Amazon Linux 2 Libgcrypt Cryptographic Module

Module Version 1.0

FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.1

Last update: 2020-Jan-27

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

© 2020 Amazon Web Services, Inc./atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table of Contents

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

List of Tables

List of Figures

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Copyrights and Trademarks

Amazon is a registered trademark of Amazon Web Services, Inc. or its affiliates.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

1 Introduction

This document is the non-proprietary FIPS 140-2 Security Policy for version 1.0 of the Amazon Linux 2 Libgcrypt Cryptographic Module. It contains the security rules under which the module must be operated 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.

1.1 Purpose of the Security Policy

There are three major reasons that a security policy is needed:

- It is required for FIPS 140 2 validation,
- It allows individuals and organizations to determine whether a cryptographic module, as implemented, satisfies the stated security policy, and
- It describes the capabilities, protection and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will meet their security requirements.

1.2 Target Audience

This document is part of the package of documents that are submitted for FIPS 140 2 conformance validation of the module. It is intended for the following audience:

- Developers.
- FIPS 140-2 testing lab.
- The Cryptographic Module Validation Program (CMVP).
- Customers using or considering integration of Amazon Linux 2 Libgcrypt Cryptographic Module.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

2 Cryptographic Module Specification

2.1 Module Overview

The Amazon Linux 2 Libgcrypt Cryptographic Module (hereafter referred to as the “module”) implements general purpose FIPS 140-2 Approved cryptographic algorithms. The module provides these cryptographic services to applications running in the user space of the underlying operating system through an application program interface (API).

2.2 FIPS 140-2 Validation Scope

Table 1 shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard.

Table 1: FIPS 140-2 Security Requirements.

Security Requirements Section		Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles and Services and Authentication	1
4	Finite State Machine 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
Overall Level		1

2.3 Definition of the Cryptographic Module

The Amazon Linux 2 Libgcrypt Cryptographic Module is defined as a Software, Multi-chip Standalone module per the requirements within FIPS 140-2. The logical cryptographic boundary of the module consists of the shared library file and its integrity test HMAC file, which are delivered through the Amazon Linux 2 yum core repository (ID amz2-core/2/x86_64) from the following RPM files:

- libgcrypt-1.5.3-14.amzn2.0.2.x86_64.rpm

Table 2 summarizes the components of the cryptographic module.

Table 2: Components of the module.

Component	Description
/usr/lib64/libgcrypt.so.11	This library provides the main interface that allows the calling applications to request cryptographic services.
/usr/lib64/.libgcrypt.so.11.hmac	HMAC-SHA-256 value of the associated library for integrity check during the power-on.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Figure 1 shows the logical block diagram of the module executing in memory on the host system. The logical cryptographic boundary is indicated with a dashed colored box.

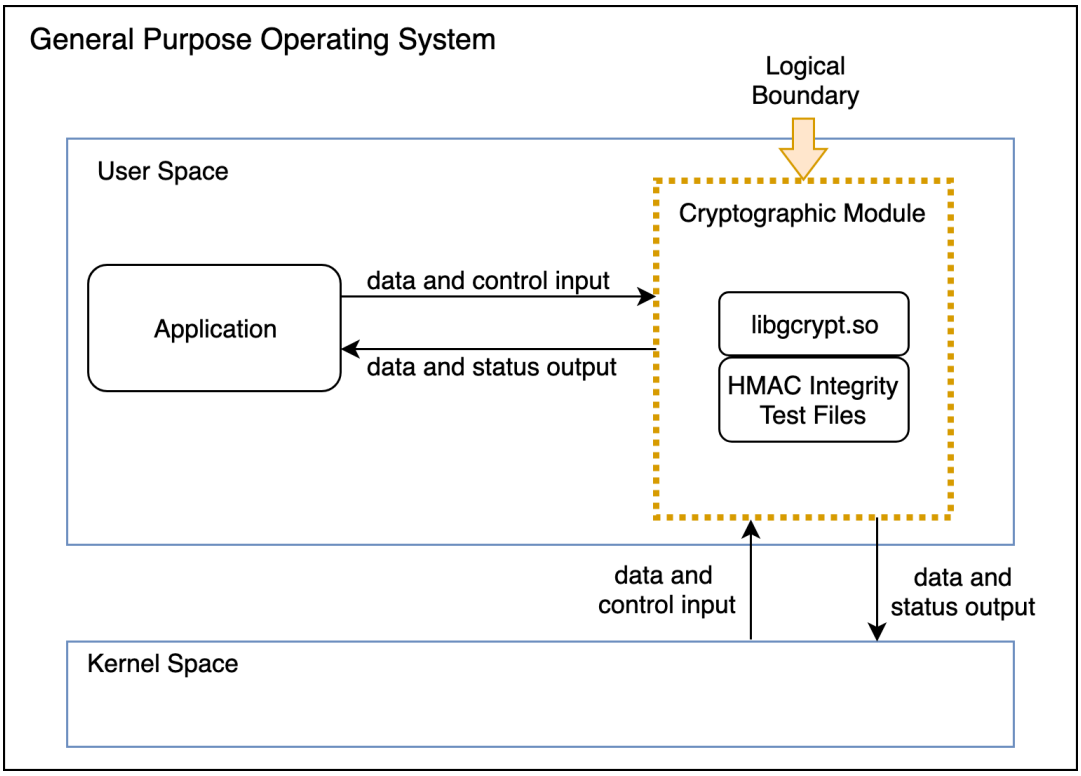

Figure 1: Logical cryptographic boundary.

2.4 Definition of the Physical Cryptographic Boundary

The physical cryptographic boundary of the module is defined as the hard enclosure of the host system on which the module runs (a general-purpose computer). Figure 2 depicts the hardware block diagram. The physical hard enclosure is indicated by the dashed colored line. No components are excluded from the requirements of FIPS 140-2.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

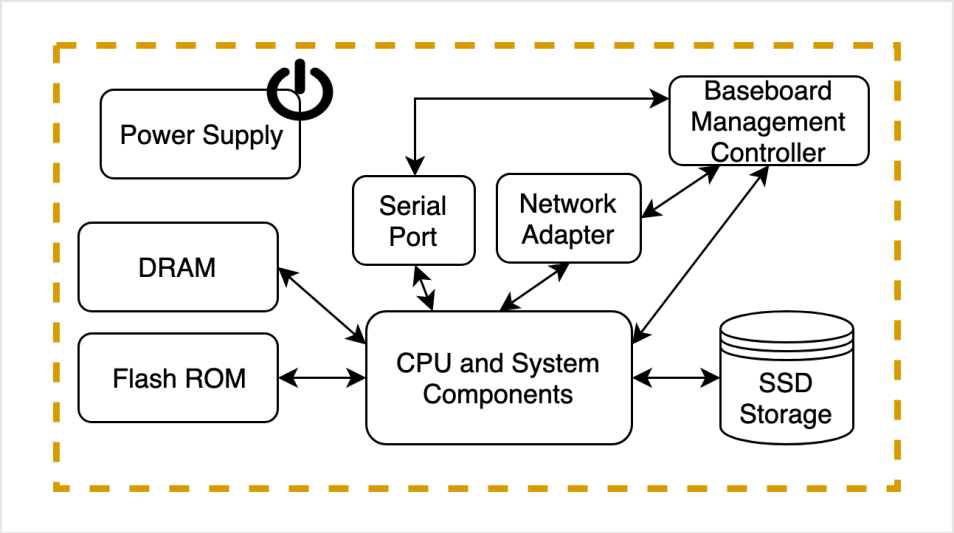

Figure 2: Hardware block diagram.

2.5 Tested Environments

The module was tested on the environment/platform listed in Table 3. The tested operational environment is not a virtualized cloud service, and was controlled such that the laboratory had full and exclusive access to the environment and module during the testing procedures.

Table 3: Tested operational environment.

Operating System	Processor	Hardware
Amazon Linux 2	Intel ® Xeon ® E5-2686 (Broadwell) x86_64bit	Amazon EC2 i3.metal 512 GiB system memory 13.6 TiB SSD storage + 8 GiB SSD boot disk 25 Gbps Elastic Network Adapter

2.6 Modes of Operation

The module supports two modes of operation.

- In "**FIPS mode**" (the Approved mode of operation), only approved or allowed security functions with sufficient security strength are offered by the module.
- In "**non-FIPS mode**" (the non-Approved mode of operation), non-approved security functions are offered by the module.

The module enters the operational mode after Power-On Self-Tests (POST) succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength[1] of the cryptographic keys chosen for the service.

If the POST or the Conditional Tests fail (Section 9), the module goes into the error state. The status of the module can be determined by the availability of the module. If the module is available, then it had passed all self-tests. If the module is unavailable, it is because any self-test failed, and the module has transitioned to the error state.

[1] See Section 5.6.1 in [SP800-57] for a definition of “security strength”.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Keys and Critical Security Parameters (CSPs) used or stored in FIPS mode shall not be used in non-FIPS mode, and vice versa.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

3 Module Ports and Interfaces

As a Software module, the module does not have physical ports. The operator can only interact with the module through the API provided by the module. Thus, the physical ports within the physical boundary are interpreted to be the physical ports of the hardware platform on which the module runs and are directed through the logical interfaces provided by the software.

The logical interfaces are the API through which applications request services and receive output data through return values or modified data referenced by pointers. Table 4 summarizes the logical interfaces and the power input.

Table 4: Ports and interfaces.

Logical Interface	Description
Data Input	API input parameters for data.
Data Output	API output parameters for data.
Control Input	API function calls.
Status Output	API return codes, error message.
Power Input	Not applicable for the Software module.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

4 Roles, Services and Authentication

4.1 Roles

The module supports the following roles:

- **User role**: performs all services (in both FIPS mode and non-FIPS mode of operation), except module installation and configuration. This role is assumed by the calling application accessing the module.
- **Crypto Officer role**: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed depending on the service requested.

4.2 Services

The module provides services to calling applications that assume the user role, and human users assuming the Crypto Officer role. Table 5 and Table 6 depict all services, which are described with more detail in the user documentation.

The tables use the following convention when specifying the access permissions that the module has for each CSP or key.

- **Create (C)**: the calling application can create a new CSP.
- **Read (R)**: the calling application can read the CSP.
- **Update (U)**: the calling application can write a new value to the CSP.
- **Zeroize (Z)**: the calling application can zeroize the CSP.
- **N/A**: the calling application does not access any CSP or key during its operation.

For the “Role” column, U indicates the User role, and CO indicates the Crypto Officer role. A checkmark symbol marks which role has access to that service.

4.2.1 Services in the FIPS-Approved Mode of Operation

Table 5 provides a full description of FIPS Approved services and the non-Approved but Allowed services provided by the module in the FIPS-approved mode of operation and lists the roles allowed to invoke each service.

Table 5: Services in the FIPS-approved mode of operation.

Service	Service Description and Algorithms	Role		Keys and CSPs	Access Types
		U	CO		
Symmetric Encryption/Decryption	Encrypts or decrypts a block of data using AES or Triple-DES.	✓		AES or Triple-DES key	R
RSA Key Generation	Generate RSA asymmetric keys using DRBG.	✓		RSA public/private keys	C, R, U
DSA Key Generation	Generate DSA asymmetric keys using DRBG.	✓		DSA public/private keys	C, R, U
RSA Digital Signature Generation and Verification	Sign and verify signature operations for RSA PCKS#1v1.5 and PSS.	✓		RSA public/private keys	R
DSA Digital Signature	Sign and verify signature operations for DSA.	✓		DSA public/private keys	R

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Service	Service Description and Algorithms	Role		Keys and CSPs	Access Types
		U	CO		
Generation and Verification					
Message Authentication Code (MAC)	Authenticate and verify authentication of data using HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512.	✓		HMAC Key	R
Message Digest	Hash a block of data with SHS (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512).	✓		None	N/A
Random Number Generation	Generate random numbers based on the SP 800-90A DRBG.	✓		Entropy input string and internal state	C, R, U
Get Key Length	Obtain key length of cipher with call to cipher_get_keylen()	✓		None	N/A
Get Block Size	Obtain block size of cipher with call to cipher_get_blocksize()	✓		None	N/A
Check Availability of Cipher	Check whether cipher is registered with call to cipher_get_blocksize()	✓		None	N/A
Other FIPS-related Services					
Show Status	Show status of the module state	✓		None	N/A
Self-Test	Initiate power-on self-tests	✓		None	N/A
Zeroization	Zeroize all critical security parameters with call to gcry_cipher_close(), gcry_sexp_release(), gcry_drbg_uninstantiate(),gcry_md_close()	✓		All keys and CSPs	Z
Module Installation	Installation of the module		✓	None	N/A
Module Configuration	Configuration of the module		✓	None	N/A

4.2.2 Services in the Non-FIPS-Approved Mode of Operation

Table 6 presents the services only available in non-FIPS-approved mode of operation.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table 6: Services in the non-FIPS approved mode of operation.

