SUSE Linux Enterprise NSS Cryptographic Module

version 3.1

FIPS 140-3 Non-Proprietary Security Policy

Version 1.1

Last update: 2024-06-25

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

1 Table of Contents

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 3.1 of the SUSE Linux Enterprise NSS Cryptographic Module. It has a one-to-one mapping to the [SP 800-140B] 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.

1.2 How this Security Policy was Prepared

The vendor has provided the non-proprietary Security Policy of the cryptographic module, which was further consolidated into this document by atsec information security together with other vendor-supplied documentation. In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

1.3 Security Levels

Table 1 describes the individual security areas of FIPS 140-3, as well as the security levels of those individual areas.

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 NSS Cryptographic Module (hereafter referred to as “the module”) is a Software multi-chip standalone cryptographic module. It provides a C language application program interface (API) designed to support cross-platform development of security-enabled client and server applications. Applications built with NSS can support SSLv3, TLS, IKEv2, PKCS#5, PKCS#7, PKCS#11, PKCS#12, S/MIME, X.509 v3 certificates, and other security standards supporting FIPS 140-3 validated cryptographic algorithms.

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.

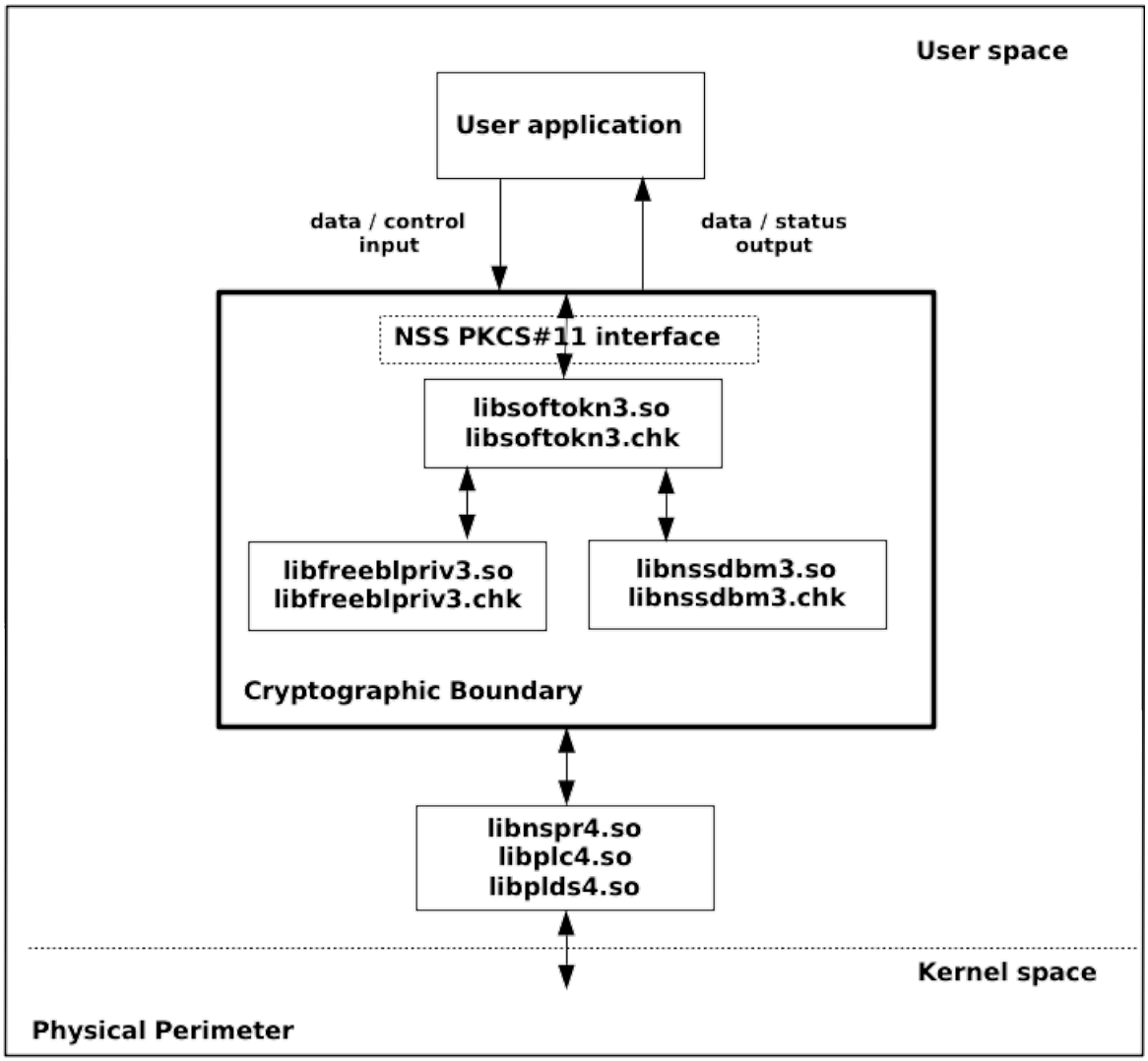

Figure 1 – Cryptographic Boundary

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

Components	Description
/usr/lib64/libsoftokn3.so	PKCS#11 wrapper shared library.
/usr/lib64/libsoftokn3.chk	DSA signature for libsoftokn3.so.
/usr/lib64/libnssdbm3.so	NSS database management shared library.
/usr/lib64/libnssdbm3.chk	DSA signature for libnssdbm3.so.
/lib64/libfreeblpriv3.so	General purpose cryptographic shared library.
/lib64/libfreeblpriv3.chk	DSA signature for libfreeblpriv3.so.

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 11. Please see section 4.1.1 for details on the service indicator provided by the module that identifies when an approved service has been requested.

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
1	SUSE Linux Enterprise Server 15 SP4	Supermicro Super Server SYS-6019P-WTR	Intel® Xeon® Silver 4215R	With and without AES-NI (PAA)
2	SUSE Linux Enterprise Server 15 SP4	GIGABYTE R181-Z90-00	AMD EPYC™ 7371	With and without AES-NI (PAA)
3	SUSE Linux Enterprise Server 15 SP4	GIGABYTE G242-P32-QZ	ARM Ampere® Altra® Q80-30	With and without Crypto Extensions (PAA)
4	SUSE Linux Enterprise Server 15 SP4	IBM z/15	z15	With and without CPACF (PAI)

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
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)

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 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.

