Red Hat Enterprise Linux 5 Libgcrypt Cryptographic Module v1.0

FIPS 140-2 Security Policy

version 1.2

Last Update: 2010-04-14

Contents

Document History....3
1 Cryptographic Module Specification....4
1.1 Description of Module....4
1.2 Description of Approved Mode....5
1.3 Cryptographic Module Boundary....6
1.3.1 Hardware Block Diagram....6
1.3.2 Software Block Diagram....7
2 Cryptographic Module Ports and Interfaces....7
3 Roles, Services and Authentication....8
3.1 Roles....8
3.2 Services....8
3.2.1 Approved Services....8
3.2.2 Non-Approved Allowed Services....11
3.3 Operator Authentication....12
4 Physical Security....12
5 Operational Environment....12
5.1 Applicability....12
5.2 Policy....12
6 Cryptographic Key Management....12
6.1 Random Number Generation....13
6.2 Electromagnetic Interference/Electromagnetic Compatibility....13
7 Self Tests....13
7.1 Power-Up Tests....13
7.1.1 Cryptographic Function....14
7.2 Conditional Tests....16
8 Guidance....17
8.1 Cryptographic Officer Guidance....17
8.2 User Guidance....18
9 Mitigation of Other Attacks....18
10 Glossary and Abbreviations....20
11 References....21

Document History

Version	Date of Change	Author	Changes to Previous Version
0.1	05/27/08	SHW - atsec	Initial
0.2	11/18/09	SHW	First full version
1.0	10/06/09	SHW	Released version
1.1	04/01/10	SHW	Updates/corrections
1.2	04/14/10	SHW	Updates/corrections

1 Cryptographic Module Specification

1.1 Description of Module

The libgcrypt module is a software only, security level 1 cryptographic module, running on a multi-chip standalone platform. The module supplies general cryptographic support for the Red Hat Enterprise Linux user space. The following table shows the overview of the security level for each of the eleven sections of validation. All components of the module will be in the libgcrypt RPM version 1.4.4-5.el5. For Documentation, please see [1].

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

Table 1, Security Level

For a complete detailed description of what libgcrypt does, how to use it, and the implications of using the FIPS 140-2 approved mode, please see the user documentation [1] that comes with the software.

The module has been tested on the following configurations:

- 32 bit x86
- 64 bit x86_64
- 64 bit Itanium

The module has been tested on the following multi-chip standalone platforms:

Manufacturer	Model	O/S & Ver.
HP	HP Integrity Server RX2660	Red Hat Enterprise Linux 5.4 (In Single User Mode)
HP	HP ProLiant Server DL585 (library in 64 bit word size and 32 bit word size)	Red Hat Enterprise Linux 5.4 (In Single User Mode)

Table 2, Tested Platforms

1.2 Description of Approved Mode

If the file `/proc/sys/crypto/fips_enabled' exists and contains a numeric of '1', libgcrypt is put into FIPS mode at initialization time.

In Approved mode, the module will support the following Approved Cryptographic functions:

- Triple-DES, encryption/decryption (ECB, CBC, OFB, CFB, CTR)
- AES 128/192/256 bits, encryption/decryption (ECB, CBC, OFB, CFB, CTR)
- RSA key generation, signature generation (PKCS #1.5), signature verification (PKCS #1.5)
- SHA 1/224/256/384/512
- DSA (PQG, Sig gen, Sig Ver)
- HMAC-SHA-1/224/256/384/512
- Random Number Generation (ANSI X9.31)

The module will support the following Allowed, but Non-Approved Cryptographic functions:

- MD5 (for TLS only)

The module will support the following Non-Approved Cryptographic functions:

- RSA (encrypt, decrypt)

1.3 Cryptographic Module Boundary

The physical module boundary is the surface of the case of the test platform. The logical module boundary is depicted in the software block diagram.

