SUSE Linux Enterprise Server libgcrypt Cryptographic Module version 3.1

FIPS 140-2 Non-Proprietary Security Policy

Doc version 3.1.4

Last update: 2021-11-23

Prepared by:
atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Table of contents

1 Cryptographic Module Specification

This document is the non-proprietary security policy for the SUSE Linux Enterprise Server libgcrypt Cryptographic Module version 3.1. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS 140-2 (Federal Information Processing Standards Publication 140-2) for a security level 1 module.

FIPS 140-2 details the requirements of the Governments of the U.S. and Canada for cryptographic modules, aimed at the objective of protecting sensitive but unclassified information. For more information on the FIPS 140-2 standard and validation program please refer to the NIST website at http://csrc.nist.gov/.

Throughout the document, “the libgcrypt module” and “the module” are also used to refer to the SUSE Linux Enterprise Server libgcrypt Cryptographic Module version 3.1.

1.1 Module Overview

The SUSE Linux Enterprise Server libgcrypt Cryptographic Module is a software library implementing general purpose cryptographic algorithms. The module provides cryptographic services to applications running in the user space of the underlying operating system through a C language application program interface (API).

For the purpose of the FIPS 140-2 validation, the module is a software-only, multi-chip standalone cryptographic module validated at overall security level 1. Table 1 shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard:

FIPS 140-2 Section		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	1

Table 1: Security Levels

Table 2 lists the software components of the cryptographic module, which defines its logical boundary. The module is provided for the 64-bit Intel architectures.

Component	Description
/usr/lib64/libgcrypt.so.20.2.2	Shared library for cryptographic algorithms.
/usr/lib64/.libgcrypt.so.20.hmac	Integrity check HMAC value for the libgcrypt shared library.

Table 2: Cryptographic Module Components

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

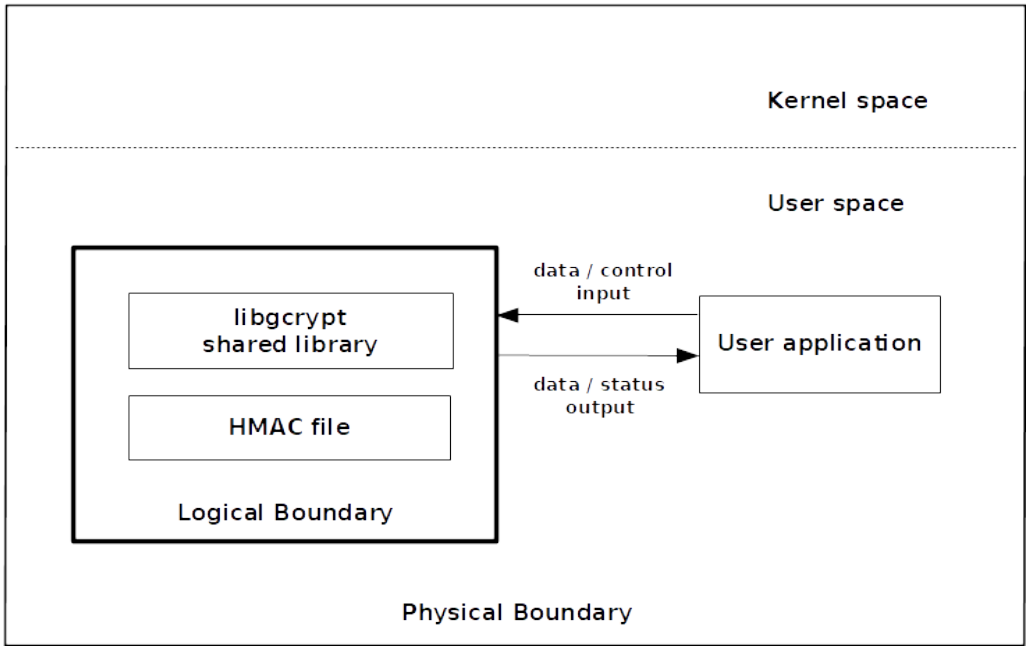

Figure 1: Software Block Diagram

The module is aimed to run on a general purpose computer (GPC). Table 3 shows the platform on which the module has been tested:

Platform	Processor	Test Configuration
Dell EMC PowerEdge 640	Intel Cascade Lake Xeon Gold 6234	SUSE Linux Enterprise Server 15 SP2 with and without PAA
IBM System Z/15	IBM z15	SUSE Linux Enterprise Server 15 SP2
Gigabyte R181-T90	Cavium ThunderX2 CN9975 ARMv8	SUSE Linux Enterprise Server 15 SP2 with and without PAA

Table 3: Tested Platforms

Note: Per FIPS 140-2 IG G.5, 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.

The physical boundary of the module is the surface of the case of the tested platform. Figure 2 shows the hardware block diagram including major hardware components of a GPC.

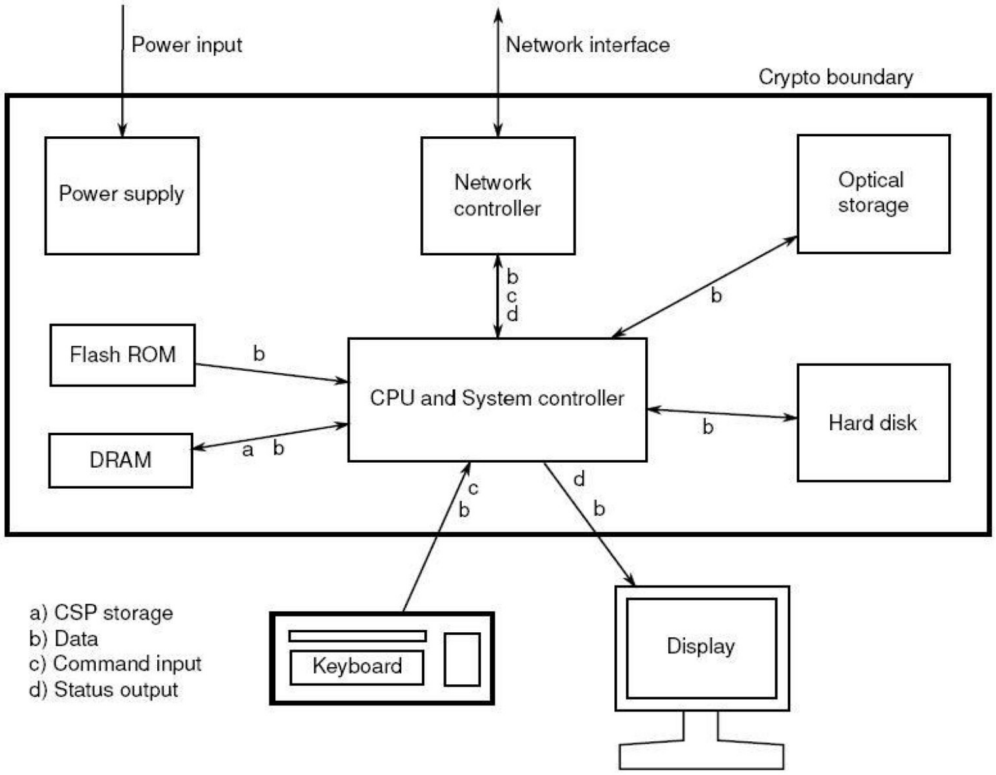

Figure 2: Hardware Block Diagram

1.2 Modes of Operation

The module supports two modes of operation:

- FIPS mode (the Approved mode of operation): only approved or allowed security functions with sufficient security strength can be used.
- non-FIPS mode (the non-Approved mode of operation): only non-approved security functions can be used.

The module enters FIPS mode after power-up tests succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.

Critical security parameters (CSPs) used or stored in FIPS mode are not used in non-FIPS mode, and vice versa.

2 Cryptographic Module Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the API through which applications request services. The ports and interfaces are shown in the following table.

FIPS Interface	Physical Port	Logical Interface
Data Input	None	API input parameters for data.
Data Output	None	API output parameters for data.
Control Input	None	API function calls, API input parameters for control input, /proc/sys/crypto/fips_enabled control file.
Status Output	None	API return codes, API output parameters for status output.
Power Input	PC Power Supply Port	N/A

Table 4: Ports and Interfaces

3 Roles, Services and Authentication

3.1 Roles

The module supports the following roles:

- User role: performs cryptographic services (in both FIPS mode and non-FIPS mode), key zeroization, get status, and on-demand self-test.
- Crypto Officer role: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed by the entity accessing the module to request services. No authentication is required.

3.2 Services

The module provides services to the users that assume one of the available roles. All services are shown in Table 5 and Table 6.

Table 5 lists the services available in FIPS mode. 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). The following convention is used to specify access rights to a CSP:

- *Create*: the calling application can create a new CSP.
- *Read*: the calling application can read the CSP.
- *Update*: the calling application can write a new value to the CSP.
- *Zeroize*: the calling application can zeroize the CSP.
- *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 7.

Service	Algorithms	Role	Keys/CSPs	Access
Cryptographic Services				
Symmetric encryption and decryption	AES	User	AES key	Read
	Three-key Triple-DES	User	Three-key Triple-DES key	Read
Symmetric decryption	Two-key Triple-DES	User	Two-key Triple-DES key	Read
RSA key generation	RSA, DRBG	User	RSA public and private keys	Create
RSA digital signature generation and verification	RSA, SHS	User	RSA public and private keys	Read
DSA key generation	DSA, DRBG	User	DSA public and private keys	Create
DSA domain parameter generation and verification	DSA	User	None	n/a
DSA digital signature generation and verification	DSA, SHS	User	DSA public and private keys	Read

Service	Algorithms	Role	Keys/CSPs	Access
ECDSA key generation	ECDSA, DRBG	User	ECDSA public and private keys	Create
ECDSA public key validation	ECDSA	User	ECDSA public key	Read
ECDSA signature generation and verification	ECDSA, DRBG, SHS	User	ECDSA public and private keys	Read
Random number generation	DRBG	User	Entropy input string, seed material	Read
			Internal state	Update
Message digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	User	None	N/A
	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256	User	None	N/A
Message authentication code (MAC)	HMAC	User	HMAC key	Read
	CMAC with AES	User	AES key	Read
	CMAC with Triple-DES	User	Triple-DES key	Read
Key encapsulation	RSA	User	RSA public and private keys	Read
Key wrapping	AES-KW	User	AES key	Read
Key derivation	PBKDF2 with HMAC	User	Password/passphrase	Read
			Derived key	Create
Other FIPS-related Services				
Show status	N/A	User	None	N/A
Zeroization	N/A	User	All CSPs	Zeroize
Self-tests	AES, DSA, ECDSA, DRBG, HMAC, RSA, SHS, Triple-DES	User	None	N/A
Module installation and configuration	N/A	Crypto Officer	None	N/A
Module initialization	N/A	Crypto Officer	None	N/A

Table 5: Services in FIPS mode of operation

Table 6 lists the services only available in non-FIPS mode of operation. The details of the non-approved cryptographic algorithms available in non-FIPS mode can be found in Table 9.

Service	Algorithm / Modes	Role	Keys	Access
Cryptographic Services				
Symmetric encryption and decryption	AES (GCM, OCB)	User	Symmetric key	Read
	ARC4, Blowfish, Camellia, CAST5, ChaCha20, DES, IDEA, RC2, RC4, Salsa20, SEED, Serpent, and Poly1305, Twofish			
Symmetric encryption	Two-key Triple-DES	User	Two-key Triple-DES key	Read
Asymmetric key generation	ElGamal	User	RSA, DSA or ECDSA public and private keys	Create
	RSA, DSA and ECDSA restrictions listed in Table 9			
Digital signature generation and verification	EC-GOST, EdDSA, El Gamal	User	RSA, DSA or ECDSA public and private keys	Read
	RSA, DSA and ECDSA and message digest restrictions listed in Table 9			
Random number generation	Hash and HMAC DRBG using SHA-384	User	Entropy input string, seed material	Read
			Internal state	Update
Message digest	Blake2, Gost, MD4, MD5, RMD160, Tiger, Whirpool	User	None	N/A
Message authentication code (MAC)	GMAC	User	HMAC key, two-key Triple-DES key	Read
	HMAC and CMAC restrictions listed in Table 9			
RSA key encapsulation	RSA keys smaller than 2048 bits.	User	RSA key pair	Read
Key derivation	KDF using OpenGPG S2K and SCRYPT	User	Password/passphrase	Read
			Derived key	Create

Table 6: Services in non-FIPS mode of operation

3.3 Operator Authentication

The module does not implement user authentication. The role of the user is implicitly assumed based on the service requested.

3.4 Algorithms

The module provides C implementation of cryptographic algorithms. It also provides multiple implementations of algorithms for the different processor architectures:

- For the Intel Xeon processor architecture:
  - use of AES-NI instructions for AES implementations (PAA);

  - use of AVX, SSSE3, SHLD and strict assembler instructions (non-PAA) for SHA implementations;
  - use of C and assembler implementations for the rest of the algorithms (non-PAA).
- For the IBM z/15 processor architecture:
  - use of C and assembler implementations (non-PAI).
- For the ARMv8 processor architecture:
  - use of the Crypto Extensions instructions for AES and SHA implementations (PAA);
  - use of NEON instructions for SHA implementation (PAA);
  - use of C and assembler implementations for the rest of the algorithms (non-PAA).

The module uses the most efficient implementation based on the processor's capability. Notice that only one algorithm implementation can be executed in runtime.

Table 7 lists the approved algorithms, the CAVP certificates, and other associated information of the cryptographic implementations in FIPS mode. Please refer to Appendix A for more detailed information about the algorithm implementations tested for each CAVP certificate.

Note: Only the CAVP certificates listed in this Table are used by the module.

Algorithm	Mode / Method	Key Lengths, Curves (in bits)	Use	Standard	CAVP Certs
AES	ECB, CBC, CFB8, CFB128, OFB, CTR	128, 192, 256	Data encryption and decryption	FIPS197, SP800-38A	A482 A485 A486 A1152
	CMAC	128, 192, 256	MAC generation and verification	SP800-38B	
	CCM	128, 192, 256	Data encryption and decryption	SP800-38C	
	XTS	128, 256	Data encryption and decryption for data storage	SP800-38E	
DRBG	CTR_DRBG: AES-128, AES-192, AES-256 with DF, with/without PR	N/A	Deterministic random bit generation	SP800-90A	A482 A485 A486 A1152
	Hash_DRBG: SHA-1, SHA-256, SHA-512 with/without PR	N/A			A482 A485 A486 A487 A1152
	HMAC_DRBG: SHA-1, SHA-256, SHA-512 with/without PR	N/A			A482 A485 A486 A487 A1152
DSA		L=2048, N=224 L=2048, N=256 L=3072, N=256	Key pair generation	FIPS186-4	A482 A485 A486 A487 A1152
	SHA-224	L=2048, N=224	Domain parameter generation		
	SHA-256	L=2048, N=256 L=3072, N=256			
	SHA-224	L=2048, N=224	Digital signature		

Algorithm	Mode / Method	Key Lengths, Curves (in bits)	Use	Standard	CAVP Certs
	SHA-224, SHA-256	L=2048, N=256 L=3072, N=256	generation		
	SHA-224	L=2048, N=224	Domain parameter verification		
	SHA-256	L=2048, N=256 L=3072, N=256			
	SHA-1	L=1024, N=160	Digital signature verification		
	SHA-1, SHA-224	L=2048, N=224			
	SHA-1, SHA-224, SHA-256	L=2048, N=256 L=3072, N=256			
ECDSA		P-256, P-384, P-521	Key pair generation Public key verification	FIPS186-4	A482 A485 A486 A487 A1152
	SHA-224, SHA-256, SHA-384, SHA-512	P-224, P-256, P-384, P-521	Digital signature generation		
	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	P-192, P-224, P-256, P-384, P-521	Digital signature verification		
HMAC	SHA-1	112 or greater	Message authentication code	FIPS198-1	A482 A484 A485 A486 A488 A1152
	SHA-224, SHA-256, SHA-384, SHA-512	112 or greater	Message authentication code	FIPS198-1	A482 A485 A486 A1152
	SHA3-224, SHA3-256, SHA3-384, SHA3-512				A482 A487 A1152
KDF PBKDF (vendor affirmed[1])	HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512		Key derivation	SP800-132	A482 A485 A486 A487 A1152
KTS	AES in KW mode	128, 192, 256	Key wrapping and unwrapping	SP800-38F	A482 A485 A486 A1152
RSA		2048, 3072, 4096	Key pair generation	FIPS186-4	A482 A485 A486 A487 A1152
	PKCS#1v1.5: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096	Digital signature generation		

1 PBKDF is vendor affirmed as the module doesn't implement the required known answer test (KAT) during self-tests.

Algorithm	Mode / Method	Key Lengths, Curves (in bits)	Use	Standard	CAVP Certs
	PSS: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096			
	PKCS#1v1.5: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096	Digital signature verification		
	PSS: SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096			
SHA-3	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256		Message Digest	FIPS202	A482 A487 A1152
SHS	SHA-1	N/A	Message digest	FIPS180-4	A482 A484 A485 A486 A488 A1152
	SHA-224, SHA-256, SHA-384, SHA-512	N/A	Message digest	FIPS180-4	A482 A485 A486 A1152
Triple-DES	ECB, CBC, CFB8, CFB64, OFB, CTR	192 (two-key Triple-DES)	Data decryption	SP800-67 SP800-38A	A482
		192 (three-key Triple-DES)	Data encryption and decryption		
	CMAC	192	MAC generation and verification	SP800-67 SP800-38B	

Table 7: Approved Cryptographic Algorithms

3.5 Allowed Algorithms

Table 8 describes the non-approved but allowed algorithms in FIPS mode.

Algorithm	Use
RSA key encapsulation with encryption and decryption primitives with keys equal or larger than 2048 bits up to 15360 or more.	Key establishment; allowed per [FIPS140-2_IG] D.9
NDRNG	The module obtains the entropy data from a NDRNG to seed the DRBG.

Table 8: Non-Approved but Allowed Algorithms

3.6 Non-Approved Algorithms

Table 9 shows the non-Approved cryptographic algorithms implemented in the module that are only available in non-FIPS mode.

Algorithm	Use
AES in OCB mode.	Data encryption and decryption

Algorithm	Use
AES in GCM mode.	Authenticated data encryption and decryption.
Arcfour (RC4), Blowfish, Camellia, CAST5, ChaCha20, DES, GOST28147, RC2, Salsa20, SEED, Serpent, Twofish.	Data encryption and decryption.
2-key Triple-DES.	Data encryption.
Poly1305.	Authenticated data encryption and decryption, message authentication code.
Blake2, MD2, MD4, MD5, RMD160, GOST, Streebog, Tiger, Whirpool.	Message digest.
CMAC with non-approved symmetric algorithms.	Message authentication code.
HMAC using keys less than 112 bits of length. HMAC with non-approved message digest algorithms.	Message authentication code.
GMAC.	Message authentication code.
SHA-1.	Message digest in digital signature generation.
DSA with L=1024, 7680, or 15360.	Key pair generation, domain parameter generation, domain parameter verification.
DSA with L=1024, 7680, or 15360. DSA with L=2048, N=224 and using SHA-1, SHA-256, SHA-384, or SHA-512. DSA with L=2048, N=256 or L=3072, N=256 and using SHA-1, SHA-384, or SHA-512.	Digital signature generation.
DSA with L=7680, 15360. DSA with L=1024, N=160 and using SHA-224, SHA-256, SHA-384, or SHA-512. DSA with L=2048, N=224 and using SHA-256, SHA-384, or SHA-512. DSA with L=2048, N=256 or L=3072, N=256 and using SHA-384, or SHA-512.	Digital signature verification
RSA with keys smaller than 2048 bits or greater than 4096 bits.	Key pair generation, digital signature generation.
RSA with keys smaller than 1024 bits or greater than 4096 bits.	Digital signature verification.
RSA with keys smaller than 2048 bits.	Key encapsulation.
ECDSA with P-192 curve.	Key pair generation, public key validation, domain parameter generation and verification, digital signature generation.
ECDSA with P-224 curve.	Key pair generation, domain parameter generation and verification.

Algorithm	Use
Non-NIST curves (i.e. Ed25519, Curve25519, brainpool and GOST curves).	Key pair generation, public key validation, domain parameter generation and verification, digital signature generation and verification.
ECC-Gost, EdDSA.	Key pair generation, public key validation, signature generation and verification.
Elgamal.	Key pair generation, public key validation, signature generation and verification, encryption and decryption.
OpenPGP S2K, SCRYPT.	Key derivation.
Hash_DRBG and HMAC_DRBG with SHA-384.	Random number generation.

Table 9: Non-Approved Cryptographic Algorithms

4 Physical Security

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

5 Operational Environment

This module operates in a modifiable operational environment per the FIPS 140-2 level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in Table 3.

The SUSE Linux Enterprise Server operating system is used as the basis of other products which include but are not limited to:

- SLES
- SLES for SAP
- SLED
- SLE Micro

Compliance is maintained for these products whenever the binary is found unchanged.

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.

5.1 Policy

The operating system is restricted to a single operator; concurrent operators are explicitly excluded.

The application that requests cryptographic services is the single user of the module.

Instrumentation tools like the ptrace system call, gdb and strace utilities, 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 Cryptographic Key Management

Table 10 summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module. Key sizes allowed in the approved mode of operation are specified in Table 7 and Table 8.

Name	Generation	Entry and Output	Zeroization
AES keys	Not applicable. Key material is entered via API parameters.	Keys are passed into the module via API input parameters in plaintext.	`gcry_cipher_close()`, `gcry_free()`
Triple-DES keys			
HMAC keys			`gcry_mac_close()`, `gcry_free()`
RSA public and private keys	Public and private keys are generated using the FIPS 186-4 key generation method; the random value used in key generation is obtained from the SP800-90A DRBG.	Keys are passed into the module via API input parameters in plaintext. Keys are passed out of the module via API output parameters in plaintext.	`gcry_sexp_release()`, `gcry_mpi_release()`, `gcry_free()`
DSA public and private keys			
ECDSA public and private keys			`gcry_sexp_release()`, `gcry_mpi_release()`, `gcry_ctx_release()`, `gcry_mpi_point_release()`, `gcry_free()`
Password or passphrase	Not applicable. Key material is entered via API parameters.	The key is passed into the module via API input parameters in plaintext.	`gcry_free()`
Derived key	Generated during the PBKDF	Keys are passed out of the module via API output parameters in plaintext.	`gcry_free()`
Entropy input string and seed material	Obtained from the NDRNG	Not applicable, it remains within the logical boundary.	`gcry_ctrl (GCRYCTL_TERM_SECMEM)`
DRBG internal state: V value, C value, key (if applicable)	Derived from entropy input as defined in SP800-90A	Not applicable, it remains within the logical boundary.	

Table 10: Life cycle of Keys or CSPs

The following sections describe how CSPs, in particular cryptographic keys, are managed during its life cycle.

6.1 Random Number Generation

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90A] for the creation of RSA, DSA and ECDSA keys, and DSA and ECDSA signature generation. In addition, the module provides a Random Number Generation service to calling applications.

The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms. The DRBG is initialized during module initialization; the module loads by default the DRBG using the HMAC_DRBG mechanism with SHA-256 and without prediction resistance. A different DRBG mechanism can be chosen by invoking the `gcry_control(GCRYCTL_DRBG_REINIT)` function.

The module uses a Non-Deterministic Random Number Generator (NDRNG) as the entropy source for seeding the DRBG. The NDRNG is provided by the operational environment (i.e., Linux RNG), which is within the module's physical boundary but outside of the module's

logical boundary. The NDRNG provides at least 128 bits of entropy to the DRBG during initialization (seed) and reseeding (reseed).

The Linux kernel performs conditional self-tests on the output of NDRNG to ensure that consecutive random numbers do not repeat. The module performs the DRBG health tests as defined in section 11.3 of [SP800-90A].

6.2 Key/CSP Generation

The module provides an SP800-90A-compliant Deterministic Random Bit Generator (DRBG) for the creation of key components of asymmetric keys, and random number generation.

The key generation methods implemented in the module for Approved services in FIPS mode is compliant with [SP800-133] (vendor affirmed).

For generating RSA, DSA and 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 [SP800-90A] DRBG.

The module generates cryptographic keys whose strengths are modified by available entropy.

6.3 Key Transport

The module provides the following key transport mechanisms:

- Key wrapping using AES-KW.
- RSA key encapsulation using private key encryption and public key decryption.

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

- AES key wrapping provides between 128 and 256 bits of encryption strength.
- RSA key wrapping[2] provides between 112 and 256 bits of encryption strength.

Note: As the module supports RSA key pairs greater than 2048 bits up to 15360 bits or more, the encryption strength 256 bits is claimed for RSA key encapsulation.

6.4 Key Derivation

The module supports password-based key derivation (PBKDF), as a vendor-affirmed security function. The implementation is compliant with option 1a of [SP-800-132]. Keys derived from passwords or passphrases using this method can only be used in storage applications.

6.5 Key/CSP Entry and Output

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

6.6 Key/CSP Storage

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 keys. The keys and CSPs are stored as plaintext in the RAM. The only exception is the HMAC key used for the Integrity Test, which is stored in the module and relies on the operating system for protection.

2 Key wrapping” is used instead of “key encapsulation” to show how the algorithm will appear in the certificate per IG G.13.

6.7 Key/CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The application is responsible for calling the appropriate zeroization functions provided in the module's API and listed in Table 10. The zeroization functions overwrite the memory occupied by keys with “zeros” and deallocate the memory with the regular memory deallocation operating system call.

7 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms as shown in Table 3 are compliant to 47 CFR FCC Part 15, Subpart B, Class A (Business use).

8 Self Tests

8.1 Power-Up Tests

The module performs power-up tests when the module is loaded into memory, without operator intervention. Power-up tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

While the module is executing the power-up tests, services are not available, and input and output are inhibited. The module is not available for use by the calling application until the power-up tests are completed successfully.

If any of the power-up test fails, the module enters the Error state. Subsequent calls to the module will also fail; no further cryptographic operations are possible. If the power-up tests complete successfully, the module will enter the Operational state and will accept cryptographic operation service requests.

In order to verify whether the self-tests have succeeded and the module is in the Operational state, the calling application may invoke the `gcry_control(GCRYCTL_OPERATIONAL_P)`. The function will return TRUE if the module is in the operational state, FALSE if the module is in the Error state.

8.1.1 Integrity Tests

The integrity of the module is verified by comparing an HMAC-SHA-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. If the HMAC values do not match, the test fails and the module enters the error state.

8.1.2 Cryptographic Algorithm Tests

The module performs self-tests on all FIPS-Approved cryptographic algorithms supported in the Approved mode of operation, using the Known Answer Tests (KAT) shown in the following table.

Algorithm	Power-Up Tests
AES	KAT AES ECB mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested).
CMAC	KAT AES CMAC with 128, 192 and 256 bit keys, MAC generation. KAT Triple-DES CMAC, MAC generation.
DRBG	KAT CTR_DRBG with AES with 128-bit key with DF, with and without PR. KAT Hash_DRBG with SHA-256 with and without PR. KAT Hash_DRBG with SHA-1 without PR. KAT HMAC_DRBG with SHA-256 with and without PR.
DSA	KAT DSA signature generation and verification with L=2048, N=256 and SHA-256 (separately tested).
ECDSA	KAT ECDSA signature generation and verification with P-256 and SHA-256 (separately tested).
HMAC	KAT HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512. KAT HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512.

Algorithm	Power-Up Tests
RSA	KAT RSA PKCS#1 v1.5 signature generation and verification with 2048-bit key and SHA-256 (separately tested). KAT RSA with 2048-bit key, public key encryption and private key decryption (separately tested).
SHS	KAT SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512.
Triple-DES	KAT Triple-DES ECB mode, encryption and decryption (separately tested).

Table 11: Self-Tests

For the KAT, the module calculates the result and compares it with the known value. If the answer does not match the known answer, the KAT fails and the module enters the Error state.

8.2 On-Demand Self-Tests

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. During the execution of the on-demand self-tests, services are not available and no data output or input is possible.

In order to verify whether the self-tests have succeeded and the module is in the Operational state, the calling application may invoke the `gcry_control(GCRYCTL_OPERATIONAL_P)`. The function will return TRUE if the module is in the operational state, FALSE if the module is in the Error state.

8.3 Conditional Tests

The module performs conditional tests on the cryptographic algorithms, using the Pair-wise Consistency Tests (PCT) shown in the following table. If the conditional test 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	Conditional Tests
DSA key generation	PCT using signature generation and verification with SHA-256.
ECDSA key generation	PCT using signature generation and verification with SHA-256.
RSA key generation	PCT using signature generation and verification with SHA-256. PCT using public encryption and private decryption.

Table 12: Conditional Tests

8.4 Error states

The Module enters the Error state on failure of power-on self-tests or conditional test, showing an error message related to the cause of failure and setting the error indicator. In order to verify whether the module is in the Operational state or the Error state, the calling application may invoke the `gcry_control(GCRYCTL_OPERATIONAL_P)`. The function will return TRUE if the module is in the operational state, FALSE if the module is in the Error state.

In the Error state, all data output is inhibited and no cryptographic operation is allowed. The error can be recovered by restart (i.e. powering off and powering on) of the module.

The module enters the Fatal Error state when random numbers are requested in the error state or when requesting cipher operations on a deallocated handle. In the Fatal Error state the module is aborted and is not available for use. The module needs to be reloaded in order to recover from this state.

9 Guidance

9.1 Crypto Officer Guidance

The binaries of the module are contained in the RPM packages for delivery. The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated as a FIPS 140-2 validated module.

The following RPM packages contain the FIPS validated module:

Processor Architecture	RPM Packages
Intel 64-bit	libgcrypt20-1.8.2-8.36.1.x86_64.rpm libgcrypt20-hmac-1.8.2-8.36.1.x86_64.rpm
IBM z15	libgcrypt20-1.8.2-8.36.1.s390x.rpm libgcrypt20-hmac-1.8.2-8.36.1.s390x.rpm
ARMv8 64-bit	libgcrypt20-1.8.2-8.36.1.aarch64.rpm libgcrypt20-hmac-1.8.2-8.36.1.aarch64.rpm

Table 13: RPM packages

9.1.1 Module Installation

The Crypto Officer can install the RPM packages containing the module as listed in Table 13 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.

9.1.2 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.

9.1.3 Operational Environment limitations

Instrumentation tools like the ptrace system call, gdb and strace utilities, 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.