Service	Service Description and Algorithms	Role		Keys	Access Types
		U	CO		
RSA key wrapping using encryption/decryption primitives	Encrypt or decrypt using RSA key sizes not listed in Table 7.	✓		RSA key pair	R
ElGamal Asymmetric Algorithm	Encrypt, decrypt, signature generation and verification	✓		ElGamal keys	C, R, U
Symmetric Encryption/Decryption	Encrypt or decrypt using symmetric algorithms not listed in Table 7.	✓		ARC4, Blowfish, Camellia, CAST5, DES, IDEA, RC2, GOST, SEED, Serpent, Twofish, 2-Key Triple-DES Key (encryption only[2])	R
Digital Signature Generation and Verification	Sign or verify operations with RSA and DSA key sizes not listed in Table 7; signature generation with SHA-1.	✓		RSA, DSA key pairs	C, R, U
Asymmetric Key Generation	Generation RSA and DSA keys in sizes no listed in Table 7 and not compliant with FIPS 186-4.	✓		RSA, DSA key pairs	C, R, U
Message Digest	Hashing using algorithms not listed in Table 7.	✓		n/a	n/a
Message Authentication Code	MAC generation using HMAC with keys not listed in Table 7.	✓		HMAC key	R
Cyclic Redundancy Code	Error detection code using CRC-32	✓		n/a	n/a
Random Number Generation	Random number generation using CSPRNG.	✓		Entropy input string, internal state	C, R, U
Password-Based Key Derivation Function	Key derivation using OpenPGP salted and iterated salted S2K	✓		Derived keys	C, R, U

4.3 Algorithms

The module implements cryptographic algorithms that are used by the services provided by the module. The cryptographic algorithms that are approved to be used in the FIPS mode of operation are tested and validated by the CAVP.

Table 7, Table 8 and Table 9 present the cryptographic algorithms in specific modes of operation. These tables include the CAVP certificates, the algorithm name, respective standards, the available modes and key sizes wherein applicable, and usage. Information from certain columns may be applicable to more than one row.

[2] 2-key Triple-DES decryption is allowed in FIPS mode for legacy use.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

4.3.1 FIPS-Approved Algorithms

Table 7 lists the cryptographic algorithms that are approved to be used in the FIPS mode of operation.

Table 7: FIPS-approved cryptographic algorithms.

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert.#
AES	[FIPS197] [SP800-38A]	CBC, CTR, ECB, CFB128, OFB	128, 192 and 256 bits	Data Encryption and Decryption	#C693
DSA	[FIPS 186-4]		L=2048, N=224; L=2048, N=256; L=3072, N=256	Key Pair Generation	#C693
		SHA-224, SHA-256	L=2048, N=224; L=2048, N=256; L=3072, N=256	Domain Parameter Generation	
		SHA-224	L=2048, N=224	Signature Generation	
		SHA-256	L=2048, N=256; L=3072, N=256		
		SHA-1	L=1024, N=160	Signature Verification	
		SHA-224	L=2048, N=224		
		SHA-256	L=2048, N=256; L=3072, N=256		
DRBG	[SP800-90A]	CTR_DRBG AES-128, AES-192, AES-256 with DF, with PR HASH_DRBG with SHA-1, SHA-256, SHA-384, SHA-512 with and without PR HMAC_DRBG with HMAC-[SHA-1, SHA-256, SHA-384, SHA-512] with and without PR	n/a	Random Number Generation	#C693
HMAC	[FIPS198-1]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 bits or greater	Message Authentication Code	#C693

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Standard	Mode/Method	Key size	Use	CAVP Cert.#
RSA	[FIPS186-4]	X9.31	2048 and 3072 bits	Key Pair Generation	#C693
		PKCS#1v1.5 and PSS with SHA-224, SHA-256, SHA-384, SHA-512	2048 and 3072 bits	Digital Signature Generation	
		PKCS#1v1.5 and PSS with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	1024, 2048, and 3072 bits	Signature Verification	
SHS	[FIPS180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest	#C693
Triple-DES	[SP800-67] [SP800-38A]	CBC, CTR, ECB, CFB64, OFB	192 bits	3-key Data Encryption and Decryption 2-key Data Decryption (legacy use)	#C693

4.3.2 Non-Approved-but-Allowed Algorithms

Table 8 lists the non-Approved-but-Allowed cryptographic algorithms provided by the module that are allowed to be used in the FIPS mode of operation.

Table 8: Non-Approved-but-allowed cryptographic algorithms.

Algorithm	Usage
RSA Key Wrapping with key size between 2048 bits and 15360 bits (or more)	Key wrapping, key establishment methodology provides between 112 and 256 bits of encryption strength.
NDRNG	Used for seeding NIST SP 800-90A DRBG.

4.3.3 Non-Approved Algorithms

Table 9 lists the cryptographic algorithms that are not allowed to be used in the FIPS mode of operation. Use of any of these algorithms (and corresponding services in Table 6) will implicitly switch the module to the non-Approved mode.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table 9: Non-FIPS approved cryptographic algorithms.

Algorithm	Usage
2-key Triple-DES	Encryption
Blowfish	Encryption/decryption
Camellia	Encryption/decryption
CAST5	Encryption/decryption
CRC32	Error detection code generation and verification
CSPRNG	Random number generator
DES	Encryption/decryption
DSA	Parameter verification; parameter/key/signature generation and verification with keys not listed in Table 7
ElGamal	Key generation; key wrapping; signature generation and verification
GOST	Key wrapping 28147-89; Hash R 34.11-94 (RFC4357); Hash R 34.11.2012 (Streebog) (RFC6986)
IDEA	Encryption, decryption
MD2	Hash function
MD4	Hash function
MD5	Hash function
RC2	Encryption/decryption
RC4	Encryption/decryption (ARCFOUR)
RIPEMD160	Hash function
RSA	Key generation/Signature generation/Signature verification with keys of length not listed in Table 7
SEED	Encryption, decryption
Serpent	Encryption/decryption
SHA-1	Signature generation
Tiger	Hash function
Twofish	Encryption/decryption
Whirlpool	Hash function

4.4 Operator Authentication

The module does not support operator authentication mechanisms. The role of the operator is implicitly assumed based on the service requested.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

5 Physical Security

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

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

6 Operational Environment

6.1 Applicability

The module operates in a modifiable operational environment per FIPS 140-2 Security Level 1 specifications. The module runs on the Amazon Linux 2 operating system executing on the hardware specified in Section 2.5.

6.2 Policy

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

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

In operational mode, the ptrace(2) system call, the 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.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

7 Cryptographic Key Management

Table 10 summarizes the keys and other CSPs that are used by the cryptographic services implemented in the module. The table describes the use of each key/CSP and, as applicable, how they are generated or established, their method of entry and output of the module, their storage location, and the method for zeroizing the key/CSP.

All key and CSP storage is done in plaintext.

Table 10: Lifecycle of keys and other Critical Security Parameters (CSPs).

Name	Use	Generation/ Establishment	Entry/ Output	Storage	Zeroization
AES key	Encryption, decryption.	Provided by the calling application.	Entered via API input parameter. No output.	RAM	gcry_cipher_close()
Triple-DES key	Encryption, decryption.	Provided by the calling application.	Entered via API input parameter. No output.	RAM	gcry_cipher_close()
HMAC key	MAC generation and verification	Provided by the calling application.	Entered via API input parameter. No output.	RAM	gcry_md_close()
RSA public/private key	RSA signature generation and verification. Key wrapping.	Keys are generated using FIPS 186-4 and the random value used in the key generation is obtained from SP800-90A DRBG.	Entered via API input parameter or generated by module. Output via API output parameters in plaintext.	RAM	gcry_sexp_release()
DSA public/private key	DSA signature generation and verification.	Keys are generated using FIPS 186-4 and the random value used in the key generation is obtained from SP800-90A DRBG.	Entered via API input parameter or generated by module. Output via API output parameters in plaintext.	RAM	gcry_sexp_release()
Entropy input string	Entropy input strings used to construct the seed for the DRBG.	Obtained from NDRNG.	N/A	RAM	gcry_drbg_uninstantiate()
DRBG Internal state (V, C, key)	Used internally by DRBG. Used to generate random bits.	During DRBG initialization.	N/A	RAM	gcry_drbg_uninstantiate()

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

7.1 Random Number Generation

The module provides a DRBG compliant with [SP800-90A] for the creation of key components of asymmetric keys, and random number generation. The DRBG implements three mechanisms: CTR_DRBG, HASH_DRBG, and HMAC_DRBG. The CTR_DRBG mechanism uses AES-128, AES-192 or AES-256 with derivation function and with/without prediction resistance. The HASH_DRBG uses SHA-1, SHA-256, SHA-384 or SHA-512 with/without prediction resistance. The HMAC_DRBG uses HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 or HMAC-SHA-512, with/without prediction resistance. The module initializes the DRBG, by default, to HMAC_DRBG with HMAC-SHA-256 without prediction resistance.

The DRBG is initialized during module initialization and seeded from the NDRNG from /dev/urandom. 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 256 bits of entropy to the DRBG.

The module performs continuous random number generator tests (CRNGT) on the output of SP800-90A DRBG to ensure that consecutive random numbers do not repeat. The operational environment, the Linux RNG, performs the continuous test on the NDRNG.

7.2 Key Generation

For generating RSA and DSA keys, the module implements asymmetric key generation services compliant with [FIPS186-4] and using a DRBG compliant with [SP800-90A]. The random value used in asymmetric key generation is obtained from the DRBG. In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per SP800-133 (vendor affirmed).

The module does not offer a dedicated service for generating keys for symmetric algorithms, or for HMAC.

7.3 Key Entry and Output

The module does not support manual key entry or intermediate key generation output. In addition, the module does not produce key output outside its physical boundary. The keys can be entered or output from the module in plaintext form via API parameters, to and from the calling application only.

7.4 Key/CSP Storage

Public and private keys are provided to the module by the calling process, and are destroyed when released by the appropriate API function calls. The module does not perform persistent storage of keys. The keys and CSPs are stored as plaintext in volatile memory (RAM). The protection of these keys and CSPs in RAM is provided by the operating system enforcement of separation of address space.

The HMAC keys used for integrity tests are stored within the module's binary files and rely on the operating system for protection.

7.5 Key/CSP Zeroization

The application is responsible for calling the appropriate destruction functions from the API to zeroize keys and CSPs. The destruction functions then overwrite the memory occupied by keys with zeros and deallocates the memory with the regular memory deallocation operating system call. In case of abnormal termination, the keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.

7.6 Key Establishment

The module provides RSA key wrapping (encapsulation) using public key encryption and private key decryption primitives as allowed by [FIPS140-2_IG] D.9.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table 7 and Table 8 specify the key sizes allowed in the FIPS mode of operation. According to “Table 2: Comparable strengths” in [SP800-57], the key sizes of RSA provide the following security strengths:

- RSA key wrapping provides between 112 and 256 bits of encryption strength.

7.7 Handling of Keys and CSPs between Modes of Operation

The module does not share CSPs between the FIPS-approved mode of operation and the non-FIPS mode of operation.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

8 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms listed in Table 3 have been tested and found to conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, FCC PART 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (i.e., Business use). These devices are designed to provide reasonable protection against harmful interference when the devices are operated in a commercial environment.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

9 Self-Tests

9.1 Power-on Self-Tests

The module performs power-up or power-on self-tests (POSTs) automatically during loading of the module by making use of default entry point (DEP). These POSTs ensure that the module is not corrupted and that the cryptographic algorithms work as expected. No operator intervention is necessary to run the POSTs.

While the module is executing the POSTs, services are not available, and input and output are inhibited. The module is not available for use until successful completion of the POSTs.

The integrity of the module binary is verified using an HMAC-SHA-256. The HMAC value is computed at build time and stored in the .hmac file. The value is recalculated at runtime and compared against the stored value in the file. If the comparison succeeds, then the remaining POSTs (consisting of the algorithm-specific Known Answer Tests) are performed. On successful completion of the all the power-on tests, the module becomes operational and crypto services are then available. If any of the tests fails, the module transitions to the error state and subsequent calls to the module will fail. Thus, in the error state, no further cryptographic operations will be possible.

Table 11 details the self-tests that are performed on the FIPS-approved cryptographic algorithms supported in the FIPS-approved mode of operation, using the Known-Answer Tests (KATs) and Pairwise Consistency Tests (PCTs).

Table 11: Self-tests.

Algorithm	Test
AES	• KAT AES-ECB with 128-bit key, encryption • KAT AES-ECB with 128-bit key, decryption • KAT AES-ECB with 192-bit key, encryption • KAT AES-ECB with 192-bit key, decryption • KAT AES-ECB with 256-bit key, encryption • KAT AES-ECB with 256-bit key, decryption
Triple-DES	• KAT Triple-DES (ECB) with 192-bit key, encryption • KAT Triple-DES (ECB) with 192-bit key, decryption
DSA	• PCT DSA with 2048-bit key and SHA-256
RSA	• KAT RSA PKCS#1v1.5 signature generation and verification with 2048-bit key and using SHA-256 • KAT RSA with 2048-bit key, public-key encryption • KAT RSA with 2048-bit key, private-key decryption
DRBG	• KAT HASH_DRBG with SHA-256, with/without PR • KAT HMAC_DRBG with HMAC-SHA-256, with/without PR • KAT CTR_DRBG with AES-128, with PR
HMAC	• KAT HMAC-SHA-1 • KAT HMAC-SHA-224 • KAT HMAC-SHA-256 • KAT HMAC-SHA-384 • KAT HMAC-SHA-512

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

SHS	• KAT SHA-1 • KAT SHA-224 • KAT SHA-256 • KAT SHA-384 • KAT SHA-512
Module Integrity	• HMAC-SHA-256

9.2 Conditional Self-Tests

Conditional tests are performed during operational state of the module when the respective crypto functions are used. If any of the conditional tests fails, module transitions to error state.

Table 12 lists the conditional self-tests performed by the functions.

Table 12: Conditional self-tests.

Algorithm	Test
DSA Key generation	PCT using signature generation and verification
RSA Key generation	PCT using signature generation and verification
DRBG	Continuous Random Number Generator Test (CRNGT)

9.3 On-Demand Self-tests

The module provides the Self-Test service to perform self-tests on demand. This service performs the same cryptographic algorithm tests executed during power-on. To invoke the on-demand self-tests, the user can make a call to the gcry_control(GCRYCTL_SELFTEST) function. During the execution of the on-demand self-tests, cryptographic services are not available and no data output or input is possible.

9.4 Error States

The module exhibits two error states: Error state and Fatal Error state.

The Module enters the Error tate (with an error message) on failure of the POST or any conditional tests. In the Error state, all data output is inhibited and no cryptographic operation is allowed. The Error state can be recovered by calling gcry_control(GCRYCTL_SELFTEST) function. This function initiates the POST. If the POST is now successful, then the module recovers from the error. Otherwise, if the POST continues to be unsuccessful, the module remains in the Error state.

The module enters the Fatal Error state when random numbers are requested in Error state or when requesting cipher operations on 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 the Fatal Error state.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

10 Guidance

This section provides guidance for the Crypto Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

10.1 Crypto-Officer Guidance

The binaries of the module are delivered via Red Hat Package Manager (RPM) packages. The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated as FIPS 140-2 validated module. The version of the RPM packages containing the FIPS validated module are listed in Section 2.3.

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. If the module is already prelinked, the prelink should be undone on all the system files using the 'prelink -u -a' command.

To configure the operating environment to support FIPS perform the following steps:

1. Install the dracut-fips package:

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

2. Recreate the INITRAMFS image:

```
# dracut -f
```

After regenerating the initramfs, the Crypto Officer must append the following string to the kernel command line by changing the setting in the boot loader:

```
fips=1
```

If /boot or /boot/efi reside on a separate partition, the kernel parameter boot=<partition of /boot or /boot/efi> must be supplied. The partition can be identified with the following command, respectively:

```
"df /boot"
```

or

```
"df /boot/efi"
```

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.

The Crypto Officer shall check whether the file /proc/sys/crypto/fips_enabled exists and whether it contains “1”. If the file does not exist or does not contain “1”, the operational environment is not configured to support FIPS and the module will not operate as a FIPS validated module. Once the operational environment has been configured to support FIPS, it is not possible to switch back to standard mode without terminating the module first.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

After performing the above configuration, the Crypto Officer should proceed to module installation. The RPM package of the module can be installed using standard tools recommended for the installation of packages on an Amazon Linux 2 system (e.g., yum, RPM). The integrity of the RPM is automatically verified during the installation of the module and the Crypto Officer shall not install the RPM file if the yum server indicates an integrity error.

10.2 User Guidance

Applications using libgcrypt need to call gcry_control(GCRYCTL_INITIALIZATION_FINISHED, 0) after initialization is done. This procedure ensures that the DRBG is properly seeded. In addition, the function gcry_control(GCRYCTL_TERM_SECMEM) needs to be called before the process is terminated. The function gcry_set_allocation_handler() shall not be used.