#	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	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)

#	Operating System	Hardware platform	Processor	PAA/Acceleration
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	IBM Power E1080 (9080-HEX)	Power10	With and without ISA (PAA)
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

2.6 Approved Algorithms

Table 5 lists all security functions of the module, including specific key strengths employed for approved services, and implemented modes of operation. The following are allowed for legacy use only: DSA signature verification with L=1024 and N=224.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3575, A3581, A3585	AES SP800-38A	CBC	128, 192, 256-bit keys with 128-256 bits of key strength	Symmetric encryption; Symmetric decryption
A3577	AES SP800-38B	CMAC	128, 192, 256-bit keys with 128-256 bits of key strength	Message authentication code (MAC)

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3575, A3581	AES SP800-38A	CTR	128, 192, 256-bit keys with 128-256 bits of key strength	Symmetric encryption; Symmetric decryption
A3580	AES SP800-38A-addendum	CTS (CS1)	128, 192, 256-bit keys with 128-256 bits of key strength	Symmetric encryption; Symmetric decryption
A3575, A3581, A3582, A3583, A3585, A3586, A3587	AES SP800-38A	ECB	128, 192, 256-bit keys with 128-256 bits of key strength	Symmetric encryption; Symmetric decryption
A3575, A3581, A3582, A3583, A3585, A3586, A3587	AES SP800-38D RFC5288 RFC8446	GCM with internal IV (8.2.1)	128, 192, 256-bit keys with 128-256 bits of key strength	Symmetric encryption
A3575, A3581, A3582, A3583, A3585, A3586, A3587	AES SP800-38D SP800-90Arev1	GCM with internal IV (8.2.2)	128, 192, 256-bit keys with 128-256 bits of key strength	Symmetric encryption
A3575, A3581, A3582, A3583, A3585, A3586, A3587	AES SP800-38D	GCM with external IV	128, 192, 256-bit keys with 128-256 bits of key strength	Symmetric decryption
A3576	AES SP800-38F	KW, KWP	128, 192, 256-bit keys with 128-256 bits of key strength	Key wrapping and unwrapping
Vendor Affirmed	CKG SP800-133rev2	Asymmetric key generation (FIPS-186-4, SP800-90Arev1)	RSA: 2048, 3072, 4096-bit keys with 112-149 bits of key strength	RSA key generation
		Asymmetric key generation (FIPS-186-4, SP800-56Arev3, SP800-90Arev1)	EC: P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	EC key generation
		Asymmetric key generation (SP800-56Arev3, SP800-90Arev1)	Safe Primes: 2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength	Safe Primes key generation
		Symmetric key generation	AES: 128, 192, 256-bit keys with 128-	Symmetric key generation

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
		(SP800-90Arev1)	256 bits of key strength HMAC: ≥ 112-bit keys with key strength of 112-256 bits	
A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	DRBG SP800-90Arev1	Hash_DRBG: SHA-256 without PR	N/A	Deterministic random bit generation
A3575, A3584, A3588	DSA FIPS186-4	SHA-224, SHA-256, SHA-384, SHA-512	L=1024, N=160 L=2048, N=224 L=2048, N=256 L=3072, N=256 keys with 80-128 of bits key strength	Digital signature verification Integrity test
A3575, A3584, A3588	ECDSA FIPS186-4	B.4.1 Extra Random Bits	P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength	EC Key pair generation EC Public key verification
		SHA-224, SHA-256, SHA-384, SHA-512	P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength	Digital signature generation
		SHA-224, SHA-256, SHA-384, SHA-512	P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength	Digital signature verification
E28, E29	Non-physical Entropy Source SP800-90B	CPU Time Jitter RNG (SHA3-256 Conditioning Component)	N/A	Random number generation
A3575, A3588	HMAC FIPS198-1	SHA-1	≥ 112-bit keys with key strength of 112-256 bits	Message authentication code (MAC)
A3575, A3584, A3588		SHA-224, SHA-256		
A3575		SHA-384, SHA-512		
A3575, A3584, A3588	KAS-ECC-SSC SP800-56Arev3	ECC Ephemeral Unified Scheme	P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength	EC Diffie-Hellman shared secret computation

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3575, A3584, A3588	KAS-FFC-SSC SP800-56Arev3	Safe Prime Groups (dhEphem): ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength	Diffie-Hellman shared secret computation
A3579	KDF IKE (CVL) SP800-135rev1	HMAC-SHA-1 HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512	IKE derived secret with 112 and 200 bits of key strength	Key derivation for IKEv1 and IKEv2
A3575, A3584, A3588	KDF TLS (CVL) SP800-135rev1	SHA-1	TLS derived secret with 112 to 256 bits key strength	Key derivation for TLS
A3575, A3584, A3588	TLS v1.2 KDF (CVL) SP800-135rev1 RFC7627	SHA-256, SHA-384, SHA-512		
A3574	KDA HKDF SP800-56Crev1	HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512		Key derivation for TLS (only TLS 1.3)
A3578	KDF SP800-108	CMAC-AES128, CMAC-AES192, CMAC-AES256 in Counter, Feedback and Double-pipeline modes	128 to 4096-bit keys with 128-256 bits of key strength	Key-based key derivation
		HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 in Counter, Feedback and Double-pipeline modes		

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3576	KTS SP800-38F	AES KW, KWP	128, 192, 256	Key wrapping and unwrapping
A3575, A3588	PBKDF SP800-132	HMAC-SHA-1	112 to 4096 derived keys with 112-256 bits of key strength	Password-based key derivation
A3575, A3584, A3588		HMAC-SHA-224, HMAC-SHA-256		
A3575		HMAC-SHA-384, HMAC-SHA-512		
A3575, A3584, A3588	RSA FIPS186-4	B.3.3 Random Probable Primes	2048, 3072, 4096-bit keys with 112-149 bits of key strength	RSA Key pair generation
		PKCS#1v1.5: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096-bit keys with 112-149 bits of key strength	Digital signature generation
		PSS: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096-bit keys with 112-149 bits of key strength	
		PKCS#1v1.5: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096-bit keys with 112-149 bits of key strength	Digital signature verification
		PSS: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096-bit keys with 112-149 bits of key strength	
A3575, A3584, A3588	Safe primes SP800-56Ar3	Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength	Safe Primes key generation
A3575, A3588	SHS	SHA-1	N/A	Message digest

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3575, A3584, A3588	FIPS180-4	SHA-224, SHA-256		
A3575		SHA-384, SHA-512		

Table 5 - 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 6 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 TLS v1.0/1.1 KDF only

Table 6 - 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 7 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 11.

