation of DES; 5.2: added commentary on key wrapping.
2006-08-31	2,2.2,2.3: Refer to RSA “key wrapping”, moved from Table 2.3 to 2.4. Added RSA key strength statement. 2.5: Added sentence noting no FIPS mode until FPS mode is initialized. 5.2: changed “the key database” to “a key database”. C.3: Added footnote on SHA1-HMAC digest determination.
2006-07-20	Added new algorithm certificate numbers, changed DES references to non-Approved (replacing May 19 2007 caveat)
2006-06-16	Removed pre-file hashes from Appendix B, changed distribution hash to match new distribution file openssl-fips-1.1.tar.gz which rearranges object code from the previous distribution.
2006-04-13	Changed distribution file pathname from “OpenSSL-fips-1.0.tar.gz” to “OpenSSLfips1.0.tar.gz” to match certificate; Section 4.3, Appendix B, Appendix C.
2006-03-24	Additional commentary on source files in Appendix B
2006-03-13	Final draft

Table of Contents

Table of Contents

TABLE OF FIGURES

Table of Contents

1. Introduction

This document is the non-proprietary FIPS 140-2 security policy for the OpenSSL FIPS software object module to meet FIPS 140-2 level 1 requirements. This Security Policy details the secure operation of the *OpenSSL FIPS Object Module v1.1.2 (OpenSSL FIPS) by the Open Source Software Institute (oss-institute.org)* as required in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

1.1 Audience

This document is required as a part of the FIPS 140-2 validation process. It describes the OpenSSL FIPS Object Module in relation to FIPS 140-2 requirements. The companion document *OpenSSL FIPS 140-2 User Guide* (Reference 14) is a technical reference for developers using, and system administrators installing, the OpenSSL FIPS software, for use in risk assessment reviews by security auditors, and as a summary and overview for program managers.

1.2 Document Organization

This Security Policy document is one part of the complete FIPS 140-2 Submission Package. The Submission Package contains:

- Non-proprietary FIPS 140-2 Security Policy: this document
- Algorithm Certificates: Listed in Table 3.3 of this document
- Design specification and functional specification of OpenSSL software: included with the standard OpenSSL software distributions and available at http://openssl.org/. Also see Reference 11 and Reference 14
- Crypto Officer and User Guidance documentation: subsumed in this document
- Finite State Machine: Appendix A of this document
- Vendor Evidence Document: *OpenSSL FIPS 140 Vendor Evidence Document*, v1.0
- Source Code Listing: included in the OpenSSL source distributions

This Security Policy document is available online at the NIST Cryptographic Module Validation website, http://csrc.nist.gov/cryptval/140-1/1401val2006.htm, and is also included in the OpenSSL distributions.

This document outlines the functionality provided by the module and gives high-level details on the means by which the module satisfies FIPS 140-2 requirements.

1.3 References

For more information on the OpenSSL FIPS Object Module please visit http://www.oss-institute.org/. For more information on the OpenSSL project see http://openssl.org/. For more information on NIST and the cryptographic module validation program, please visit http://csrc.nist.gov/cryptval/.

1.4 Relationship to the OpenSSL API

The FIPS object module is designed for use in conjunction with the separate API libraries provided by the OpenSSL project. Applications linked with the FIPS object module and with the separate OpenSSL libraries can use the FIPS validated cryptographic functions of the FIPS object module and the high level support and encapsulation features of OpenSSL.

2. Module Specification

For the purposes of FIPS 140-2 validation the OpenSSL FIPS Object Module v1.1.2 is defined as a specific discrete unit of binary object code (the "*FIPS Object Module*") generated from a specific set and revision level of C language source files embedded within a source distribution. These platform portable source files are compiled to create this object code in an isolated and separated form that is used to provide a cryptographic API (Application Programming Interface) to external applications. The term Module elsewhere in this document refers to this OpenSSL FIPS Object Module object code ("*FIPS Object Module*").

The source code may be used to generate Modules for use on a wide variety of hardware and operating system platforms. The Module provides an API for invocation of FIPS approved cryptographic functions from calling applications. The Module is designed for use in conjunction with separate libraries provided by the OpenSSL project.

The Module was tested by the FIPS 140-2 Cryptographic Module Testing (CMT) laboratory for a specific test platform. The OpenSSL FIPS Object Module, when generated from the identical unmodified source code, is "Vendor Affirmed" to be FIPS 140-2 compliant when running on other supported computer systems provided the conditions described in §4, "Operational Environment", are met. On any platform the Module generated from the Module source code is *not* validated if that source code is modified in any way.

The Module was designed and implemented to meet FIPS 140-2 requirements. As such, there are no special steps, other than building the binary FIPS Object Module file from the OpenSSL FIPS approved and HMAC-SHA-1 verified source code, and loading and initializing with a runtime executable application, required to ensure FIPS 140-2 compliant operation of the Module. This process of generating the FIPS Object Module and the runtime application from source code is the same for all platforms and is documented in the *User Guide*, Reference 14.

The Module provides confidentiality, integrity, and message digest services. It natively supports the following algorithms: DES[1], Triple DES, AES, RSA (for digital signatures and key wrapping), DH, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, and HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-384, HMAC-SHA-512. OpenSSL FIPS performs ANSI X9.31 compliant pseudo-random number generation.

2.1 The FIPS Object Module

The generation of the Module from source code introduces some new challenges in complying with the requirements of the FIPS 140-2 standard.

[1]Non-Approved algorithm. Note DES is provided for backward compatibility with legacy applications and must not be used in FIPS 140-2 compliant operation. The use of DES is not allowed in the Approved FIPS mode of operation, and its use will result in the module operating in a non-Approved state.

The concept of the FIPS Object Module and the documented process for creating it from source code were developed to satisfy those requirements. The single integrity test of a validated binary executable is replaced by a chain of integrity tests beginning with the source files used for the CMVP validation testing, and ending with the runtime executable application generated from that source code.

The purpose of the FIPS Object Module is to maintain the separate and distinct identity of the object code within the logical boundary as required by FIPS 140-2. Since the Module is generated from source code and not distributed as pre-built binary code, this separation must be maintained for all phases of the software from source code to intermediate object code to executing memory mapped runtime code. The process of generating the binary FIPS Object Module must assure that the same unmodified source is used with the same build-time options, and that unmodified object code is used in the generation of runtime executable applications.

The concept of the FIPS Object Module and the documented process for creating it from source code were developed to address the following specific areas of concern:

- Integrity of Source Code
- Integrity of Object Code
- Integrity of Executable Code
- Exclusivity of Integrity Tests

2.1.1 Integrity of Source Code

The integrity of the sequestered source files is protected by a HMAC-SHA-1 digest of the entire source distribution. This digest is published in this document (see Appendix B), and should be checked manually as specified in Appendix C. Since this document is posted on the CMVP validation list website the digests recorded there cannot be arbitrarily changed.

2.1.2 Integrity of Object Code

Once generated the FIPS Object Module file is protected by a HMAC-SHA-1 digest. This digest protects only object code belonging to the Module, and is verified at the time the FIPS Object Module is incorporated into an executable application.

Fixed Object Code Order and Content

The presence and relative order of the generated object code in the FIPS Object Module must be fixed and invariant.

The sequestered source files are used to generate the FIPS Object Module. The instructions in this document forbid user specified build-time options when building the FIPS Object Module, hence all the object code derived from the sequestered source code is always included in the FIPS Object Module.

The usual compilation and linking process does not care about the relative order of individual object modules, and can omit object code not needed to satisfy link references. When generating or linking against the validated binary code we are required to prevent any such omission or rearrangement of the object code derived from the sequestered source files.

The FIPS Object Module is created by compiling all the sequestered source code into a single monolithic object module[2], the FIPS Object Module. The object code within the FIPS Object Module cannot be removed, replaced, rearranged, or extended by the standard tools used for the management of software libraries or the creation of executable application code. All subsequent references to this monolithic object module will preserve the relative order, and presence, of the original object code. The FIPS Object Module is not a static library. It may be incorporated into shared library files or runtime executable application files, but in any event can only be incorporated intact and in its entirety.

Isolation of Object Code

The object code generated by the compilation of these sequestered source files is carefully isolated from all other OpenSSL and application object code. This isolation is accomplished by collecting all of this sequestered object code into a single discrete unit of object code. We refer to this discrete unit as the FIPS Object Module, which resides in the FIPS Object Module file, `fipscanister.o`.

The FIPS Object Module contains only the object code belonging to the Module. The integrity of the FIPS Object Module file is protected by a HMAC-SHA-1 digest that is calculated over the file at the time it is created, and stored in a separate file, `fipscanister.o.sha1`, that is installed along with `fipscanister.o`. This digest is checked whenever an application is linked against the FIPS Object Module file.

2.1.3 Integrity of Executable Code

The design of the FIPS Object Module includes the definition of reference points within the object code that are used to define the object code to be protected by the runtime integrity test.

At application link time a HMAC-SHA-1 digest of the memory mapped object code within the FIPS Object Module is created and stored in the FIPS Object Module. This digest is calculated entirely within the confines of the FIPS Object Module and so will never include extraneous object code. This digest is independent of other object code, data, memory or file areas, etc. that may adjoin, surround, embed, link to, or otherwise reference the FIPS Object Module.

In order to test the integrity of the FIPS Object Module, an integrity test is required over the object code within the logical boundary only. This requirement is satisfied with the runtime in-core integrity test over object code within the FIPS Object Module that carefully excludes any non-sequestered object code from the digest calculation and verification.

[2]Here the technical term *object module* is used in the context of software development and computer science, not FIPS 140-2.

Note this integrity testing technique is conceptually similar to the case of firmware integrity testing: not only is non-sequestered code omitted from the digest, but also non-executing ancillary data specific to the particular executable format. Only the actual machine code byte sequence directly executed by the CPU and the actual data referred to in the course of execution are included in the digest; i.e. only what really matters to the general purpose computer at runtime, pure code as with firmware.

2.1.4 Exclusivity of Integrity Tests

Each of the checks in the chain of integrity tests is exclusive to components specific to the Module.

- The HMAC-SHA-1 digests that protect the sequestered source files are specific to only those files.
- The HMAC-SHA-1 digest that protects the FIPS Object Module in the FIPS Object Module file is specific to only that file which contains only object code generated from the sequestered source files.
- The HMAC-SHA-1 digest that protects the FIPS Object Module in a executable module is specific to object code within the Module.

2.2 Ports and Interfaces

For the purposes of this FIPS-140-2 validation the Module is considered a multi-chip standalone module. Although the Module is software the physical embodiment is a general purpose computer that consists of multiple components, considered to be a multi-chip standalone module by FIPS-140-2.

The logical cryptographic boundary for the Module is the discrete contiguous block of object code (the FIPS Object Module) containing the machine instructions and data generated from the OpenSSL FIPS source, as used by the calling application. The physical cryptographic boundary contains the general purpose computing hardware of the system executing the application. This system hardware includes the central processing unit(s), cache and main memory (RAM), system bus, and peripherals including disk drives and other permanent mass storage devices, network interface cards, keyboard and console and any terminal devices.

The Module provides a logical interface via an Application Programming Interface (API). This logical interface exposes services that applications may utilize directly or extend to add support for new data sources or protocols. The API provides functions that may be called by the referencing application.

The API interface provided by the Module is mapped onto the FIPS 140-2 logical interfaces: data input, data output, control input, and status output. Each of the FIPS 140-2 logical interfaces relates to the Module's callable interface, as follows:

- Data input - input parameters to all functions that accept input from Crypto-Officer or User entities
- Data output - output parameters from all functions that return data as arguments or return values from Crypto-Officer or User entities
- Control input – all API function input into the module by the Crypto-Officer and User entities
- Status output - information returned via exceptions (return/exit codes) to Crypto-Officer or User entities

The API function specifications are included in the OpenSSL project documentation which covers both FIPS and non-FIPS functions.

2.3 Approved Cryptographic Algorithms

The Module supports the following FIPS-approved cryptographic algorithms:
Triple DES, Rivest Shamir Adleman (RSA) PKCS #1 digital signatures, Diffie-Hellman, Advanced Encryption Standard (AES – FIPS 197), Secure Hashing Algorithm (SHA-1, SHA-2 - FIPS 180-2), and Keyed-Hash Message Authentication Code (HMAC - FIPS 198). Triple DES may be used, for example, to protect the contents of a key database when used with a mode of operation that requires an initialization vector (IV), i.e. Electronic Codebook (ECB) mode is not permitted for key database protection purposes.

The Module performs ANSI X9.31 pseudo-random number generation.

Approved Algorithms				
Algorithm Type	**Algorithm**	**Standard**	**FIPS Validation Certificate #**	**Use**
asymmetric keys	RSA	ALG[ANSIX9.31]; SIG(gen); SIG(ver); ALG[RSASSA-PKCS1_V1_5]; SIG(gen); SIG(ver) ALG[RSASSA-PSS]; SIG(gen); SIG(ver);	**#310**	sign and verify operations
				sign and verify operations
symmetric key	3DES - CBC, CFB8, CFB64, ECB, OFB modes	FIPS 46-3	**#613**	encrypt/decrypt operations
	AES - CBC, CFB8, CFB128, ECB, OFB each with 128, 192, or 256 bit keys	FIPS 197	**#668**	
HMAC	HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	FIPS 198	**#352**	module integrity code integrity message integrity
hashing	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	FIPS 180-2	**#701**	hashing
RNG	ANSI X9.31	ANSI X9.31	**#387**	random number generation

Table 2.3 - Approved Cryptographic Algorithms

The RSA implementation includes known answer tests at startup which consist of a sign and verify operation and pair-wise consistency test which is also a sign and verify operation when keys are generated.

2.4 Non-Approved Cryptographic Algorithms

With the exception of Diffie-Hellman (DH), all non-FIPS algorithms (those algorithms not listed in Table 2.2) are not included in the Module. As a convenience for application developers the use of non-FIPS algorithms in the separate OpenSSL library is disabled in FIPS mode.

Diffie-Hellman (key agreement, key establishment) methodology supports 80 bits to 256 bits of encryption strength.

RSA (key wrapping; key establishment) methodology provides between 80 and 150 bits of encryption strength)

Algorithm Type	Algorithm	Standard	FIPS Approval #	Use
key agreement	DH	N/A	none	key agreement
symmetric	DES	FIPS 46-3	none	encrypt, decrypt
asymmetric	RSA	ANSI X9.31	none	key wrapping
asymmetric	DSA[3]	FIPS 186-2	#250	signing, signature verification

Table 2.4 - Non-Approved Cryptographic Algorithms

Note that FIPS 140-2 does not strictly require that all non-FIPS approved algorithms be disabled, and other validated cryptographic libraries are available which support non-FIPS approved algorithms. However, since the Module is intended for general use with the OpenSSL library by applications that will not require separate FIPS 140-2 validations, this choice was made as a deliberate design decision to maintain confidence that those applications are operating in Approved mode when using the FIPS Object Module in FIPS mode in conjunction with the OpenSSL library.

2.5 Approved Mode of Operation

A single initialization call, `FIPS_mode_set()`, is required to initialize the Module for operation in the FIPS 140-2 Approved mode, referred to herein as "FIPS mode". When the Module is in FIPS mode all security functions and cryptographic algorithms are performed in Approved mode, with the exception of DES which is not allowed in the Approved FIPS mode of operation, and its use will result in the module operating in a non-Approved state. Use of the `FIPS_mode_set()` function call is described in the *User Guide*, Reference 14. The Module is not in FIPS mode until FIPS mode is initialized.

The FIPS mode initialization is performed when the application invokes the `FIPS_mode_set()` call. Prior to this invocation the Module is uninitialized with the internal global flag `FIPS_mode` set to FALSE indicating non-FIPS mode by default.

[3]Note that the DSA implementation is no longer compliant for use in FIPS mode because a Pairwise Consistency Test is required on a minimum 1024-bit modulus size, and that test is currently performed on a 512-bit modulus size.

The `FIPS_mode_set()` function verifies the integrity of the runtime executable using a HMAC-SHA-1 digest computed at build time. If this computed HMAC-SHA-1 digest matches the stored known digest then the power-up self-test, consisting of the algorithm specific Pairwise Consistency and Known Answer tests, is performed. If any component of the power-up self-test fails the internal global error flag `FIPS_selftest_fail` is set to prevent subsequent invocation of any cryptographic function calls. If all components of the power-up self-test are successful then `FIPS_mode_set()` sets the `FIPS_mode` flag to TRUE and the Module is in FIPS mode.

2.6 Test Environment

The Module was tested by the FIPS 140-2 CMT laboratory on the following computer system:

- IBM NetVista Pentium 4 (x86 architecture) running the SUSE Linux 10.2 operating system and gcc v4.1.2

The OpenSSL FIPS Object Module, when generated from the identical unmodified source code, is "Vendor Affirmed" to be FIPS 140-2 compliant when running on other supported computer systems provided the conditions described in IG G.5 (Reference 3), are met. On any platform the Module generated from the Module source code (the source files identified in Appendix B) is *not* validated if that source code is modified in any way.

3. Roles, Services and Authentication

3.1 Roles and Services

The User and Crypto Officer roles are implicitly assumed by any entity that can access services implemented in the Module. In addition the Crypto Officer role can install and initialize the Module; this role is implicitly entered when installing the Module or performing system administration functions on the host operating system:

Role	Authorized Services
User role	All services except installation and initialization
Crypto Officer role	All services including installation and initialization

Table 3.1- Services Authorized for Roles

The Module meets the FIPS 140-2 level 1 requirements for Roles and Services for User and Crypto Officer roles. As a library and as allowed by FIPS 140-2 the Module does not support user identification or authentication for those roles.

3.2 Authentication

The Module does not provide identification or authentication mechanisms that would distinguish between the two supported roles. These roles are implicitly assumed by the services that are accessed, and can be differentiated by assigning module installation and configuration services to the Crypto Officer. Only a single user in a specific role may access Module services at the same time.

3.3 Authorized Services

The services provided by the Module are listed in the following table. All services may be performed in both User and Crypto Officer roles except for the Module installation and Initialization services which may only be performed by in the Crypto Officer role:

Roles	Service	Critical Security Parameters	Algorithm	API Functions	Access
User, Crypto Officer	Symmetric Encryption/ Decryption	symmetric key	AES 3DES (2 key)[4] 3DES (3 key)[5]	AES_cbc_encrypt AES_decrypt AES_encrypt AES_set_decrypt_key AES_set_encrypt_key AES_Td AES_Te _x86_AES_decrypt _x86_AES_encrypt DES_check_key_parity DES_decrypt3 DES_ede3_cbc_encrypt DES_encrypt1 DES_encrypt2 DES_encrypt3 DES_is_weak_key DES_key_sched DES_ncbc_encrypt DES_set_key DES_set_key_checked DES_set_key_unchecked DES_set_odd_parity	Read Write Execute
User, Crypto Officer	Message Digest	none HMAC key	SHA-1, SHA-2 HMAC	SHA1 sha1_block_asm_data_order sha1_block_asm_host_order SHA1_Final SHA1_Init SHA1_Transform SHA1_Update SHA224 SHA224_Final SHA224_Init SHA224_Update SHA256 SHA256_block_data_order SHA256_block_host_oeder SHA256_Final SHA256_Init SHA256_Transform SHA256_Update SHA384 SHA384_Final SHA358_Init SHA384_Update SHA512 SHA512_Final SHA512_Init SHA512_Transform SHA512_Update	Read Write Execute

[4] *DES_encrypt2* is used exclusively with 2 Key Triple DES for encryption and decryption.
[5] *DES_decrypt3*, *DES_encrypt3* are used exclusively with 3DES for decryption and encryption respectively.

Roles	Service	Critical Security Parameters	Algorithm	API Functions	Access
User, Crypto Officer	Random Number Generation	seed key	ANSI X9.31	FIPS_rand_method FIPS_rand_seed FIPS_rand_seeded FIPS_set_prng_key	Read Write Execute
User, Crypto Officer	Show Status	none	N/A	FIPS_mode FIPS_mode_set	Execute
Crypto Officer	Module Initialization	none	N/A	FIPS_check_incore_fingerprint FIPS_check_rsa FIPS_incore_fingerprint FIPS_mode_set FIPS_rand_check ERR_load_FIPS_strings	Read Execute
User, Crypto Officer	Self Test (includes integrity, known answer, and pair-wise consistency tests)	none	N/A	FIPS_selftest FIPS_selftest_aes FIPS_selftest_des FIPS_selftest_failed FIPS_selftest_hmac FIPS_selftest_rng FIPS_selftest_rsa FIPS_selftest_sha1	Execute
User, Crypto Officer	Key Establishment	asymmetric public and private keys	RSA DH	RSA_generate_key RSA_PKCS1_SSLeay RSA_X931_derive RSA_X931_generate_key DH_check DH_compute_key DH_generate_key DH_generate_parameters DH_OpenSSL	Read Write Execute

Table 3.3 - Authorized Services

See the Vendor Evidence document, Appendix A, "API Entry points by Source File" for the correspondence between the API and the source files which implement that API.

3.4 Finite State Machine Model

The Module implements the finite state machine detailed in Appendix A.

4. Operational Environment

The FIPS Object Module is generated from source code available for use on a wide variety of computer hardware and operating system platforms. Applications referencing the FIPS Object Module run as processes under the control of the host system operating system.

Modern operating systems segregate running processes into virtual memory areas that are logically separated from all other processes by the operating system and CPU. The FIPS Object Module functions completely within the process space of the process which loads it. It does not communicate with any processes other than the one that loads it, and satisfies the FIPS 140-2 requirement for a single user mode of operation.

The FIPS Object Module was built from source and tested on specific hardware/software environments (See §2.6) . As stated in §G.5 of Reference 3 ("*Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program*") the Module maintains FIPS 140-2 validation on other hardware and operating systems provided that:

1. The source code that is compiled into the FIPS Object Module is bitwise identical to[6] the source code used for the validation testing, and is compiled in the same way.
2. The host operating system satisfies the rules of operation outlined in the following section, §4.1.

The CMVP allows the re-compilation of the software cryptographic module utilizing the unchanged source files specified in Appendix B with compilers different than the listed compilers that were used for validation testing. The validation status is maintained utilizing the different compilers without re-testing the newly compiled cryptographic module. However, the CMVP makes no statement as to the correct operation of the module utilizing compilers not listed in Appendix B.

The CMVP allows user porting of a validated software cryptographic module on an OS(s) and/or GPC(s) which were not included as part of the validation testing. The validation status is maintained on the new OS(s) and/or GPC without re-testing the cryptographic module on the new OS(s) and/or GPC(s). However, the CMVP makes no statement as to the correct operation of the module when executed on an OS(s) and/or GPC(s) not listed on the validation certificate.

The module validated by the CMVP and which assurance is provided is based as caveated on the validation certificate and when compiled with the reference compilers and operated on the reference operating systems annotated on the certificate. FIPS 140-2 IG G.5 and the above caveats allow re-compilation and porting but without CMVP assurance as to the correct operation of the module.

4.1 Rules of Operation

[6]Note the verification of the source distribution file with the HMAC-SHA-1 digest as described in Appendix B assures this bitwise identity.

1. The Module is initialized in the FIPS mode of operation using the `FIPS_mode_set()` function call.
2. The replacement or modification of the Module by unauthorized intruders is prohibited.
3. The Operating System enforces authentication method(s) to prevent unauthorized access to Module services
4. All Critical Security Parameters are verified as correct and are securely generated, stored, and destroyed.
5. All host system components that can contain sensitive cryptographic data (main memory, system bus, disk storage) must be located in a secure environment.
6. The referencing application accessing the Module runs in a separate virtual address space with a separate copy of the executable code.
7. The unauthorized reading, writing, or modification of the address space of the Module is prohibited.
8. The writable memory areas of the Module (data and stack segments) are accessible only by a single application so that the Module is in "single user" mode, i.e. only the one application has access to that instance of the Module.
9. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the Module.
10. Secret or private keys that are input to or output from an application must be input or output in encrypted form using a FIPS Approved algorithm. Note that keys exchanged between the application and the FIPS Object Module may not be encrypted.
11. The developer shall use entropy sources that contain at least 64 bits of entropy to seed the RNG as the module is not capable of detecting the randomness or quality of the seeding material provided.

4.2 Compatible Platforms

The Module is designed to run on a very wide range of hardware and software platforms as long as the conditions in IG G.5 (Reference 3) are met. Any such computing platform that meets the conditions listed above can be used to host a FIPS 140-2 validated Module generated in accordance with this Security Policy[7]. At the time the *OpenSSL FIPS Object Module v 1.1.2* was developed all platforms supported by the full OpenSSL distribution were covered by the FIPS validated source files included in the Module. However, successful compilation of the Module for all such platforms was not verified[8]. If any platform specific errors occur that can only be corrected by modification of the Module source files (Appendix B) then the Module will not be validated for that platform.

Note also that future releases of OpenSSL may add support for additional platforms requiring new platform specific source replacing parts of the current sequestered source, in which case the modified Module will not be validated for those new platforms.

[7]See §2.4 and §G.5 of Reference 3 ("Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program")
[8]In particular cross-platform compilation has not been addressed.

Note there is a possibility that the introduction of a new platform may be incompatible with the design of the integrity test, preventing a valid verification. The implementation of the integrity test is designed to fail for any such unrecognized platforms.

4.3 Software Security

Several separate integrity checks are performed in the process of generating and running an application using the Module:

1. First, the integrity of the entire source distribution file *openssl-fips-1.1.2.tar.gz* must be checked as documented in Appendix C. If that digest does not match then any software generated from the distribution cannot be considered validated.

   The cryptographic algorithm implementations are from the OpenSSL project (www.openssl.org), and have been validated separately by NIST/CSE. All of this code is located in the `openssl-fips-1.1.2.tar.gz` distribution file. The OpenSSL source distribution is identified by a HMAC-SHA-1 digest which is published in this document in Appendix B. Because of this isolation from the other software the files contained in that distribution are known as sequestered files. The software generated from this source code has been given the version designation "*OpenSSL FIPS Object Module v1.1.2*". The code in this cryptographic module cannot be changed without affecting the FIPS 140-2 validation, hence the isolation of the code within the distribution and the distinct version number.

2. The integrity of the binary FIPS Object Module file is checked before generating the runtime executable application .

   When the FIPS Object Module file has been built, a HMAC-SHA-1 digest for that file is generated, and is installed along with the file itself. When the application is generated this HMAC-SHA-1 digest is used to check the integrity of the installed FIPS Object Module file, and a HMAC-SHA-1 digest of the resulting runtime executable application is created and embedded within the FIPS Object Module.

3. The integrity of the FIPS Object Module in the runtime executable application is checked at runtime prior to performing the power-up self-tests.

   At runtime the `FIPS_mode_set()` function uses the embedded HMAC-SHA-1 digest to check the integrity of the memory mapped contents of the FIPS Object Module[9].

This chain of integrity checks assures that applications using OpenSSL, such as Apache-ssl, mod_ssl, OpenSSH, stunnel, etc., will use FIPS 140-2 validated cryptography when built using the validated FIPS Object Module.

[9]Specifically, the object code (text segment) and initialized read-only data (rodata segment) areas of memory as mapped by the runtime linker/loader from the FIPS Object Module file.

4.4 Critical Security Parameters

A Critical Security Parameter (CSP) is information, such as passwords, symmetric keys, asymmetric private keys, etc., that must be protected from unauthorized access. Since the Module is accessed via an API from a referencing application[10], the Module does not manage CSPs. In fact, for most applications CSPs will be found in multiple locations external to the Module, such as in application buffers, primary (RAM) memory, secondary disk storage, CPU registers, and on the system bus. In the case of networked client-server applications some CSPs will be found on both the client and server system and on the network infrastructure in between (Ethernet and WAN communication lines, routers, switches).

The application designer and the end user share a responsibility to ensure that CSPs are always protected from unauthorized access. This protection will generally make use of the security features of the host hardware and software which is outside of the cryptographic boundary defined for this Module.

As an example of the relationship between OpenSSL FIPS Object Module and the calling application, consider the OpenSSH application. The FIPS Object Module is used for actual cryptographic operations (random number generation, encryption, decryption) but the CSPs are stored and managed by the OpenSSH application. The persistent per-user CSPs (public and private keys) are stored in the *~/.ssh/* subdirectory and the application enforces the presence of appropriate permissions (private key owned by the user account and not world readable or group writable).

4.5 Self-Tests

The Module performs a number of power-up and conditional self-tests to ensure proper operation of the Module. Power-up tests include cryptographic algorithm known answer tests and integrity tests. The integrity tests are performed using a HMAC-SHA-1 digest calculated over the object code in the FIPS Object Module. Power-up tests are run automatically when the Module is initialized. Additionally, power-up tests may be executed at any time by calling the `FIPS_selftest()` function and verifying it returns true. No FIPS mode cryptographic functionality will be available until after successful execution of all power-up tests. No authentication is required to perform self-tests either automatically or upon demand.

The failure of any power-up self-test or continuous test causes the Module to enter the Self-Test Failure state (see Appendix A), and all cryptographic operations are disabled until the Module is reinitialized with a successful `FIPS_mode_set()` call. Note the most likely cause of a self-test failure is memory or hardware errors. In practice a self-test failure means the application must exit and be restarted.

4.5.1 Power-up Tests

Known Answer Tests (KATs) are tests where a cryptographic value is calculated and compared with a stored previously determined answer. The power-up self-tests for the following algorithms use a KAT:

[10]Either directly, or indirectly via a separate library referenced by the application.

Algorithm	Known Answer Test
AES	encryption and decryption with 128 bit key
2 Key Triple DES	encryption and decryption
3DES	encryption and decryption
RSA	pairwise consistency test with 1024 bit key public encryption and private decryption with 1024 bit key sign and verify test with 1024 bit key
HMAC	HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512
RNG	random number generation from known IV

Table 4.5.1 - Power-up Self-Tests

4.5.2 Conditional Tests

In addition to the power-up tests, the Module performs several conditional tests including pair-wise consistency tests on newly generated public and private key pairs.

Conditional tests are performed automatically as necessary and cannot be turned off. Currently, all conditional tests relate to services available only to users. Thus, conditional and critical function tests are not performed at any time in response to Crypto Officer actions.

Algorithm	Conditional Test
RSA	pairwise consistency test (public encryption and private decryption with 1024 bit key)
RNG	Continuous RNG test per FIPS 140-2 §4.9.2

Table 4.5.2 - Conditional Tests

Pair-wise Consistency Test

A pairwise consistency test is performed when RSA key pairs are generated by applying a private key to the ciphertext and verifying that the result equals the original plaintext.

Software/Firmware Load Test

Not applicable; the Module does not utilize externally loaded cryptographic modules.

Manual Key Entry Test

Not applicable; keys are not manually entered into the Module.

Bypass Test

Not applicable; the Module does not implement a bypass capability.

4.5.3 Critical Function Tests

The Module does not implement any critical function tests.

4.6 Physical Security

The Module does not claim to enforce any physical security as it is implemented entirely in software.

4.7 Mitigation of Other Attacks

The Module does not mitigate against any specific attacks.

5. Design Assurance

The Module is managed in accordance with the established configuration management and source version control procedures of the OpenSSL project, with additional mechanisms to assure the integrity of source code as delivered and used to create applications.

5.1 Source Code Control

Software development functions for OpenSSL software (configuration management, version control, change control, software defect tracking) are managed by the OpenSSL group. The source code revisions are maintained in a CVS[11] repository (http://cvs.openssl.org/) with public read access but with write access restricted to the core OpenSSL team. Individually packaged revisions are released periodically in "tarball" (compressed tar archive) form. The integrity of the Module is based on HMAC-SHA-1.

The OpenSSL group also maintains several mailing lists for developers and end users (http://www.openssl.org/support/), accessible on a subscription basis or as searchable archives.

5.2 Application Management of Critical Security Parameters

Identifying CSPs

All CSPs must be created, stored, and destroyed in an approved manner as described by Reference 1, FIPS 140-2. CSPs are those items of information which must be protected from disclosure, such as symmetric keys, asymmetric private keys, etc. Note that the application designer and end user/system administrator/Crypto Officer share a responsibility for protection of CSPs; the former to include appropriate technical protections and the latter to install and configure the application correctly. Technical protections include checks to require that files storing CSPs have appropriate permissions (not group writable or world readable, for example). Administrative protections include installation of the runtime software (executables and configuration files) in protected locations. End users have a responsibility to refrain from comprising CSPs (as by sending a password in clear text or copying an encryption key to an unprotected location).

Key Generation

The Module API provides cryptographic functions used for key generation. As with other validated cryptographic libraries, API function calls from the calling application that lies outside of the logical cryptographic boundary are used to generate keys. For example, a call to the *RSA_generate_key()* API function would be used to generate RSA keys, *AES_set_encrypt_key()* for AES, *DES_set_key()* for

[11] See http://www.cvshome.org/.

3DES, and so forth. The control input that drives the invocation of the Module API functions comes from the calling application.

Output of Keys Used for Key Establishment

Secret keys used for key establishment must be wrapped with RSA by the calling application (the application using the Module to perform cryptographic operations) before being output. For key wrapping using RSA, the key used to wrap the transported key should be larger than the key that is being transported. The minimum key size that must be used for this operation is 1024 bits which provides 80 bits of encryption strength.

Storage of CSPs

The Module does not store any critical security parameters (CSPs) in persistent media; while the Module is initialized any CSPs reside temporarily in RAM and are destroyed at the end of the session. Any keys or other CSPs otherwise stored in persistent media must be protected in accordance with FIPS requirements in Reference 1, FIPS 140-2.

Destruction of CSPs

When no longer needed, CSPs contained within the application must be deleted by overwriting in a way that will make them unrecoverable. The `fips_rand_bytes()` function in the Module can be used to generate random data to overwrite the storage location of a CSP.

Note the `OPENSSL_cleanse()` function call which is typically used to perform key wiping functions is not part of the Module. This function overwrites a memory storage area to ensure destruction of data, using the random data generation functions of the Module.

6. Glossary

AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CSE	Communications Security Establishment (Canada)
CSP	Critical Security Parameter
DES	Digital Encryption Standard
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
ECB	Electronic Codebook
FIPS	Federal Information Processing Standard
FSM	Finite State Machine
HMAC-SHA-1	Hash based Message Authentication Code
IV	Initialization Vector
KAT	Known Answer Test
NIST	National Institute of Standards and Technology (United States)
OFB	Output Feedback
OpenSSH	The open source SSH implementation
OpenSSL	The open source cryptographic library implementation
OS	Operating System
RSA	Rivest, Shamir and Adleman
SHA-1	Secure Hash Algorithm
SSH	Secure Shell application layer protocol
SSL	Secure Socket Layer transport layer protocol
tar	Tape Archive command common to Unix® based and Linux® systems
tarball	Compressed tar archive, a common format for software distribution
TLS	Transport Layer Security
XOR	Exclusive Or
3DES	Triple DES

7. References

1. FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*, May 2001, National Institute of Standards and Technology

2. *Derived Test Requirements for FIPS PUB 140-2, Security Requirements for Cryptographic Modules*, 15 November 2001 (draft), National Institute of Standards and Technology

3. *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*, January 21, 2005, National Institute of Standards and Technology

4. FIPS PUB 197, *Advanced Encryption Standard (AES)*, 26 November 2001, National Institute of Standards and Technology

5. FIPS PUB 46-3, *Data Encryption Standard (DES)*, 25 October 25 1999, National Institute of Standards and Technology

6. FIPS PUB 81, *DES Modes of Operation*, 2 December 1980, National Institute of Standards and Technology

7. The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), 15 November 2002, National Institute of Standards and Technology

8. NIST Special Publication 800-20, *Modes of Operation Validation System for the Triple Data Encryption Algorithm (TMOVS): Requirements and Procedures*, April 2000, National Institute of Standards and Technology

9. NIST Special Publication 800-17, *Modes of Operation Validation System (MOVS): Requirements and Procedures*, February 1998, National Institute of Standards and Technology

10. FIPS 180-1, Secure Hash Standard (SHS), 17 April 1995, National Institute of Standards and Technology

11. *Network Security with OpenSSL*, John Viega et. al., 15 June 2002, O'Reilly & Associates

12. FIPS 171, National Institute of Standards and Technology, 27 April 1992, http://csrc.nist.gov/publications/fips/fips171/fips171.txt

13. RFC 2246, *The TLS Protocol*, T. Dierks, C. Allen, January 1999, http://www.ietf.org/rfc/rfc2246.txt

14. *OpenSSL FIPS 140-2 User Guide, v1.0*, included in OpenSSL distributions and available at http://www.openssl.org/.

15. *Handbook of Applied Cryptography,* Alfred Menezes, October 1996, CRC Press. The relevant page describing a RNG implementation is available online at http://www.cacr.math.uwaterloo.ca/hac/about/chap5.pdf.

16. *X9.31-1988, Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA),* September 9, 1998, American National Standards Institute.

Appendix A Finite State Model

This Appendix describes the Finite State Machine (FSM) model for an application utilizing the OpenSSL FIPS Object Module. Figure A.1 is a finite state diagram showing the states and transitions between states. At any point in time the Module is in one and only one state. Various software or operating system driven events can cause a transition to another state.

A.1 Diagram

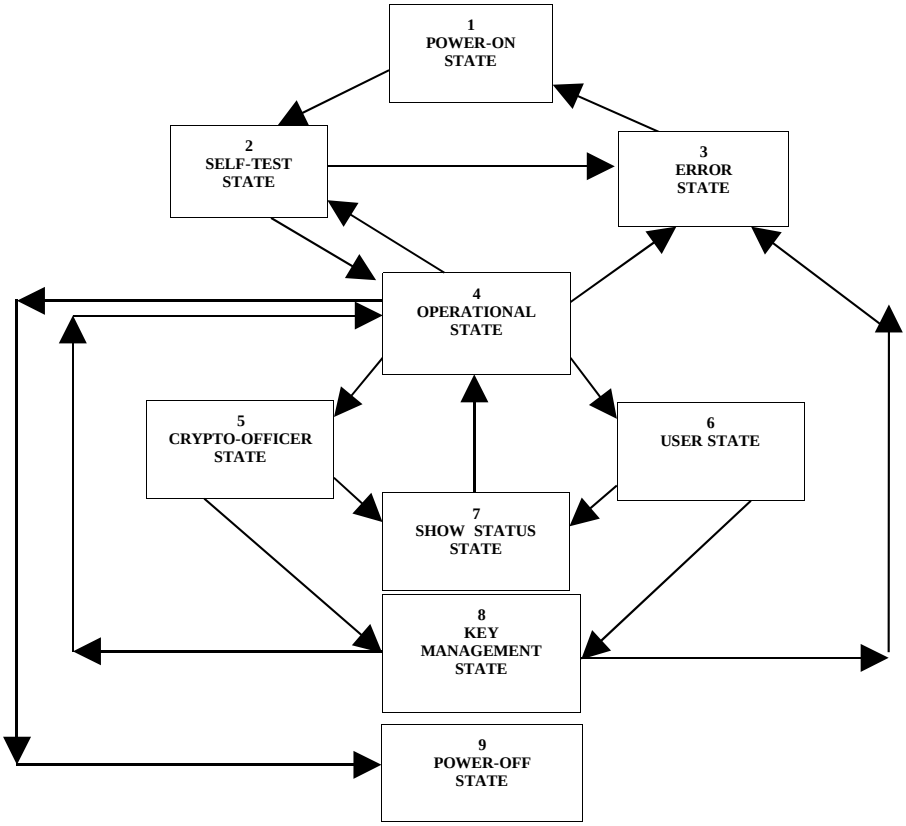

Figure A.1 - Finite State Machine Diagram

A.2 State Descriptions

State 1: Power-On State

The application containing the FIPS Object Module has not been loaded into memory by the host operating system. The Module transitions to the Power-On State when the application is invoked as a process by the host operating system.

State 2: Self-Test State

The application has been loaded into memory for a process created by the host operating system, but the power-up self-tests and FIPS mode initialization (`FIPS_mode_set()` call) have not yet been performed. The `FIPS_mode_set()` call will transition to either the Error or Operational state. Any of the following errors can occur during the power-up self test, all cause a transition to the Error state:

```
FIPS_R_FINGERPRINT_DOES_NOT_MATCH       "fingerprint does not match"
FIPS_R_FINGERPRINT_DOES_NOT_MATCH_NONPIC_RELOCATED
                                        "fingerprint does not match, possibly because of non-PIC relocatation"
FIPS_R_FINGERPRINT_DOES_NOT_MATCH_SEGMENT_ALIASING
                                        “fingerprint does not match, because of invalid segment aliasing"
FIPS_R_FIPS_MODE_ALREADY_SET            "fips mode already set"
FIPS_R_FIPS_SELFTEST_FAILED             "fips selftest failed"
FIPS_R_NON_FIPS_METHOD                  "non fips method"
FIPS_R_PAIRWISE_TEST_FAILED             "pairwise test failed"
FIPS_R_SELFTEST_FAILED                  "selftest failed"
FIPS_R_UNSUPPORTED_PLATFORM             "unsupported platform"
```

State 3: Error State

The initial power-up self-test or subsequent optional self-test has failed. The application and Module will typically terminate on detection of the power-up self-test error. While not likely in practice, a successful re-invocation of the power-up self-test could transition to the Operational state.

State 4: Operational State

The power-up self-test has executed successfully. The cryptographic algorithms in the Module can now be accessed by the application. The Module will remain in the Operational state until the application is terminated and enters the Power-Off state.

State 5: Crypto-Officer State

The application is in crypto-officer state.

State 6: User State

The application is in user state.

State 7: Show Status State

The application is performing a show status operation.

State 8: Key Management State

The application is performing a key management operation.

State 9: Power-Off State

The host operating system has terminated the application process and released all memory.

Appendix B Controlled Source File Fingerprint

The *OpenSSL FIPS Object Module v1.1.2* consists of the FIPS Object Module (the *fipscanister.o* contiguous unit of binary object code) generated from the specific source files found in the specific special OpenSSL distribution o*penssl-fips-1.1.2.tar.gz* with HMAC-SHA-1 digest[12] of

```
e0a9c4b06ecae197084ae152524dd39fcaab695d
```

at http://www.openssl.org/source/openssl-fips-1.1.2.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 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.

Note the reference compiler for the tested platform was *gcc v4.1.2* (SUSE).

[12]The command "*openssl sha1 -hmac etaonrishdlcupfm openssl-fips-1.1.2.tar.gz*" will display this HMAC-SHA-1 digest value.

Appendix C Installation and Initialization

These instructions assume the following:

1. The reader has the basic knowledge of how to unpack a source distribution; and
2. The reader has a functional development environment to execute the instructions below.

C.0 Validating the Source Distribution File

The HMAC-SHA-1 digest of the o*penssl-fips-1.1.2.tar.gz* source distribution file is published in Appendix B. Generate a HMAC-SHA-1 digest of that file using the same key and compare it to the published value to confirm that the file is authentic and unmodified. The following command can be used on any system that contains a recent version of the standard OpenSSL product:

```
openssl sha1 -hmac etaonrishdlcupfm openssl-fips-1.1.2.tar.gz
```

The displayed digest value must exactly match the published value in Appendix B, or the distribution cannot be used to generate any FIPS 140-2 validated software.

C.1 Building the FIPS Object Module from Source

Build the OpenSSL FIPS Object Module from source after unpacking the source distribution o*penssl-fips-1.1.2.tar.gz.* The FIPS specific code is incorporated into the generated FIPS Object Module file when the `fips` configuration option is specified as:

```
$ ./config fips
```

Note that no other configuration options may be specified by the user.

Then:

```
$ make
```

to generate the FIPS Object Module file *fipscanister.o*, and the digest for the FIPS Object Module file, *fipscanister.o.sha1*. The *fipscanister.o*, and *fipscanister.o.sha1* files are intermediate files. The object code in the *fipscanister.o* file is incorporated into the runtime executable application at the time the binary executable is generated.

C.2 Installing and Protecting the FIPS Object Module

The Crypto Officer should install the generated files in a location protected by the host operating system security features. These protections should allow write access only to Crypto Officers and read access only to authorized users.

The usual

```
# make install
```

will install both the *fipscanister.o* and *fipscanister.o.sha1* files in the default location for the type of system[13] with the appropriate permissions to satisfy the security requirement. In addition the source file used to generate the embedded digest, *fips_premain.c*, and the digest that protects that file, *fips_premain.c.sha1*, are installed as well. The *fips_premain.c* source file is used at application link time to create the embedded digest.

After installation of the four files (*fipscanister.o*, *fipscanister.o.sha1*, *fips_premain.c*, *fips_premain.c.sha)* the unpacked distribution and all other generated files should be discarded.

C.3 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 OpenSSL distribution contains a reference utility[14] which 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.

[13] Typically */usr/local/lib/* for Unix® based or Linux® systems.

[14] This utility is the "*openssl sha*" command with the *-hmac* option switch. It is included in the FIPS Object Module distribution and also in all recent OpenSSL distributions. The version of this utility generated from the FIPS Object Module distribution can be used to check the validity of the distribution tarball digest after the fact. Note that in principle a software distribution could be corrupted in such a way as to incorrectly return the expected digest. This risk is present for all validated products, of course, and would be even harder to detect without visible source code.

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.

C.4 FIPS Mode Initialization

Somewhere very early in the execution of the application FIPS mode must be enabled. This should be done by invocation of the `FIPS_mode_set()` function call as in this example:

```
#ifdef OPENSSL_FIPS
      if(options.no_fips <= 0)
            {
            if(!FIPS_mode_set(1))
                  {
                  ERR_load_crypto_strings();
                  ERR_print_errors(BIO_new_fp(stderr,BIO_NOCLOSE));
                  exit(1);
                  }
            else
                  fprintf(stderr,"*** IN FIPS MODE ***\n");
            }
            #endif
```

Example 4.C - Invocation of FIPS_mode_set()

Note that it is permitted to *_not_* enable FIPS mode, in which case, the FIPS Object Module used in conjunction with the OpenSSL API should function as the OpenSSL API alone always has. The application will not, of course, be FIPS 140-2 validated.