The user must not call malloc/free to create/release space for keys. The libgcrypt must be utilized to manage space for keys, which will ensure that the key memory is zeroized before it is released. See the documentation file doc/gcrypt.texi within the source code tree for complete instructions for use.

The information pages are included within the developer package. The user can find the documentation at the following location after having installed the developer package:

- /usr/share/info/gcrypt.info-1.gz
- /usr/share/info/gcrypt.info-2.gz
- /usr/share/info/gcrypt.info.gz

10.2.1 Triple-DES Data Encryption

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES (64-bit) blocks, in accordance to [SP800-67] and IG A.13 in [FIPS140-2-IG].

10.2.2 Key Usage and Management

In general, a single key shall be used for only one purpose (e.g., encryption, integrity, authentication, key wrapping, random bit generation, or digital signatures) and be disjoint between the modes of operations of the module. Thus, if the module is switched between its FIPS mode and non-FIPS mode or vice versa, the following procedures shall be observed:

- The DRBG engine shall be reseeded.
- CSPs and keys shall not be shared between security functions of the two different modes.

The DRBG shall not be used for key generation for non-approved services in the non-FIPS mode.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

11 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·re mod n) is decrypted and the unblinded value (x' = y'·r-1 mod n) is returned. The blinding value "r" is a random value with the size of the modulus "n" and generated with `GCRY_WEAK_RANDOM' random level.

The module also implements a Weak Triple-DES key detection algorithm. 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 weak keys are displayed below, and have all their parity bits cleared.

```
static byte weak_keys[64][8] =
{
        { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }, /*w weak keys*/
        { 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 semi-weak keys*/
        { 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 },
```

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

```
        { 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*/
};
```

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

12 Acronyms, Terms and Abbreviations

Term	Definition
AES	Advanced Encryption Standard
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
EDC	Error Detection Code
FIPS	Federal Information Processing Standard
FSM	Finite State Model
HMAC	(Keyed) Hash Message Authentication Code
KAT	Known Answer Test
KDF	Key Derivation Function
MAC	Message Authentication Code
NDRNG	Non-Deterministic Random Number Generator
NIST	National Institute of Standards and Technology
OFB	Output Feedback Mode
OS	Operating System
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptographic Standard
POST	Power On Self-Test
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
PUB	Publication
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

13 References

FIPS140-2 **FIPS PUB 140-2 - Security Requirements For Cryptographic Modules**
May 2001
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS140-2_IG **Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program**
May 7, 2019
https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips140-2/FIPS1402IG.pdf

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

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

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

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

PKCS#1 **Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2**
November 2016
https://tools.ietf.org/rfc/rfc8017.txt

RFC3711 **The Secure Real-time Transport Protocol (SRTP)**
March 2004
https://tools.ietf.org/html/rfc3711

RFC4347 **Datagram Transport Layer Security**
April 2006
https://tools.ietf.org/html/rfc4347

RFC4357 **Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms**
January 2006
https://tools.ietf.org/html/rfc4357

RFC5246 **The Transport Layer Security (TLS) Protocol Version 1.2**
August 2008
https://tools.ietf.org/html/rfc5246

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

RFC5288	**AES Galois Counter Mode (GCM) Cipher Suites for TLS** August 2008 https://tools.ietf.org/html/rfc5288
RFC5764	**Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)** May 2010 https://tools.ietf.org/html/rfc5764
RFC6986	**GOST R 34.11-2012: Hash Function** August 2013 https://tools.ietf.org/html/rfc6986
SP800-131A	**NIST Special Publication 800-131A Revision 2- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP800-135	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-52	**NIST Special Publication 800-52 Revision 1 - Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** April 2014 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf
SP800-56A	**NIST Special Publication 800-56A - Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised)** March, 2007 http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf
SP800-67	**NIST Special Publication 800-67 Revision 2 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** November 2017 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** June 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

The FIPS 140-2 standard, and information on the CMVP, can be found at http://csrc.nist.gov/groups/STM/cmvp/index.html. More information describing the module can be found on the vendor web site at aws.amazon.com .

This Security Policy contains non-proprietary information. All other documentation submitted for FIPS 140-2 conformance testing and validation is proprietary and is releasable only under appropriate non-disclosure agreements.

© 2020 Amazon Web Services, Inc.; atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.