Algorithm/Functions	Use/Function
AES in CBC-MAC and XCBC-MAC modes	Symmetric encryption and decryption
AES-GCM with external IV	Symmetric encryption

[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
Camellia, CAST, CAST3, CAST5, ChaCha20, DES, DES2, Triple-DES CDMF, IDEA, RC2, RC4, RC5, SEED	Symmetric key generation, encryption and decryption
Poly1305	Symmetric encryption and decryption, message authentication code (MAC)
MD2, MD5	Message digest
HMAC using keys less than 112 bits of length HMAC with non-approved message digest algorithms	Message authentication code (MAC)
DSA with any key size	Key pair generation, domain parameter generation and verification, digital signature generation
DSA with non-approved message digest algorithms	Digital signature verification
DSA with keys smaller than 1024 bits or greater than 3072 bits	Digital signature verification
RSA with pre-hashed message	Digital signature generation and verification
RSA PSS with non-approved message digest algorithms	Digital signature generation and verification
RSA PSS with keys smaller than 2048 bits or greater than 4096 bits	Key pair generation, digital signature generation and verification
RSA PKCS#1v1.5 with non-approved message digest algorithms	Digital signature generation and verification
RSA PKCS#1v1.5 with keys smaller than 2048 bits or greater than 4096 bits	Key pair generation, digital signature generation and verification
RSA encryption and decryption with any key size	Key encapsulation
ISO/IEC 9796 RSA	Digital signature generation and verification with and without message recovery
RSA X.509	RSA X.509 certificate generation
ECDSA with pre-hashed message	Digital signature generation and verification
ECDSA using non-approved message digest algorithms	Digital signature generation and verification

Algorithm/Functions	Use/Function
ECDSA with P-192 and P-224 curves, K curves, B curves and non-NIST curves	Key pair generation, digital signature generation and verification
Curve25519	Key pair generation, domain parameter generation and verification, digital signature generation and verification
J-PAKE	Key agreement
HKDF (outside of the TLS 1.3 protocol), PBKDF1	Key derivation
Diffie-Hellman with keys generated with domain parameters other than safe primes	Diffie-Hellman shared secret computation
EC Diffie-Hellman with P-192 and P-224 curves, K curves, B curves and non-NIST curves	EC Diffie-Hellman shared secret computation

Table 7 - Non-Approved 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.

All data output via data output interface is inhibited when the module is performing pre-operational test, conditional cryptographic algorithm self-tests, zeroization, or when the module enters the error state.

Logical Interface[2]	Data that passes over port/interface
Data Input	API input parameters for data
Data Output	API output parameters for data
Control Input	API function calls, API input parameters for control input, /proc/sys/crypto/fips_enabled control file
Status Output	API return codes, API output parameters for status output

Table 8 - Ports and Interfaces

[2] The control output interface is omitted on purpose because the module does not implement it.

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. No support is provided for multiple concurrent operators or a maintenance role.

Role	Service	Input	Output
Crypto Officer (CO)	Asymmetric Key Generation	Key size	Module generated key pair
	Diffie-Hellman shared secret computation	Diffie-Hellman private key (owner), Diffie-Hellman public key from peer	Diffie-Hellman shared secret
	Digital signature generation	Message, hash algorithm, private key	Digital signature
	Digital signature verification	Message, Signature, hash algorithm, public key	Verification result
	EC Diffie-Hellman shared secret computation	EC Private key (owner), EC public key from peer	EC Diffie-Hellman shared secret
	Key derivation for TLS	(EC) Diffie-Hellman Shared secret	TLS derived secret
	Key derivation for IKEv1 and IKEv2	(EC) Diffie-Hellman Shared secret	IKE derived secret
	Key-based key derivation	Key derivation key	KBKDF derived key
	Key encapsulation	Key to be encapsulated, Key encapsulating key	Encapsulated key
	Key unencapsulation	Encapsulated key, Key encapsulating key	Unencapsulated key
	Key unwrapping	Wrapped key, Key unwrapping key	Unwrapped key
	Key wrapping	Key to be wrapped, Key wrapping key	Wrapped key
	Message authentication code (MAC)	Message, HMAC key or AES key	Message authentication code
	Message digest	Message	Digest of the message
	Module initialization	None	None
	Module installation and configuration	Configuration parameters	Return codes and/or log messages
	On-Demand integrity tests	None	Return codes
	Password-based key derivation	Password/Passphrase, Salt, Key size, Iteration Count	PBKDF derived key

Role	Service	Input	Output
	Public key verification	Key	Return codes/log messages
	Random number generation	Number of bits	Random number
	Self-tests	Module reset	Result of self-test (pass/fail)
	Symmetric decryption	Key, IV (for AEAD), Ciphertext	Plaintext
	Symmetric encryption	Key, IV (for AEAD), Plaintext	Ciphertext
	Symmetric key generation	Key size	Module generated key
	Show module name and version	None	Name and version information
	Show status	None	Return codes and/or log messages
	Zeroization	Context containing SSPs	N/A

Table 9 - Roles, Service Commands, Input and Output

4.1.1 Approved Services

The module provides services to the operators that assume the available role. Table 10 lists approved services. For each service, the table lists the associated cryptographic algorithm(s), the role to perform the service, the cryptographic keys or CSPs involved, and their access type(s). In addition to the CSPs listed in Table 10, any hash values of passwords and RBG state information are considered to be CSPs. No support of intermediate key generation is provided. The following convention is used to specify access rights to a CSP:

- **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 CSP or key during its operation.

The details of the approved cryptographic algorithms including the CAVP certificate numbers can be found in Table 5.

The module implements the method NSC_NSSGetFIPSStatus() to indicate whether the last service requested was approved (1).

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Cryptographic Services						
Symmetric key generation	Generate AES or HMAC key	DRBG	Module generated AES key	CO	G, R	NSC_NSSGetFIPSStatus = 1
			Module generated HMAC key			
Symmetric encryption	Perform AES encryption	AES-CBC, AES-CMAC, AES-CTR, AES-CTS (CS1), AES-ECB, AES-GCM	AES key		W, E	NSC_NSSGetFIPSStatus = 1
Symmetric decryption	Perform AES decryption	AES-CBC, AES-CMAC, AES-CTR, AES-CTS (CS1), AES-ECB, AES-GCM	AES key		W, E	NSC_NSSGetFIPSStatus = 1
Asymmetric key generation	Generate key pairs	RSA key generation DRBG	Module generated RSA public and private keys		G, R	NSC_NSSGetFIPSStatus = 1
		EC key generation DRBG	Module generated EC public and private keys			
		Safe Primes key generation DRBG	Module generated Diffie-Hellman public and private keys			
Digital signature generation	Generate a signature	RSA digital signature generation SHS	RSA private key		W, E	NSC_NSSGetFIPSStatus = 1
		ECDSA digital signature generation SHS	EC private key			
Digital signature verification	Verify a signature	RSA digital signature verification SHS	RSA public key		W, E	NSC_NSSGetFIPSStatus = 1
		ECDSA digital signature verification SHS	EC public key			
		DSA digital signature verification SHS	DSA public key			

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Public key verification	Verify a public key	EC public key verification	EC public key		W, E	NSC_NSSGetFIPSStatus= 1
Random number generation	Generate random bitstrings	DRBG	Entropy input		W, E	NSC_NSSGetFIPSStatus= 1
			DRBG seed		G, E	
			DRBG internal state		G, E	
Message digest	Compute SHA hashes	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	None		N/A	NSC_NSSGetFIPSStatus= 1
Message authentication code (MAC)	Compute a MAC tag	AES-CMAC	AES key		W, E	NSC_NSSGetFIPSStatus= 1
		HMAC	HMAC key			
Key wrapping	Perform AES-based key wrapping	AES-KW AES-KWP	AES key		W, E	NSC_NSSGetFIPSStatus= 1
Diffie-Hellman shared secret computation	Perform DH shared secret computation	KAS-FFC-SSC	Diffie-Hellman private key (owner)		W, E	NSC_NSSGetFIPSStatus= 1
			Diffie-Hellman public key (peer)		W, E	
			Diffie-Hellman shared secret		G, R	
EC Diffie-Hellman shared secret computation	Perform ECDH shared secret computation	KAS-ECC-SSC	EC private key (owner)		W, E	NSC_NSSGetFIPSStatus= 1
			EC public key (peer)		W, E	
			EC Diffie-Hellman shared secret		G, R	
Key derivation for TLS	Perform key derivation for TLS	TLS 1.0/1.1 KDF TLS 1.2 KDF KDA HKDF	(EC) Diffie-Hellman shared secret		W, E	NSC_NSSGetFIPSStatus= 1
			TLS derived secret		G, R	
Key derivation for IKEv1 and IKEv2	Perform key derivation for IKEv1 and IKEv2	IKE KDF	(EC) Diffie-Hellman shared secret		W, E	
			IKE derived secret		G, R	
Password-based key derivation	Perform key derivation from a password/passphrase	PBKDF KDF	PBKDF password or passphrase		W, E	
			PBKDF derived key		G, R	
Key-based key derivation	Perform key derivation from a key	SP800-108 KDF with HMAC and CMAC-AES in Counter, Feedback, and Double-pipeline modes	Key derivation key		W, E	
			KBKDF derived key		G, R	

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Other FIPS-related Services						
Show status	Show module status	N/A	None	CO	N/A	N/A
Zeroization	Zeroize CSPs	N/A	All CSPs		Z	
Self-tests	Perform self-tests	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 AES ECB, CBC, KW KAS-FFC-SSC, KAS-ECC-SSC Hash_DRBG DSA ECSDA RSA KDA HKDF HMAC IKE KDF TLS KDF PBKDF See Table 14 for specfics	None		N/A	
On-Demand integrity tests	Perform self-tests	See Table 14	None		N/A	
Module installation and configuration	Install and configure module	N/A	None		N/A	
Module initialization	Initialize module	N/A	None		N/A	
Show module name and version	Show module name and version	N/A	None		N/A	

Table 10 - Approved Services

Table 11 lists the non-approved services. The details of the non-approved cryptographic algorithms available in non-approved mode can be found in Table 7.

Service	Description	Algorithms Accessed	Role	Indicator
Cryptographic Services				

Service	Description	Algorithms Accessed	Role	Indicator
Symmetric key generation	Generate symmetric key	DRBG When key length is less than 112 bits	CO	N/A
Symmetric encryption and decryption	Compute the cipher for encryption and decryption	Camellia, CAST, CAST3, CAST5, ChaCha20, DES, DES2, Triple-DES, CDMF, IDEA, RC2, RC4, RC5, SEED		
	Compute AES GCM using external IV	AES GCM with external IV		
Asymmetric key generation	Generate RSA and EC key pairs	RSA and EC with restrictions listed in Table 7		
Digital signature generation and verification	Generate and verify RSA and ECDSA signatures	RSA and ECDSA and message digest restrictions listed in Table 7		
DSA domain parameter generation	Generate DSA domain parameters	DSA		
DSA key generation	Generate DSA key pairs	DSA		
DSA digital signature generation	Generate DSA signatures	DSA		
DSA digital signature verification	Verify DSA signatures	DSA and message digest and key restrictions listed in Table 7		
Message digest	Compute message digest	MD2, MD5		
Message authentication code (MAC)	Compute HMAC	HMAC with restrictions listed in Table 7		
Key encapsulation	Perform RSA key encapsulation	RSA		
Key unencapsulation	Perform RSA key unencapsulation	RSA		
Diffie-Hellman shared secret computation	Perform DH shared secret computation	Diffie-Hellman restrictions listed in Table 7		
EC Diffie-Hellman shared secret computation	Perform ECDH shared secret computation	Restrictions listed in Table 7		
Key derivation	Perform key derivation	HKDF (outside of the TLS 1.3 protocol)		
		PBKDF1		
Key agreement	Perform key agreement	J-PAKE		

Table 11 - Non-Approved Services

5 Software/Firmware security

5.1 Integrity Techniques

The integrity of the module is verified by performing a DSA signature verification for each component that comprises the module. The module uses DSA signature verification with a 2048-bit key and SHA-256. If the DSA signature for any of the components cannot be verified, then the test fails, and the module enters the error state.

5.2 On-Demand Integrity Test

The module provides the Self-Test service to perform self-tests on demand which includes the pre-operational test (i.e., integrity test) and the cryptographic algorithm self-tests (CASTs). The Self-Tests service can be called on demand by invoking the sftk_FIPSRepeatIntegrityCheck() function which will perform integrity tests and the cryptographic algorithms self-tests. Additionally, the Self-Test service can be invoked by powering-off and reloading the module. During the execution of the on-demand self-tests, services are not available, and no data output is possible.

5.3 Executable Code

The module consists of executable code in the form of libsoftokn3.so, libnssdbm3.so and libfreeblpriv3.so shared libraries as stated in the Table 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 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, the debugger gdb and strace, as well as other tracing mechanisms offered by the Linux environment (ftrace, systemtap) shall not be used. 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 12 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[3]	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related keys
Module generated AES key (CSP)	128, 192, 256 bits	Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	Generated using the SP800-90Arev1 DRBG.	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form.	N/A	RAM	FC_DestroyObject	**Use:** Symmetric key generation **Related SSPs:** DRBG internal state
AES key (CSP)	128, 192, 256 bits	AES-CBC, AES-CMAC, AES-CTR, AES-CTS (CS1), AES-ECB, AES-GCM, AES-KW, AES-KWP A3575, A3576, A3577, A3580, A3581, A3582, A3583, A3585, A3586, A3587	N/A	MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in wrapped form.	N/A	RAM	FC_DestroyObject	**Use:** Symmetric encryption; Symmetric decryption; Message authentication code (MAC); Key wrapping and unwrapping **Related SSPs:** N/A
Module generated HMAC key (CSP)	112-256 bits	Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	Generated using the SP800-90Arev1 DRBG.	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** Symmetric key generation **Related SSPs:** DRBG internal state
HMAC key (CSP)	112-256 bits	HMAC A3575, A3584, A3588	N/A	MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in wrapped format.	N/A	RAM	FC_DestroyObject	**Use:** Message Authentication Code (MAC) **Related SSPs:** N/A
Module generated RSA private key (CSP)	112, 128, 149 bits	RSA A3575, A3584, A3588 Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	Generated using the FIPS 186-4 key generation method; the random value used in key generation is	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** RSA key generation **Related SSPs:** DRBG internal state; Module generated RSA public key

[3] see Table 5 for the certificate number of each algorithm listed in this column.

