Red Hat Enterprise Linux 6.2 Libgcrypt Cryptographic Module v2.0

FIPS 140-2 Security Policy

version 1.1

Last Update: 2012-07-19

Contents

1 Cryptographic Module Specification 3
  1.1 Description of Module 3
  1.2 Description of Approved Mode....4
  1.3 Cryptographic Module Boundary....4
    1.3.1 Hardware Block Diagram....4
    1.3.2 Software Block Diagram....6
  1.4 Red Hat Enterprise Linux 6.2 Cryptographic Modules and FIPS 140-2 Certification....6
    1.4.1 Platforms....7
    1.4.2 FIPS Approved Mode....7
2 Cryptographic Module Ports and Interfaces 8
3 Roles, Services, and Authentication 9
  3.1 Roles....9
  3.2 Services....9
    3.2.1 Approved Services....9
    3.2.2 Non-Approved Allowed Services....12
  3.3 Operator Authentication....13
  3.4 Mechanism and Strength of Authentication....13
4 Physical Security....14
5 Operational Environment 15
  5.1 Applicability 15
  5.2 Policy 15
6 Cryptographic Key Management....16
  6.1 Random Number Generation 16
7 Electromagnetic Interference/Electromagnetic Compatibility 17
8 Self-Tests 18
  8.1 Power-Up Tests....18
    8.1.1 Cryptographic Function....18
  8.2 Conditional Tests....19
9 Guidance....21
  9.1 Cryptographic Officer Guidance....21
  9.2 User (Application Developer) Guidance....22
  9.3 Handling Self-Test Errors....22
10 Mitigation of Other Attacks....24
11 Glossary and Abbreviations....26
12 References....28
Appendix A: Code for Invoking the libgcrypt Module....29

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 1.4.5-9.el6_2.2.

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 that comes with the software. The user documentation is provided with the “gcrypt” info page installed with the libgcrypt-devel RPM package.

The module has been tested on a 64 bit and a 32 bit configuration.

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

Manufacturer	Model	O/S & Ver.
HP	Proliant DL585	Red Hat Enterprise Linux 6.2 (In Single User Mode)
IBM	HS22	Red Hat Enterprise Linux 6.2 (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 approved mode at initialization time.

If the file '/etc/gcrypt/fips_enabled' exists, libgcrypt is put into FIPS approved mode at initialization time.

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

- Triple-DES, encryption/decryption (ECB, CBC, OFB, CFB64, CTR)
- AES 128/192/256 bits, encryption/decryption (ECB, CBC, OFB, CFB128, 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)
- RSA (encrypt, decrypt, key wrapping)

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

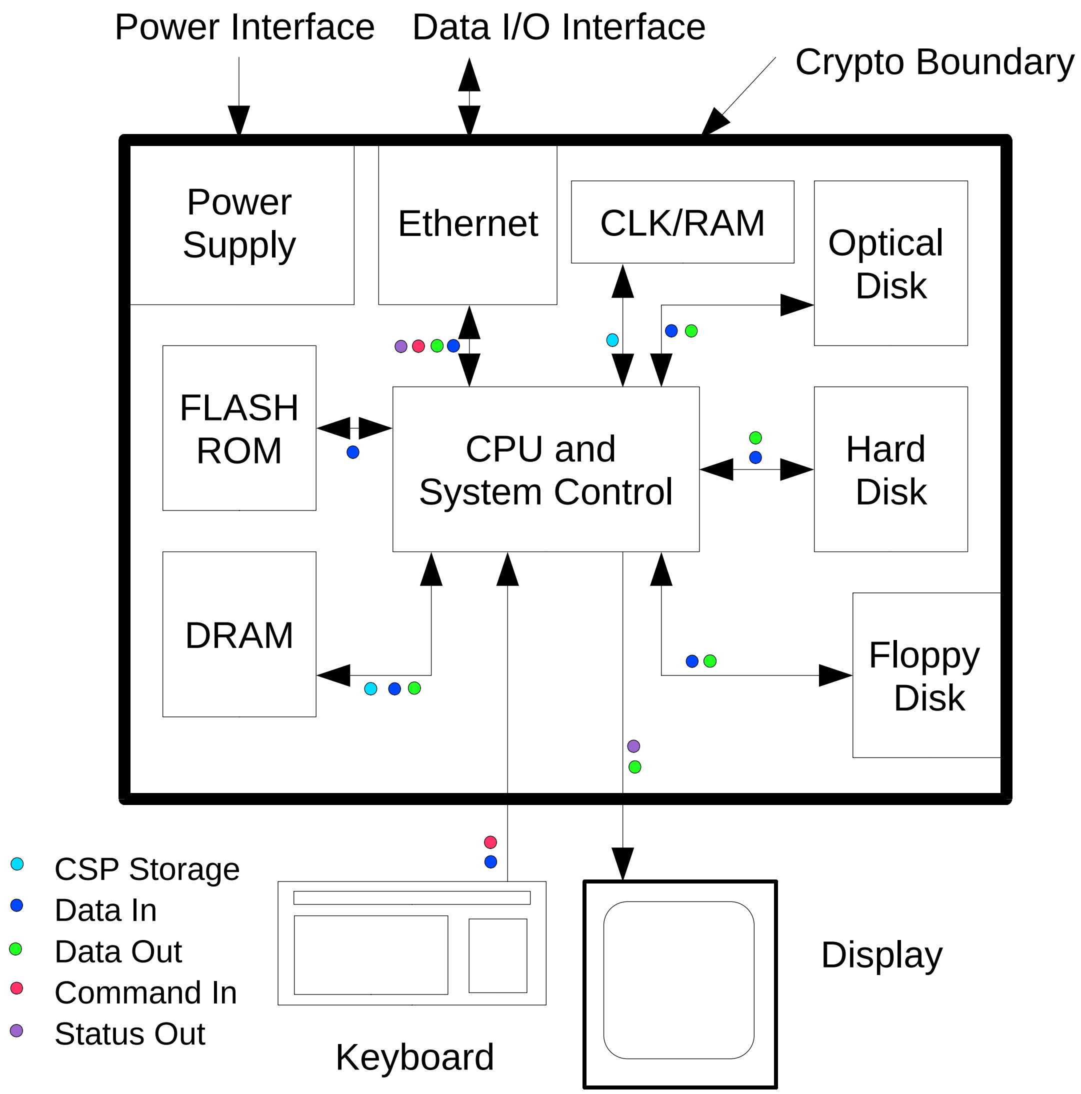

Figure 1. Hardware Block Diagram

1.3.2 Software Block Diagram

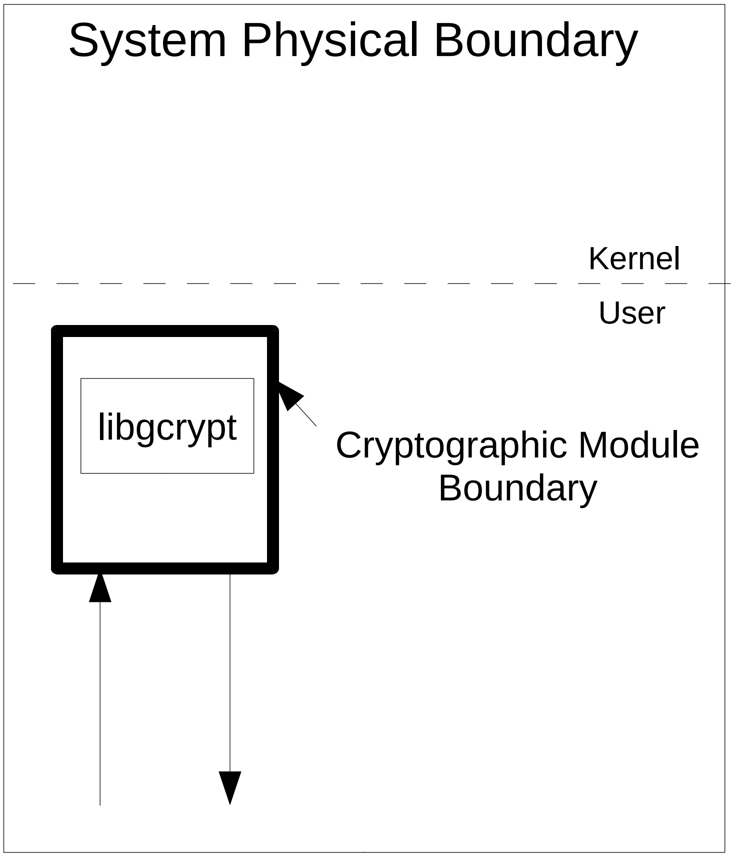

Figure 2. Software Block Diagram

1.4 Red Hat Enterprise Linux 6.2 Cryptographic Modules and FIPS 140-2 Certification

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

The certification is performed at FIPS 140-2 level 1, a software only certification that does not make any claims about the hardware enclosure. This allows vendors to develop their own higher level FIPS-certified modules using cryptographic modules.

The following cryptographic modules are included in the RHEL6.2 certification, although are validated as independent modules:

- Kernel Crypto API - a software only cryptographic module that provides general-purpose cryptographic services to the remainder of the Linux kernel
- Disk Volume Encryption - provides disk management and transparent partial or full disk encryption. Partial disk encryption encrypts only one or more partitions, leaving at least one partition as plaintext. Full disk encryption, on the other hand, encrypts the entire disk with the exception of the partitions of either /boot or /boot/efi.
- Libgcrypt - supplies general cryptographic support for the Red Hat Enterprise Linux user space

- OpenSSL - a software library supporting FIPS 140-2 approved cryptographic algorithms for general use by vendors
- OpenSSH-Server - supplies cryptographic support for the SSH protocol
- OpenSSH-Client - supplies cryptographic support for the SSH protocol
- Openswan - provides the IKE protocol version 1 key agreement services required for IPSec

1.4.1 Platforms

The certification was performed on a 64-bit system, which are capable of executing both 32 and 64-bit code concurrently. Vendors can "transfer" the FIPS 140-2 certificate to the other similar platforms, provided the binary is only recompiled without changing the code. This is called a vendor assertion.

1.4.2 FIPS Approved Mode

Any vendor-provided FIPS 140-2 certified cryptographic modules and services that use the RHEL6.2 underlying services to provide cryptographic functionality must use the RHEL6.2 services in their approved mode. The approved mode ensures that FIPS 140-2 required self-tests are executed and that ciphers are restricted to those that have been FIPS 140-2 certified by independent testing.

The kernel services (Kernel Crypto API) must be in FIPS 140-2 approved mode as many services rely on these underlying services for their cryptographic capabilities. See the Kernel Crypto API Security Policy for instructions.

If dm-crypt is the disk volume encryption mentioned above, used to encrypt a disk or partition, particularly when other modules may store cryptographic keys or other CSPs (critical security parameters) there, it must be in FIPS 140-2 approved mode as described in the dm-crypt Security Policy.

If the kernel is in the approved mode, the remaining RHEL6.2 FIPS services start up in the approved mode by default.

The approved mode for a module becomes effective as soon as the module power on self-tests complete successfully and the module loads into memory. Self-tests and integrity tests triggered in RHEL6.2 at start-up or on the first invocation of the crypto module:

- kernel: during boot, before dm-crypt becomes available via initramfs
- user space: before the user space module is available to the caller (i.e., for libraries, during the initialization call; for applications, at load time)

See each module security policy for descriptions of self-tests performed by each module and descriptions of how self-test errors are reported for each module.

2 Cryptographic Module Ports and Interfaces

Interface	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	Initialization, use of the cryptographic services listed below
Crypto Officer	Initialization, use of the cryptographic services listened below, 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

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

R – The item is read or referenced by the service.
W – The item is written or updated by the service.
Z – The persistent item is zeroized by the service.

Role	Service	CSP	Algo/Mode(s)	API/Function Call	Access
User, Crypto Officer	Triple-DES ENC/DEC (Cert #1225, 1228, 1229,1230)	168 bit Triple- DES Key	Triple-DES (ECB,CBC,CFB64, OFB,CTR)	GCRY_CIPHER_3DES	R, W, Z
User, Crypto Officer	AES128 ENC/DEC (Cert #1886, 1890, 1891, 1892)	128 bit AES Key	AES (ECB,CBC,CFB128,CTR, OFB)	GCRY_CIPHER_AES128	R, W, Z
User, Crypto Officer	AES192 ENC/DEC (Cert #1886, 1890, 1891, 1892)	192 bit AES Key	AES (ECB,CBC,CFB128,CTR, OFB)	GCRY_CIPHER_AES192	R, W, Z
User, Crypto Officer	AES256 ENC/DEC (Cert #1886, 1890, 1891, 1892)	256 bit AES Key	AES (ECB,CBC,CFB128,CTR, OFB)	GCRY_CIPHER_AES256	R, W, Z

Role	Service	CSP	Algo/Mode(s)	API/Function Call	Access
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 #1657, 1660, 1661, 1662)	none	N/A	GCRY_MD_SHA1	R, W, Z
User, Crypto Officer	SHA-224 (Cert #1657, 1660, 1661, 1662)	none	N/A	GCRY_MD_SHA224	R, W, Z
User, Crypto Officer	SHA-256 (Cert #1657, 1660, 1661, 1662)	none	N/A	GCRY_MD_SHA256	R, W, Z
User, Crypto Officer	SHA-384 (Cert #1657, 1660, 1661, 1662)	none	N/A	GCRY_MD_SHA384	R, W, Z
User, Crypto Officer	SHA-512 (Cert #1657, 1660, 1661, 1662)	none	N/A	GCRY_MD_SHA512	R, W, Z
User, Crypto Officer	HMAC-SHA-1 (Cert #1128, 1131, 1132, 1133)	MAC-key	N/A	GCRY_MD_SHA1, GCRY_MD_FLAG_HMAC	R, W, Z
User, Crypto Officer	HMAC-SHA-224 (Cert #1128, 1131, 1132, 1133)	MAC-key	N/A	GCRY_MD_SHA224, GCRY_MD_FLAG_HMAC	R, W, Z
User, Crypto Officer	HMAC-SHA-256 (Cert #1128, 1131, 1132, 1133)	MAC-key	N/A	GCRY_MD_SHA256, GCRY_MD_FLAG_HMAC	R. W. Z
User, Crypto Officer	HMAC-SHA-384 (Cert #1128, 1131, 1132, 1133)	MAC-key	N/A	GCRY_MD_SHA384, GCRY_MD_FLAG_HMAC	R, W, Z
User, Crypto Officer	HMAC-SHA-512 (Cert #1128, 1131, 1132, 1133)	MAC-key	N/A	GCRY_MD_SHA512, GCRY_MD_FLAG_HMAC	R, W, Z

Role	Service	CSP	Algo/Mode(s)	API/Function Call	Access
User, Crypto Officer	DSA (verify,sign & pqg) (Cert #591, 594, 595, 596)	Key Length L=1024, N=160, L=2048, N=224, L=2048, N=256, L=3072, N=256	DSA	GCRY_PK_DSA	R, W, Z
User, Crypto Officer	Fill buffer with length random bytes (Cert #988, 991, 992, 993)	Seed, Seed Key	RNG	gcry_randomize	W, Z
User, Crypto Officer	Convenience function to allocate a memory block consisting of nbytes of random bytes (Cert #988, 991, 992, 993)	Seed, Seed Key	RNG	gcry_random_bytes	W, Z
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 #988, 991, 992, 993)	Seed, Seed Key	RNG	gcry_random_bytes_secure	W, Z
User, Crypto Officer	Fill buffer with random bytes. (Cert #988, 991, 992, 993)	Seed, Seed Key	RNG	gcry_create_nonce	R, Z
User, Crypto Officer	Initialize Module	N/A	N/A	gcry_check_version	Z

Role	Service	CSP	Algo/Mode(s)	API/Function Call	Access
User, Crypto Officer	Self-Test	N/A	All	GCRYCTL_SELFTEST	Z
User, Crypto Officer	Zeroize secure memory	All	N/A	GCRYCTL_TERM_SECMEM	W,Z
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, Z
User, Crypto Officer	Release all resources of hash context	Context based	All hash functions	GCRY_MD_CLOSE	R, W,Z
User, Crypto Officer	Release the S-expression object SEXP (RSA/DSA keys)	Context based	SEXP	GCRY_SEXP_RELEASE	R, W, Z
User, Crypto Officer	RSA (verify, sign) (Cert #963, 966, 967, 968)	Variable Length Key 1024 – 4096 bits	RSA	GCRY_PK_RSA	R, W, Z
User, Crypto Officer	Get Status	N/A	N/A	GCRYCTL_SET_VERBOSITY	R, Z

Table 5. *Approved Services*

CAVEATS:

1) The strength of AES encryption keys is between 128 and 160 bits
2) The module generates cryptographic keys whose strengths are modified by available entropy – 160 bits

3.2.2 Non-Approved Allowed Services

Role	Service	CSP	Algo/Mode(s)	API call	Access
User, Crypto Officer	RSA (encrypt, decrypt, key wrapping) Key	Variable Length Key 1024 – 4096 bits	RSA	GCRY_PK_RSA	R, W, Z

Role	Service	CSP	Algo/Mode(s)	API call	Access
	establishment methodology provides between 80 and 150 bits of encryption strength				
User, Crypto Officer	MD5 (for TLS only)	N/A	N/A	GCRY_PK_MD5	R, W, Z

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.

3.4 Mechanism and Strength of Authentication

At security level 1, authentication is not required.

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 140-2 approved mode, the ptrace(2) system call, debugger(gdb(1)), and strace(1) shall not be used. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or systemtap 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 does support 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 two internal values, one key and one seed, which are stored in the encryption contexts used by the RNG. These 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 also the encryption contexts. 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.

The random number generator is keyed and seeded with data that is loaded from the /etc/gcrypt/rngseed if the device or symlink to device exists xored with the data from the /dev/urandom device. This allows the system administrator to always seed the RNGs from /dev/random if it is required.

7 Electromagnetic Interference/Electromagnetic Compatibility

Product Name and Model: HP ProLiant Server DL585 Series
Regulatory Model Number: HSTNS-1025
Product Options: All
EMC: Class A

Product Name and Model: IBM BladeCenter HS22 Series
Regulatory Model Number: 09-EMCRTP-0008
Product Options: All
EMC: Class A

8 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.

No operator intervention is required during the running of self-tests.

See section 8.1.3 for descriptions of possible self-test errors and recovery procedures.

8.1 Power-Up Tests

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

8.1.1 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. 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

Algorithm	Modes	Comments
		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	N/A	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 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

8.2 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 first 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 than 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

Algorithm	Comments
	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

9 Guidance

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

9.1 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. Existing prelinking, if any, should be undone on all system files using the 'prelink -u -a' command.

To bring the module into FIPS mode, perform the following:

1. Install the dracut-fips package:

   ```
   # yum install dracut-fips
   ```

2. Recreate the INITRAMFS image:

   ```
   # dracut -f
   ```

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

```
fips=1
```

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 to the kernel command line:

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

Reboot to apply these settings.

Because FIPS 140-2 has certain restrictions on the use of cryptography which are not always wanted, libgcrypt needs to be put into FIPS approved 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 approved mode at initialization time. This is the mechanism recommended for ordinary use, activated by using the fips=1 option in the boot loader, as described above.
- If the file /etc/gcrypt/fips_enabled exists, libgcrypt is put into a FIPS approved mode suitable for TLS

usage at initialization time. If the stricter non-TLS mode is needed, then the file must exist and have a non-zero number on the first line. This setting is global for all applications linked to libgcrypt that can read the file. Therefore it is not suitable in general but is useful for chroot environments.

Note that this file name is hardwired and does not depend on any configuration options.

- The application can request to enter the FIPS approved mode using the control command 'GCRYCTL_FORCE_FIPS_MODE'. This will put it in a mode suitable for TLS usage. If you are not needing TLS support, then you must instead use the stricter mode that disables MD5 by using the control command 'GCRYCTL_SET_ENFORCED_FIPS_FLAG'. This must be done prior to any initialization (i.e., before 'gcry_check_version').

If an application that uses libgcrypt for its cryptography is put into a chroot environment, the crypto officer must ensure one of the three above methods is available to the module from within the chroot environment to ensure entry into FIPS mode. Failure to do so will not allow the application to properly enter 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.

To operate the libgcrypt crypto module, the operating system must be restricted to a single operator mode of operation. (This should not be confused with single user mode which is runlevel 1 on RHEL. This refers to processes having access to the same cryptographic instance which RHEL ensures this cannot happen by the memory management hardware.)

See section 9.3 for information about handling self-test errors.

9.2 User (Application Developer) Guidance

See user guide for complete instructions for use of the library. Note especially the “FIPS Mode” appendix.

Applications using libgcrypt need to call “gcry_control(GCRYCTL_TERM_SECMEM)” before the process is terminated. This includes both normal process termination and termination caused by a signal.

The function “gcry_set_allocation_handler” may not be used.

The user must not call malloc/free to create/release space for keys. Instead, keys should be allocated in secure memory using functions 'gcry_malloc_secure' and 'gcry_calloc_secure', and freed using 'gcry_free', which will ensure that the key memory is overwritten before it is released. Each application is individually responsible for management and zeroization of keys outside of the libgcrypt module.

See section 9.3 for information about handling self-test errors.

See Appendix A for skeleton code that illustrates how to invoke the libgcrypt module.

9.3 Handling Self-Test Errors

The libgcrypt module performs a power-on self-test that includes an integrity check and known answer tests for the available cryptographic algorithms. It also runs conditional tests on the RNG whenever it is used and on asymmetric cryptographic keys whenever they are generated.

The effects of self-test failures in the libgcrypt module differ depending on the type of self-test that failed.

If the integrity test or the power-on self-test fails, the module will enter the error (non-operational) state. In the error (non-operational) state, any crypto calls to the module will abort the application. The self-test failure message is written to the system log for an integrity test failure. Other messages are returned and may be handled by the calling application as appropriate.

Power-On Test Failure Message	Description
Integrity check using `<hmac file>' failed	This message occurs if the integrity test fails. This message is printed to syslog.
None	A failing KAT test failure forces the module into an error state
Conditional Test Error Messages	
Duplicate 128 bit block returned by RNG	If the RNG continuous self-test fails, the RNG generated a duplicate block of random numbers.
Self-test after key generation failed	This occurs after RSA or DSA key generation test failure

Table 9. Test Failure and Error Messages

If you call a cryptographic function while the module is in a non-operational state (before initialization, or after a self-test failure), the function returns the error “called in non-operational state.” The calling application should be programmed to avoid this error or handle it accordingly.

An integrity check failure will make the module enter the error (non-operational) state. In this state, any crypto calls to the module will abort the application. For an integrity check failure or a KAT failure, try restarting the module (or the application that uses the module). For persistent failures, reinstall the module. If you downloaded the software, verify the package hash to confirm a proper download.

A KAT failure puts the module in an error state. Subsequent calls to cryptographic functions print the message "called in non-operational state" to the syslog and cause the calling application to abort. An application could be modified to receive an error message by calling the following function after initialization of libgcrypt:

```
if (gcry_control(GCRYCTL_OPERATIONAL_P, 0)) {
/* Succeeded */
} else {
/* Failed - subsequent crypto calls will make the application abort */
}
```

For an RNG continuous test failure, try obtaining another random number to get past the error and continue. If the test continues to fail, try restarting the module (or the application that uses the module). For persistent failures, reinstall the module. If you downloaded the software, verify the package hash to confirm a proper download.

For an RSA or DSA key generation error, try regenerating the key pair. If the failure persists, try restarting the module (or the application that uses the module). For persistent failures, reinstall the module. If you downloaded the software, verify the package hash to confirm a proper download.

10 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*/
};
```

11 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
CC	Common Criteria
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cypher Feedback
CMT	Cryptographic Module Testing
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode for block encryption
CVT	Component Verification Testing
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	Random 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 10. Glossary and Abbreviations

12 References

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

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

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

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

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

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

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

[8] 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

Appendix A: Code for Invoking the libgcrypt Module

The following skeleton code demonstrates how to invoke the libgcrypt module:

```
/* Example of starting libgcrypt for a non-TLS usage */
gcry_control(GCRYCTL_SET_ENFORCED_FIPS_FLAG, 0);
if (!gcry_check_version(GCRYPT_VERSION))
    error out
gcry_control(GCRYCTL_INITIALIZATION_FINISHED, 0);

/* example of storing a symmetric cipher context in secure memory */
if (gcry_cipher_open(&cipher_hd, GCRY_CIPHER_AES128,
GCRY_CIPHER_MODE_CBC, GCRY_CIPHER_SECURE)) {
/* Error */
}

/* zeroization and release of the symmetric cipher context */
gcry_cipher_close(cipher_hd);

/* example of storing a hash context in secure memory */
if (gcry_md_open(&hash_hd, GCRY_MD_SHA256, GCRY_MD_SECURE)) {
/* Error */
}

/* zeroization and release of the hash context */
gcry_md_close(hash_hd);

/* arbitrary secure memory allocation */
secure_buffer = gcry_malloc_secure(size);

/* arbitrary secure memory zeroization and release */
gcry_free(secure_buffer);

/* Note that if the public/private key created with
gcry_sexp_new(), gcry_sexp_sscan() or any part of the s-expression
created with gcry_sexp_build() is stored in secure memory,
the resulting s-expression will be stored in secure memory and zeroized
on release with gcry_sexp_release(). */
```