ORACLE
Linux

FIPS 140-2 Non-Proprietary Security Policy

Oracle Linux 7 OpenSSL Cryptographic Module

FIPS 140-2 Level 1 Validation

Software Version: R7-7.8.0

Date: May 9th, 2022

Title: Oracle Linux 7 OpenSSL Cryptographic Module Security Policy

Date: May 9th, 2022

Author: Oracle Security Evaluations – Global Product Security

Contributing Authors:

Oracle Linux Engineering

atsec information security

Oracle Corporation
World Headquarters
2300 Oracle Way
Austin, TX 78741
U.S.A.
Worldwide Inquiries:
Phone: +1.650.506.7000
www.oracle.com

TABLE OF CONTENTS

List of Tables

List of Figures

1. Introduction

1.1 Overview

Oracle Linux is a set of cryptographic libraries, services, and user level cryptographic applications that are validated at FIPS 140-2 level 1, providing a secure foundation for vendor use in developing dependent services, applications, and even purpose-built appliances that may be FIPS 140-2 validated.

This section is informative to the reader to reference other cryptographic services of Oracle Linux. Only the software listed in section 3.1 is subject to the FIPS 140-2 validation. The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when supported if the specific operational environment is not listed on the validation certificate.

The FIPS 140-2 validation is performed at Security Level 1, on software only modules that do not make any claims about the hardware enclosure. This allows vendors to develop their own modules using these basic cryptographic modules and validate the new modules at a higher FIPS 140-2 security level.

The following cryptographic modules are included in Oracle Linux:

- OpenSSL– a software cryptographic module supporting FIPS 140-2-approved cryptographic algorithms for protocols like TLS 1.2, OpenSSH, and HTTPS
- OpenSSH-Server – supplies key derivation support for the SSH protocol
- OpenSSH-Client – supplies key derivation support for the SSH protocol
- NSS Softokn Cryptographic Module – supplies cryptographic support for TLS, PKCS #5, PKCS #7, PKCS #11 (version 3.0), PKCS #12, S/MIME, X.509 v3 certificates, and other security standards supporting FIPS 140-2 validated cryptographic algorithms
- Oracle Linux Unbreakable Enterprise Kernel (UEK) – a software only cryptographic module via the Kernel Crypto API that is an optimized kernel with a wide range of advanced features and improvements for enterprise workloads. The UEK provides general-purpose cryptographic services and block storage encryption.
- GnuTLS – general purpose cryptographic module to support TLS network protocols
- Libgcrypt – supplies general cryptographic support for GPG.
- Libreswan – provides the IKE protocol version 2 key agreement services required for IPSEC.

This Security Policy describes the features and design of the Oracle Linux OpenSSL Cryptographic Module using the terminology contained in the FIPS 140-2 specification. FIPS 140-2, Security Requirements for Cryptographic Module specifies the security requirements that will be satisfied by a cryptographic module utilized within a security system protecting sensitive but unclassified information. The NIST/CSE Cryptographic Module Validation Program (CMVP) validates cryptographic module to FIPS 140-2. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

1.2 Document Organization

The Security Policy document is one document in a FIPS 140-2 Submission Package. In addition to this document, the Submission Package contains:

- Oracle Linux 7 OpenSSL Cryptographic Module Non-Proprietary Security Policy
- Other supporting documentation as additional references

With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to Oracle and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact Oracle.

2. Oracle Linux OpenSSL Cryptographic Module

2.1 Functional Overview

The Oracle Linux 7 OpenSSL Cryptographic Module (hereafter referred to as the “Module”) is a software module supporting FIPS 140-2 Approved cryptographic algorithms within Oracle Linux. The code base of the Module is formed in a combination of standard OpenSSL shared Library, OpenSSL FIPS Object Module, and development work by Oracle Linux engineering. The scope of testing for this validation covers versions R7-7.8.0 running Oracle Linux 7.8. The Oracle Linux 7 OpenSSL Module is distributed with an open-source distribution. The Module provides a C language application program interface (API) for use by other processes that require cryptographic functionality.

Oracle Linux 7 OpenSSL supports following three types of cryptographic implementations. The implementations available for an algorithm are listed in Table 2 and they can be selected based on the environment variable OPENSSL_ia32cap. Please refer to its man page for the details.

a) OpenSSL in Native C Programming Language;
b) AES-NI hardware acceleration for X86 processors;
c) Assembler implementation.

2.2 FIPS 140-2 Validation Scope

The following table shows the security level for each of the eleven sections of the validation. See Table 1 below.

Security Requirements Section	Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles and Services and Authentication	1
Finite State Machine Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	1

Table 1: FIPS 140-2 Security Requirements

3. Cryptographic Module Specification

3.1 Definition of the Cryptographic Module

The Oracle Linux 7 OpenSSL Cryptographic Module is defined as a multi-chip standalone module as defined by the requirements within FIPS PUB 140-2. The logical cryptographic boundary of the module consists of shared library files and their integrity check HMAC files, which are delivered through the Oracle Linux Yum Server as listed below:

The module version R7-7.8.0 was tested on Oracle Linux 7.8 and is contained within the RPM file with versions openssl-libs-1.0.2k-21.0.1.el7_9.x86_64.rpm for x86_64 or openssl-libs-1.0.2k-21.0.1.el7_9.aarch64.rpm for aarch64, which contains the following files:

- /usr/lib64/.libcrypto.so.1.0.2k.hmac (64 bits)
- /usr/lib64/.libssl.so.1.0.2k.hmac (64 bits)
- /usr/lib64/libcrypto.so.1.0.2k (64 bits)
- /usr/lib64/libssl.so.1.0.2k (64 bits)

The module instantiation for version R7-8.0.0 is provided by the dracut-fips package with the version of the RPM file of dracut-fips-033-572.0.5.el7.x86_64.rpm for x86_64 or dracut-fips-033-572.0.5.el7.aarch64.rpm for aarch64.

The AES-NI configuration of the UEK kernel for module version R7-7.8.0 for Oracle Linux 7.8 is provided by the dracut-fips package with the version of the RPM file of dracut-fips-AESNI-033-572.0.5.el7.x86_64.rpm for x86_64 or dracut-fips-aesni-033-572.0.5.el7.aarch64.rpm for aarch64.

The Oracle OpenSSL package includes the binary files, integrity check HMAC files, Man Pages and the OpenSSL engines provided by the standard OpenSSL shared library. The OpenSSL engines and their shared object files are not part of the Module, and therefore they must not be used when operating the Module.

The Module shall be instantiated by the dracut-fips package with the RPM file version specified above. The dracut-fips RPM package is only used for the installation and instantiation of the Module. This code is not active when the Module is operational and does not provide any services to users interacting with the Module. Therefore, the dracut-fips RPM package is outside the modules logical boundary.

Figure 1 shows the logical block diagram of the module executing in memory on the host system.

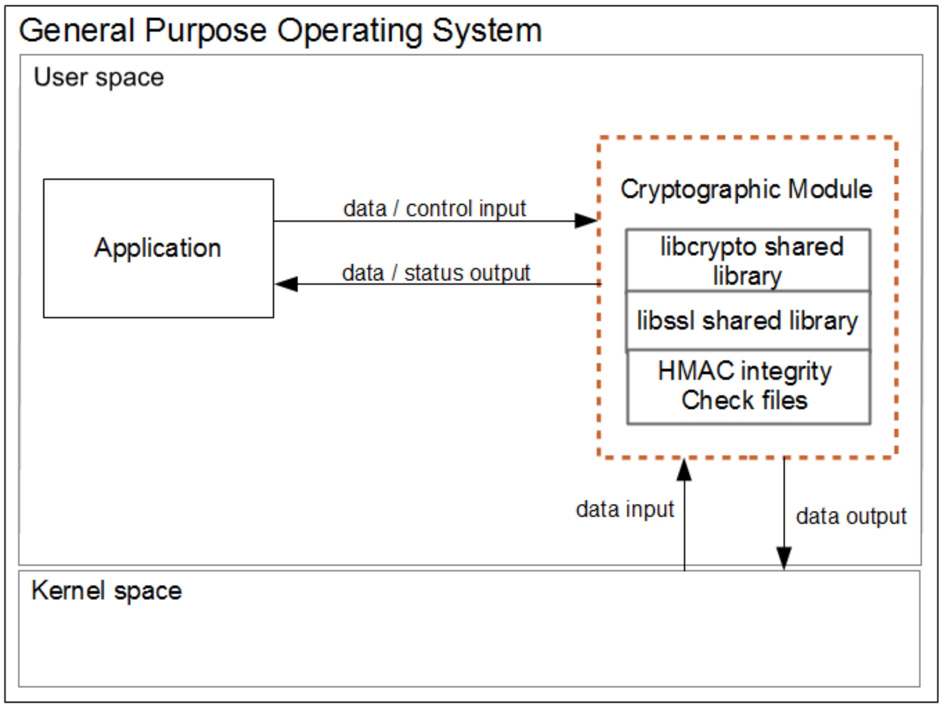

Figure 1: Oracle Linux OpenSSL Logical Cryptographic Boundary

3.2 Definition of the Physical Cryptographic Boundary

The physical cryptographic boundary of the module is defined as the hard enclosure of the host system on which it runs. See figure 2 below. No components are excluded from the requirements of FIPS PUB 140-2.

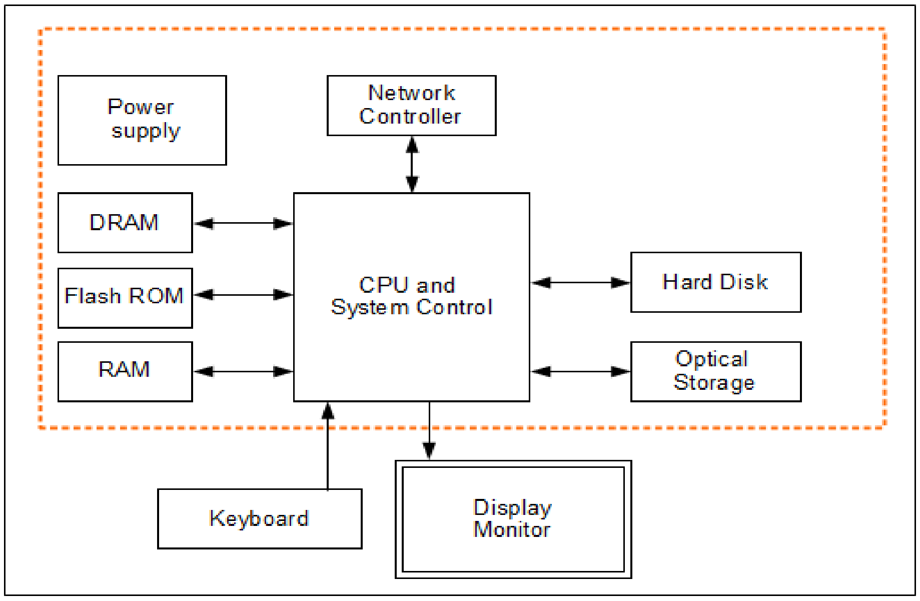

Figure 2: Oracle Linux OpenSSL Hardware Block Diagram

3.3 Approved or Allowed Security Functions

The Oracle Linux 7 OpenSSL Cryptographic Module contains the following FIPS Approved Algorithms listed in Table 2.

Once the module is operational, the mode of operation is implicitly assumed depending on the services/security function invoked. By default, the module enters Approved mode after the power-up tests succeed. In Approved mode, only approved or allowed security functions can be used as specified in Table 2 and Table 3. FIPS approved services are listed in Table 8.

Approved Security Functions		Cert #
Symmetric Algorithms		
AES	**AESNI:** AES in CBC, ECB, CFB1, CFB8, CFB128, OFB, CTR, CCM, KW, KWP, CMAC (MAC generation and verification), XTS Modes; (E/D; Key Sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 1207
	AESNI_AVX: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1210
	AESNI_CLMULNI: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1211
	AESNI_ASM: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1212
	AESASM: AES in CBC, ECB, CFB1, CFB8, CFB128, OFB, CTR, CCM, KW, KWP, CMAC (MAC generation and verification), XTS Modes; (E/D; Key Sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 1208
	AESASM_AVX: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1213
	AESASM_CLMULNI: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1214
	AESASM_ASM: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1215
	BAES_CTASM: AES in CBC, ECB, CFB1, CFB8, CFB128, OFB, CTR, CCM, KW, KWP, CMAC (MAC generation and verification), XTS Modes; (E/D; Key Sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 1209
	BAES_CTASM_AVX: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1216
	BAES_CTASM_CLMULNI: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1217

Approved Security Functions		Cert #
	BAES_CTASM_ASM: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 1218
	AES_C: AES in CBC, ECB, CFB1, CFB8, CFB128, OFB, CTR, CCM, KW, KWP, CMAC (MAC generation and verification), XTS Modes; (E/D; Key Sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 2394
	CE: AES in CBC, ECB, CFB1, CFB8, CFB128, OFB, CTR, CCM, KW, KWP, CMAC (MAC generation and verification), XTS Modes; (E/D; Key Sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 2396
	VPAES: AES in CBC, ECB, CFB1, CFB8, CFB128, OFB, CTR, CCM, KW, KWP, CMAC (MAC generation and verification), XTS Modes; (E/D; Key Sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 2397
	AES_C_GCM: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 2398
	CE_GCM: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 2399
	VPAES_GCM: AES in GCM and GMAC Modes; External IV (D; Key Sizes 128, 192, 256) AES in GCM Mode; Internal IV (E; Key Sizes 128, 192, 256)	A 2400
Triple-DES	**TDES_C:** Triple-DES in CBC, ECB, CFB1, CFB8, CFB64, OFB, CMAC (MAC generation and verification) Modes; (KO1, D/E)	A 1206
Secure Hash Standard (SHS)		
SHS	**SHA_AVX2:** SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	A 1224
	SHA_AVX: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	A 1225
	SHA_SSSE3: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	A 1226
	SHA_ASM : SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	A 1227
	NEON : SHA-256	A 2395
Data Authentication Code		
HMAC	**SHA_AVX2:** HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	A 1224
	SHA_AVX: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	A 1225

Approved Security Functions		Cert #
	SHA_SSSE3: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	A 1226
	SHA_ASM: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	A 1227
	NEON: HMAC-SHA-256	A 2395
Asymmetric Algorithms		

RSA	**SHA_AVX2:**
	FIPS 186-4:
	PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096;
	PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PKCS 1.5 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
	X9.31 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-256, SHA-384, SHA-512;
	X9.31 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-256, SHA-384, SHA-512;
		A 1224

	SHA_AVX:
	FIPS 186-4:
	PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096;
	PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PKCS 1.5 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
	X9.31 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-256, SHA-384, SHA-512;
	X9.31 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-256, SHA-384, SHA-512;
		A 1225

	SHA_SSSE3:
	FIPS 186-4:
	PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096;
	PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PKCS 1.5 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
		A 1226

Approved Security Functions		Cert #

	X9.31 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-256, SHA-384, SHA-512;
	X9.31 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-256, SHA-384, SHA-512;	

	SHA_ASM:
	FIPS 186-4:
	PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096;
	PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PKCS 1.5 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512;
	PSS (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-224, SHA-256, SHA-384, SHA-512;
	X9.31 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-256, SHA-384, SHA-512;
	X9.31 (Sig Ver) Modulus Sizes 1024, 2048, 3072, 4096 with hash sizes SHA-1, SHA-256, SHA-384, SHA-512;
		A 1227

DSA	**SHA_AVX2:** **FIPS 186-4:** Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1224
	SHA_AVX: **FIPS 186-4:** Key Gen, PQG Gen, PQG Ver**,** Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1225
	SHA_SSSE3: **FIPS 186-4:** Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1226
	SHA_ASM: Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1227
ECDSA	**SHA_AVX2:** **FIPS 186-4:** Key Gen, Key Ver, Sig Gen, Sig Ver; Curves P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1224
	SHA_AVX: **FIPS 186-4:** Key Gen, Key Ver, Sig Gen, Sig Ver; Curves P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1225
	SHA_SSSE3: **FIPS 186-4:**	A 1226

Approved Security Functions		Cert #
	Key Gen, Key Ver, Sig Gen, Sig Ver; Curves P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	
	SHA_ASM: **FIPS 186-4:** Key Gen, Key Ver, Sig Gen, Sig Ver; Curves P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1227
Random Number Generation (NIST SP 800-90A)		
DRBG	**AESNI:** **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)]	A 1207
	AESASM: **CTR_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)]	A 1208
	BAES_CTASM: **CTR_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)]	A 1209
	DRBG_10X: **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)] **HMAC_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; (HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512)] **Hash_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)]	A 1228
	AES_C: **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)]	A 2394
	CE: **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)]	A 2396
	VPAES: **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)]	A 2397
Key Agreement (NIST SP 800-56Ar3)		
KAS-FFC-SSC-SP800-56Ar3	**FFC_DH:** **KAS-FFC-SSC SP 800-56Ar3:** dhEphem scheme, Domain Parameter Generation and Mod P Methods (ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192) KAS role: initiator, responder	A 1322
KAS-ECC-SSC-SP800-56Ar3	**SHA_AVX2:** **KAS-ECC-SSC SP 800-56Ar3:** ephemeralUnified scheme for EC Diffie-Hellman shared secret computation (Curves P-256, P-384, P-521). KAS role: initiator, responder	A 1224
	SHA_AVX: **KAS-ECC-SSC SP 800-56Ar3:** ephemeralUnified scheme for EC Diffie-Hellman shared secret computation (Curves P-256, P-384, P-521). KAS role: initiator, responder	A 1225

Approved Security Functions		Cert #
	SHA_SSSE3: **KAS-ECC-SSC SP 800-56Ar3:** ephemeralUnified scheme for EC Diffie-Hellman shared secret computation (Curves P-256, P-384, P-521). KAS role: initiator, responder	A 1226
	SHA_ASM: **KAS-ECC-SSC SP 800-56Ar3:** ephemeralUnified scheme for EC Diffie-Hellman shared secret computation (Curves P-256, P-384, P-521). KAS role: initiator, responder	A 1227
Safe Primes Key Generation and Verification	**FFC_DH:** **Safe Primes Key Generation and Verification:** Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	A 1322
Key Transport Scheme (KTS)		
KTS	**AESNI:** AES-KW (D/E; Key Sizes 128 , 192, 256) AES-KWP (D/E; Key Sizes 128, 192, 256)	A 1207
	AESASM: AES-KW (D/E; Key Sizes 128 , 192, 256) AES-KWP (D/E; Key Sizes 128, 192, 256)	A 1208
	BAES_CTASM: AES-KW (D/E; Key Sizes 128 , 192, 256) AES-KWP (D/E; Key Sizes 128, 192, 256)	A 1209
	AES-GCM key wrapping with 128, 192, 256 bit keys	A 1210 A 1211 A 1212 A 1213 A 1214 A 1215 A 1216 A 1217 A 1218 A 2398 A 2399 A 2400
	AES-CCM key wrapping with 128, 192, 256 bit keys	A 1207 A 1208 A 1209 A 2394 A 2396 A 2397
	AES-CBC with 128 and 256 bit keys and HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-512 key wrapping.	A 1207 (AES) A 1208 (AES) A 1209 (AES) A 2394 (AES) A 2396 (AES) A 2397 (AES)

Approved Security Functions		Cert #
		A 1224 (HMAC) A 1225 (HMAC) A 1226 (HMAC) A 1227 (HMAC) A 2395 (HMAC-SHA-256 Only)
	Triple-DES CBC with 192[1] bit key and HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-512 key wrapping.	A 1206 (Triple-DES) A 1224 (HMAC) A 1225 (HMAC) A 1226 (HMAC) A 1227 (HMAC) A 2395 (HMAC-SHA-256 Only)
Key Derivation Function (NIST SP 800-135)		
KDF TLS	**SHA_AVX2:** TLS1.0/1.1, TLS 1.2 (SHA 256, 384)	A 1224
	SHA_AVX: TLS1.0/1.1, TLS 1.2 (SHA 256, 384)	A 1225
	SHA_SSSE3: TLS1.0/1.1, TLS 1.2 (SHA 256, 384)	A 1226
	SHA_ASM: TLS1.0/1.1, TLS 1.2 (SHA 256, 384)	A 1227
Entropy		
ENT (NP)	NIST SP 800-90B	N/A

Table 2: FIPS Approved Security Functions

[1] Though the key size is 192 bits, the strength of that key and therefore the KTS implementation for Triple-DES is 112 bits only according to IG 7.5.

3.4 Non-Approved but Allowed Security Functions

The following are considered non-Approved but allowed security functions provided by the Module:

Algorithm	Usage
RSA PKCS#1-v1.5 Key Wrapping with key sizes greater than 2048-bit up to 16384 bits	Key wrapping using PKCS#1-v1.5 padding scheme method, key establishment methodology provides between 112 and 256 bits of encryption strength.
MD5 (no security claimed) Allowed to be used in FIPS mode per IG 1.23	Message digest used in TLS only

Table 3: Non-Approved but Allowed Security Functions

3.5 Non-Approved Security Functions

Security functions listed in the table below, make use of non-approved cryptographic algorithms. Use of any of these algorithms and services in Table 9 will put the module in the non-Approved mode implicitly. The services associated with these algorithms are specified in section 7.2.

Algorithm	Usage
AES-GCM with external IV	Encryption[2]
ANSI X9.31 RNG	Random Number Generation
Blowfish	Encryption/Decryption
Camellia	Encryption/Decryption
CAST	Encryption/Decryption
CAST5	Encryption/Decryption
DES	Encryption/Decryption
Diffie-Hellman with non-compliant key size	Key agreement and shared secret computation using primes not listed in Table 2
Diffie-Hellman keys generated with domain parameters other than safe primes.	Key agreement, Shared Secret computation
DSA with non-compliant key size or hash size	DSA key pair generation and domain parameter generation with key size less than 2048 bits or greater than 3072 bits.
	DSA signature generation with key size less than 2048 bits or greater than 3072 bits. DSA signature generation with SHA-1.
	DSA domain parameter generation/verification less than 2048 bits or greater than 3072 bits. DSA domain parameter generation/verification with SHA-1
	DSA signature verification with key size less than 1024 bits or greater than 3072 bits.
EC Diffie-Hellman with P-192 curve, K curves, B curves and non-NIST curves.	Key agreement, Shared Secret computation

[2] The GCM encryption with external IV has been tested with CAVP certs #A1210, #A1211, #A1212, #A1213, #A1214, #A1215, #A1216, #A1217 and #A1218, #2398, #2399, #2400 however, it does not meet the IG A.5 compliance and hence it is listed as non-approved.

ECDSA with P-192 curve, K curves, B curves and non-NIST curves.	Key generation/Key verification/Signature generation/Signature Verification with NIST curve P-192, K curves, B curves and non-NIST curves
	Signature generation using SHA-1
GHASH	Hash Function
HMAC with non-compliant key size	HMAC with less than 112-bit keys
IDEA	Encryption/Decryption
J-PAKE	Password Authenticated Key Exchange
MD2	Hash Function
MD4	Hash Function
MD5	Hash Function
MDC2	Hash Function
PBKDF[3]	Key Derivation
RC2	Encryption/Decryption
RC4	Encryption/Decryption
RC5	Encryption/Decryption
RIPEMD	Hash Function
RSA with non-compliant key size or hash size	RSA key pair generation with key size less than 2048 bits or greater than 4096 bits.
	RSA Signature generation with SHA-1
	RSA signature generation with key size less than 2048 bits or greater than 4096 bits.
	RSA signature verification with key size less than 1024 bits or greater than 4096 bits.
SEED	Encryption/Decryption
Whirlpool	Hash Function

Table 4: Non-Approved Disallowed Functions

[3] Even though the PBKDF has CAVP certificates #A1224, #A1225, #A1226 and #A1227, the KAT is not implemented.

4. Module Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs. The logical interfaces are the API through which applications request services, and the TLS protocol internal state and messages sent and received from the TCP/IP protocol.

The module interfaces can be categorized as follows:

- Data Input Interface
- Data Output Interface
- Control Input interface
- Status Output Interface

The module can be accessed by utilizing the API function it provides. Table 5 below, shows the mapping of interfaces as per FIPS 140-2 Standard.

FIPS 140 Interface	Module Interfaces
Data Input	API Input Parameters
Data Output	API Output Parameters
Control Input	API Function Calls
Status Output	API Return Values, error message

Table 5: Mapping of FIPS 140 Logical Interfaces

5. Physical Security

The Module is comprised of software only and thus does not claim any physical security.

6. Operational Environment

6.1 Tested Environments

The Modules were tested on the following environments with and without PAA i.e. AES-NI:

Module Version	Operating Environment	Processor	Hardware
R7-7.8.0	Oracle Linux 7.8 64 bit	Intel® Xeon® Platinum 8167M	Oracle Server X7-2C
R7-7.8.0	Oracle Linux 7.8 64 bit	AMD EPYC™ 7551	Oracle Server E1-2C
R7-7.8.0	Oracle Linux 7.8 64 bit	Ampere® Altra® Neoverse-N1	Oracle Server A1-2C

Table 6: Tested Operating Environment

6.2 Vendor Affirmed Environments

The following platforms have not been tested as part of the FIPS 140-2 level 1 certification however Oracle "vendor affirms" that these platforms are equivalent to the tested and validated platforms. Additionally, Oracle affirms that the module will function the same way and provide the same security services on any of the systems listed below.

Operating Environment	Hardware
Oracle Linux 7 64-bit	Oracle X Series Servers
Oracle Linux 7 64-bit	Oracle E Series Servers
Oracle Linux 7 64-bit	Oracle A Series Servers
Oracle Linux 7 64-bit	Marvell CN23XX OCTEON (MIPS) SmartNIC
Oracle Linux 7 64-bit	Marvell CN93XX LiquidIO III (ARM) SmartNIC
Oracle Linux 7 64-bit	Pensando DSC-200 (ARM) SmartNIC

Table 7: Vendor Affirmed Operating Environment

6.3 Operational Environment Policy

The operating system is restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded).

The application that makes calls to the Modules is the single user of the Modules, even when the application is serving multiple clients.

During module operation, the ptrace(2) system call, the debugger (gdb(1)), and strace(1) shall not be used. In addition, other tracing mechanisms offered by the Linux environment, such as Dtrace, ftrace or systemtap, shall not be used.

7. Roles, Services and Authentication

7.1 Roles

The Oracle Linux 7 OpenSSL Cryptographic Module supports 2 roles as required by FIPS PUB 140-2. These roles are a Crypto Officer Role and a User Role. Both roles are implicitly assumed by the entity accessing services implemented by the Module. The module does not support authentication mechanisms.

7.2 FIPS Approved Operator Services and Descriptions

The below table provides a full description of FIPS Approved services provided by the module and lists the roles allowed to invoke each service. In the table below, the “U” represents a User Role, and “CO” denotes a Crypto Officer role.

U	CO	Service Name	Service Description	Keys and CSP(s)	Access Type(s)
X		Symmetric Encryption/Decryption	Encrypts or decrypts a block of data using AES or 3-Key Triple-DES listed in the Table 2	AES or 3-Key Triple-DES Key listed in the Table 2	R, W, X
X		Asymmetric Encryption/Decryption	Encrypts or decrypts using Approved RSA key size	RSA key pair keys listed in Table 2	R, W, X
X		Certificate Management Handling	Management of key properties within certificates.	RSA, DSA, and ECDSA private keys listed in Table 2	R, W, X
X		Asymmetric Key Generation	Generate RSA, DSA and ECDSA asymmetric keys	RSA, DSA and ECDSA keys listed in Table 2	R, W, X
X		Asymmetric Key Verification	Verify ECDSA asymmetric keys	ECDSA keys listed in Table 2	R, W, X
X		Digital Signature Generation and Verification	Sign and verify operations	RSA, DSA, and ECDSA keys listed in Table 2	R, W, X
X		Asymmetric domain parameter Generation/Verification	Generate and verify DSA domain parameters	DSA public and private keys listed in Table 2	R, W, X
X		Diffie-Hellman shared secret computation	Shared secret computation for Diffie-Hellman	Diffie-Hellman public and private keys, Shared secret	R, W, X
X		Diffie-Hellman key generation and verification using safe primes	Safe Primes Key Generation and Verification	Diffie-Hellman public and private keys	R, W, X

U	CO	Service Name	Service Description	Keys and CSP(s)	Access Type(s)
X		EC Diffie-Hellman shared secret computation	Shared secret computation for EC Diffie-Hellman	EC Diffie-Hellman public and private keys, Shared secret	R, W, X
X		Key wrapping	AES-KW and AES-KWP	AES key	R, W, X
X		TLS Network Protocol	Provide data encryption and authentication over TLS network protocol	AES, Triple-DES, and HMAC keys listed in Table 2.	R, W, X
X		TLS Key Agreement	Negotiate a TLS key agreement secure channel	AES or Triple-DES key, RSA, DSA or ECDSA private key, HMAC Key, shared secrete, Diffie-Hellman and EC Diffie-Hellman Private keys listed in Table 2	R, W, X
X		Key Derivation	TLS KDF	Shared secret, derived key	R, W, X
X		Keyed Hash (HMAC)	Sign and or authenticate data using HMAC-SHA	HMAC Keys listed in Table 2	R, W, X
X		Keyed Hash (CMAC)	Encrypt and sign data using CMAC.	AES or 3-Key Triple-DES Key	R, W, X
X		Hash (SHS)	Hash a block of data.	None	R, W, X
X		Random Number Generation	Generate random numbers based on the NIST SP 800-90A Standard	Entropy input string, seed and internal State	R, W, X
X		Show Status	Show status of the module state	None	X
X		Self-Test	Initiate power-on self-tests	None	R, X
X		Zeroize	Zeroize all critical security parameters	All keys and CSP's	Z
	X	Module Installation	Installation of the module	None	R, W

R – Read, W – Write, X – Execute, Z - Zeroize

Table 8: FIPS Approved Services and Descriptions

7.3 Non-FIPS Approved Services and Descriptions

The following table lists the non-Approved services available in non-FIPS mode. Security services listed in the table below, make use of non-approved cryptographic algorithms. Use of any of these services in Table 9 will put the module in the non-Approved mode implicitly. The algorithms associated with these services are specified in section 3.5.

U	CO	Service Name	Service Description	Keys and CSP(s)	Access Type(s)
X		Asymmetric Encryption/Decryption	Encrypts or decrypts using non-Approved RSA key size as listed in Table 4	RSA key pair	R, W, X

U	CO	Service Name	Service Description	Keys and CSP(s)	Access Type(s)
X		Symmetric Encryption/Decryption	Encrypts or decrypts using non-Approved algorithms	AES-GCM (encryption only with external IV), Blowfish, Camellia, CAST, CAST5, DES, IDEA, RC2, RC4, RC5, SEED keys	R, W, X
X		Digital Signature Generation	Sign operations with RSA, DSA and ECDSA restrictions listed in Table 4	RSA key < 2048 and RSA key > 4096 DSA key < 2048 and DSA key > 3072 ECDSA with NIST curve P-192, K curves, B curves and non-NIST curves Signature Generation with SHA-1	R, W, X
X		Digital Signature Verification	Verify operations with RSA, DSA and ECDSA restrictions listed in Table 4	RSA key < 1024 and RSA key > 4096 DSA key < 1024 and DSA key > 3072 ECDSA with NIST curve P-192, K curves, B curves and non-NIST curves	R, W, X
X		TLS Key Agreement	Negotiate a TLS key agreement secure channel with non-Approved key sizes	RSA/ Diffie-Hellman key < 2048 EC Diffie-Hellman with P-192	R, W, X
X		Asymmetric Key Generation	Generation of non-Approved RSA, DSA and ECDSA keys	RSA key < 2048 DSA key < 2048 ECDSA using NIST curve P-192	R, W, X
X		Random Number Generation	Generation of random numbers using the ANSI X9.31 PRNG	seed, seed key	R, W, X
X		Keyed Hash (HMAC)	HMAC restriction listed in Table 4	HMAC with less than 112-bit keys	R, W, X
X		Hash	Hashing using non-Approved hash functions that include GHASH, MD2, MD4, MD5, MDC2, RIPEMD, Whirlpool	None	R, W, X
X		J-PAKE Key Agreement	Password authenticated key agreement using J-PAKE	J-PAKE key pair	R, W, X
X		Diffie-Hellman with non-compliant key size	Key agreement and shared secret computation using primes not listed in Table 2	Diffie-Hellman public and private keys	R, W, X
X		Diffie-Hellman shared secret computation	Diffie-Hellman restrictions listed in Table 4	Diffie-Hellman public and private keys	R, W, X
X		EC Diffie-Hellman shared secret computation	EC Diffie-Hellman restrictions listed in Table 4	EC Diffie-Hellman public and private keys	R, W, X
X		Key derivation (PBKDF)	PBKDF	Password/passphrase, Derived key	R, W, X

Table 9: Non-FIPS Approved Services and Descriptions

7.4 Operator Authentication

The module does not support operator authentication mechanisms.

8. Key and CSP Management

The following keys, cryptographic key components and other critical security parameters are contained in the module.

CSP Name	Generation/Input	Use	Zeroization
AES Key (128, 192, 256 bits)	The Key is passed into the module via API input parameter	Encrypt/Decrypt operations Used to generate and verify MAC's with AES as part of the CMAC algorithm.	EVP_CIPHER_CTX_cleanup()
Triple-DES Keys (192 bits)		Used for Encrypt/Decrypt operations. Used for generating and verifying MAC's with Triple-DES as part of the CMAC algorithm.	
HMAC Key (≥ 112 bits)		HMAC keys used to generate and verify MAC's on data.	HMAC_CTX_cleanup()
RSA Key pair (2048, 3072, 4096 bits)	Keys are generated using FIPS 186-4 and the random value used in the key generation is generated using SP800- 90A DRBG	RSA public/private keys used to sign and verify data. RSA private key used for key decryption as part of key wrapping.	RSA_free()
DSA Key pair (2048, 3072, 4096 bits)		DSA public/private keys used to sign and verify data.	DSA_free()
ECDSA Key pair (P-224, P-256, P-384, P-521)		ECDSA public/private keys used to sign and verify data.	EC_KEY_free()
Diffie-Hellman Key pair	Public and private keys are generating using the SP 800-56Arev3 Safe Primes key generation method, random values are obtained from the SP800-90A DRBG.	Diffie-Hellman key pair used as part of the key agreement protocol.	DH_free()
EC Diffie-Hellman Key pair	Public and private keys are generated using the FIPS 186-4 key generation method, random values are obtained from the SP800 90A DRBG.	EC Diffie-Hellman key pair used as part of the key agreement protocol.	EC_KEY_free()
Shared secret	Generated during the Diffie-Hellman or EC Diffie-Hellman key agreement.	Used to derive the required keys by applying key derivation function.	DH_free(), EC_KEY_free()
Derived key	Generated during TLS KDF	Derive the master secret from the pre-master secret, and to derive the session keys from the master secret	EVP_PKEY_free()
DRBG Entropy input string	Obtained from CPU jitter source	Entropy input strings used as part of the DRBG process.	FIPS_drbg_free()
DRBG seed, Internal state (V, C, Key value)	Derived from entropy input during DRBG initialization.	The seed is used as input into the DRBG mechanism. V and key are used as part of HMAC and CTR DRBG process. V and C are used as part of HASH DRBG process.	FIPS_drbg_free()
TLS Pre-Master Secret and Master Secret	Established during the TLS handshake	Used as part of the TLS key establishment protocol	SSL_free(), SSL_clear()

Table 10: CSP Table

8.1 Random Number Generation

The module employs the Deterministic Random Bit Generator (DRBG) based on [SP 800-90A] for the random number generation. The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms with SHA-256. The module performs the DRBG health tests as defined in section 11.3 of [SP 800-90A]. The module uses CPU jitter as a noise source provided by the operational environment which is within the module's physical boundary but outside of the module's logical boundary. The source is compliant with [SP 800-90B] and marked as ENT on the certificate. The entropy source provides at least 223 bits of entropy, and the following caveat applies: The module generates keys whose strength are modified by available entropy.

8.2 Key Generation

For generating RSA, DSA and ECDSA keys, the module implements asymmetric key generation services compliant with [FIPS 186-4] and using DRBG compliant with [SP 800-90A]. A seed (i.e. the random value) used in asymmetric key generation is obtained from [SP 800-90A] DRBG. In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per NIST SP 800-133. The asymmetric seed is an unmodified output from a DRBG.

The public and private keys used in the EC Diffie-Hellman key agreement schemes are generated internally by the module using the ECDSA key generation method compliant with [FIPS186-4] and [SP 800-56Arev3]. The Diffie-Hellman key agreement scheme is also compliant with [SP 800-56Arev3], and generates keys using safe primes defined in RFC 7919 and RFC 3526, as described in the next section.

The module does not support manual key entry or intermediate key generation output.

8.3 Key/CSP Storage

Public and private keys are provided to the Module by the calling process and are destroyed when released by the appropriate API function calls. The Module does not perform persistent storage of keys.

8.4 Key/CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The application is responsible for calling the appropriate zeroization functions provided in the module's API and listed in Table 10. Calling the SSL_free() and SSL_clear() will zeroize the keys and CSPs stored in the TLS protocol internal state and also invoke the corresponding API functions listed in Table 10 to zeroize keys and CSPs. The zeroization functions overwrite the memory occupied by keys with "zeros" and deallocate the memory with the regular memory deallocation operating system call. In case of abnormal termination, or swap in/out of a physical memory page of a process, the keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.

8.5 Key/ Establishment / Key Transport

The module provides Diffie-Hellman and EC Diffie-Hellman shared secret computation compliant with SP 800-56Arev3, in accordance with scenario X1 (1) of IG D.8.

Additionally, the module provides Diffie-Hellman and EC Diffie-Hellman key agreement schemes compliant with SP 800-56rev3 and used as part of the TLS protocol key exchange in accordance with scenario X1 (2) of IG D.8;

that is, the shared secret computation (KAS-FFC-SSC and KAS-ECC-SSC) followed by the derivation of the keying material using SP 800-135 KDF.

For Diffie-Hellman, the module supports the use of safe primes from RFC 7919 for domain parameters and key generation, which are used in the TLS key agreement implemented by the module.

- TLS (RFC7919)
  - ffdhe2048 (ID = 256)
  - ffdhe3072 (ID = 257)
  - ffdhe4096 (ID = 258)
  - ffdhe6144 (ID = 259)
  - ffdhe8192 (ID = 260)

The module also supports the use of safe primes from RFC 3526, which are part of the Modular Exponential (MODP) Diffie-Hellman groups that can be used for Internet Key Exchange (IKE). Note that the module only implements key generation and verification, and shared secret computation using safe primes, but no part of the IKE protocol.

- IKEv2 (RFC3526)
  - MODP-2048 (ID=14)
  - MODP-3072 (ID=15)
  - MODP-4096 (ID=16)
  - MODP-6144 (ID=17)
  - MODP-8192 (ID=18)

According to Table 2: Comparable strengths in [SP 800-57], the key sizes of AES, Triple-DES, RSA, Diffie-Hellman and EC Diffie-Hellman provide the following security strength in FIPS mode of operation:

- Diffie-Hellman shared secret computation provides between 112 and 200 bits of encryption strength.
- EC Diffie-Hellman shared secret computation provides between 128 and 256 bits of encryption strength.
- Diffie-Hellman key agreement with TLS KDF provides between 112 and 200 bits of encryption strength.
- EC Diffie-Hellman key agreement with TLS KDF provides between 128 and 256 bits of encryption strength.
- RSA key wrapping with PKCS#1-v1.5 provides between 112 and 256 bits of encryption strength; Allowed per IG D.9
- AES key wrapping with KW and KWP key establishment methodology provides between 128 and 256 bits of encryption strength.
- AES key wrapping with CCM and GCM key establishment methodology provides between 128 and 256 bits of encryption strength.
- AES key wrapping with AES CBC and HMAC key establishment methodology provides 128 or 256 bits of encryption strength.
- Triple-DES Key wrapping with Triple-DES CBC and HMAC key establishment methodology provides 112 bits of encryption strength.

8.6 Key Derivation

The module supports the following key derivation method according to [SP 800-135]:

- KDF for the TLS protocol, used as pseudo-random functions (PRF) for TLSv1.0/1.1 and TLSv1.2

8.7 Key/CSP Entry and Output

The module does not support manual key entry or intermediate key generation key output. The keys are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form. This is allowed by [FIPS140-2_IG] IG 7.7, according to the "CM Software to/from App Software via GPC INT Path" entry on the Key Establishment Table.

9. Self-Tests

FIPS 140-2 requires that the Module performs self-tests to ensure the integrity of the Module, and the correctness of the cryptographic functionality at start up. In addition, conditional tests are required during operational stage of the module. All of these tests are listed and described in this section. See section 10.3 for descriptions of possible self-test errors and recovery procedure.

9.1 Power-Up Self-Tests

The Module performs power-up self-tests automatically during loading of the module by making use of default entry point (DEP) and no operator intervention is required. The module is not available for use until successful completion of power-up self-tests. Hence input, output, or any cryptographic functions cannot be performed while the Module is executing self-tests. The integrity of the module binary is verified using a HMAC SHA-256. The HMAC value is computed at build time and stored in the hmac file. The value is recalculated at runtime and compared against the stored value. If the comparison succeeds, then the remaining power-up self-test (consisting of the algorithm-specific Known Answer Tests) are performed. On successful completion of the power-up tests, the module becomes operational and crypto services are available. If any of the tests fails module transitions to error state and subsequent calls to the Module will fail - thus no further cryptographic operations will be possible.

Algorithm	Test
AES	KAT AES ECB mode with 128-bit key, encryption and decryption are tested separately. KAT AES CCM mode with 192-bit key, encryption and decryption are tested separately. KAT AES GCM mode with 256-bit key, encryption and decryption are tested separately. KAT AES XTS mode with 128 and 256 bit keys, encryption and decryption are tested separately.
Triple-DES	KAT Triple-DES ECB mode, encryption and decryption are tested separately.
DSA	PCT, sign and verify with L=2048 and SHA-256.
RSA	KAT RSA with 2048-bit key, PKCS#1 v1.5 scheme with SHA-224, SHA-256, SHA-384, SHA-512, signature generation tested separately. KAT RSA with 2048-bit key, PKCS#1 v1.5 scheme with SHA1, SHA-224, SHA-256, SHA-384, SHA-512, signature verification tested separately. KAT RSA with 2048-bit key, PSS scheme and SHA-224, SHA-256, SHA-384, SHA-512, signature generation tested separately. KAT RSA with 2048-bit key, PSS scheme and SHA1, SHA-224, SHA-256, SHA-384, SHA-512, signature verification tested separately. KAT RSA with 2048-bit key, public key encryption and private key decryption tested separately.
ECDSA	PCT ECDSA with P-256 and SHA-256, sign and verify
Diffie-Hellman	Primitive "Z" Computation KAT with 2048-bit key
EC Diffie-Hellman	Primitive "Z" Computation KAT with P-256 curve
TLS KDF	KAT with SHA-256
SP 800-90A CTR_DRBG	KAT CTR_DRBG with AES with 128, 192 and 256-bit keys
SP 800-90A Hash_DRBG	KAT Hash_DRBG with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
SP 800-90A HMAC_DRBG	KAT HMAC_DRBG with HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512
SP 800-90A DRBG	Health test per section 11.3 of SP 800-90A DRBG specification
HMAC	(SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) KAT
SHA	(1, 256, 512) KAT
CMAC	KAT AES CMAC with 128, 192 and 256 bit keys, MAC generation KAT Triple-DES CMAC, MAC generation
Module Integrity	HMAC-SHA-256

Table 11: Power-On Self-Tests

9.2 Conditional Self-Tests

Conditional tests are performed during operational state of the module when the respective crypto functions are used. If any of the conditional tests fails, module transitions to error state.

Algorithm	Test
DSA Key generation	Pairwise Consistency Test: signature generation and verification
ECDSA Key generation	Pairwise Consistency Test: signature generation and verification
RSA Key generation	Pairwise Consistency Test: signature generation and verification, encryption and decryption
DRBG NIST SP 800-90A	Continuous Random Number Generator Test

Table 12: Conditional Self-Tests

9.3 On-Demand self-tests

The module provides the Self-Test service to perform self-tests on demand. On demand self-tests can be invoked by powering-off and reloading the module. This service performs the same cryptographic algorithm tests executed during power-up. During the execution of the on-demand self-tests, crypto services are not available, and no data output or input is possible.

10. Crypto-Officer and User Guidance

This section provides guidance for the Cryptographic Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

10.1 Crypto-Officer Guidance

The version of the RPM containing the validated module is stated in section 3.1 above. The RPM package of the Module can be downloaded from the Oracle Linux 7 "Security Validation (Update 8)" yum repository and installed by standard tools recommended for the installation of Oracle packages on an Oracle Linux system (for example, yum, RPM, and the RHN remote management tool). The integrity of the RPM is automatically verified during the installation of the Module and the Crypto Officer shall not install the RPM file if the Oracle Linux Yum Server indicates an integrity error. The RPM files listed in section 3 are signed by Oracle and during installation; Yum performs signature verification which ensures as secure delivery of the cryptographic module. If the RPM packages are downloaded manually, then the CO should run 'rpm –K <rpm-file-name>' command after importing the builder's GPG key to verify the package signature. In addition, the CO can also verify the hash of the RPM package to confirm a proper download.

The OpenSSL static libraries libcrypto.a and libssl.a in openssl-static package are not approved to be used. The applications must be dynamically linked to run the OpenSSL.

To configure the operating environment to support FIPS Approved mode, perform the following steps:

1. Install RPM file, e.g for x86_64 use yum command:
   ```
   # yum install openssl-libs-1.0.2k-21.0.1.el7_9.x86_64.rpm
   ```
2. Ensure that the system is registered with the unbreakable Linux Network (ULN) and that the OL7_X86_64_latest channel is enabled
   ```
   # yum-config-manager --enable ol7_latest
   ```
3. Install the dracut-fips package:
   ```
   # yum install dracut-fips
   ```
4. Install the dracut-fips-aesni package (if AES-NI or ARM optimizations are supported):
   To check if AES-NI is supported run:
   ```
   # grep aes /proc/cpuinfo
   ```
   If it is supported, run:
   ```
   # yum install dracut-fips-aesni
   ```
5. Recreate the INITRAMFS image:
   ```
   # dracut -f
   ```
6. Perform the following steps to configure the boot loader so that the module is installed in FIPS validated manner:

   Method 1: Using Kernel Command Line Parameter:

   a) Identify the boot partition and the UUID of the partition. If /boot or /boot/efi resides on a separate partition, the kernel parameter boot=<partition of /boot or /boot/efi> must be supplied. The partition can be identified with the command:

   ```
   # df /boot or df /boot/efi
   ```

```
Filesystem   1K-blocks   Used    Available   Use%   Mounted on
/dev/sda1    233191      30454   190296      14%    /boot
```

```
# blkid /dev/sda1
```

```
/dev/sda1: UUID="6046308a-75fc-418e-b284-72d8bfad34ba" TYPE="xfs"
```

b) As the root user, edit the /etc/default/grub file as follows:

i. Add the fips=1 option to the boot loader configuration.

```
GRUB_CMDLINE_LINUX="vconsole.font=latarcyrheb-sun16
rd.lvm.lv=ol/swap rd.lvm.lv=ol/root crashkernel=auto
vconsole.keymap=uk rhgb quiet fips=1"
```

ii. If the contents of /boot reside on a different partition to the root partition, you must use the boot=UUID=boot_UUID line to the boot loader configuration to specify the device that should be mounted onto /boot when the kernel loads.

```
GRUB_CMDLINE_LINUX="vconsole.font=latarcyrheb-sun16
    rd.lvm.lv=ol/swap rd.lvm.lv=ol/root crashkernel=auto
    vconsole.keymap=uk rhgb quiet
    boot=UUID=6046308a-75fc-418e-b284-72d8bfad34ba fips=1"
```

iii. Save the changes.

This is required for FIPS to perform kernel validation checks, where it verifies the kernel against the provided HMAC file in the /boot directory.

Note:
On systems that are configured to boot with UEFI, /boot/efi is located on a dedicated partition as this is formatted specifically to meet UEFI requirements. This does not automatically mean that /boot is located on a dedicated partition.

Only use the boot= parameter if /boot is located on a dedicated partition. If the parameter is specified incorrectly or points to a non-existent device, the system may not boot.

If the system is no longer able to boot, you can try to modify the kernel boot options in grub to specify an alternate device for the boot=UUID=boot_UUID parameter, or remove the parameter entirely.

c) Rebuild the GRUB configuration as follows:

On BIOS-based systems, run the following command:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

On UEFI-based systems, run the following command:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

To ensure proper operation of the in-module integrity verification, prelinking must be disabled on all system files. By default, the prelink package is not installed on the system. However, if it is installed, disable prelinking on all libraries and binaries as follows:

Set PRELINKING=no in the /etc/sysconfig/prelink configuration file.

If the libraries were already prelinked, undo the prelink on all of the system files as follows:

```
# prelink –u –a
```

d) Reboot the system

e) Verify that FIPS Mode is enabled by running the command:

```
# cat /proc/sys/crypto/fips_enabled
```

The response should be "1"

<u>Method 2: Using Environment Variable:</u>

To support FIPS validated module, the operating environment can also be configured via environment variable OPENSSL_FORCE_FIPS_MODE. To enable it globally for users, use the steps below:

a) Open: /etc/bashrc
b) Set: export OPENSSL_FORCE_FIPS_MODE=1
c) Save and close the bashrc file.
d) Run: # source /etc/bashrc

The version of the RPM containing the validated Modules is the version listed in Section 3. The integrity of the RPM is automatically verified during the installation of the Modules and the Crypto Officer shall not install the RPM file if the RPM tool indicates an integrity error.

10.2 User Guidance

In order to run the module in FIPS mode, only the FIPS approved or allowed services listed in Table 8 or the validated or allowed cryptographic algorithms/security functions listed in Table 2 and
Table 3: Non-Approved but Allowed Security Functions
should be used. Interpretation of the return code is the responsibility of the host application. ENGINE_register_*, ENGINE_set_default_* and FIPS_mode_set(0) function calls are prohibited.

10.2.1 TLS and Diffie-Hellman

The TLS protocol implementation provides both server and client sides. In order to operate in FIPS mode, digital certificates used for server and client authentication shall comply with the restrictions of key size and message digest algorithms imposed by [SP 800-131A]. In addition, for Diffie-Hellman only the safe prime groups listed in RFC 7919 are approved to be used in FIPS mode.

10.2.2 Random Number Generator

The OpenSSL API call of RAND_cleanup must not be used. This call will clean up the internal DRBG state. This call also replaces the DRBG instance with the non-FIPS Approved SSLeay Deterministic Random Number Generator when using the RAND_* API calls.

10.2.3 AES GCM IV

The GCM encryption using internal IV generation method is compliant to IG A.5, scenario #1 (for TLS 1.2): Comply with the provision of a peer-to-peer industry standard. Specifically, following RFC 5288 for TLS. The counter portion of the IV is set by the module within its cryptographic boundary. When the IV exhausts the maximum number of possible values for a given session key, the first party, client, or server to encounter this condition will trigger a handshake to establish a new encryption key in accordance with RFC 5246.

In case the Modules' power is lost and then restored, the key used for the AES GCM encryption/decryption shall be re-distributed.

10.2.4 AES-XTS Guidance

The length of a single data unit encrypted or decrypted with the AES-XTS shall not exceed 2^{20} AES blocks that is 16MB of data per AES-XTS instance. An XTS instance is defined in section 4 of SP 800-38E.

The AES-XTS mode shall only be used for the cryptographic protection of data on storage devices. The AES-XTS shall not be used for other purposes, such as the encryption of data in transit. The module implements the check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

10.2.5 Triple-DES Keys

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES blocks (2GB of data), in accordance to SP 800-67 and IG A.13.

[SP 800-67] imposes a restriction on the number of 64-bit block encryptions performed under the same three-key Triple-DES key.

When the three-key Triple-DES is generated as part of a recognized IETF protocol, the module is limited to 2^{20} 64-bit data block encryptions. This scenario occurs in the following protocols:

- Transport Layer Security (TLS) versions 1.1 and 1.2, conformant with [RFC 5246]

In any other scenario, the module cannot perform more than 2^{16} 64-bit data block encryptions. The user is responsible for ensuring the module's compliance with this requirement.

10.2.6 Environment Variables

Setting the environment variable OPENSSL_ENFORCE_MODULUS_BITS can restrict the module to only generate the acceptable key sizes of RSA. If the environment variable is set, the module enforces the generation of keys of 2048 bits or more.

10.3 Handling Self-Test Errors

The Module transition to error state when any of self-tests or conditional tests fails. The application must be restarted to recover from these errors. Following are the error messages specific to self-test failure:

FIPS_R_FINGERPRINT_DOES_NOT_MATCH – The integrity verification check failed

FIPS_R_SELFTEST_FAILED – a known answer test failed
FIPS_R_TEST_FAILURE – a known answer test failed (RSA); pairwise consistency test failed (DSA)
FIPS_R_PAIRWISE_TEST_FAILED – a pairwise consistency test failed during EC/DSA or RSA key generation
FIPS_R_DRBG_STUCK – the DRBG generated two same consecutive values

These errors are reported through the regular ERR interface of the Module and can be queried by functions such as ERR_get_error(). See the OpenSSL manual page for the function description.

When the Module is in error state, output is inhibited, and no crypto operations are available. Any calls to the crypto functions in error state will return error message: 'FATAL FIPS SELFTEST FAILURE' on stderr and the application is terminated with the abort() call.

The only way to recover from the error state is to reload the module and restart the application. If failures persist, the Module must be reinstalled. If downloading the software, make sure to verify the package hash to confirm a proper download.

11. Mitigation of Other Attacks

RSA is vulnerable to timing attacks. In a setup where attackers can measure the time of RSA decryption or signature operations, blinding must be used to protect the RSA operation from that attack.

The API function of RSA_blinding_on turns blinding on for key rsa and generates a random blinding factor. The random number generator must be seeded prior to calling RSA_blinding_on.

Weak Triple-DES keys are detected as follows:

```
/* Weak and semi week keys as taken from
* %A D.W. Davies
* %A W.L. Price
* %T Security for Computer Networks
* %I John Wiley & Sons
* %D 1984
* Many thanks to smb@ulysses.att.com (Steven Bellovin) for the reference
* (and actual cblock values).
*/
#define NUM_WEAK_KEY 16
static const DES_cblock weak_keys[NUM_WEAK_KEY]={
/* weak keys */
        {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},
        {0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE},
        {0x1F,0x1F,0x1F,0x1F,0x0E,0x0E,0x0E,0x0E},
        {0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},
        /* semi-weak keys */
        {0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},
        {0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},
        {0x1F,0xE0,0x1F,0xE0,0x0E,0xF1,0x0E,0xF1},
        {0xE0,0x1F,0xE0,0x1F,0xF1,0x0E,0xF1,0x0E},
        {0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},
        {0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01},
        {0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E,0xFE},
        {0xFE,0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E},
        {0x01,0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E},
        {0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E,0x01},
        {0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1,0xFE},
        {0xFE,0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1}};
```

Please note that there is no weak key detection by default. The caller can explicitly set the DES_check_key to 1 or call DES_check_key_parity() and/or DES_is_weak_key() functions on its own.

Acronyms, Terms and Abbreviations

Term	Definition
AES	Advanced Encryption Standard
AVX	Advanced Vector Extensions
CAVP	Cryptographic Algorithm Validation Program
CKG	Cryptographic Key Generation
CMVP	Cryptographic Module Validation Program
CTASM	Constant time assembler
CSE	Communications Security Establishment
CSP	Critical Security Parameter
DH	Diffie-Hellman
DHE	Diffie-Hellman Ephemeral
DRBG	Deterministic Random Bit Generator
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EDC	Error Detection Code
GPG	Gnu Privacy Guard
HMAC	(Keyed) Hash Message Authentication Code
IKE	Internet Key Exchange
IPSEC	Internet Protocol Security
KAT	Known Answer Test
KDF	Key Derivation Function
LRNG	Linux Random Number Generator
NIST	National Institute of Standards and Technology
PAA	Processor Algorithm Acceleration
POST	Power On Self-Test
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
PUB	Publication
SHA	Secure Hash Algorithm
SSSE3	Supplemental Streaming SIMD Extensions 3
UEK	Oracle Linux Unbreakable Enterprise Kernel
TLS	Transport Layer Security

Table 13: Acronyms

References

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

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

Document	Author	Title
FIPS PUB 140-2	NIST	FIPS PUB 140-2: Security Requirements for Cryptographic Modules
FIPS IG	NIST	Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program
FIPS PUB 140-2 Annex A	NIST	FIPS 140-2 Annex A: Approved Security Functions
FIPS PUB 140-2 Annex B	NIST	FIPS 140-2 Annex B: Approved Protection Profiles
FIPS PUB 140-2 Annex C	NIST	FIPS 140-2 Annex C: Approved Random Number Generators
FIPS PUB 140-2 Annex D	NIST	FIPS 140-2 Annex D: Approved Key Establishment Techniques
DTR for FIPS PUB 140-2	NIST	Derived Test Requirements (DTR) for FIPS PUB 140-2, Security Requirements for Cryptographic Modules
NIST SP 800-67	NIST	Recommendation for the Triple Data Encryption Algorithm TDEA Block Cipher
FIPS PUB 197	NIST	Advanced Encryption Standard
FIPS PUB 198-1	NIST	The Keyed Hash Message Authentication Code (HMAC)
FIPS PUB 186-4	NIST	Digital Signature Standard (DSS)
FIPS PUB 180-4	NIST	Secure Hash Standard (SHS)
NIST SP 800-131Ar1	NIST	Recommendation for the Transitioning of Cryptographic Algorithms and Key Sizes
PKCS#1	RSA Laboratories	PKCS#1 v2.1: RSA Cryptographic Standard
RFC 5288	https://tools.ietf.org/html/rfc5288	AES Galois Counter Mode (GCM) Cipher Suites for TLS

Table 14: References