Key / SSP Name / Type	Strength	Security Function and Cert. Number[3]	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related keys
Module generated RSA public key (PSP)			obtained from the SP800-90Arev1 DRBG.	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) form				**Use:** RSA key generation **Related SSPs:** DRBG internal state; Module generated RSA private key
RSA private keys (CSP)	112, 128, 149 bits	RSA A3575, A3584, A3588	N/A	MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** Digital signature generation **Related SSPs:** RSA public key
RSA public key (PSP)				MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) form				**Use:** Digital signature verification **Related SSPs:** RSA private key
Module generated EC private key (CSP)	128, 192, 256 bits	KAS-ECC-SSC ECDSA A3575, A3584, A3588 Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	Generated using the FIPS 186-4 key generation method; the random value used in key generation is obtained from the SP800-90Arev1 DRBG.	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** EC key generation **Related SSPs:** DRBG internal state; Module generated EC public key
Module generated EC public key (PSP)				MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) form				**Use:** EC key generation and verification **Related SSPs:** DRBG internal state; Module generated EC private key
EC private key (CSP)	128, 192, 256 bits	KAS-ECC-SSC ECDSA A3575, A3584, A3588	N/A	MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** Digital signature generation; EC Diffie-Hellman shared secret computation **Related SSPs:** EC public key; EC Diffie-Hellman shared secret

Key / SSP Name / Type	Strength	Security Function and Cert. Number[3]	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related keys
EC public key (PSP)				MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) form				**Use:** Digital signature verification; EC Public key verification; EC Diffie-Hellman shared secret computation **Related SSPs:** EC private key; EC Diffie-Hellman shared secret
Module generated Diffie-Hellman private key (CSP)	112 to 200 bits	Safe primes A3575, A3584, A3588 Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	Generated using the SP 800-56Arev3 Safe Primes key generation method; random values are obtained from the SP800-90Arev1 DRBG.	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** Safe Primes key generation **Related SSPs:** DRBG internal state; Module generated Diffie-Hellman public key
Module generated Diffie-Hellman public key (PSP)				MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) form				**Use:** Safe Primes key generation and verification **Related keys:** DRBG internal state; Module generated Diffie-Hellman private key
Diffie-Hellman private key (CSP)	112 to 200 bits	KAS-FFC-SSC A3575, A3584, A3588	N/A	MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** Diffie-Hellman shared secret computation **Related SSPs:** Diffie-Hellman shared secret; Diffie-Hellman public key
Diffie-Hellman public key (PSP)				MD/EE **Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) form				**Use:** Diffie-Hellman shared secret computation **Related keys:** Diffie-Hellman shared secret; Diffie-Hellman private key
DSA public key (PSP)	80 to 128 bits	DSA A3575, A3584, A3588	N/A	MD/EE **Import:** CM from TOEPP Path.	N/A	RAM	FC_DestroyObject	**Use:** Digital signature verification; Integrity test

Key / SSP Name / Type	Strength	Security Function and Cert. Number[3]	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related keys
				Passed to the module via API parameters in plaintext (P) form				**Related SSPs:** N/A
Intermediate key generation value (CSP)	112 to 256 bits	CKG Vendor affirmed	SP 800-133r2 Section 4	N/A	N/A	RAM	Automatic	**Use:** RSA key generation; EC key generation; Safe primes Key generation **Related SSPs:** Module generated RSA public key; Module generated RSA private key; Module generated EC public key; Module generated EC private key; Module generated Diffie-Hellman public key; Module generated Diffie-Hellman private key
Diffie-Hellman shared secret (CSP)	112 to 200 bits	KAS-FFC-SSC A3575, A3584, A3588	N/A	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form.	SP 800-56Ar3 (DH shared secret computation)	RAM	FC_DestroyObject	**Use:** Diffie-Hellman shared secret computation **Related SSPs:** Diffie-Hellman public and private keys; TLS derived secret; IKE derived secret
EC Diffie-Hellman shared secret (CSP)	128 to 256 bits	KAS-ECC-SSC A3575, A3584, A3588	N/A	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form.	SP 800-56Ar3 (ECDH shared secret computation)	RAM	FC_DestroyObject	**Use:** EC Diffie-Hellman shared secret computation **Related SSPs:** EC Diffie-Hellman public and private keys; TLS derived secret; IKE derived secret
PBKDF password or passphrase (CSP)	N/A	PBKDF A3575, A3584, A3588	N/A	MD/EE **Import:** CM from TOEPP Path. Passed to the module via API	N/A	RAM	N/A	**Use:** Password-based key derivation **Related SSPs:** PBKDF derived key

Key / SSP Name / Type	Strength	Security Function and Cert. Number[3]	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related keys
				parameters in plaintext (P) form				
PBKDF derived key (CSP)	112 to 256 bits	PBKDF A3575, A3584, A3588	SP 800-133r2, Section 6.2 Generated during the PBKDF	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form	N/A	RAM	FC_DestroyObject	**Use:** Password-based key derivation (for storage purposes) **Related SSPs:** PBKDF password or passphrase
Entropy input (CSP)	256, 384 bits	ESV (Cert. E28, E29) Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	N/A	N/A	N/A	RAM	FC_Finalize	**Use:** Random number generation **Related SSPs:** DRBG seed
DRBG seed (CSP) IG D.L compliant	256 bits	Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	Generated from the entropy input as defined in SP800-90Arev1	N/A	N/A	RAM	FC_Finalize	**Use:** Random number generation **Related SSPs:** Entropy input; DRBG internal state
DRBG internal state: V, C (CSP) IG D.L compliant	256 bits	Hash_DRBG A3575, A3582, A3583, A3584, A3585, A3586, A3587, A3588	Generated from the DRBG seed as defined in SP800-90Arev1	N/A	N/A	RAM	FC_Finalize	**Use:** Random number generation **Related SSPs:** DRBG seed
TLS derived secret (CSP)	112 to 256 bits	KDF TLS, TLSv1.2 KDF A3575, A3584, A3588 KDA HKDF A3574	SP 800-133r2, Section 6.2 Derived during the TLS KDF per SP800-135rev1 and KDA HKDF per SP800-56Crev1	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form.	N/A	RAM	FC_DestroyObject	**Use:** Key derivation for TLS **Related SSPs:** Diffie-Hellman or EC Diffie-Hellman shared secret
IKE derived secret (CSP)	112 to 200 bits	IKE KDF A3579	SP 800-133r2, Section 6.2 Derived during the IKEv1 and IKEv2 KDF per SP800-135rev1	MD/EE **Export:** CM to TOEPP Path. Passed from the module via API parameters in wrapped form.	N/A	Stored in the module	FC_DestroyObject	**Use:** Key derivation for IKEv1 and IKEv2 **Related SSPs:** Diffie-Hellman or EC Diffie-Hellman shared secret
Key derivation key (CSP)	CMAC-AES: 128-256 bits of key strength HMAC: 112-256 bits	KBKDF SP800-108 A3578	N/A	MD/EE **Import:** CM from TOEPP Path. Passed to the module via API	N/A	RAM	FC_DestroyObject	**Use:** Key-based key derivation **Related SSPs:** KBKDF derived key