9.2 User Guidance

In order to run in FIPS mode, the module must be operated using the FIPS Approved services, with their corresponding FIPS Approved and FIPS allowed cryptographic algorithms provided in this Security Policy (see section 3.2). In addition, key sizes must comply with [SP800-131A].

The user can find the documentation at the following location once the module is installed:

/usr/share/info/gcrypt.info.gz

9.2.1 Memory Management

The user shall only use the memory management functions provided by the libgcrypt API. Critical security parameters (e.g. keys) which are used as input or output parameters shall be managed using the `gcry_malloc_secure()`, `gcry_calloc_secure()` and `gcry_free()` functions.

The function `gcry_set_allocation_handler()` shall not be used; the user shall not change the libgcrypt memory handlers. The use of this API function implies that the cryptographic module is being executed in an invalid configuration.

9.2.2 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 A.9, 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.

9.2.3 Triple-DES encryption

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES blocks (2GB of data), in accordance to SP800-67 and IG A.13.

[SP800-67] imposes a restriction on the number of 64-bit block encryptions performed under the same three-key Triple-DES key.

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

- Transport Layer Security (TLS) versions 1.1 and 1.2, conformant with [RFC5246]
- Secure Shell (SSH) protocol, conformant with [RFC4253]
- Internet Key Exchange (IKE) versions 1 and 2, conformant with [RFC7296]

In any other scenario, the module cannot perform more than 2^{16} 64-bit data block encryptions.

The user is responsible for ensuring the module’s compliance with this requirement.

9.2.4 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 to [SP800-132] and IG D.6, 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.
- A portion of the salt, with a length of at least 128 bits, shall be generated randomly using the SP800-90A 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.
- 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, 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].

10 Mitigation of Other Attacks

10.1 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.

By default, 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*.

10.2 Weak Triple-DES Key Detection

There are 64 known Triple-DES keys which are weak because they produce only one, two, or four different subkeys in the subkey scheduling process. The module can detect these weak keys; the calling application shall invoke the `gcry_cipher_ctl()` function with the `PRIV_CIPHERCTL_DISABLE_WEAK_KEY` command (this feature is disabled by default).

Appendix A - CAVP certificates

The tables below show the certificates obtained from the CAVP for all the target platforms included in Table 3. The CAVP certificates validate all algorithm implementations used as approved or allowed security functions in FIPS mode of operation. The tables include the certificate number, the label used in the CAVP certificate for reference and a description of the algorithm implementation.

Cert#	CAVP Label	Algorithm Implementation	PAA
A482	Full Acceleration	Assembler block mode mode using Intel AES-NI AES and assembler SHA implementation.	Yes
		Generic C implementation for Triple-DES	No
A1152	No Acceleration	Generic C implementation	No
A484	AESNI_BMI2	Intel AES-NI and BMI SHA-1 implementation	Yes
A485	AESNI_AVX	Intel AES-NI and AVX SHA implementation	Yes
A486	SSSE3	SSSE3 SHA implementation	No
A487	SHLD	SHLD assembler SHA implementation	No

Table 14: CAVP certificates for the Intel Xeon processor

Cert#	CAVP Label	Algorithm Implementation	PAI
A482	Full Acceleration	Generic C implementation	No

Table 15: CAVP certificates for the z15 processor

Cert#	CAVP Label	Algorithm Implementation	PAA
A482	Full Acceleration	Assembler block mode using ARM Cryptographic Extension AES and SHA implementation.	Yes
		Generic C implementation for Triple-DES	No
A1152	No Acceleration	Generic C implementation	No
A488	NEON	ARM NEON SHA implementation	Yes

Table 16: CAVP certificates for the ARMv8 processor

Appendix B - Glossary and Abbreviations

AES	Advanced Encryption Specification
AES_NI	Intel® Advanced Encryption Standard (AES) New Instructions
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining Message Authentication Code
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PKCS	Public Key Cryptography Standards
RNG	Random Number Generator
RPM	Red hat Package Manager
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
TDES	Triple-DES
XTS	XEX Tweakable Block Cipher with Ciphertext Stealing

Appendix C - References

FIPS 140-2 **FIPS PUB 140-2 - Security Requirements for Cryptographic Modules**
https://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS 140-2_IG **Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program**
December 3, 2019
https://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

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

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

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

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

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

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

SP800-38A **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
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**
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**
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**
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**
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** https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-67	**NIST Special Publication 800-67 Revision 2 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-67r2.pdf
SP800-90A	**NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-131A	**NIST Special Publication 800-131A Revision 1- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf