SUSE Linux Enterprise OpenSSL Cryptographic Module

version 4.2

FIPS 140-3 Non-Proprietary Security Policy

Version 1.2

Last update: 2024-06-17

Prepared by:

atsec information security corporation

4516 Seton Center Parkway, Suite 250

Austin, TX 78759

www.atsec.com

1 Table of Contents

1 General

This document is the non-proprietary FIPS 140-3 Security Policy for version 4.2 of the SUSE Linux Enterprise OpenSSL Cryptographic Module. It has a one-to-one mapping to the [SP800-140B] starting with section B.2.1 named “General” that maps to section 1 in this document and ending with section B.2.12 named “Mitigation of other attacks” that maps to section 12 in this document.

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic Module Specification	1
3	Cryptographic Module Interfaces	1
4	Roles, Services, and Authentication	1
5	Software/Firmware Security	1
6	Operational Environment	1
7	Physical Security	N/A
8	Non-invasive Security	N/A
9	Sensitive Security Parameter Management	1
10	Self-tests	1
11	Life-cycle Assurance	1
12	Mitigation of Other Attacks	1

Table 1 - Security Levels

2 Cryptographic Module Specification

2.1 Module Embodiment

The SUSE Linux Enterprise OpenSSL Cryptographic Module (hereafter referred to as “the module”) is a Software multi-chip standalone cryptographic module.

2.2 Module Design, Components, Versions

The software block diagram below shows the cryptographic boundary of the module, and its interfaces with the operational environment.

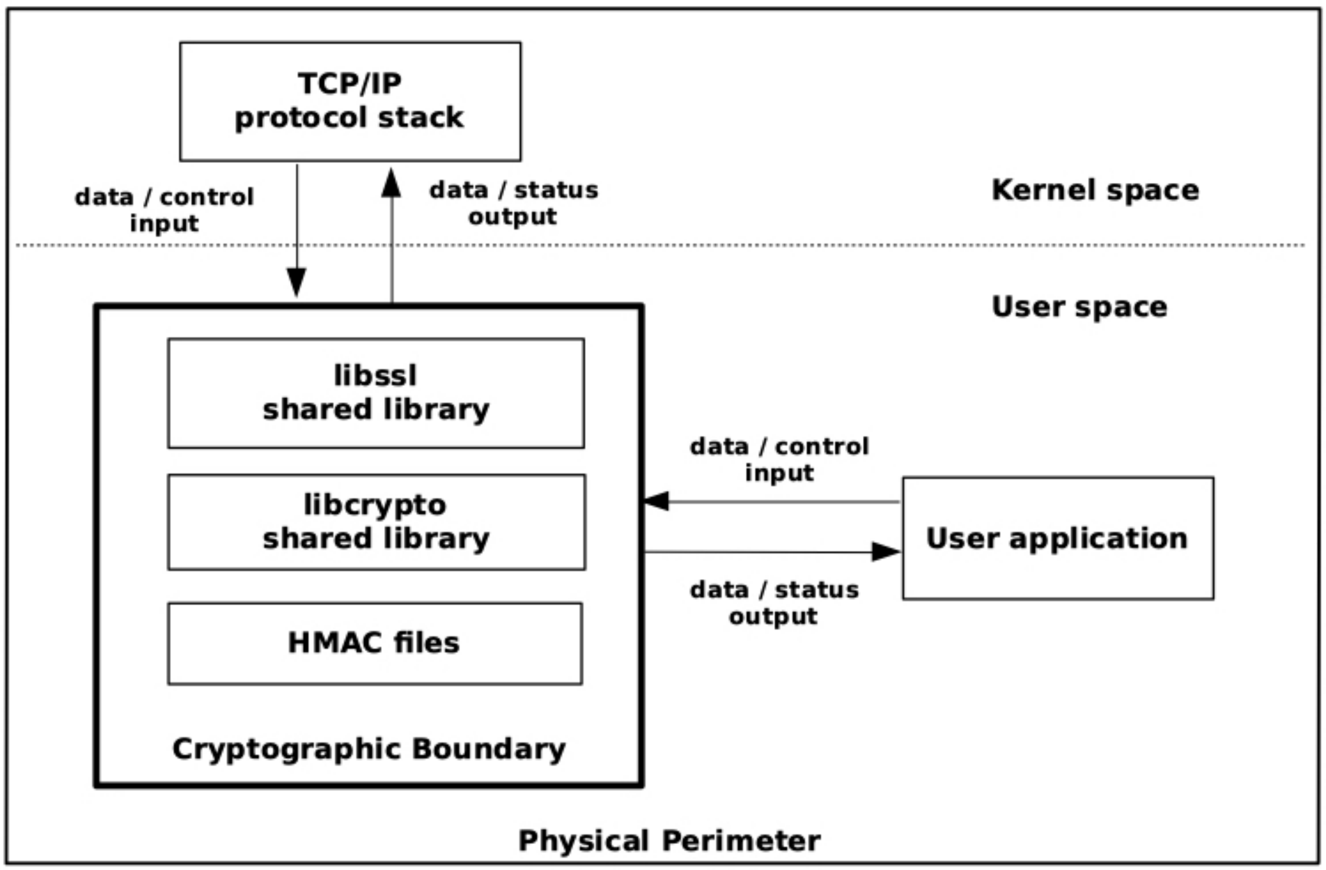

Figure 1 - Cryptographic boundary

Table 2 lists the software components of the cryptographic module, which defines its cryptographic boundary.

Components	Description
libcrypto.so.1.1	Shared library for cryptographic algorithms.
.libcrypto.so.1.1.hmac	Integrity check HMAC value for the libcrypto shared library.
libssl.so.1.1	Shared library for TLS/DTLS network protocols.
.libssl.so.1.1.hmac	Integrity check HMAC value for the libssl shared library.

Table 2 – Cryptographic Module Components

2.3 Modes of operation

When the module starts up successfully, after passing all the pre-operational and conditional cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved services listed in Table 12. Please see section 4 for the details on service indicator provided by the module that identifies when an approved service is called.

2.4 Tested Operational Environments

The module has been tested on the following platforms with the corresponding module variants and configuration options:

#	Operating System	Hardware Platform	Processor	PAA/Acceleration	Module version
1	SUSE Linux Enterprise Server 15 SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)	32-bit 64-bit
2	SUSE Linux Enterprise Server 15 SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)	64-bit
3	SUSE Linux Enterprise Server 15 SP4	GIGABYTE G242-P32-QZ	ARM Ampere® Altra® Q80-30	With and without Cryptography Extensions (PAA)	64-bit
4	SUSE Linux Enterprise Server 15 SP4	IBM z/15	z15	With and without CPACF (PAI)	64-bit
5	SUSE Linux Enterprise Server 15 SP4 on PowerVM (VIOS 3.1.4.00)	IBM Power E1080 (9080-HEX)	Power10	With and without ISA (PAA)	64-bit

Table 3 - Tested Operational Environments

2.5 Vendor-Affirmed Operational Environments

In addition to the platforms listed in Table 3, SUSE has also tested the module on the platforms in Table 4 and Table 5, and claims vendor affirmation on them.

Note: the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.5.1 OpenSSL 64-bit Vendor Affirmed Operational Environments

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	SUSE Linux Enterprise Server 15SP4	IBM LinuxONE III LT1	z15	With and without CPACF (PAI)
2	SUSE Linux Enterprise Micro 5.3	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
3	SUSE Linux Enterprise Micro 5.3	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
4	SUSE Linux Enterprise Micro 5.3	GIGABYTE G242-P32-QZ	ARM Ampere® Altra® Q80-30	With and without Cryptography Extensions (PAA)
5	SUSE Linux Enterprise Micro 5.3	IBM z/15	z15	With and without CPACF (PAI)
6	SUSE Linux Enterprise Micro 5.3	IBM LinuxONE III LT1	z15	With and without CPACF (PAI)
7	SUSE Linux Enterprise Server for SAP 15SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
8	SUSE Linux Enterprise Server for SAP 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
9	SUSE Linux Enterprise Server for SAP 15SP4 on PowerVM (VIOS 3.1.4.00)	IBM Power E1080 (9080-HEX)	Power10	With and without ISA (PAA)
10	SUSE Linux Enterprise Base Container Image 15SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
11	SUSE Linux Enterprise Base Container Image 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
12	SUSE Linux Enterprise Base Container Image 15SP4	GIGABYTE G242-P32-QZ	ARM Ampere® Altra® Q80-30	With and without Cryptography Extensions (PAA)
13	SUSE Linux Enterprise Base Container Image 15SP4	IBM z/15	z15	With and without CPACF (PAI)
14	SUSE Linux Enterprise Base Container Image 15SP4	IBM LinuxONE III LT1	z15	With and without CPACF (PAI)
15	SUSE Linux Enterprise Base Container Image 15SP4 on PowerVM (VIOS 3.1.4.00)	IBM Power E1080 (9080-HEX)	Power10	With and without ISA (PAA)