Key / SSP Name / Type	Strength	Security Function and Cert. Number[3]	Generation	Import/Export	Establishment	Storage	Zeroization	Use & related keys
				parameters in wrapped form.				
KBKDF derived key (CSP)	128 to 256 bits	KBKDF SP800-108 A3578	SP 800-133r2, Section 6.2 Generated during the KBKDF	MD/EE **Export:** CM to TOEPP Path. Passed to the module via API parameters in wrapped form.	N/A	RAM	FC_DestroyObject	**Use:** Key-based key derivation **Related SSPs:** Key derivation key

Table 12 - 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 symmetric keys, asymmetric keys, RSA signature generation and ECDSA signature generation. In addition, the module provides a Random Number Generation service to calling applications.

The DRBG supports the Hash_DRBG mechanism using SHA-256 and without prediction resistance. The module uses an SP800-90B-compliant entropy source specified in Table 13. 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
SP800-90B compliant Userspace Standalone CPU Time Jitter RNG (64-bit with internal timer) and Userspace Standalone CPU Time Jitter RNG (64-bit with external timer) (ESV Cert. E28[4], E29[5])	256 bits of entropy in the 256-bit output	Standalone Userspace CPU Time Jitter RNG version 3.4.0 entropy source (using SHA-3 as the vetted conditioning component) is located within the physical perimeter of the operational environment but outside the module cryptographic boundary

Table 13 - Non-Deterministic Random Number Generation Specification

[4] E28 Public Use Document: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E28_PublicUse.pdf

[5] E29 Public Use Document: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E29_PublicUse.pdf

9.2 SSP Generation

In accordance with FIPS 140-3 IG D.H, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys.

- For generating RSA and ECDSA keys, the module implements asymmetric cryptographic key generation (CKG) services compliant with [FIPS186-4], providing 112 to 149 bits of key strength for RSA and 128 to 256 bits for ECDSA.
- The public and private keys used in the EC Diffie-Hellman shared secret computation schemes are generated internally by the module using the EC key generation method compliant with [FIPS186-4] and [SP800-56Arev3], providing 128 to 256 bits of key strength.
- The public and private keys used in the Diffie-Hellman shared secret computation scheme are also compliant with [SP800-56Arev3]. The module generates keys using safe primes defined in RFC7919 and RFC3526, providing 112 to 200 bits of key strength as described in the next section.

Additionally, for AES and HMAC keys, the module provides key derivation service compliant with section 4 of [SP800-133rev2], providing 128 to 256 bits of key strength for AES and 112-256 bits of key strength for HMAC.

Random values used for symmetric and asymmetric key generation are obtained directly from an approved [SP800-90Arev1] 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), in compliance with section 4 of [SP800-133rev2].

The module supports the following key derivation methods:

- KDF for the TLS protocol, used as pseudo-random functions (PRF) for TLSv1.0/1.1 and TLSv1.2, compliant with [SP800-135rev1].
- KDF for the Internet Key Exchange (IKE) versions 1 and 2 protocol, compliant with [SP800-135rev1].
- HKDF for the TLS protocol TLSv1.3, compliant with [SP800-56Crev2].
- KBKDF, compliant with [SP800-108rev1]. This implementation can be used to generate secret keys from a pre-existing key-derivation-key.
- PBKDF2, compliant with option 1a of [SP800-132]. This implementation can only be used to derive keys for storage applications.

9.3 SSP establishment

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

For Diffie-Hellman, the module supports the use of safe primes from RFC7919 for domain parameters and key generation.

- 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)

For Elliptic Curve Diffie-Hellman, the module supports the NIST-defined P-256, P-384, and P-521 curves.

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

- Diffie-Hellman shared secret computation provides between 112 and 200 bits of strength.
- EC Diffie-Hellman shared secret computation provides between 128 and 256 bits of strength.

The module also provides the following key transport mechanisms in compliance with IG D.G:

- AES key wrapping using AES in KW, KWP modes with 128, 192, or 256 bit keys, providing 128, 192, or 256 bits of strength respectively.
- AES key wrapping using AES in GCM mode (in the context of the TLS protocol) with 128, 192, or 256 bit keys, providing 128, 192, or 256 bits of strength respectively.

These algorithms have been CAVP tested and the obtained certificates are listed in Table 5 of this security policy. These algorithms can be used to wrap SSPs with a security strength of 128, 192, or 256 bits, depending on the wrapping key size.

9.4 SSP Entry and Output

The module does not support manual key entry or intermediate key generation key output. The SSPs has to be provided to the module via API input parameters in encrypted form (using the FC_UnwrapKey function) and output via API output parameters also in encrypted form (using the FC_WrapKey function). The module uses AES with KW/KWP compliant with SP800-38F as the approved key wrapping method. PSPs can be imported and exported in plaintext.

9.5 SSP Storage

The module employs the cryptographic keys and CSPs in the approved mode of operation as listed in Table 12. The module does not perform persistent storage of keys. Note that the private key database (provided with the files key3.db/key4.db) is outside the cryptographic boundary.

Symmetric keys, HMAC keys, public and private keys are provided to the module by the calling application via API input parameters and are destroyed by the module when invoking the appropriate API function calls.

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 12.

- The FC_Finalize, FC_CloseSession or FC_CloseAllSession functions overwrite the memory occupied by keys with “zeros” and deallocate the memory with the regular memory deallocation operating system call.
- The FC_DestroyObject function overwrites with "zeros" the area occupied by the secret key in the private key database.

10 Self-tests

The module performs the pre-operational self-test and CASTs automatically when the module is loaded into memory. The 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 self-tests, 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 tests and 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.

In order to verify whether the self-tests have succeeded, the calling application may invoke the FC_Initialize function. The function will return CKR_OK if the module is operational, CKR_DEVICE_ERROR if the module is in the Error state.

10.1 Pre-Operational Tests

The module performs pre-operational tests automatically when the module is powered on. The pre-operational self-tests ensure that the module is not corrupted. The module transitions to the operational state only after the pre-operational self-tests are passed successfully.
The types of pre-operational self-tests are described in the next sub-sections.

10.1.1 Pre-Operational Software Integrity Test

The module performs the integrity test using DSA signature verification with a 2048-bit key and SHA-256. The details of integrity test are provided in section 5.1.

10.2 Conditional Tests

10.2.1 Cryptographic algorithm tests

Table 14 specifies all the CASTs. 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.

Algorithm	Test
AES	KAT AES in ECB mode with 128-, 192- and 256-bit keys, encryption and decryption (separately tested). KAT AES in CBC mode with 128-, 192- and 256-bit keys, encryption and decryption (separately tested). KAT AES in KW mode with 128-, 192- and 256-bit keys, encryption and decryption (separately tested).
Diffie-Hellman	Primitive “Z” Computation KAT with 2048-bit key.
DRBG	KAT Hash_DRBG with SHA-256 without PR.
DSA	KAT DSA signature verification with L=2048, N=224 and SHA-224.

Algorithm	Test
EC Diffie-Hellman	Primitive “Z” Computation KAT with P-256 curve.
ECDSA	KAT ECDSA signature generation and verification with P-256 and SHA-224 (separately tested).
HKDF	KAT HKDF with HMAC-SHA2-256, HMAC-SHA2-384.
HMAC	KAT HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512.
IKE KDF	KAT IKE PRF using HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384 and HMAC-SHA2-512.
KDF	KAT SP800-108 Counter KDF with HMAC-SHA-256.
PBKDF2 KDF	KAT with HMAC-SHA-1 and HMAC-SHA2-256.
RSA	KAT RSA PKCS#1 v1.5 signature generation and verification with 2048-bit key and SHA-256, SHA-384 and SHA-512 (separately tested).
SHS	KAT SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512.
TLS KDF	KAT TLS KDF for v1.0/v1.1 KAT TLS KDF for v1.2 with SHA-224, SHA-256, SHA-384, SHA-512

Table 14 - Conditional Cryptographic Algorithms Self-Tests

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 SHA-224, signature generation and verification.
RSA key generation	PCT using SHA-224, signature generation and verification.
Safe primes key generation	PCT according to section 5.6.2.1.4 of [SP800-56Arev3].
EC Diffie-Hellman key generation	PCT using SHA2-224, signature generation and verification (covered by pre-requisite algorithm ECDSA’s PCT).

Table 15 - Pairwise Consistency Test

10.2.3 Periodic/On-Demand Self-Test

The module provides the Self-Test service to perform self-tests on demand which includes the pre-operational test (i.e., integrity test) and the cryptographic algorithm self-tests (CASTs). The Self-Tests service can be called on demand by invoking the sftk_FIPSRepeatIntegrityCheck() function which will perform integrity tests and the cryptographic algorithms self-tests. Additionally, the Self-Test service can be invoked by powering-off and reloading the module. During the execution of the on-demand self-tests, services are not available, and no data output is possible.

10.3 Error States

The Module enters the Error state returning the CKR_DEVICE_ERROR error code, on failure of pre-operational self-tests or conditional test. In the Error state, all data output is inhibited and no cryptographic operation is allowed. The error can be recovered by powering-off and reloading the module.

Error State	Cause of Error	Status Indicator
Error state	Failure of pre-operational tests or conditional tests.	CKR_DEVICE_ERROR error code

Table 16 - Error States

Self-test errors transition the module into an error state that keeps the module operational but prevents any cryptographic related operations. The module must be restarted and perform the per-operational self-test and the CASTs to recover from these errors. If failures persist, the module must be re-installed.

11 Life-cycle assurance

11.1 Delivery and Operation

11.1.1 Module Installation

The Netscape Portable Runtime (NSPR) package (mozilla-nspr-4.23-3.9.1.x86_64.rpm) is a prerequisite for the module. The mozilla-nspr package must be installed in the operating environment.

The Crypto Officer can install the RPM packages containing the module as listed in Table 18 using the zypper tool. 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 FIPS, 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 FIPS 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 FIPS and the module will not operate as a FIPS validated module properly.

11.1.3 Access to Audit Data

The module may use the Unix syslog function and the audit mechanism provided by the operating system to audit events. Auditing is turned off by default. Auditing capability must be turned on as

part of the initialization procedures by setting the environment variable NSS_ENABLE_AUDIT to 1. The Crypto Officer must also configure the operating system's audit mechanism.

The module uses the syslog function to audit events, so the audit data are stored in the system log. Only the root user can modify the system log. On some platforms, only the root user can read the system log; on other platforms, all users can read the system log. The system log is usually under the /var/log directory. The exact location of the system log is specified in the /etc/syslog.conf file. The module uses the default user facility and the info, warning, and err severity levels for its log messages.

The module can also be configured to use the audit mechanism provided by the operating system to audit events. The audit data would then be stored in the system audit log. Only the root user can read or modify the system audit log. To turn on this capability it is necessary to create a symbolic link from the library file /usr/lib64/libaudit.so.1 to /usr/lib64/libaudit.so.1.0.0.

11.1.4 Module Installation for Vendor Affirmed Platforms

Table 17 includes the information on module installation process for the vendor affirmed platforms that are listed in Table 4.

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 17 - 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.

11.1.5 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 18 lists the RPM packages that contain the FIPS validated module. The "Show module name and version" is implemented by accessing the CKA_NSS_VALIDATION_MODULE_ID attribute of the CKO_NSS_VALIDATION object in the default slot. The object attribute contains the value “SUSE Linux Enterprise NSS 3.79.4-150400.3.29.1”, which matches the service output and the version information provided in the RPM packages where the module is distributed, and map to version 3.1 of the cryptographic module.

Processor Architecture	RPM Packages
Intel 64-bit	libsoftokn3-3.79.4-150400.3.29.1.x86_64.rpm libsoftokn3-hmac-3.79.4-150400.3.29.1.x86_64.rpm libfreebl3-3.79.4-150400.3.29.1.x86_64.rpm libfreebl3-hmac-3.79.4-150400.3.29.1.x86_64.rpm
AMD 64-bit	libsoftokn3-3.79.4-150400.3.29.1.x86_64.rpm libsoftokn3-hmac-3.79.4-150400.3.29.1.x86_64.rpm libfreebl3-3.79.4-150400.3.29.1.x86_64.rpm libfreebl3-hmac-3.79.4-150400.3.29.1.x86_64.rpm
IBM z15	libsoftokn3-3.79.4-150400.3.29.1.s390x.rpm libsoftokn3-hmac-3.79.4-150400.3.29.1.s390x.rpm libfreebl3-3.79.4-150400.3.29.1.s390x.rpm libfreebl3-hmac-3.79.4-150400.3.29.1.s390x.rpm
ARMv8 64-bit	libsoftokn3-3.79.4-150400.3.29.1.aarch64.rpm libsoftokn3-hmac-3.79.4-150400.3.29.1.aarch64.rpm libfreebl3-3.79.4-150400.3.29.1.aarch64.rpm libfreebl3-hmac-3.79.4-150400.3.29.1.aarch64.rpm
IBM Power10 64-bit	libsoftokn3-3.79.4-150400.3.29.1.aarch64.rpm libsoftokn3-hmac-3.79.4-150400.3.29.1.aarch64.rpm libfreebl3-3.79.4-150400.3.29.1.aarch64.rpm libfreebl3-hmac-3.79.4-150400.3.29.1.aarch64.rpm

Table 18 - RPM packages

11.2.1 Considerations for the approved mode

In order to run in in the approved mode, the module must be operated using the approved services, with their corresponding approved and allowed cryptographic algorithms provided in this Security Policy (see section 2). In addition, key sizes must comply with [SP800-131Arev2].

The following module initialization steps must be followed before starting to use the NSS module:

- Set the environment variable NSS_ENABLE_AUDIT to 1 before using the module.
- Use the FC_GetFunctionList function to obtain pointer references to the API. The function returns a CK_FUNCTION_LIST structure containing function pointers named as the API functions but with the "C_" prefix (e.g. C_Initialize and C_Finalize). The function pointers reference the "FC_" prefixed functions.

- Use FC_Initialize (function pointer C_Initialize) to initialize the module. Ensure that the function returns CKR_OK, which means that the module was properly configured and the power-on self-tests were successful. If the function returns a different code, the module must be reset and initialized again.

The module can be configured to use different private key database formats: key3.db or key4.db. “key3.db” format is based on the Berkeley DataBase engine and should not be used by more than one process concurrently. “key4.db” format is based on SQL DataBase engine and can be used concurrently by multiple processes. Both databases are considered outside the cryptographic boundary and all data stored in these databases are considered stored in plaintext. The interface code of the NSS cryptographic module that accesses data stored in the database is considered part of the cryptographic boundary.

Secret and private keys, plaintext passwords, and other security-relevant data items are maintained under the control of the cryptographic module. Secret and private keys must be entered to the module from the calling application and output from the module to the calling application in encrypted form using the FC_WrapKey and FC_UnwrapKey functions, respectively. The cryptographic algorithms allowed for this purpose in the approved mode of operation are AES in KW mode.

All cryptographic keys used in the approved mode of operation must be generated in the approved mode or imported while running in the approved mode.

11.2.2 AES GCM IV

The AES GCM IV generation is in compliance with section 8.2.2 of [SP800-38D] and IG C.H scenario 2 [FIPS140-3_IG], in which the GCM IV is generated internally at its entirety randomly. The module uses the DRBG that is compliant with SP800-90A-rev1, for generating the IV. The DRBG is fully seeded with entropy provided by the SP800-90B compliant entropy source that is not within the cryptographic boundary of the module but within its physical perimeter.

The GCM IV must be at least 96 bits in length, which is enforced by the module.

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

The module also implements AES GCM for being used in the TLS v1.2 and v1.3 protocols. AES GCM IV generation is in compliance with [FIPS140-3_IG] IG C.H for both protocols as follows:

- For TLS v1.2, IV generation is in compliance with scenario 1.a of IG C.H and [RFC5288]. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of [SP800-52rev2].
- For TLS v1.3, IV generation is in compliance with scenario 5 of IG C.H and [RFC8446]. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of [SP800-52rev2].

Additionally, the module offers an internal deterministic IV generation mode compliant with Scenario 3 of FIPS 140-3 IG C.H. The size of the fixed (name) field used by this IV generation mode is at least 32 bits. The module then internally generates a 32 bit or longer deterministic non-repetitive counter. The module explicitly ensures that this counter is monotonically increasing at each invocation of the AES-GCM for the same encryption key, and that this counter does not exhaust all its possible values. The generated GCM IV is at least 96 bits in length.

The IV generated in both scenarios is only used within the context of the TLS protocol. The design of the TLS protocol 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.

11.2.3 Key derivation using SP800-132 PBKDF

The module provides password-based key derivation (PBKDF), compliant with SP800-132. 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] and IG D.N, 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}. In a worst-case scenario where the user selects a password consisting of only digits, the guessing probability is estimated to be 10^{-20}.

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

11.2.4 SP 500-56Ar3 Assurances

To comply with the assurances found in Section 5.6.2 of SP 800-56Ar3, the operator must use the module together with an application that implements the TLS protocol. Additionally, the module's approved Key Pair Generation service (see Section 4.1.1) must be used to generate ephemeral Diffie-Hellman or EC Diffie-Hellman key pairs, or the key pairs must be obtained from another FIPS-validated module. As part of this service, the module will internally perform the full public key validation of the generated public key.

11.2.5 IG C.F Compliance

All of the RSA modulus sizes used by the cryptographic module have been CAVP tested and the certificates are listed in Table 5 of this security policy. There are no untested RSA modulus sizes used by the cryptographic module.

12 Mitigation of other attacks

12.1 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 uses the following blinding technique: instead of using the RSA decryption directly, a blinded value $y = x\, r^e \ mod\ n$ is decrypted and the unblinded value $x' = y'\, r^{-1} \ mod\ n$ returned. The blinding value r is a random value with the size of the modulus *n*.

12.2 Cache invariant modular exponentiation

Modular exponentiation used in DSA and RSA is vulnerable to cache-timing attacks. The module implements a variant of the modular exponentiation proposed by Colin Percival to defend against these attacks.

12.3 Double-checking RSA signatures

Arithmetic errors in RSA signatures might leak the private key. The module verifies the RSA signature generated after the cryptographic operation is performed.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
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
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
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Schema
KAT	Known Answer Test
KW	AES Key Wrap
KWP	AES Key Wrap with Padding
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

Appendix B. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** March 2024 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 (Draft)** 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)** August 2015 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
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography** Specifications Version 2.1 February 2003 https://www.ietf.org/rfc/rfc3447.txt
RFC5288	**AES Galois Counter Mode (GCM) Cipher Suites for TLS** August 2008 https://datatracker.ietf.org/doc/html/rfc5288
RFC8446	**The Transport Layer Security (TLS) Protocol Version 1.3** August 2018 https://datatracker.ietf.org/doc/html/rfc8446

SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-38A-Addendum	**NIST Special Publication 800-38A-Addendum - Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode** October 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.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
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.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
SP800-52rev2	**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 2 - 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	**NIST Special Publication 800-56C Revision 2 - 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
SP800-108rev1	**NIST Special Publication 800-108 Revision 1 - Recommendation for Key Derivation Using Pseudorandom Functions (Revised)** August 2022 https://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
SP800-131Arev2	**NIST Special Publication 800-131A Revision 1- 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://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.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