1.3.1 Hardware Block Diagram

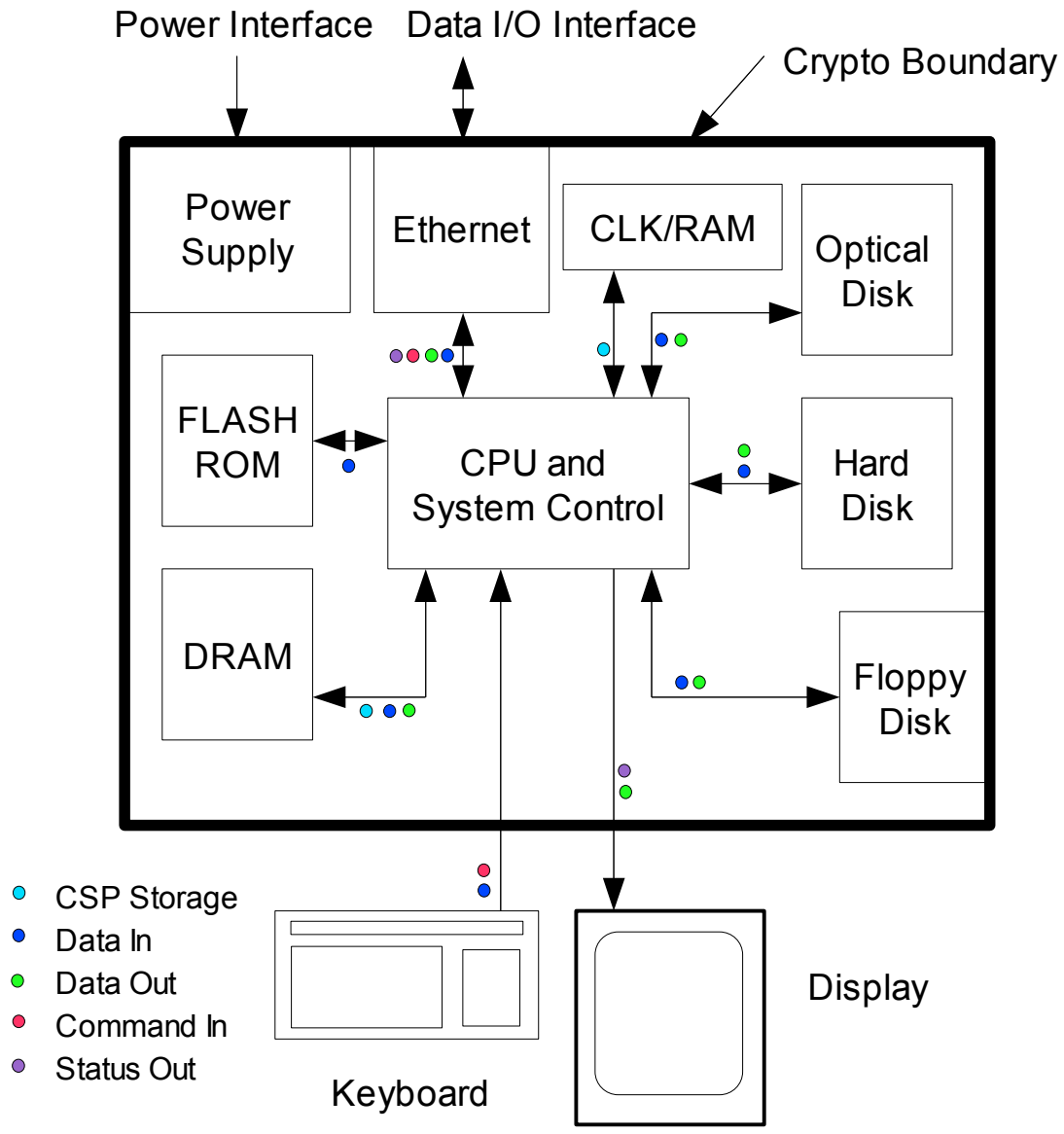

Figure 1. Hardware Block Diagram

1.3.2 Software Block Diagram

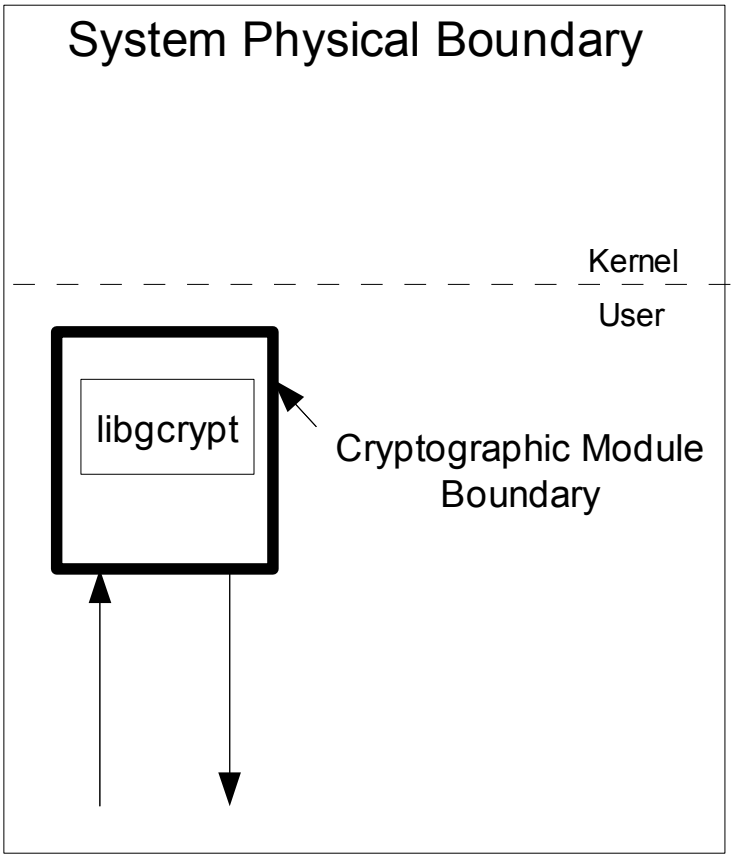

Figure 2. Software Block Diagram

2 Cryptographic Module Ports and Interfaces

Function	Port	Comments
Command in	API, disk	Configuration comes from disk files
Status Out	API	
Data IN	API	
Data Out	API	

Table 3. Ports and Interfaces

3 Roles, Services and Authentication

This section defines the roles, services and authentication mechanisms and methods with respect to the applicable FIPS 140-2 requirements.

3.1 Roles

Role	Services (see list below)
User	Encryption, Decryption (symmetric and public/private), Random Numbers
Crypto Officer	Encryption, Decryption (symmetric and public/private), Random Numbers, Configuration of Approved Mode

Table 4. Roles

3.2 Services

This section defines the Approved services with respect to the applicable FIPS 140-2 requirements.

3.2.1 Approved Services

Role	Service	CSP	Algo/Mode(s)	API call	Access
User, Crypto Officer	Triple-DES ENC/DEC (Cert #851, 859, 860)	168 bit Triple-DES Key	Triple-DES (ECB,CBC,CFB64, OFB,CTR)	GCRY_CIPHER_3DES	R, W, EX
User, Crypto Officer	AES128 ENC/DEC (Cert #1180, 1192, 1193)	128 bit AES Key	AES (ECB,CBC,CFB128,CTR, OFB)	GCRY_CIPHER_AES128	R, W, EX
User, Crypto Officer	AES192 ENC/DEC (Cert #1180, 1192, 1193)	192 bit AES Key	AES (ECB,CBC,CFB128,CTR, OFB)	GCRY_CIPHER_AES192	R, W, EX
User, Crypto Officer	AES256 ENC/DEC (Cert #1180, 1192, 1193)	256 bit AES Key	AES (ECB,CBC,CFB128,CTR, OFB)	GCRY_CIPHER_AES256	R, W, EX
User, Crypto Officer	Get Key Length	none	All symmetric	GCRYCTL_GET_KEYLEN	R
User, Crypto Officer	Get Block Length	none	All symmetric	GCRYCTL_GET_BLKLEN	R
User, Crypto Officer	Check availability of Algorithm	none	All symmetric	GCRYCTL_TEST_ALGO	R
User, Crypto Officer	SHA-1 (Cert #1089, 1098, 1099)	none	N/A	GCRY_MD_SHA1	R, W, EX
User, Crypto Officer	SHA-224 (Cert #1089, 1098, 1099)	none	N/A	GCRY_MD_SHA224	R, W, EX
User, Crypto Officer	SHA-256 (Cert #1089, 1098, 1099)	none	N/A	GCRY_MD_SHA256	R, W, EX
User, Crypto Officer	SHA-384 (Cert #1089, 1098, 1099)	none	N/A	GCRY_MD_SHA384	R, W, EX
User, Crypto Officer	SHA-512 (Cert #1089, 1098, 1099)	none	N/A	GCRY_MD_SHA512	R, W, EX
User, Crypto Officer	HMAC-SHA-1 (Cert #680, 691, 692)	MAC-key	N/A	GCRY_MD_SHA1,GCRY_MD_FLAG_HMAC	R, W, EX
User, Crypto Officer	HMAC-SHA-224 (Cert #680, 691, 692)	MAC-key	N/A	GCRY_MD_SHA224,GCRY_MD_FLAG_HMAC	R, W, EX
User, Crypto	HMAC-SHA-256	MAC-key	N/A	GCRY_MD_SHA	R. W. EX

Officer	(Cert #680, 691, 692)			256,GCRY_MD_FLAG_HMAC	
User, Crypto Officer	HMAC-SHA-384 (Cert #680, 691, 692)	MAC-key	N/A	GCRY_MD_SHA384,GCRY_MD_FLAG_HMAC	R, W, EX
User, Crypto Officer	HMAC-SHA-512 (Cert #680, 691, 692)	MAC-key	N/A	GCRY_MD_SHA512,GCRY_MD_FLAG_HMAC	R, W, EX
User, Crypto Officer	DSA (verify,sign & pqg) (Cert #389, 393, 394)	Key Length1024 bits	DSA	GCRY PK DSA	R, W, EX
User, Crypto Officer	Fill buffer with length random bytes (Cert #651, 658, 659)	Seed, Seed Key	RNG	GCRY_RANDOMIZE	W, EX
User, Crypto Officer	Convenience function to allocate a memory block consisting of nbytes of random bytes Cert #651, 658, 659)	Seed, Seed Key	RNG	GCRY_RANDOM_BYTES	W, EX
User, Crypto Officer	Convenience function to allocate a memory block consisting of nbytes fresh random bytes using a random quality as defined by level. This function differs from gcry_random_bytes in that the returned buffer is allocated in a “secure" area of the memory. Cert #651, 658, 659)	Seed, Seed Key	RNG	GCRY_RANDOM_BYTES_SECURE	W, EX
User, Crypto Officer	Fill buffer with random bytes.Cert #651,	Seed, Seed Key	RNG	GCRY_CREATE_NONCE	R, EX

	658, 659)				
User, Crypto Officer	Initialize Module	N/A	N/A	gcry_check_version	EX
User, Crypto Officer	Self Test	N/A	All	GCRYCTL_SELFTEST	EX
User, Crypto Officer	Zeroize secure memory	All	N/A	GCRYCTL_TERM_SECMEM	W,EX
User, Crypto Officer	Release all resources of context created by gcry_cipher_open zeroises all sensitive information associated with this cipher handle	Context based	All cryptographic functions	GCRY_CIPHER_CLOSE	R,W,EX
User, Crypto Officer	Release all resources of hash context	Context based	All hash functions	GCRY_MD_CLOSE	R,W,EX
User, Crypto Officer	Release the S-expression object SEXP (RSA/DSA keys)	Context based	SEXP	GCRY_SEXP_RELEASE	R,W,EX
User, Crypto Officer	Get Status	N/A	N/A	GCRYCTL_SET_VERBOSITY	R,EX

Table 5. Approved Services

This are the basic crypto functions. Please see libgcrypt user manual for the complete list, including additional non-crypto utility functions.

3.2.2 Non-Approved Allowed Services

Role	Service	CSP	Algo/Mode(s)	API call	Access
User, Crypto Officer	RSA (encrypt, deccrypt, verify, sign & keygen) (Cert #561, 570, 571)	Variable Length Key 1024 – 4096 bits	RSA	GCRY PK RSA	R, W, EX
User, Crypto Officer	MD5 (for TLS only)	N/A	N/A	GCRY PK MD5	R, W, EX

Table 6. Non-Approved Allowed Services

3.3 Operator Authentication

There is no operator authentication. The assumption of a role is implicit by the action taken.

4 Physical Security

This module is a security level 1 software module and offers no physical security.

5 Operational Environment

5.1 Applicability

This module will operate in a modifiable operational environment per the FIPS 140-2 specifications.

5.2 Policy

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

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

In the FIPS Approved mode, the ptrace(2) system call, debugger(gdb(1)), and strace(1)) shall not be used.

6 Cryptographic Key Management

Libgcrypt considers all keys to be ephemeral. They are received for use or generated by the module only at the command of the calling application.

Libgcrypt does not perform key management. However it supports key management by the calling application.

Keys which are passed to libgcrypt should be allocated in secure memory as being available with the functions `gcry_malloc_secure' and `gcry_calloc_secure.'

At the end of use, calling `gcry_free' on this memory, the memory (and thus the keys) are overwritten with zeros before releasing the memory back to the operating system.

For use with the random number generator, libgcrypt generates three internal keys which are stored in the encryption contexts used by the RNG. These keys are stored in secure memory for the lifetime of the process. Applications are required to use 'GCRYCTL_TERM_SECMEM' before process termination. This will zero out the entire secure memory and thus also the encryption contexts with these keys. This call should be added to the signal handlers of all signals leading to a termination of the process.

6.1 Random Number Generation

A FIPS 140-2, ANSI X9.31-approved pseudo random number generation mechanism using AES 128 will be used in the module.

/dev/random will be used to seed the pseudo random number generator.

6.2 Electromagnetic Interference/Electromagnetic Compatibility

Product Name and Model: HP Proliant Server DL585 Series
Regulatory Model Number: 1) HSTNS-1025
Product Options: All
conforms to the following Product Specifications and Regulations:
EMC: Class A
CISPR 22:2005
EN 55022:2006
EN 55024:1998 +A1:2001 +A2:2003
EN 61000-3-2:2006
EN 61000-3-3:1995 +A1:2001 +A2:2005

Product Name and Model: HP Integrity Server rx2660
Regulatory Model Number: 1) RSVLA-0503
Product Options: All
conforms to the following Product Specifications and Regulations :
EMC: Class A
CISPR22:1997 / EN 55022:1998
CISPR 24:1997 + A1:2001 + A2: 2002 / EN 55024:1998 + A1:2001 + A2:2003
EN 61000-3-2:2000
EN 61000-3-3:1995 +A1:2001

7 Self Tests

FIPS 140-2 requires that the module perform self tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition, some functions require continuous verification of function, such as the random number generator. All of these tests are listed and described in this section.

7.1 Power-Up Tests

The following tests are performed each time the module starts and must be completed successfully for the module to operate.

7.1.1 Cryptographic Function

Algorithm	Modes	Comments
Triple-DES	2 & 3 key	KAT (Known Answer Test)
AES 128	ECB	KAT
AES 192	ECB	KAT
AES 256	ECB	KAT
SHA-1	N/A	KAT
SHA-224	N/A	KAT
SHA-256	N/A	KAT
SHA-384	N/A	KAT
SHA-512	N/A	KAT
HMAC SHA-1	N/A	KAT
HMAC SHA-224	N/A	KAT
HMAC SHA-256	N/A	KAT
HMAC SHA-384	N/A	KAT
HMAC SHA-512	N/A	KAT
RNG	N/A	KAT
RSA	N/A	A pre-defined 1024-bit RSA key is used and these tests are run in turn: 1. Conversion of S-expression to internal format. (cipher/rsa.c:selftests_rsa) 2. Private key consistency check. (cipher/rsa.c:selftests_rsa) 3. A pre-defined 20 byte value is signed with PKCS#1 padding for SHA-1. The result is verified using the public key against the original data and against modified data. (cipher/rsa.c:selftest_sign_1024) 4. A 1000-bit random value is encrypted and checked that it does not match the original random value. The encrypted result is then decrypted and checked that it matches the original random value. (cipher/rsa.c:selftest_encr_1024)
DSA		A pre-defined 1024-bit DSA key is used and these tests are run in turn: 1. Conversion of S-expression to internal format. (cipher/dsa.c: selftests_dsa) 2. Private key consistency check. (cipher/dsa.c:selftests_dsa) 3. A pre-defined 20 byte value is signed with PKCS#1 padding for SHA-1. 4.The result is verified using the public key against the original data and against the modified data.

		(cipher/dsa.c:selftest_sign_1024)
Module Integrity test		The integrity of the libgcrypt module is tested during power-up. The check works by computing a HMAC SHA-256 checksum over the file used to load libgcrypt into memory. That checksum is compared against a checksum stored in a file of the same name but with a single dot as a prefix and a suffix of `.hmac'.

Table 7. Self Tests

7.2 Conditional Tests

Algorithm	Comments
RNG	The continuous random number test is only used in FIPS mode. The RNG generates blocks of 128-bit size; the rst block generated per context is saved in the context and another block is generated to be returned to the caller. Each block is compared against the saved block and then stored in the context. If a duplicated block is detected, an error is signaled and the library is put into the “\Fatal-Error" state. (random/random-fips.c:x931_aes_driver)
RSA	The test uses a random number 64 bits less the size of the modulus as plaintext and runs an encryption and decryption operation in turn. The encrypted value is checked to not match the plaintext, and the result of the decryption is checked to match the plaintext. A new random number of the same size is generated, signed, and verified to test the correctness of the signing operation. As a second signing test, the signature is modified by incrementing its value and then verified with the expected result that the verification fails. (cipher/rsa.c:test_keys)
DSA	The test uses a random number of the size of the Q parameter to create a signature and then checks that the signature verifies. As a second signing test, the data is modified by incrementing its value and then is verified against the signature with the expected result that the verification fails. (cipher/dsa.c:test_keys)

Table 8. Conditional Self Tests

8 Guidance

The following guidance items are to be used for assistance in maintaining the module's validated status while in use.

8.1 Cryptographic Officer Guidance

The RPM package of the module can be installed by standard tools recommended for the installation of RPM packages on a Red Hat Enterprise Linux system (for example, yum, rpm, and the RHN remote management tool). For proper operation of the in-module integrity verification, the prelink has to be disabled. This can be done by setting PRELINKING=no in the /etc/sysconfig/prelink configuration file.

To bring the module into FIPS mode, regenerate initrd by performing the following:

For the x86_64 platform, the command is:

```
mkinitrd --with-fips -f /boot/initrd-$(uname -r).img $(uname -r)
```

For the IA64, the command is:

```
mkinitrd --with-fips -f /boot/efi/efi/redhat/initrd-$(uname -r).img $(uname -r)
```

Then modify the kernel command line of the current kernel in the boot loader, by appending the following string:

```
fips=1
```

Because FIPS 140-2 has certain restrictions on the use of cryptography which are not always wanted, libgcrypt

needs to be put into FIPS mode explicitly. Three alternative mechanisms are provided to switch libgcrypt into this mode:

- If the file /proc/sys/crypto/fips_enabled exists and contains a numeric value other than 0, libgcrypt is put into FIPS mode at initialization time.
- If the file /etc/gcrypt/fips_enabled exists, libgcrypt is put into FIPS mode at initialization time. Note that this filename is hardwired and does not depend on any configuration options. If a non-zero value is stored in that file, libgcrypt is put into Enforced FIPS mode to help detection of applications which don't fulfill all requirements for using libgcrypt in FIPS mode.
- If the application requests FIPS mode using the control command 'GCRYCTL_FORCE_FIPS_MODE.' This must be done prior to any initialization (i.e. before 'gcry_check_version').
- In addition to the standard FIPS mode, libgcrypt may also be put into an Enforced FIPS mode by writing a non-zero value into the file /etc/gcrypt/fips_enabled. The Enforced FIPS mode helps to detect applications which don't fulfill all requirements for using libgcrypt in FIPS mode.

Once libgcrypt has been put into FIPS mode, it is not possible to switch back to standard mode without terminating the process first. If the logging verbosity level of libgcrypt has been set to at least 2, the state transitions and the self tests are logged.

The version of the RPM containing the validated module is stated in section 1 above. The integrity of the RPM is automatically verified during the installation and the Crypto officer shall not install the RPM file if the RPM tool indicates an integrity error.

8.2 User Guidance

Applications using libgcrypt need to call “gcry_control(GCRYCTL_TERM_SECMEM)” before the process is terminated. A signal handler may be used to achieve this.

The function gcry set_allocation_handler may not be used.

The user must not call malloc/free to create/release space for keys, let libgcrypt manage space for keys, which will ensure that the key memory is overwritten before it is released.

See user guide [1] for complete instructions for use.

9 Mitigation of Other Attacks

Libgcrypt uses a blinding technique for RSA decryption to mitigate real world timing attacks over a network: 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” and generated with `GCRY_WEAK_RANDOM' random level.

Weak Triple-DES keys are detected as follows:

In DES there are 64 known keys which are weak because they produce only one, two, or four different subkeys in the subkey scheduling process. The keys in this table have all their parity bits cleared.

```
static byte weak_keys[64][8] =
{
  { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }, /*w*/
  { 0x00, 0x00, 0x1e, 0x1e, 0x00, 0x00, 0x0e, 0x0e },
  { 0x00, 0x00, 0xe0, 0xe0, 0x00, 0x00, 0xf0, 0xf0 },
```

```
{ 0x00, 0x00, 0xfe, 0xfe, 0x00, 0x00, 0xfe, 0xfe },
{ 0x00, 0x1e, 0x00, 0x1e, 0x00, 0x0e, 0x00, 0x0e }, /*sw*/
{ 0x00, 0x1e, 0x1e, 0x00, 0x00, 0x0e, 0x0e, 0x00 },
{ 0x00, 0x1e, 0xe0, 0xfe, 0x00, 0x0e, 0xf0, 0xfe },
{ 0x00, 0x1e, 0xfe, 0xe0, 0x00, 0x0e, 0xfe, 0xf0 },
{ 0x00, 0xe0, 0x00, 0xe0, 0x00, 0xf0, 0x00, 0xf0 }, /*sw*/
{ 0x00, 0xe0, 0x1e, 0xfe, 0x00, 0xf0, 0x0e, 0xfe },
{ 0x00, 0xe0, 0xe0, 0x00, 0x00, 0xf0, 0xf0, 0x00 },
{ 0x00, 0xe0, 0xfe, 0x1e, 0x00, 0xf0, 0xfe, 0x0e },
{ 0x00, 0xfe, 0x00, 0xfe, 0x00, 0xfe, 0x00, 0xfe }, /*sw*/
{ 0x00, 0xfe, 0x1e, 0xe0, 0x00, 0xfe, 0x0e, 0xf0 },
{ 0x00, 0xfe, 0xe0, 0x1e, 0x00, 0xfe, 0xf0, 0x0e },
{ 0x00, 0xfe, 0xfe, 0x00, 0x00, 0xfe, 0xfe, 0x00 },
{ 0x1e, 0x00, 0x00, 0x1e, 0x0e, 0x00, 0x00, 0x0e },
{ 0x1e, 0x00, 0x1e, 0x00, 0x0e, 0x00, 0x0e, 0x00 }, /*sw*/
{ 0x1e, 0x00, 0xe0, 0xfe, 0x0e, 0x00, 0xf0, 0xfe },
{ 0x1e, 0x00, 0xfe, 0xe0, 0x0e, 0x00, 0xfe, 0xf0 },
{ 0x1e, 0x1e, 0x00, 0x00, 0x0e, 0x0e, 0x00, 0x00 },
{ 0x1e, 0x1e, 0x1e, 0x1e, 0x0e, 0x0e, 0x0e, 0x0e }, /*w*/
{ 0x1e, 0x1e, 0xe0, 0xe0, 0x0e, 0x0e, 0xf0, 0xf0 },
{ 0x1e, 0x1e, 0xfe, 0xfe, 0x0e, 0x0e, 0xfe, 0xfe },
{ 0x1e, 0xe0, 0x00, 0xfe, 0x0e, 0xf0, 0x00, 0xfe },
{ 0x1e, 0xe0, 0x1e, 0xe0, 0x0e, 0xf0, 0x0e, 0xf0 }, /*sw*/
{ 0x1e, 0xe0, 0xe0, 0x1e, 0x0e, 0xf0, 0xf0, 0x0e },
{ 0x1e, 0xe0, 0xfe, 0x00, 0x0e, 0xf0, 0xfe, 0x00 },
{ 0x1e, 0xfe, 0x00, 0xe0, 0x0e, 0xfe, 0x00, 0xf0 },
{ 0x1e, 0xfe, 0x1e, 0xfe, 0x0e, 0xfe, 0x0e, 0xfe }, /*sw*/
{ 0x1e, 0xfe, 0xe0, 0x00, 0x0e, 0xfe, 0xf0, 0x00 },
{ 0x1e, 0xfe, 0xfe, 0x1e, 0x0e, 0xfe, 0xfe, 0x0e },
{ 0xe0, 0x00, 0x00, 0xe0, 0xf0, 0x00, 0x00, 0xf0 },
{ 0xe0, 0x00, 0x1e, 0xfe, 0xf0, 0x00, 0x0e, 0xfe },
{ 0xe0, 0x00, 0xe0, 0x00, 0xf0, 0x00, 0xf0, 0x00 }, /*sw*/
{ 0xe0, 0x00, 0xfe, 0x1e, 0xf0, 0x00, 0xfe, 0x0e },
{ 0xe0, 0x1e, 0x00, 0xfe, 0xf0, 0x0e, 0x00, 0xfe },
{ 0xe0, 0x1e, 0x1e, 0xe0, 0xf0, 0x0e, 0x0e, 0xf0 },
{ 0xe0, 0x1e, 0xe0, 0x1e, 0xf0, 0x0e, 0xf0, 0x0e }, /*sw*/
{ 0xe0, 0x1e, 0xfe, 0x00, 0xf0, 0x0e, 0xfe, 0x00 },
{ 0xe0, 0xe0, 0x00, 0x00, 0xf0, 0xf0, 0x00, 0x00 },
{ 0xe0, 0xe0, 0x1e, 0x1e, 0xf0, 0xf0, 0x0e, 0x0e },
{ 0xe0, 0xe0, 0xe0, 0xe0, 0xf0, 0xf0, 0xf0, 0xf0 }, /*w*/
{ 0xe0, 0xe0, 0xfe, 0xfe, 0xf0, 0xf0, 0xfe, 0xfe },
{ 0xe0, 0xfe, 0x00, 0x1e, 0xf0, 0xfe, 0x00, 0x0e },
{ 0xe0, 0xfe, 0x1e, 0x00, 0xf0, 0xfe, 0x0e, 0x00 },
{ 0xe0, 0xfe, 0xe0, 0xfe, 0xf0, 0xfe, 0xf0, 0xfe }, /*sw*/
{ 0xe0, 0xfe, 0xfe, 0xe0, 0xf0, 0xfe, 0xfe, 0xf0 },
{ 0xfe, 0x00, 0x00, 0xfe, 0xfe, 0x00, 0x00, 0xfe },
{ 0xfe, 0x00, 0x1e, 0xe0, 0xfe, 0x00, 0x0e, 0xf0 },
{ 0xfe, 0x00, 0xe0, 0x1e, 0xfe, 0x00, 0xf0, 0x0e },
{ 0xfe, 0x00, 0xfe, 0x00, 0xfe, 0x00, 0xfe, 0x00 }, /*sw*/
{ 0xfe, 0x1e, 0x00, 0xe0, 0xfe, 0x0e, 0x00, 0xf0 },
{ 0xfe, 0x1e, 0x1e, 0xfe, 0xfe, 0x0e, 0x0e, 0xfe },
{ 0xfe, 0x1e, 0xe0, 0x00, 0xfe, 0x0e, 0xf0, 0x00 },
{ 0xfe, 0x1e, 0xfe, 0x1e, 0xfe, 0x0e, 0xfe, 0x0e }, /*sw*/
```

```
  { 0xfe, 0xe0, 0x00, 0x1e, 0xfe, 0xf0, 0x00, 0x0e },
  { 0xfe, 0xe0, 0x1e, 0x00, 0xfe, 0xf0, 0x0e, 0x00 },
  { 0xfe, 0xe0, 0xe0, 0xfe, 0xfe, 0xf0, 0xf0, 0xfe },
  { 0xfe, 0xe0, 0xfe, 0xe0, 0xfe, 0xf0, 0xfe, 0xf0 }, /*sw*/
  { 0xfe, 0xfe, 0x00, 0x00, 0xfe, 0xfe, 0x00, 0x00 },
  { 0xfe, 0xfe, 0x1e, 0x1e, 0xfe, 0xfe, 0x0e, 0x0e },
  { 0xfe, 0xfe, 0xe0, 0xe0, 0xfe, 0xfe, 0xf0, 0xf0 },
  { 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe }  /*w*/
};
```

10 Glossary and Abbreviations

3DES	Another name for TDES or Triple-DES (the Data Encryption Standard)
AES	Advanced Encryption Specification
CAVP	Cryptographic Algorithm Validation Program
CBC	Cypher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cypher Feedback
CC	Common Criteria
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CVT	Component Verification Testing
CTR	Counter Mode for block encryption
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
FSM	Finite State Model
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
LDAP	Lightweight Directory Application Protocol
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OFB	Output Feedback

O/S	Operating System
PP	Protection Profile
RNG	Randome Number Generator
RSA	Rivest, Shamir, Addleman
SAP	Service Access Points
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SOF	Strength of Function
SSH	Secure Shell
SVT	Scenario Verification Testing
TDES	Triple-DES
TOE	Target of Evaluation
UI	User Interface

Table 7. Glossary and Abbreviations

11 References

[1] libgcrypt user guide (provided with libgcrypt-devel RPM, libgcrypt-devel-1.4.4-5.el5.x86_64.rpm) the documentation iuncludes:

```
/usr/bin/libgcrypt-config
/usr/include/gcrypt-module.h
/usr/include/gcrypt.h
/usr/lib64/libgcrypt.a
/usr/lib64/libgcrypt.so
/usr/share/aclocal/libgcrypt.m4
/usr/share/info/gcrypt.info.gz (The user guide)
```

[2] rx2660_EMIEMC_cert.pdf (On file at Red Hat)

[3] DL585_EMIEMC_CEcert.pdf (On file at Red Hat)

[4] FIPS 140-2 Standard, http://csrc.nist.gov/groups/STM/cmvp/standards.html

[5] FIPS 140-2 Implementation Guidance, http://csrc.nist.gov/groups/STM/cmvp/standards.html

[6] FIPS 140-2 Derived Test Requirements,http://csrc.nist.gov/groups/STM/cmvp/standards.html

[7] FIPS 197 Advanced Encryption Standard, http://csrc.nist.gov/publications/PubsFIPS.html

[8] FIPS 180-3 Secure Hash Standard, http://csrc.nist.gov/publications/PubsFIPS.html

[9] FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC), http://csrc.nist.gov/publications/PubsFIPS.html

[10] FIPS 186-3 Digital Signature Standard (DSS), http://csrc.nist.gov/publications/PubsFIPS.html

[11] ANSI X9.52:1998 Triple Data Encryption Algorithm Modes of Operation, http://webstore.ansi.org/FindStandards.aspx?Action=displaydept&DeptID=80&Acro=X9&DpName=X9,%20Inc.