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
16	SUSE Linux Enterprise Desktop 15SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
17	SUSE Linux Enterprise Desktop 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
18	SUSE Linux Enterprise Real Time 15SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
19	SUSE Linux Enterprise Real Time 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)

Table 4 - Vendor-Affirmed Operational Environments for OpenSSL (64-bit)

2.5.2 OpenSSL 32-bit Vendor Affirmed Operational Environments

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	SUSE Linux Enterprise Server 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
2	SUSE Linux Enterprise Server for SAP 15SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
3	SUSE Linux Enterprise Server for SAP 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
4	SUSE Linux Enterprise Desktop 15SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
5	SUSE Linux Enterprise Desktop 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
6	SUSE Linux Enterprise Real Time 15SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
7	SUSE Linux Enterprise Real Time 15SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)

Table 5 - Vendor-Affirmed Operational Environments for OpenSSL (32-bit)

2.6 Approved Algorithms

Table 6 lists all the approved security functions of the module, including specific key strengths employed for approved services.

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	AES FIPS197, SP800-38A, SP800-38C	CBC, CCM, CFB1, CFB8, CFB128, CTR, OFB	128, 192, 256-bit keys with 128-256 bits of security strength	Symmetric encryption; Symmetric decryption
A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	AES SP800-38B	CMAC	128, 192, 256-bit keys with 128-256 bits of security strength	Message authentication code (MAC)
A3136, A3137, A3138, A3140, A3141, A3142, A3143, A3149, A3150, A3154, A3157, A3158, A3160, A3162, A3163, A3165, A3166, A3167, A3169, A3170, A3171, A3172	AES FIPS197, SP800-38A	ECB	128, 192, 256-bit keys with 128-256 bits of security strength	Symmetric encryption; Symmetric decryption
A3151, A3152, A3153, A3155, A3159, A3176, A3177, A3178, A3179, A3180, A3181, A3182, A3183, A3184, A3190, A3194, A3195, A3196, A3197, A3198, A3199, A3200, A3201, A3204, A3205, A3206	AES SP800-38D	GCM with internal IV (IV Gen Mode 8.2.1)	128, 192, 256-bit keys with 128-256 bits of security strength	Symmetric encryption; Symmetric decryption

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3151, A3152, A3153, A3155, A3159, A3176, A3177, A3178, A3179, A3180, A3181, A3182, A3183, A3184, A3190, A3194, A3195, A3196, A3197, A3198, A3199, A3200, A3201, A3204, A3205, A3206	AES SP800-38D	GCM with external IV (IV Gen Mode 8.2.1)	128, 192, 256-bit keys with 128-256 bits of security strength	Symmetric decryption
A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	AES SP800-38F	KW, KWP	128, 192, 256-bit keys with 128-256 bits of security strength	Key wrapping and unwrapping
A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	AES SP800-38E	XTS	128, 256-bit keys with 128-256 bits of security strength	Symmetric encryption; Symmetric decryption (for data storage)
Vendor Affirmed	CKG SP800-133rev2	FIPS186-4, SP800-56Arev3, SP800-90Arev1	RSA: 2048 to 16384-bit keys with 112-256 bits of security strength ECDSA: P-224, P-256, P-384, P-521-bit keys with 112-256 bits of security strength Safe Primes: 2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of security strength	Key pair generation
A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	DRBG SP800-90Arev1	CTR_DRBG: AES-128, AES-192, AES-256 with/without DF, with/without PR	128, 192, 256-bit keys with 128, 192 and 256 bits of security strength	Random number generation
A3147, A3156, A3185, A3186, A3187, A3188,	ECDSA FIPS186-4	B.4.2 Testing Candidates	P-224, P-256, P-384, P-521 with 112-256 bits of security strength	Key pair generation

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3193, A3202, A3203, A3210		N/A	P-224, P-256, P-384, P-521 with 112-256 bits of security strength	Key pair validation, ECDSA Public key validation
A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210		SHA2-224, SHA2-256, SHA2-384, SHA2-512 SHA3-224, SHA3-256, SHA3-384, SHA3-512	P-224, P-256, P-384, P-521 with 112-256 bits of security strength	Digital signature generation
		SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512 SHA3-224, SHA3-256, SHA3-384, SHA3-512	P-192, P-224, P-256, P-384, P-521 with 80-256 bits of security strength	Digital signature verification (usage of P-192 curve or SHA-1 are considered Legacy Use)
E22, E28, E29, E30	ESV SP800-90B	N/A	N/A	Entropy source
A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	HMAC FIPS198-1	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	≥ 112-bit keys with 112-256 bits of security strength	Message authentication code (MAC)
A3161		SHA2-256		
A3144, A3145, A3146, A3148, A3173, A3174, A3175		SHA3-224, SHA3-256, SHA3-384, SHA3-512	≥ 112-bit keys with 112 -256 bits of security strength	Message authentication code (MAC)
A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	KAS-ECC-SSC SP800-56Arev3	ECC Ephemeral Unified Scheme	P-224, P-256, P-384, P-521 with 112-256 bits of security strength	(EC Diffie-Hellman) Key agreement
A3207, A3211	KAS-FFC-SSC SP800-56Arev3	dhEphem Scheme with safe prime groups	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with keys with 112-200 bits of security strength	(Diffie-Hellman) key agreement

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
CVL. A3139, A3168	KDA HKDF SP800-56Crev1	SHA2-224, SHA2-256, SHA2-384, SHA2-512	N/A	Key derivation for TLS v1.3 used in the TLS protocol service
CVL. A3140, A3141, A3142, A3143, A3149, A3157, A3169, A3170, A3171, A3172	KDF SSH SP800-135rev1	AES with SHA-1, SHA2-256, SHA2-384, SHA2-512	128, 192, 256-bit keys with 128-256 bits of security strength	Key derivation
CVL. A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	KDF TLS SP800-135rev1 RFC7627	TLS v1.0, v1.1, v1.2	N/A	Key derivation
A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	KTS SP800-38F	AES KW, KWP	128, 192, 256-bit keys with 128-256 bits of security strength	Key wrapping; Key unwrapping
A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	KTS SP800-38C	AES CCM	128, 256-bit keys with 128, 256 bits of security strength	Key wrapping and key unwrapping (as part of the cipher suites in the TLS protocol)
A3151, A3152, A3153, A3155, A3159, A3176, A3177, A3178, A3179, A3180, A3181, A3182, A3183, A3184, A3190, A3194, A3195, A3196, A3197, A3198, A3199, A3200, A3201, A3204, A3205, A3206	KTS SP800-38D	AES GCM	128, 256-bit keys with 128, 256 bits of security strength	

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
(AES) A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167 (HMAC) A3144, A3145, A3146, A3147, A3148, A3156, A3161, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	KTS SP800-38A, FIPS198-1	AES CBC and HMAC	128, 256-bit keys with 128, 256 bits of security strength	
A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	PBKDF SP800-132	HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	N/A	Key derivation
A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210 Key sizes other than mentioned here and up to 16384 bits are not CAVP tested but approved per IG C.F	RSA FIPS186-4	B.3.3 Random Probable Primes	2048, 3072 and 4096-bit keys with 112-149 bits of security strength Key sizes greater than 4096 bits provide 150-256 bits of security strength.	Key pair generation
		PKCS#1v1.5: SHA2-224, SHA2-256, SHA2-384, SHA2-512	2048, 3072 and 4096-bit keys with 112-149 bits of security strength	Digital signature generation
		PSS: SHA2-224, SHA2-256, SHA2-384, SHA2-512	2048, 3072 and 4096-bit keys with 112-149 bits of security strength	
		X9.31: SHA2-256, SHA2-384, SHA2-512	2048, 3072 and 4096-bit keys with 112-149 bits of security strength	

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
		PKCS#1v1.5: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	1024, 2048, 3072 and 4096-bit keys with 80-149 bits of security strength	Digital signature verification (usage of 1024-bit keys or SHA-1 is considered Legacy Use)
		PSS: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	1024, 2048, 3072 and 4096-bit keys with 80-149 bits of security strength	
		X9.31: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	1024, 2048, 3072 and 4096 keys with 80-149 bits of security strength	
A3207, A3211	Safe Primes SP800-56Arev3	Section 5.6.1.1.4 Testing Candidates	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 keys with 112-200 bits of security strength	Key pair generation, Diffie-Hellman public key validation
A3144, A3145, A3146, A3148, A3173, A3174, A3175	SHA-3 FIPS202	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256	N/A	Message digest
A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	SHS FIPS180-4	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	N/A	Message digest
A3161		SHA2-256		

Table 6 – Approved Algorithms

2.7 Non-Approved Algorithms Allowed in the Approved Mode of Operation

The module does not implement non-approved algorithms that are allowed in the approved mode of operation.

2.8 Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed

Table 7 lists the non-approved algorithms that are allowed in the approved mode of operation with no security claimed. These algorithms are used by the approved services listed in Table 10.

Algorithm[1]	Caveat	Use/Function
MD5	Only allowed as the PRF in TLSv1.0 and v1.1 per IG 2.4.A	Message digest used in TLSv1.0 / v1.1 KDF only

Table 7 - Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed

2.9 Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

Table 8 lists non-approved algorithms that are not allowed in the approved mode of operation. These algorithms are used by the non-approved services listed in Table 12.

Algorithm/Functions	Use/Function
AES(GCM) with external IV	Symmetric encryption
ARIA	Symmetric encryption; Symmetric decryption
Blake2	Message digest
Blowfish	Symmetric encryption; Symmetric decryption
Camellia	Symmetric encryption; Symmetric decryption
CAST	Symmetric encryption; Symmetric decryption
CAST5	Symmetric encryption; Symmetric decryption
ChaCha20	Symmetric encryption; Symmetric decryption
DES	Symmetric encryption; Symmetric decryption
Chacha20 and Poly1305	Authenticated encryption; Authenticated decryption
CMAC with Triple-DES	Message authentication code (MAC)
Diffie-Hellman with keys generated with domain parameters other than safe primes	Key pair generation; Diffie-Hellman public key validation; Key agreement; Shared secret computation
DSA with any key sizes	Key pair generation; Domain parameter generation, Digital signature generation; Digital signature verification

[1] These algorithms do not claim any security and are not used to meet FIPS 140-3 requirements. Therefore, SSPs do not map to these algorithms.

Algorithm/Functions	Use/Function
EC Diffie-Hellman with P-192 curve, K curves, B curves and non-NIST curves	Key agreement; Shared secret computation
ECDSA with P-192 curve, K curves, B curves and non-NIST curves	Key pair generation; Key pair validation; ECDSA public key validation
ECDSA with P-192 curve, K curves, B curves and non-NIST curves	Digital signature generation; Digital signature verification
GHASH	Message digest
Gost	Message digest
HKDF	Key derivation as a standalone service
HMAC with less than 112-bit keys	Message authentication Code (MAC)
KDF SSH using Triple-DES	Key derivation
MD4	Message digest
MD5	Message digest
MDC2	Message digest
Multiblock ciphers using AES in CBC mode with 128- and 256-bit keys and HMAC SHA-1 and SHA2-256 (available only in Intel processors with AES-NI capability)	Authenticated encryption; Authenticated decryption
PBKDF with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 11.2.4	Key derivation
RC2	Symmetric encryption; Symmetric decryption
RC4	Symmetric encryption; Symmetric decryption
RMD160	Message digest
RSA with keys smaller than 2048 bits	Key pair generation; Domain parameter verification; Digital signature generation
RSA with keys smaller than 1024 bits	Digital signature verification
RSA encryption and decryption with any key sizes	Key encapsulation
SEED	Symmetric encryption; Symmetric decryption
SHA-1	Digital signature generation
SipHash	Message authentication code (MAC)
SM3	Message digest
SM4	Symmetric encryption; Symmetric decryption
Triple-DES	Symmetric encryption; Symmetric decryption

Table 8 - Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

3 Cryptographic Module Ports and Interfaces

As a software-only module, the module does not have physical ports. The operator can only interact with the module through the API provided by the module. Thus, the physical ports are interpreted to be the physical ports of the hardware platform on which the module runs. The following table shows the logical interfaces implemented in the module.

All data output via data output interface is inhibited when the module is performing pre-operational test or zeroization or when the module enters error state.

Logical Interface	Data that passes over port/interface
Data Input	API input parameters, kernel I/O network or files on filesystem, TLS protocol input messages.
Data Output	API output parameters, kernel I/O network or files on filesystem, TLS protocol output messages.
Control Input	API function calls, API input parameters for control.
Status Output	API return codes, API output parameters for status output.

Table 9 - Ports and Interfaces

Note: The module does not implement a control output interface.

4 Roles, services, and authentication

4.1 Services

The module supports the Crypto Officer role only. This sole role is implicitly assumed by the operator of the module when performing a service. The module does not support authentication.

Role	Service	Input	Output
Crypto Officer (CO)	Symmetric encryption	Plaintext, key	Ciphertext
	Symmetric decryption	Ciphertext, key	Plaintext
	Authenticated encryption	Plaintext, key	Ciphertext, authentication tag
	Authenticated decryption	Ciphertext, authentication tag, key	Plaintext
	Key pair generation	Key size	Key pair
	Domain parameter generation	Key size	Domain parameters
	Domain parameter verification	Domain parameters	Return codes/log messages
	Key pair validation	Key pair	Return codes/log messages
	Key agreement	Key pair	Shared secret
	Digital signature generation	Message, hash algorithm, private key	Signature
	Digital signature verification	Signature, hash algorithm, public key	Signature verification result
	Random number generation	Size	Random number
	Message digest	Message	Message digest
	Message authentication code (MAC)	Message, key	Message authentication code
	Key wrapping	Key to be wrapped, key wrapping key	Wrapped key
	Key unwrapping	Wrapped key, key unwrapping key	Unwrapped key

Role	Service	Input	Output
	Key encapsulation	Key to be encapsulated, key encapsulating key	Encapsulated key
	Shared secret computation	Private key, public key from peer	Shared secret
	Diffie-Hellman key generation using safe primes	Safe prime	Key pair
	Public key validation	Public key	Return codes/log messages
	TLS Key derivation	TLS pre-master secret	Derived key
	SSH Key Derivation	Shared secret	Derived key
	PBKDF Key Derivation	Password/passphrase	Derived key
	HKDF Key Derivation	Shared secret	Derived key
	Show status	None	Return codes/log messages
	Zeroization	Context containing SSPs	None
	Self-test	Module reset	Self-test results
	On-demand integrity test	None	Self-test results
	Module installation and configuration	None	Log messages
	Module initialization	None	Log messages
	Show module name and version	None	Name and version of the module
	Transport Layer Security (TLS) network protocol	Application data	Application data

Table 10 - Roles, Service Commands, Input and Output

The module provides services to the users that assume one of the available roles. All services are shown in Table 11 and Table 12.

4.2 Approved Services

Table 11 lists the approved services. For each service, the table lists the associated cryptographic algorithm(s), the role to perform the service, the cryptographic keys or SSPs involved, and their access type(s). No support of intermediate key generation is provided. The following convention is used to specify access rights to an SSP:

- **G** = **Generate**: The module generates or derives the SSP.
- **R** = **Read**: The SSP is read from the module (e.g., the SSP is output).
- **W** = **Write**: The SSP is updated, imported, or written to the module.
- **E** = **Execute**: The module uses the SSP in performing a cryptographic operation.
- **Z** = **Zeroize**: The module zeroizes the SSP.
- **N/A**: the calling application does not access any SSP or key during its operation.

The details of the approved cryptographic algorithms including the CAVP certificate numbers can be found in Table 6.

The “Indicator” column shows the service indicator API functions that must be used to verify the service indicator for each of the services. A value of 1 indicates that the service is approved, and 0 indicates that the service is non-approved.

Additionally there is a separate indicator used for the following services.

- The API function used to determine the indicator for the “TLS network protocol” service returns the cipher suite established for the TLS session. If the returned cipher suite ID belongs to one of the cipher suites listed in Table 20 of Appendix A, then the service is approved, otherwise, it is non-approved.
- The “Show status”, “Zeroization”, “Self-tests” and “Show module name and version” services listed under the “Other FIPS-related Services” group are always approved. The service level indicator is implicit.

For more information, see the “FIPS server level indicator” man pages.

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Cryptographic Services						
Symmetric encryption	Perform AES encryption	AES	AES key	CO	W, E	fips_sli_is_approved_EVP_CIPHER_CTX returns 1
Symmetric decryption	Perform AES decryption	AES	AES key		W, E	fips_sli_is_approved_EVP_CIPHER_CTX returns 1
Key pair generation	Generate RSA key pairs	RSA, DRBG	RSA public key, RSA private key		E, G, R	fips_sli_is_approved_EVP_PKEY_CTX returns 1
	Generate ECDSA key pairs	ECDSA, DRBG	ECDSA public key, ECDSA private key		E, G, R	fips_sli_is_approved_EVP_PKEY_CTX returns 1
Digital signature generation	Sign using RSA	RSA, SHS	RSA private key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1
	Sign using ECDSA	ECDSA, DRBG, SHS	ECDSA private key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1
Digital signature verification	Verify RSA signatures	RSA, SHS	RSA public key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1
	Verify ECDSA signatures	ECDSA, SHS	ECDSA public key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Key pair validation	Validate ECDSA public key	ECDSA	ECDSA public key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1
			ECDSA private key		W, E	
ECDSA public key validation	Validate ECDSA public key	ECDSA	ECDSA public key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1
Random number generation	Generate random bitstrings	DRBG	Entropy input		W, E	fips_sli_RAND_bytes_is_approved returns 1 fips_sli_RAND_priv_bytes_is_approved returns 1
			DRBG seed , DRBG internal state (V, Key)		E, G	
Message digest	Compute SHA hashes	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	None		N/A	fips_sli_SHA*_is_approved returns 1
		SHA3-224, SHA3-256, SHA3-384, SHA3-512				
Message authentication code (MAC)	Compute HMAC	HMAC	HMAC key		W, E	fips_sli_HMAC_is_approved returns 1
	Compute and AES-based CMAC	CMAC with AES	AES key			fips_sli_is_approved_CMAC_CTX returns 1
Key wrapping	Perform AES-based key wrapping	AES-KW, AES-KWP	AES key		W, E	fips_sli_is_approved_EVP_CIPHER_CTX returns 1
Key unwrapping	Perform AES-based key unwrapping	AES-KW, AES-KWP	AES key		W, E	fips_sli_is_approved_EVP_CIPHER_CTX returns 1
Shared secret computation	Diffie-Hellman shared secret computation	KAS-FFC-SSC	Diffie-Hellman public key, Diffie-Hellman private key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1
			Diffie-Hellman shared secret		G, R	
	EC Diffie-Hellman shared secret computation	KAS-ECC-SSC	EC Diffie-Hellman public key, EC Diffie-Hellman private key		W, E	fips_sli_is_approved_EVP_PKEY_CTX returns 1
			EC Diffie-Hellman shared secret		G, R	

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Diffie-Hellman key generation	Perform Diffie-Hellman key generation with safe primes	Safe Primes Key Generation	Diffie-Hellman public key, Diffie-Hellman private key		E, G, R	`fips_sli_is_approved_EVP_PKEY_CTX` returns 1
Diffie-Hellman public key validation	Perform Diffie-Hellman public key validation	Safe Primes Public key validation	Diffie-Hellman public key		W, E	`fips_sli_is_approved_EVP_PKEY_CTX` returns 1
Key derivation	Perform key derivation	TLS KDF	TLS pre-master secret		W, E	`fips_sli_is_approved_EVP_KDF_CTX` returns 1
			TLS master secret		W, E, G	
			TLS derived key		G, R	
		SSH KDF	Diffie-Hellman or EC Diffie-Hellman shared secret		W, E	`fips_sli_is_approved_EVP_KDF_CTX` returns 1
			SSH derived key		G, R	
		PBKDF KDF	Password/passphrase		W, E	`fips_sli_PKCS5_PBKDF2_HMAC_is_approved` returns 1
			PBKDF Derived key		G, R	
Transport Layer Security (TLS) network protocol	Provide supported cipher suites in the approved mode	Supported cipher suites in the approved mode (see Appendix A for the complete list of valid cipher suites)	RSA public key, RSA private key, ECDSA public key, ECDSA private key		W, E	`SSL_CIPHER_get_protocol_id` or `SSL_get_current_cipher` return a two-byte ID matching an approved cipher suite (listed in Appendix C).
			TLS pre-master secret, TLS master secret, , Diffie-Hellman private key, EC Diffie-Hellman private key, TLS derived key		E, G	
			Diffie-Hellman public key, EC Diffie-Hellman public key,		W, E, G, R	
Other FIPS-related Services						
Show status	Show module status	N/A	None	CO	N/A	Implicit (always approved)
Zeroization	Zeroize SSPs	N/A	All SSPs		Z	Implicit (always approved)

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Self-tests	Perform self-tests	AES, Diffie-Hellman, EC Diffie-Hellman, ECDSA, DRBG, HMAC, RSA, SHS	None		N/A	Implicit (always approved)
Show module name and version	Show module name and version	N/A	None		N/A	Implicit (always approved)

Table 11 - Approved Services

Table 12 lists the non-approved services. The details of the non-approved cryptographic algorithms available in non-approved mode can be found in Table 8.

Service	Description	Algorithms Accessed	Roles
Cryptographic Services			
Symmetric encryption	Compute the cipher for encryption	ARIA, Blowfish, Camellia, ChaCha20, CAST, CAST5, DES, RC2, RC4, SEED, Triple-DES	CO
Symmetric decryption	Compute the plaintext for decryption		
Authenticated encryption	Compute authenticated encryption cipher	AES-GCM with external IV	
Authenticated encryption	Compute authenticated encryption cipher	AES and SHA from multi-buffer or stitch implementations listed in Table 8, ChaCha20 and Poly1305	
Authenticated decryption	Compute plaintext from authenticated encryption		
Key pair generation	Generate RSA, DSA, and ECDSA key pairs	RSA, DSA and ECDSA with restrictions listed in Table 8	
Digital signature generation	Sign using RSA, DSA or ECDSA	RSA, DSA and ECDSA and message digest restrictions listed in Table 8	
Digital signature verification	Verify RSA, DSA, ECDSA signatures		
Message digest	Compute message digest	Blake2, Gost, MD4, MD5, MDC2, RMD160	
Message authentication code (MAC)	Compute HMAC and CMAC	HMAC and CMAC with restrictions listed in Table 8	
Key encapsulation	Perform RSA key encapsulation	RSA encryption and decryption with any key sizes	
Shared secret computation	Perform Diffie-Hellman or EC Diffie-Hellman key agreement	Diffie-Hellman and EC Diffie-Hellman restrictions listed in Table 8	

Service	Description	Algorithms Accessed	Roles
Key derivation	Perform key derivation	KDF SSH using Triple-DES	
		HKDF (as a standalone service)	
		PBKDF using non-approved message digest or input parameters not meeting requirements stated in section 11.2.4	
Network Protocol Services			
Transport Layer Security (TLS) network protocol	Provide non-supported cipher suites	Non-supported cipher suites (see Appendix A for the complete list of valid cipher suites)	CO

Table 12 - Non-Approved Services

5 Software/Firmware security

5.1 Integrity Techniques

The integrity of the module is verified by comparing an HMAC-SHA2-256 value calculated at run time with the HMAC value stored in the .hmac file that was computed at build time for each software component of the module listed in section 2. If the HMAC values do not match, the test fails, and the module enters the error state.

5.2 On-Demand Integrity Test

On-Demand integrity tests can be invoked by powering-off and reloading the module.

5.3 Executable Code

The module consists of executable code in the form of libcrypto and libssl shared libraries as stated in section 2.

6 Operational Environment

6.1 Applicability

This module operates in a modifiable operational environment per the FIPS 140-3 level 1 specifications. The SUSE Linux Enterprise Server operating system is used as the basis of other products. Compliance is maintained for SUSE products whenever the binary is found unchanged per the vendor affirmation from SUSE based on the allowance FIPS 140-3 management manual [FIPS140-3_MM] section 7.9.1 bullet 1 a i).

Note: 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.

6.2 Policy

Instrumentation tools like the `ptrace` system call, `gdb` and `strace` utilities, userspace live patching, as well as other tracing mechanisms offered by the Linux environment such as `ftrace` or `systemtap`, shall not be used in the operational environment. The use of any of these tools implies that the cryptographic module is running in a non-tested operational environment.

6.3 Requirements

The module shall be installed as stated in section 11. The operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data and uncontrolled access to the data of other processes is prevented.

7 Physical Security

The module is comprised of software only, and therefore this section is not applicable.

8 Non-invasive Security

This module does not implement any non-invasive security mechanism and therefore this section is not applicable.

9 Sensitive Security Parameter Management

Table 13 summarizes the Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module.

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
AES key	128, 192, 256	AES-CBC, AES-CCM, AES-CFB1, AES-CFB128, AES-CFB8, AES-CMAC, AES-CTR, AES-GCM, AES-KW, AES-KWP, AES- OFB, AES-XTS, CTR-DRBG A3136, A3137, A3138, A3140, A3141, A3142, A3143, A3149, A3150, A3151, A3152, A3153, A3154, A3155, A3157, A3158, A3159, A3160, A3162, A3163, A3165, A3166, A3167, A3169, A3170, A3171, A3172, A3176, A3177, A3178, A3179, A3180, A3181, A3182, A3183, A3184, A3190, A3194, A3195, A3196, A3197, A3198, A3199, A3200, A3201, A3204, A3205, A3206	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EVP_CIPHER_CTX_free, EVP_CIPHER_CTX_reset	**Use:** Symmetric encryption and decryption; Key wrapping and unwrapping; Message authentication code (MAC) generation and verification **Related keys:** N/A
HMAC key	112 to 256	HMAC A3144, A3145, A3146, A3147, A3148, A3156, A3161, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	HMAC_CTX_free	**Use:** Message authentication code (MAC) generation and verification **Related keys:** N/A

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
Module-generated RSA public key	112, to 256[2]	RSA A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	Generated using the random probable primes method (B.3.3) specified in FIPS 186-4; random values are obtained from the SP800-90Arev1 DRBG.	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	RSA_free	**Use:** Key generation **Related keys:** DRBG internal state; Module-generated RSA private key
Module-generated RSA private key	112 to 256	RSA A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	Generated using the random probable primes method (B.3.3) specified in FIPS 186-4; random values are obtained from the SP800-90Arev1 DRBG.	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	RSA_free	**Use:** Key generation **Related keys:** DRBG internal state; Module-generated RSA public key
RSA public key	80[3] to 256	RSA A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	RSA_free	**Use:** Digital signature verification **Related keys:** RSA private key
RSA private key	112 to 256	RSA A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	RSA_free	**Use:** Digital signature generation **Related keys:** RSA public key

[2] The security strength of RSA is based on key sizes between 2048 and up to 16384 bits allowed by IG C.F.

[3] RSA public key with less than 2048 bits and security strength of 80-111 bits is allowed for legacy use.

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
Module-generated ECDSA public key	112, 128, 192, 256	ECDSA A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	B.4.2 Testing Candidates Generated using the testing candidates method specified in FIPS 186-4; random values are obtained from the SP800-90Arev1 DRBG	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EC_KEY_free	**Use:** Key generation **Related keys**: DRBG internal state, Module-generated ECDSA private key
Module-generated ECDSA private key	112, 128, 192, 256	ECDSA A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	B.4.2 Testing Candidates Generated using the testing candidates method specified in FIPS 186-4; random values are obtained from the SP800-90Arev1 DRBG	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EC_KEY_free	**Use:** Key generation **Related keys**: DRBG internal state, Module-generated ECDSA public key
ECDSA public key	112, 128, 192, 256	ECDSA A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EC_KEY_free	**Use:** Key pair validation; ECDSA public key validation; Digital signature verification **Related keys**: ECDSA private key

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
ECDSA private key	112, 128, 192, 256	ECDSA A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EC_KEY_free	**Use:** Key pair validation; Digital signature generation **Related keys**: ECDSA public key
Module-generated EC Diffie-Hellman public key	112 to 256	KAS-ECC-SSC A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	Generated using the testing candidates method specified in SP800-56Arev3; random values are obtained from the SP800 90Arev1 DRBG.	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EC_KEY_free	**Use:** Key generation; Shared secret computation; Transport Layer Security (TLS) network protocol **Related SSPs:** DRBG internal state; EC Diffie-Hellman private key; EC Diffie-Hellman shared secret
Module-generated EC Diffie-Hellman private key	112 to 256	KAS-ECC-SSC A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	Generated using the testing candidates method specified in SP800-56Arev3; random values are obtained from the SP800 90Arev1 DRBG.	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EC_KEY_free	**Use:** Key generation; Shared secret computation; Transport Layer Security (TLS) network protocol **Related SSPs:** DRBG internal state; EC Diffie-Hellman public key; EC Diffie-Hellman shared secret

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
EC Diffie-Hellman public key	112 to 256	KAS-ECC-SSC A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EC_KEY_free	**Use:** Key pair validation; Shared secret computation; Transport Layer Security (TLS) network protocol **Related SSPs:** EC Diffie-Hellman shared secret
EC Diffie-Hellman private key	112 to 256	KAS-ECC-SSC A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EC_KEY_free	**Use:** Key pair validation; Shared secret computation **Related SSPs:** EC Diffie-Hellman shared secret
Module-generated Diffie-Hellman public key	112 to 200	KAS-FFC-SSC A3207, A3211	Generated using safe prime key generation method specified in SP800-56Arev3; random values are obtained from the SP800-90Arev1 DRBG.	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	DH_free	**Use:** Key generation; Shared secret computation; Transport Layer Security (TLS) network protocol **Related SSPs:** DRBG internal state; Module-generated Diffie-Hellman private key; Diffie-Hellman shared secret
Module-generated Diffie-Hellman private key	112 to 200	KAS-FFC-SSC A3207, A3211	Generated using safe prime key generation method specified in SP800-56Arev3;	**Import:** N/A **Export:** CM to TOEPP Path. Passed rom the module via API	N/A	RAM	DH_free	**Use:** Key generation; Shared secret computation; Transport Layer Security

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
			random values are obtained from the SP800-90Arev1 DRBG.	parameters in plaintext (P) format.				(TLS) network protocol **Related SSPs:** DRBG internal state; Module-generated Diffie-Hellman public key; Diffie-Hellman shared secret
Diffie-Hellman public key	112 to 200	KAS-FFC-SSC A3207, A3211	N/A.	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	`DH_free`	**Use:** Diffie-Hellman public key validation; Shared secret computation; Transport Layer Security (TLS) network protocol **Related SSPs:** Diffie-Hellman shared secret
Diffie-Hellman private key	112 to 200	KAS-FFC-SSC A3207, A3211	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	`DH_free`	**Use:** Shared secret computation **Related SSPs:** Diffie-Hellman shared secret
EC Diffie-Hellman shared secret	112 to 256	KAS-ECC-SSC A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import/Export:** CM to/from TOEPP Path. Passed to/from the module via API parameters in plaintext (P) format.	Computed during the EC Diffie-Hellman key agreement and shared secret computation per SP800-56Arev3.	RAM	`EC_KEY_free`	**Use:** Key derivation; EC Diffie-Hellman shared secret computation **Related SSPs:** EC Diffie-Hellman public key; EC Diffie-Hellman private key; TLS derived key; SSH derived key
Diffie-Hellman shared secret	112 to 200	KAS-FFC-SSC A3207, A3211	N/A	**Import/Export:** CM to/from TOEPP Path. Passed to/from the module via API parameters in plaintext (P) format.	Computed during the Diffie-Hellman key agreement and shared secret computatio	RAM	`DH_free`	**Use:** Key derivation; DH shared secret computation **Related SSPs:** Diffie-Hellman public key; Diffie-Hellman

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
					n per SP800-56Arev3.			private key; TLS derived key; SSH derived key
Password/passphrase	N/A	PBKDF A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM to TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	EVP_PKEY_free	**Use:** Key derivation. **Related SSPs:** PBKDF derived key
TLS Derived key	AES 128, 192, 256; HMAC 112 to 256	TLS KDF A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210 KDA HKDF A3139, A3168	Generated during the TLS KDF	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_PKEY_free	**Use:** Transport Layer Security (TLS) network protocol **Related SSPs:** TLS pre-master secret, TLS master secret
SSH Derived key	AES 128, 192, 256; HMAC 112 to 256	SSH KDF A3140, A3141, A3142, A3143, A3149, A3157, A3169, A3170, A3171, A3172	Generated during the SSH KDF	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_PKEY_free	**Use:** Key derivation **Related SSPs:** Shared secret
PBKDF Derived key	112 to 256	PBKDF A3144, A3145, A3146, A3147, A3148, A3156, A3173, A3174, A3175, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	Generated during the PBKDF compliant with SP800-132.	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	EVP_PKEY_free	**Use:** Key derivation **Related SSPs:** Password/passphrase
Entropy input IG D.L compliant	192 to 384	CTR_DRBG A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	Obtained from the SP800-90B entropy source	**Import/Export:** N/A; it remains within the cryptographic boundary.	N/A	RAM	FIPS_drbg_free	**Use:** Random number generation **Related SSPs:** DRBG seed
DRBG seed IG D.L compliant	192 to 384	CTR_DRBG A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162,	Derived from the entropy input as defined in SP800-90Arev1	**Import/Export:** N/A; it remains within the cryptographic boundary.	N/A	RAM	FIPS_drbg_free	**Use:** Random number generation **Related SSPs:** Entropy

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related SSPs
		A3163, A3165, A3166, A3167						input, DRBG Internal state
DRBG internal state (V, key) IG D.L compliant	128 to 256	CTR_DRBG A3136, A3137, A3138, A3150, A3154, A3158, A3160, A3162, A3163, A3165, A3166, A3167	Derived from seed as defined in SP800-90Arev1	**Import/Export:** N/A; it remains within the cryptographic boundary.	N/A	RAM	FIPS_drbg_free	**Use:** Random number generation **Related SSPs:** Entropy input, DRBG seed
TLS pre-master secret	DH 112 to 200 ECDH 128 to 256	TLS KDF A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	N/A	**Import:** CM to TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	Computed during key agreement for Diffie-Hellman or EC Diffie-Hellman cipher suites.	RAM	SSL_free, SSL_clear	**Use:** Transport Layer Security (TLS) network protocol **Related SSPs:** TLS master secret
TLS master secret	384	TLS KDF A3147, A3156, A3185, A3186, A3187, A3188, A3193, A3202, A3203, A3210	Derived from TLS pre-master secret using TLS KDF per SP800-135rev1.	**Import/Export:** N/A; it remains within the cryptographic boundary.	N/A	RAM	SSL_free, SSL_clear	**Use:** Transport Layer Security (TLS) network protocol **Related SSPs:** TLS pre-master secret; TLS derived key

Table 13 - *SSPs*

9.1 Random Number Generation

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90Arev1] for the creation of seeds for asymmetric keys, random numbers for security functions (e.g. ECDSA signature generation), and server and client random numbers for the TLS protocol. In addition, the module provides a Random Number Generation service to calling applications.

The DRBG supports the CTR_DRBG mechanisms. The DRBG is initialized during module initialization; the module loads by default the DRBG using the CTR_DRBG mechanism with AES-256, with derivation function, and without prediction resistance. A different DRBG mechanism can be chosen through an API function call.

The module uses an SP800-90B-compliant entropy source specified in Table 14. This entropy source is located within the physical perimeter, but outside of the cryptographic boundary of the module. The module obtains 384 bits to seed the DRBG, and 256 bits to reseed it, sufficient to provide a DRBG with 256 bits of security strength.

Entropy Sources	Minimum number of bits of entropy	Details
ESV certs. E22, E28, E29, E30	256 bits of entropy in the 256-bit output	Standalone Userspace CPU Time Jitter RNG version 3.4.0 entropy source with SHA-3 as the vetted conditioning component is located within the physical perimeter of the operational environment but outside the module cryptographic boundary.

Table 14 - Non-Deterministic Random Number Generation Specification

9.2 SSP Generation

The SSP generation methods implemented in the module are compliant with [SP800-133rev2].

For generating RSA, ECDSA keys, the module implements asymmetric key generation services compliant with [FIPS186-4]. A seed (i.e., the random value) used in asymmetric key generation is directly obtained from the [SP800-90Arev1] 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 [SP800-56Arev3]. The Diffie-Hellman key agreement scheme is also compliant with [SP800-56Arev3] and generates keys using safe primes defined in RFC7919 and RFC3526, as described in the next section. In accordance with FIPS 140-3 IG D.H, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per section 5.1 of SP800-133rev2 (vendor affirmed) by obtaining a random bit string directly from an approved DRBG and that can support the required security strength requested by the caller (without any V, as described in Additional Comments 2 of IG D.H).

The module supports the following key derivation methods according to [SP800-135rev1]:

- KDF for the TLS protocol, used as pseudo-random functions (PRF) for TLSv1.0/1.1, TLSv1.2;
- HKDF for the TLS protocol, used as pseudo-random functions (PRF) for TLSv1.3; and
- KDF for the SSHv2 protocol.

The module also supports password-based key derivation (PBKDF). The implementation is compliant with option 1a of [SP800-132]. Keys derived from passwords or passphrases using this method can only be used in storage applications.

9.3 SSP establishment

The module provides Diffie-Hellman and EC Diffie-Hellman shared secret computation compliant with SP800-56Arev3, in accordance with scenario 2 (1) of IG D.F.

The module also provides Diffie-Hellman and EC Diffie-Hellman key agreement schemes compliant with SP800-56rev3 and used as part of the TLS protocol key exchange in accordance with scenario 2 (2) of IG D.F; that is, the shared secret computation (KAS-FFC-SSC and KAS-ECC-SSC) followed by the derivation of the keying material using SP800-135rev1 KDF.

For Diffie-Hellman, the module supports the use of safe primes from RFC7919 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 RFC3526, 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)

The module also provides the following key transport mechanisms:

- Key wrapping using AES-KW and AES-KWP modes.
- Key wrapping using AES-CCM, AES-GCM, and AES-CBC with HMAC, used in the context of the TLS protocol cipher suites with 128-bit or 256-bit keys.

According to Table 2: Comparable strengths in [SP800-57rev5], the key sizes of AES, RSA, Diffie-Hellman and EC Diffie-Hellman provides the following security strength in the approved mode of operation:

- AES key wrapping using AES in KW, KWP provides between 128 and 256 bits of encryption strength.
- AES key wrapping using AES-CCM, AES-GCM, and AES in CBC mode and HMAC, provides between 128 or 256 bits of encryption strength.
- Diffie-Hellman key agreement provides between 112 and 200 bits of encryption strength.
- EC Diffie-Hellman key agreement provides between 112 and 256 bits of encryption strength.

9.4 SSP Entry and Output

The module does not support manual SSP entry or intermediate SSP generation output. The SSPs are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form within the physical perimeter of the operational environment. This is allowed by [FIPS140-3_IG] IG 9.5.A, according to the “CM Software to/from App via TOEPP Path” entry in the Key Establishment Table.

9.5 SSP Storage

The module does not perform persistent storage of SSPs. The SSPs are temporarily stored in the RAM in plaintext form. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

9.6 SSP Zeroization

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The application that is acting as the CO is responsible for calling the appropriate zeroization functions provided in the module's API and listed in Table 13. Calling the SSL_free() and SSL_clear() will zeroize the SSPs stored in the TLS protocol internal state and also invoke the corresponding API functions listed in Table 13 to zeroize SSPs. The zeroization functions overwrite the memory occupied by SSPs with “zeros” and deallocate the memory with the regular memory deallocation operating system call. The completion of a zeroization routine(s) will indicate that a zeroization procedure succeeded.

10 Self-Tests

The module performs the pre-operational self-test and CASTs automatically when the module is loaded into memory. Pre-operational self-test ensure that the module is not corrupted, and the CASTs ensure that the cryptographic algorithms work as expected. While the module is executing the pre-operational test and the CASTs, the module services are not available, and input and output are inhibited. The module is not available for use by the calling application until the pre-operational self-test and the CASTs are completed successfully. After the pre-operational test and the CASTs succeed, the module becomes operational. If any of the pre-operational test or any of the CASTs fail an error message is returned, and the module transitions to the error state.

Algorithm	Test
AES	KAT AES ECB mode with 128-bit key, encryption and decryption (separately tested) KAT AES CCM mode with 192-bit key, encryption and decryption (separately tested) KAT AES GCM mode with 256-bit key, encryption and decryption (separately tested) KAT AES XTS mode with 128 and 256-bit keys, encryption, and decryption (separately tested)
CMAC	KAT AES CMAC with 128-,192-, and 256-bit keys, MAC generation
Diffie-Hellman	Primitive “Z” computation KAT with 2048-bit key
DRBG	KAT CTR_DRBG with AES with 256-bit keys with and without DF, with and without PR Health tests according to section 11.3 of [SP800-90Arev1]
EC Diffie-Hellman	Primitive “Z” computation KAT with P-256 curve
ECDSA	KAT ECDSA with P-256 and SHA2-256, signature generation and verification (separately tested)
HMAC	KAT HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 KAT HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512
PBKDF	KAT with SHA2-256
RSA	KAT RSA with 2048-bit key, PKCS#1 v1.5 scheme and SHA2-256, signature generation and verification (separately tested) KAT RSA with 2048-bit key, PSS scheme and SHA2-256, signature generation and verification (separately tested)
SHA-3	KAT SHA3-256, SHA3-512, SHAKE-128 and SHAKE-256
SHA	KAT SHA-1, SHA2-224, SHA2-256, SHA2-384 and SHA2-512.

Algorithm	Test
SSH KDF	KAT with SHA2-256
TLSv1.2 KDF	KAT with SHA2-256
HKDF	KAT with SHA2-256

Table 15 – Cryptographic Algorithms Self-Tests

10.1 Pre-Operational Tests

The module performs the integrity test of the shared libraries that comprise the module. The details of integrity test are provided in section 5.1.

10.2 Conditional Tests

10.2.1 Cryptographic Algorithm Self-Tests

Table 15 specifies all the CASTs. All the CASTs are performed in the form of the Known Answer Tests (KATs) and are run prior to performing the integrity test. A KAT includes the comparison of a calculated output with an expected known answer, hard coded as part of the test vectors used in the test. If the values do not match, the KAT fails.

10.2.2 Pairwise Consistency Test

The module performs the Pair-wise Consistency Tests (PCT) shown in the following table. If at least one of the tests fails, the module returns an error code and enters the Error state. When the module is in the Error state, no data is output, and cryptographic operations are not allowed.

Algorithm	Test
ECDSA key generation	PCT using SHA2-256, signature generation and verification
RSA key generation	PCT using SHA2-256, signature generation and verification
Diffie-Hellman key generation	PCT according to section 5.6.2.1.4 of [SP800-56Arev3]
ECDH key generation	Covered by ECDSA PCT as allowed by IG 10.3, additional comment 1.

Table 16 - Pairwise Consistency Test

10.3 Periodic/On-Demand Self-Test

On-Demand self-tests can be invoked by powering-off and reloading the module which cause the module to run the power-up tests again.

10.4 Error States

When the module fails the pre-operational self-test or any conditional test, the module returns an error code to indicate the error and enter the “Abort” error state, showing the following message to stderr: “OpenSSL internal error, assertion failed: FATAL FIPS SELFTEST FAILURE” and stopping functioning. The only way to recover from this error is to restart the application. If the failure persists, the module must be reinstalled.

When a PCT fails during conditional tests, the module returns an error code to indicate the error and enter the “Error” error state. Any further cryptographic operation is inhibited. The calling application can obtain the module state by requesting the “Show status” service by calling the `FIPS_selftest_failed()` API function. The function returns 1 if the module is in the “Error” state, 0 if the module is in the Operational state.

Some cryptographic services cannot handle the return value of the “Error” state, and when the module is in that state and receives a service request, shows an error message and transitions to the “Abort” state, showing the following message to stderr: “OpenSSL internal error, assertion failed: FATAL FIPS SELFTEST FAILURE” and stopping functioning. The only way to recover from this error is to restart the application.

Table 17 shows the error codes and the corresponding condition:

State	Cause of Error	Status Indicator
Abort	The integrity test fails at power-up.	FIPS_R_FINGERPRINT_DOES_NOT_MATCH (110). Module stops functioning.
Abort	Any of the AES, CMAC, DRBG, HMAC, or SHA KATs fails during CAST.	FIPS_R_SELFTEST_FAILED (101). Module stops functioning.
Abort	Any of the KATs for RSA or ECDSA fails during CAST.	FIPS_R_TEST_FAILURE (117). Module stops functioning.
Abort	The KAT of a DRBG fails during CAST.	FIPS_R_NOPR_TEST1_FAILURE (145) FIPS_R_NOPR_TEST2_FAILURE(146) FIPS_R_PR_TEST1_FAILURE (147) FIPS_R_PR_TEST2_FAILURE (148) Module stops functioning.
Error	The PCT of a newly generated RSA, ECDSA, Diffie-Hellman or EC Diffie-Hellman key pair fails during conditional tests.	FIPS_R_PAIRWISE_TEST_FAILED (127)
Error	The module is in the Error state and a cryptographic service other than the following is invoked: Message Digest, Encryption/Decryption, Diffie-Hellman.	FIPS_R_FIPS_SELFTEST_FAILED (106)
Abort	The module is in the Error state and one of the following cryptographic services is invoked: Message Digest, Encryption/Decryption, Diffie-Hellman.	Error message “OpenSSL internal error, assertion failed: FATAL FIPS SELFTEST FAILURE” shown in stderr. Module stops functioning.

Table 17 - Error States

In the “Error” state, errors are reported through the regular ERR interface of the modules and can be queried by functions such as ERR_get_error(). See the OpenSSL man pages for the function description.

11 Life-cycle assurance

11.1 Delivery and Operation

11.1.1 Module Installation

The Crypto Officer can install the RPM packages containing the module as listed in Table 19 using the zypper tool as follows:

```
# zypper install libopenssl1_1
# zypper install libopenssl1_1-hmac
```

If the use of certified 32-bit Openssl libraries on Intel x86 is required, then use the following to install the 32bit libraries and hmac packages:

```
# zypper install libopenssl1_1-32bit
# zypper install libopenssl1_1-hmac-32bit
```

The integrity of the RPM package is automatically verified during the installation, and the Crypto Officer shall not install the RPM package if there is any integrity error.

11.1.2 Operating Environment Configuration

The operating environment needs to be configured to support the approved mode of operation, so the following steps shall be performed with the root privilege:

1. Install the dracut-fips RPM package:

```
# zypper install dracut-fips
```

2. Recreate the INITRAMFS image:

```
# dracut -f
```

3. After regenerating the initrd, the Crypto Officer has to append the following parameter in the /etc/default/grub configuration file in the GRUB_CMDLINE_LINUX_DEFAULT line:

```
fips=1
```

4. After editing the configuration file, please run the following command to change the setting in the boot loader:

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

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" respectively. For example:

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

The partition of /boot is located on /dev/sda1 in this example. Therefore, the following string needs to be appended in the aforementioned grub file:

```
"boot=/dev/sda1"
```

5. Reboot to apply these settings.

Now, the operating environment is configured to support the approved mode of operation. The Crypto Officer should check the existence of the file /proc/sys/crypto/fips_enabled, and verify it contains a numeric value “1”. If the file does not exist or does not contain “1”, the operating environment is not configured to support the approved mode of operation and the module will not operate as a FIPS validated module properly.

11.1.3 Module Installation for Vendor Affirmed Platforms

Table 18 includes the information on module installation process for the vendor affirmed platforms that are listed in Table 4 and Table 5.

Product	Link
SUSE Linux Enterprise Micro 5.3	https://documentation.suse.com/sle-micro/5.3/single-html/SLE-Micro-security/#sec-fips-slemicro-install
SUSE Linux Enterprise Server for SAP 15SP4	https://documentation.suse.com/sles/15-SP4/html/SLES-all/book-security.html
SUSE Linux Enterprise Base Container Image 15SP4	https://documentation.suse.com/smart/linux/html/concept-bci/index.html
SUSE Linux Enterprise Desktop 15SP4	https://documentation.suse.com/sled/15-SP4/html/SLED-all/book-security.html
SUSE Linux Enterprise Real Time 15SP4	https://documentation.suse.com/sle-rt/15-SP4

Table 18 - Installation for Vendor Affirmed Platforms

Note: Per section 7.9 in the FIPS 140-3 Management Manual [FIPS140-3_MM], the Cryptographic Module Validation Program (CMVP) makes no statement as to the correct operation of the module or the security strengths of the generated keys when this module is ported and executed in an operational environment not listed on the validation certificate.

11.1.4 End of Life Procedure

For secure sanitization of the cryptographic module, the module needs first to be powered off, which will zeroize all keys and CSPs in volatile memory. Then, for actual deprecation, the module shall be upgraded to a newer version that is FIPS 140-3 validated.

The module does not possess persistent storage of SSPs, so further sanitization steps are not needed.

11.2 Crypto Officer Guidance

The binaries of the module are contained in the RPM packages for delivery. The Crypto Officer shall follow section 11.1.1 and 11.1.2 to configure the operational environment and install the module to be operated as a FIPS 140-3 validated module.

Table 19 lists the RPM packages that contain the FIPS validated module and the OE directory where the components are installed. The "Show module name and version" service returns the value "OpenSSL 1.1.1l 24 Aug 2021 SUSE release 150400.7.28.1”, which matches the version included in the RPM package filenames.

Processor Architecture	RPM Packages	Location in the OE
Intel 64-bit	libopenssl1_1-1.1.1l-150400.7.28.1.x86_64.rpm libopenssl1_1-hmac-1.1.1l-150400.7.28.1.x86_64.rpm	/usr/lib64
Intel 32-bit	libopenssl1_1-32bit-1.1.1l-150400.7.28.1.x86_64.rpm libopenssl1_1-hmac-32bit-1.1.1l-150400.7.28.1.1.x86_64.rpm	/usr/lib
AMD 64-bit	libopenssl1_1-1.1.1l-150400.7.28.1.x86_64.rpm libopenssl1_1-hmac-1.1.1l-150400.7.28.1.x86_64.rpm	/usr/lib64
IBM z15	libopenssl1_1-1.1.1l-150400.7.28.1.s390x.rpm libopenssl1_1-hmac-1.1.1l-150400.7.28.1.s390x.rpm	/usr/lib64
ARMv8 64-bit	libopenssl1_1-1.1.1l-150400.7.28.1.aarch64.rpm libopenssl1_1-hmac-1.1.1l-150400.7.28.1.aarch64.rpm	/usr/lib64
IBM Power10 64-bit	libopenssl1_1-1.1.1l-150400.7.28.1.ppc64le.rpm libopenssl1_1-hmac-1.1.1l-150400.7.28.1.ppc64le.rpm	/usr/lib64

Table 19 - RPM packages

11.2.1 AES XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E]. The length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks, that is 16MB of data.

To meet the requirement stated in IG C.I, the module implements a check that ensures, before performing any cryptographic operation, that the two AES keys used in AES XTS mode are not identical.

Note: AES-XTS shall be used with 128 and 256-bit keys only. AES-XTS with 192-bit keys is not an Approved service.

11.2.2 AES GCM IV

The AES GCM IV generation is in compliance with the [RFC5288] and shall only be used for the TLS protocol version 1.2 to be compliant with [FIPS140-3_IG] IG C.H, provision 1 (“TLS protocol IV generation”); in addition, the module is compliant with section 3.3.1 of [SP800-52rev2].

The nonce_explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all of its possible values.

In case the module's power is lost and then restored, the key used for the AES GCM encryption or decryption shall be redistributed.

When a GCM IV is used for decryption, the responsibility for the IV generation lies with the party that performs the AES GCM encryption.

11.2.3 Environment Variables

OPENSSL_ENFORCE_MODULUS_BITS

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.

Notice that even if this environment variable is not set, the module will provide the corresponding value of the service indicator depending on the size of the key generated.

11.2.4 Key derivation using SP800-132 PBKDF

The module provides password-based key derivation (PBKDF), compliant with SP800-132 and IG D.N. The module supports option 1a from section 5.4 of [SP800-132], in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK).

In accordance with [SP800-132], the following requirements shall be met.

- Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The length of the MK or DPK shall be of 112 bits or more (this is verified by the module to determine the service is approved).
- A portion of the salt, with a length of at least 128 bits (this is verified by the module to determine the service is approved), shall be generated randomly using the SP800-90Arev1 DRBG.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The minimum value shall be 1000 (this is verified by the module to determine the service is approved).
- Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.
- The length of the password or passphrase shall be of at least 20 characters (this is verified by the module to determine the service is approved), and shall consist of lower-case, upper-case and numeric characters. The probability of guessing the value is estimated to be $1/62^{20} = 10^{-36}$, which is less than 2^{-112}.

The calling application shall also observe the rest of the requirements and recommendations specified in [SP800-132].

12 Mitigation of other attacks

The module implements blinding against RSA timing 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 module provides the API functions `RSA_blinding_on()` and `RSA_blinding_off()` to turn the blinding on and off for RSA. When the blinding is on, the module generates a random value to form a blinding factor in the RSA key before the RSA key is used in the RSA cryptographic operations.

Appendix A. TLS Cipher Suites

The module supports the following cipher suites for the TLS protocol versions 1.0, 1.1, 1.2 and 1.3 compliant with section 3.3.1 of [SP800-52rev2]. Each cipher suite defines the key exchange algorithm, the bulk encryption algorithm (including the symmetric key size) and the MAC algorithm.

Cipher Suite	ID	Reference
TLS_DH_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x31 }	RFC3268
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x33 }	RFC3268
TLS_DH_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x37 }	RFC3268
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x39 }	RFC3268
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x3F }	RFC5246
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x67 }	RFC5246
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x69 }	RFC5246
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x6B }	RFC5246
TLS_PSK_WITH_AES_128_CBC_SHA	{ 0x00, 0x8C }	RFC4279
TLS_PSK_WITH_AES_256_CBC_SHA	{ 0x00, 0x8D }	RFC4279
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0x9E }	RFC5288
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0x9F }	RFC5288
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0xA0 }	RFC5288
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0xA1 }	RFC5288
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x04 }	RFC4492
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x05 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x09 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0A }	RFC4492
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x0E }	RFC4492
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0F }	RFC4492
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x13 }	RFC4492
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x14 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x23 }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x24 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x25 }	RFC5289

Cipher Suite	ID	Reference
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x26 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x27 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x28 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x29 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x2A }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2B }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x2C }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2D }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x2E }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2F }	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x30 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x31 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x32 }	RFC5289
TLS_DHE_RSA_WITH_AES_128_CCM	{ 0xC0, 0x9E }	RFC6655
TLS_DHE_RSA_WITH_AES_256_CCM	{ 0xC0, 0x9F }	RFC6655
TLS_DHE_RSA_WITH_AES_128_CCM_8	{ 0xC0, 0xA2 }	RFC6655
TLS_DHE_RSA_WITH_AES_256_CCM_8	{ 0xC0, 0xA3 }	RFC6655
TLS_AES_128_GCM_SHA256	{ 0x13, 0x01 }	RFC8446
TLS_AES_256_GCM_SHA384	{ 0x13, 0x02 }	RFC8446
TLS_AES_128_CCM_SHA256	{ 0x13, 0x04 }	RFC8446
TLS_AES_128_CCM_8_SHA256	{ 0x13, 0x05 }	RFC8446

Table 20 - TLS Cipher Suites

Appendix B. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAST	Cryptographic Algorithm Self-Tests
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CPACF	Central Processor Assist for Cryptographic Function
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DF	Derivation Function
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
FSM	Finite State Model
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
ISA	Instruction Set Architecture
KAS	Key Agreement Schema
KAT	Known Answer Test
KW	AES Key Wrap
KWP	AES Key Wrap with Padding
MAC	Message Authentication Code
NDF	No Derivation Function
NIST	National Institute of Science and Technology
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PR	Prediction Resistance

PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSH	Secure Shell
SSP	Sensitive Security Parameter
TDES	Triple-DES
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix C. References

FIPS140-3 **FIPS PUB 140-3 - Security Requirements For Cryptographic Modules**
March 2019
https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf

FIPS140-3_IG **Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program**
October 2022
https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf

FIPS140-3_MM **FIPS 140-3 Cryptographic Module Validation Program - Management Manual**
April 2024
https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS-140-3-CMVP%20Management%20Manual.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
July 2013
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
November 2001
https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

FIPS202 **SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions**
August 2015
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

PKCS#1 **Public Key Cryptography Standards (PKCS) #1: RSA Cryptography**
Specifications Version 2.1
February 2003
https://www.ietf.org/rfc/rfc3447.txt

RFC3394 **Advanced Encryption Standard (AES) Key Wrap Algorithm**
September 2002
https://www.ietf.org/rfc/rfc3394.txt

RFC5649	**Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm** September 2009 https://www.ietf.org/rfc/rfc5649.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38B.pdf
SP800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-38G	**NIST Special Publication 800-38G - Recommendation for Block Cipher Modes of Operation: Methods for Format - Preserving Encryption** March 2016 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38G.pdf
SP800-52rev2	**NIST Special Publication 800-52 Revision 2 - Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf

SP800-56Arev3	**NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP800-56Crev2	**Recommendation for Key Derivation through Extraction-then-Expansion** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP800-57rev5	**NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management Part 1: General** May 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
SP800-90Arev1	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-90B	**NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP800-131Arev2	**NIST Special Publication 800-131A Revision 2 - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf
SP800-133rev2	**NIST Special Publication 800-133 Revision 2 - Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP800-135rev1	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf

SP800-140B **NIST Special Publication 800-140B - CMVP Security Policy Requirements**
March 2020
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf