

AWS-LC Cryptographic Module

**version AWS-LC** FIPS 1.0.0

FIPS 140-2 Non-Proprietary Security Policy

**Version 1.0**

Last update: 2021-12-07

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

[1 General 4](#_Toc89105164)

[2 Cryptographic Module Specification 5](#_Toc89105165)

[3 Cryptographic Module Ports and Interfaces 9](#_Toc89105166)

[4 Roles, services, and authentication 10](#_Toc89105167)

[5 Software/Firmware security 14](#_Toc89105168)

[5.1 Integrity Techniques 14](#_Toc89105171)

[5.2 On-Demand Integrity Test 14](#_Toc89105172)

[6 Operational Environment 15](#_Toc89105173)

[6.1 Applicability 15](#_Toc89105175)

[6.2 Policy 15](#_Toc89105176)

[7 Physical Security 16](#_Toc89105177)

[8 Non-invasive Security 17](#_Toc89105189)

[9 Sensitive Security Parameter Management 18](#_Toc89105190)

[ Specify the RBG entropy source(s) 18](#_Toc89105191)

[10 Self-tests 19](#_Toc89105192)

[10.1 Pre-Operational Tests 19](#_Toc89105193)

[10.1.1 Integrity Test 19](#_Toc89105194)

[10.2 Conditional Tests 19](#_Toc89105199)

[10.2.1 Cryptographic algorithm tests 19](#_Toc89105200)

[10.2.2 Pairwise Consistency Test 20](#_Toc89105201)

[10.2.3 Periodic/On-Demand Self-Test 20](#_Toc89105210)

[10.3 Error States 20](#_Toc89105211)

[11 Life-cycle assurance 21](#_Toc89105212)

[11.1 Delivery and Operation 21](#_Toc89105213)

[11.2 Crypto Officer Guidance 21](#_Toc89105214)

[11.3 User Guidance 21](#_Toc89105215)

[12 Mitigation of other attacks 22](#_Toc89105218)

# Copyrights and Trademarks

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

# General

This document is the non-proprietary FIPS 140-3 Security Policy for version AWS-LC FIPS 1.0.0 of the AWS-LC Cryptographic Module. It has a one to one mapping to the [SP 800-140B] starting with section B.2.1 named “General” that maps to section 1 in this document and ending with section B.2.12 named “Mitigation of other attacks” that maps to section 12 in this document.

|  |  |  |
| --- | --- | --- |
| ISO/IEC 24759 Section 6. [Number Below] | FIPS 140-3 Section Title | Security Level |
| 1 | General | 1 |
| 2 | Cryptographic Module Specification | 1 |
| 3 | Cryptographic Module Interfaces | 1 |
| 4 | Roles, Services, and Authentication | 1 |
| 5 | Software/Firmware Security | 1 |
| 6 | Operational Environment | 1 |
| 7 | Physical Security | N/A |
| 8 | Non-invasive Security | N/A |
| 9 | Sensitive Security Parameter Management | 1 |
| 10 | Self-tests | 1 |
| 11 | Life-cycle Assurance | 1 |
| 12 | Mitigation of Other Attacks | N/A |

Table 1 - Security Levels

# Cryptographic Module Specification

The AWS-LC Cryptographic Module cryptographic module (hereafter referred to as “the module”) is a Software Multichip standalone cryptographic module. The module provides cryptographic services to applications running in the user space of the underlying operating system through a C language Application Program Interface (API).

The module has been tested on the following platforms with the corresponding module variants and configuration options with and without PAA:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Operating System | Hardware Platform | Processor | PAA/Acceleration |
| 1 | Ubuntu 20.04 | Amazon EC2 c5.metal with 192 GiB system memory and Elastic Block Store (EBS) 200 GiB | Intel Xeon Platinum 8275CL | AES-NI and SHA extensions |
| 2 | Amazon Linux 2 |
| 3 | Ubuntu 20.04 | Amazon EC2 c6g.metal with 128 GiB system memory and Elastic Block Store (EBS) 200 GiB | Graviton 2 | Neon and Crypto Extension (CE) |
| 4 | Amazon Linux 2 |

Table 2 - Tested Operational Environments

The table below lists all security functions of the module, including specific key strengths employed for approved services, and implemented modes of operation.

| CAVP Cert | Algorithm and  Standard | Mode / Method | Description / Key Size(s) / Key Strength(s) | Use / Function |
| --- | --- | --- | --- | --- |
| XX | AES  [FIPS197]  [SP 800-38A] | CBC | 128,192,256 bits | Encryption/ Decryption using *AES\_cbc\_encrypt*, *AES\_ctr128\_encrypt, AES\_ecb\_encrypt* |
| ECB | 128,192,256 bits |
| CTR | 128,192,256 bits |
| XX | AES [FIPS197] [SP800-38C] | CCM | 128 bits | Authenticated Encryption/ Decryption using *EVP\_aead\_aes\_128\_\** |
| XX | AES [FIPS197] [SP 800-38D] | GCM with Internal IV 8.2.2 | 128,256 bits | Authenticated Encryption/ Decryption *EVP\_aead\_aes\_#[[1]](#footnote-1)\_gcm\_randnonce* |
| XX | GCM with external IV | 128,256 bits | Authenticated Encryption/ Decryption  *EVP\_aead\_aes\_#\_gcm\_tls12*  *EVP\_aead\_aes\_#\_gcm\_tls13* |
|  | GMAC | 128,192,256 bits | Authentication with *EVP\_aead\_aes\_\** |
| XX | AES [FIPS197] [SP 800-38B] | CMAC | 128,256 bits |
| XX | AES [FIPS197]  [SP 800-38F] | KW and KWP | 128,192,256 bits | Key wrapping |
| XX | HMAC [FIPS198-1] | HMAC-SHA-1,224,256,384,512 | 112 bits or greater | Authentication |
| XX | SHA | SHA-1,224,256,384,512 |  | Message Digest |
| XX | DRBG [SP 800-90A] | CTR with AES | 256 bits | Random number generation |
| XX | RSA [FIPS 186-4] | B.3.3 Random Probably Primes | 2048,3072,4096 bits | Key Generation *RSA\_generate\_key\_fips* or *EVP\_PKEY\_keygen* |
| PKCS#1v1.5 and PSS | 2048,3072,4096 bits with SHA-224,256,384,512 | Signature Generation *EVP\_DigestSign or*  *EVP\_DigestSignInit*  *EVP\_DigestSignUpdate*  *EVP\_DigestSignFinal* |
| 2048,3072,4096 bits with SHA-1,224,256,384,512 | Signature Verification  *EVP\_DigestVerify or*  *EVP\_DigestVerifyInit*  *EVP\_DigestVerifyUpdate*  *EVP\_DigestVerifyFinal* |
| XX | ECDSA [FIPS 186-4] | B.4.2 Testing Candidate | P-224, P-256, P-384, P-521 | Key Generation *EC\_KEY\_generate\_key\_fips* or *EVP\_PKEY\_keygen* |
| N/A | Key Verification |
| P-224, P-256, P-384, P-521 with SHA-224, SHA-256, SHA-384, SHA-512 | Signature Generation *EVP\_DigestSign or*  *EVP\_DigestSignInit*  *EVP\_DigestSignUpdate*  *EVP\_DigestSignFinal* |
| Signature Verification *EVP\_DigestVerify or*  *EVP\_DigestVerifyInit*  *EVP\_DigestVerifyUpdate*  *EVP\_DigestVerifyFinal* |
| XX | KAS ECC SSC [SP 800-56ARev3] | ECC Ephemeral Unified scheme | P-224, P-256, P-384, P-521 | Shared secret computation |
| XX | KDF [SP 800-135] | TLS 1.0/1.1/1.2 | With SHA-256,384,512 | Key Derivation |
| N/A | ENT(NP)  [SPE 800-90B] | CPU Jitter source | N/A | Random number generation |
| See RSA/ECDSA rows | CKG [SP 800-133] | FIPS 186-4 | RSA: 2048,3072,4096 bits  ECDSA: P-224, P-256, P-384, P-521 | Key Generation with *RSA\_generate\_key\_fips,* *EC\_KEY\_generate\_key\_fips* *EVP\_PKEY\_keygen* |

Table 3 - Approved Algorithms

|  |  |  |
| --- | --- | --- |
| Algorithm | Caveat | Use/Function |
| MD5 | Allowed per IG 2.4.A | Message digest used in TLS 1.0 KDF only |

Table 4 - Non-Approved Allowed in the Approved Mode of Operation

|  |  |
| --- | --- |
| Algorithm/Functions | Use/Function |
| DES and Triple-DES | Encryption/Decryption |
| AES with OFB or CFB modes | Encryption/Decryption |
| AES with 192 bits | Encryption/Decryption |
| AES using *aes\_\*\_generic* function | Encryption/Decryption |
| AES GMAC using *aes\_\*\_generic* | Message Authentication |
| Diffie Hellman | shared secret computation |
| MD4 | Message digest |
| MD5 | Message digest outside TLS 1.0 |
| SHA-1 | Signature Generation |
| RSA using *RSA\_generate\_key\_ex* | Key Generation |
| ECDSA using *EC\_KEY\_generate\_key* | Key Generation |
| RSA using keys less than 2048 bits | Signature Generation/verification |
| RSA | Key wrapping, sign/verify primitive operations without hashing |
| Diffie-Hellman | Shared secret computation |
| TLS KDF using SHA-224 | Key Derivation |

Table 5 - Non-Approved Not Allowed in the Approved Mode of Operation

The software block diagram in Figure 1 shows the cryptographic boundary of the module, and its interfaces with the operational environment.

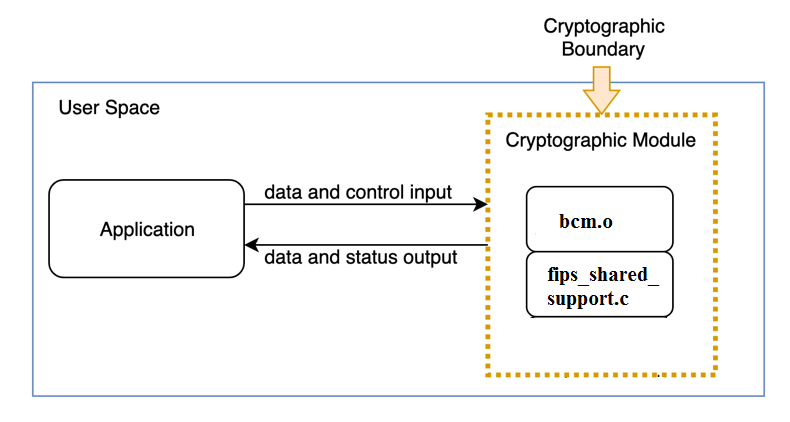


Figure 1 – Module Block Diagram

# Cryptographic 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.

| Logical Interface | Data that passes over port/interface |
| --- | --- |
| Data Input | API input parameters for data. |
| Data Output | API output parameters for data. |
| Control Input | API function calls. |
| Status Output | API return codes, error message. |
| Control output | Not applicable |

Table 6 - Ports and Interfaces

# Roles, services, and authentication

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed   
by the operator of the module. The module does not support authentication.

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Service | Input | Output |
| Crypto Officer | Encryption | Plaintext, key | Ciphertext |
| Decryption | Ciphertext, key | Plaintext |
| Authenticated Encryption | Plaintext, key | Ciphertext, authentication tag |
| Authenticated Decryption | Ciphertext, authentication tag, key | Plaintext |
| Key wrapping | Key wrapping key, key to be wrapped | Wrapped key |
| Message Authentication | Message, HMAC key | Authenticated Message |
| Message Digest | Message | Hashed Message |
| Random number generation | Amount of random number | Random numbers |
| Key Generation | Key size | Key pair |
| Signature Generation | Message, Hash algorithm, private key | Signature |
| Signature Verification | Signature, Hash algorithm, public key | Verification result |
| Shared secret computation | Key pair and Public key for remote peer | Shared secret |
| Key Derivation | PRF algorithm, TLS pre-master secret | TLS Derived key |
| Zeroization | Context containing SSPs | zeroised memory space |
| Self-test | Module Reset | result |
| Show Status | None | Return codes and/or log  messages |
| Show version info | None | Name and version  information |

Table 7 - Roles, Service Commands, Input and Output

The module provides following approved services that utilize approved and allowed security functions. The convention below applies when specifying the access permissions (types) that the service has for each SSP.

**G** = Generate: The module generates or derives the SSP.

**R** = Read: The SSP is read from the module (e.g. the SSP is output).

**W** = Write: The SSP is updated, imported, or written to the module.

**E** = Execute: The module uses the SSP in performing a cryptographic operation.

**Z** = Zeroise: The module zeroises the SSP.

The approved service indicator is a return value 1 from the FIPS\_service\_indicator\_check\_approved function. This function is used together with two other functions. The usage is as follows:

* STEP 1: Should be called before invoking the service

int before = FIPS\_service\_indicator\_before\_call();

* STEP 2: Make a service call

*func;*

* STEP 3: Should be called after invoking the service

int after = FIPS\_service\_indicator\_after\_call();

* STEP 4: Return value 1 indicates approved service was invoked

int Return= FIPS\_service\_indicator\_check\_approved(before, after);

| Service | Description | Approved Security Functions | Keys and/or SSPs | Roles | Access rights to SSPs | Indicator |
| --- | --- | --- | --- | --- | --- | --- |
| Encryption | Encryption | AES CBC, CTR, ECB with  listed in Table 3 | AES Key | CO | W | Return value 1 from the function  FIPS\_ service\_ indicator\_ check\_ approved() |
| Decryption | Decryption |
| Authenticated Encryption | Authenticated Encryption | AES CCM  AES GCM listed in Table 3 |
| Authenticated Decryption | Authenticated Decryption |
| AES Key wrapping[[2]](#footnote-2) | Encrypting/decrypting key | AES KW, KWP | AES key | R |
| Message Authentication | Mac computation | AES CMAC | AES Key  HMAC Key | W |
| AES GMAC |
| HMAC |
| Message Digest | Generating key pair | SHA | N/A | N/A |
| Random number generation | Generating random numbers | CTR\_DRBG | Entropy Input | W |
| Seed, V, Key | G |
| Key Generation | Generating key pair using | RSA listed in Table 3 | RSA key pair | G |
| ECDSA listed in Table 3 | ECDSA key pair |
| Key Verification | Verifying the public key | ECDSA listed in Table 3 | ECDSA Public key | W |
| Signature Generation | Generating signature | RSA, ECDSA listed in Table 3 | RSA private key  ECDSA private key | W |
| Signature Verification | Verifying signature | RSA, ECDSA listed in Table 3 | RSA public key  ECDSA public key | W |
| Shared secret computation | Calculating share secret | KAS ECC SSC | EC Key pair | W |
| Shared secret | G |
| Key Derivation | Deriving a key | TLS KDF 1.0/1.1/1.2 | TLS pre-mast secret | CO | W |
| TLS master secret  TLS Derived key | G |
| Zeroization | Zeroize PSP in  volatile  memory | None | All above SSPs | Z |
| Self-test | Initiate power-on self-tests  by reset | AES, HMAC, SHA, DRBG, RSA, DSA, ECDSA, KAS ECC SSC, TLS KDF | N/A | N/A |
| Show Status | Show status of the module state | N/A | N/A | N/A |
| Show version | Show the version of the module using *awslc\_version\_string* | N/A | N/A | N/A |

Table 8 - Approved Services

The module provides following approved services that utilize approved and allowed security functions, and the services that utilize non-approved security functions.

| Service | Description | Algorithms Accessed | Role | Indicator |
| --- | --- | --- | --- | --- |
| Encryption and Decryption | Encryption and Decryption | AES, DES, Triple-DES  listed in Table 5 | CO | Return value 0 from the function  FIPS\_ service\_ indicator\_ check\_ approved() |
| Message Authentication | Mac computation | AES GMAC listed in Table 5 |
| Message digest | Hash computation | MD4, MD5 outside TLS 1.0 usage |
| Signature Generation | Generating signature | Using SHA-1 |
| RSA listed in Table 5 |
| Signature verification | Verifying signature | RSA listed in Table 5 |
| Key Generation | Generating key pair | RSA or ECDSA listed in Table 5 |
| Shared secret computation | Calculating share secret | Diffie-Hellman |
| Key Derivation | Deriving key | TLS KDF listed in Table 5 |
| RSA Key Wrapping | Encrypting/decrypting key | RSA |

Table 9 - Non-Approved Services

# Software/Firmware security

## Integrity Techniques

The integrity of the module is verified by comparing a value calculated at run time with the HMAC-SHA-256 value stored in the module file fips\_shared\_support.c that was computed at build time.

## On-Demand Integrity Test

Integrity tests are performed as part of the Pre-Operational Self-Tests. The module provides the Self-Test service to perform self-tests on demand. This service can be invoked by powering-off and reloading the module and it performs the same cryptographic algorithm tests executed during power-up.

# Operational Environment

## The module operates in a modifiable operational environment. The module runs on a commercially available general-purpose operating system executing on the hardware specified in section 2. The application that requests cryptographic services is the single user of the module. The module does not support concurrent operators.

# Physical Security

The module is comprised of software only and therefore this section is Not applicable.

# Non-invasive Security

The module claims no non-invasive security techniques.

# Sensitive Security Parameter Management

Table 10 summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module..

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Key/SSP Name  /Type | Strength | Security Function and Cert. Number | Generation | Import  /Export | Establish-ment | Storage | Zero-ization | Use and related keys |
| AES key | 128 to 256 bits | AES Cert XXXX | N/A | Passed into the module via API input parameters in plaintext. | Automated | RAM | OPENSSL\_cleanse | See  Table 8 |
| EVP\_AEAD\_CTX\_zero |
| HMAC key | 112 or greater | HMAC Cert XXXX | N/A | RAM | HMAC\_CTX\_cleanup |
| DRBG Entropy Input | 256 | DRBG Cert XXXX | N/A | Obtained from the ENT(NP) | N/A | RAM | CTR\_DRBG\_clear |
| DRBG Seed, V, Key | 256 | DRBG Cert XXXX | Per SP 800-90A DRBG | N/A | N/A | RAM | CTR\_DRBG\_clear |
| RSA key pair | 112 to 150 bits | RSA Cert XXXX | Per FIPS 186-4; random values generated using DRBG | passed into or out of the module via API input or output parameters in plaintext. | Automated | RAM | RSA\_free |
| ECDSA key pair | 112 to 256 bits | ECDSA Cert XXXX | RAM | EC\_GROUP\_free,  EC\_POINT\_free  EC\_KEY\_free |
| ECDH key pair | ECDH Cert XXXX | RAM |
| Shared secret | ECDH Cert XXXX | Per SP 800-56Arev3 | RAM | OpenSSL\_clenase |
| TLS pre-mast secret | 112 to 256 bits | TLS KDF Cert XXXX | N/A | Passed into the module via API input parameters in plaintext. | Automated | RAM | OPENSSL\_cleanse |
| TLS master secret | XXXX | TLS KDF Cert XXXX | Generated using SP 800-135 TLS KDF | N/A | N/A | RAM |
| TLS Derived key (AES/ HMAC) | AES: 128 to 256 bits  HMAC: 112 or greater | TLS KDF Cert XXXX | Passed out of the module via API output parameters in plaintext. | Automated | RAM |

Table 10 - SSPs

## The module uses the entropy source specified in Table 11

|  |  |  |
| --- | --- | --- |
| Entropy Source | Minimum number of bits of entropy | Details |
| NISP SP800-90B compliant ENT (NP\_ | 256 | CPU Jitter entropy source with SHA-3 as the vetted conditioning component. |

Table 11 - Non-Deterministic Random Number Generation Specification

# Self-tests

The module performs pre-operational tests automatically when the module is loaded into memory; power-up tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected. While the module is executing the power-up tests, services are not available, and input and output are inhibited.

## Pre-Operational Tests

Table 12 lists the tests performed as part of pre-operational tests. All the known answer tests are run prior to performing the integrity test.

| Algorithm | Test |
| --- | --- |
| SHA-256 | Integrity Test |
| AES | CBC encrypt/decrypt KAT using 128 bit key  GCM encrypt/decrypt KAT using 128 bit key |
| HMAC | Covered with TLS KDF |
| SHS | SHA-1, SHA-256 and SHA-512 KAT |
| DRBG | CTR\_DRBG with AES 256 |
| RSA | Sign and Verify KAT with 2048 bit key and SHA-256 |
| ECDSA | Sign and Verify KAT with P-256 curve and SHA-256 |
| KAS ECC SSC | Z computation KAT with P-256 curve |
| TLS KDF | TLS 1.2 using SHA-256 |

Table 12 - Pre-Operational Self-Tests

## Conditional 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.

### Pairwise Consistency Test

The module implements RSA and ECDSA key generation service and performs the respective pairwise consistency test using sign and verify functions, when the keys are generated.

### Periodic/On-Demand Self-Test

On demand self-tests can be invoked by powering-off and reloading the module. This service performs the same cryptographic algorithm tests executed during power-up. During the execution of the on-demand self-tests, cryptographic services are not available and no data output or input is possible.

## Error States

If the module fails any of the self-tests, the module enters the error state. In the error state, the module outputs the error through the status output interface. In the error state, the data output interface is inhibited.

| Error State | Cause of Error | Status Indicator |
| --- | --- | --- |
| Error | Pre-operational test failure | Module is aborted. |
| Conditional test failure | Module sends an error and then regenerates new key, If the PCT still does not pass, eventually the module will be aborted after 5 trials. |

Table 13 - Error States

# Life-cycle assurance

Delivery and Operation

The module bcm.o is distributed embedded into the shared library libcrypto.so which can be obtained at [TODO: insert URL] with the following checksum: [TODO: insert sha256 output (or something else?)] which can be checked by running the command:

sha256sum [TODO: insert filename]  
  
The shared library does not require installation. It only needs to reside in a path accessible to the loader. A user application would use the library by invoking any of the FIPS API functions.  
  
The module was built using the following steps:

1. Gather the following tools
   * Clang compiler version 7, clang-7 (<http://releases.llvm.org/download.html>) or GCC compiler version 7, gcc-7 (<https://gcc.gnu.org/gcc-7/>)
   * Go programming language version 1.12.7 (<https://golang.org/dl/>)
   * Ninja build system version 1.90 (<https://github.com/ninja-build/ninja/releases>)
2. Once the above tools have been obtained, issue the following command to create a CMake toolchain file to specify the use of Clang (or GCC):

printf "set(CMAKE\_C\_COMPILER \"clang-7\")\nset(CMAKE\_CXX\_COMPILER \"clang++-7\")\n" > ${HOME}/toolchain

1. Having the source code in aws-lc folder, the following commands are used to compile the module:
   1. cd boringssl
   2. mkdir build && cd build && cmake -GNinja - DCMAKE\_TOOLCHAIN\_FILE=${HOME}/toolchain -DFIPS=1 -DBUILD\_SHARED\_LIBS=1 -DCMAKE\_BUILD\_TYPE=Release ..
   3. ninja
   4. ninja run\_tests
2. Upon completion of the build process, the module’s status can be verified by the below command. If the value obtained is “1” then the module i.e. the bcm.o has been installed and configured in to operate as FIPS validated module.  
   ./tool/bssl isfips
3. Lastly, the user can call the “show version” service using awslc\_version\_string function and the output AWS-LC FIPS 1.0.0 will confirm that the module is FIPS validated.

Crypto Officer Guidance

The module offers two AES GCM implementations.

* **IG C.H scenario 2**: The GCM implementations available via *EVP\_aead\_aes\_128\_gcm\_randnonce()* OR *EVP\_aead\_aes\_256\_gcm\_randnonce()*functions make use of 96 bit IV generated internally by the module’s Approved DRBG per NIST SP 800-38D, Section 8.2.2. The DRBG used is SP 800-90A compliant and is seeded with SP 800-90B compliant entropy source.
* **IG C.H scenario 1**: The GCM implementations available via *EVP\_aead\_aes\_128\_gcm\_tls12(),*

*EVP\_aead\_aes\_128\_gcm\_tls13(), EVP\_aead\_aes\_256\_gcm\_tls12(),*

*EVP\_aead\_aes\_256\_gcm\_tls13()* are used in the context of the TLS protocol version 1.2 or 1.3. The mechanism for IV generation is compliant with RFC 5288. The module ensures that it's strictly increasing and when the IV exhausts the maximum number of possible values for a given session key, the module errors out.

Per IG C.H, in the event module power is lost and restored the consuming application must ensure that any of its AES-GCM keys used for encryption or decryption are re-distributed.

# Mitigation of other attacks

RSA is vulnerable to timing attacks. In a setup where attackers can measure the time of RSA

decryption or signature operations, blinding must be used to protect the RSA operation from

that attack.

The module provides the machoism to use the blinding for RSA. When the blinding is on, the module generates a random value to form a blinding factor in the RSA key before the RSA key is used in the RSA cryptographic

operations.

1. Glossary and Abbreviations

|  |  |
| --- | --- |
| **AES** | Advanced Encryption Standard |
| **AES-NI** | Advanced Encryption Standard New Instructions |
| **CAVP** | Cryptographic Algorithm Validation Program |
| **CBC** | Cipher Block Chaining |
| **CCM** | Counter with Cipher Block Chaining-Message Authentication Code |
| **CFB** | Cipher Feedback |
| **CMAC** | Cipher-based Message Authentication Code |
| **CMT** | Cryptographic Module Testing |
| **CMVP** | Cryptographic Module Validation Program |
| **CSP** | Critical Security Parameter |
| **CTR** | Counter Mode |
| **CVT** | Component Verification Testing |
| **DES** | Data Encryption Standard |
| **DSA** | Digital Signature Algorithm |
| **DRBG** | Deterministic Random Bit Generator |
| **ECB** | Electronic Code Book |
| **ECC** | Elliptic Curve Cryptography |
| **FIPS** | Federal Information Processing Standards Publication |
| **FSM** | Finite State Model |
| **GCM** | Galois Counter Mode |
| **HMAC** | Hash Message Authentication Code |
| **KAT** | Known Answer Test |
| **KW** | AES Key Wrap |
| **KWP** | AES Key Wrap with Padding |
| **MAC** | Message Authentication Code |
| **NIST** | National Institute of Science and Technology |
| **OFB** | Output Feedback |
| **O/S** | Operating System |
| **PAA** | Processor Algorithm Acceleration |
| **PR** | Prediction Resistance |
| **PSS** | Probabilistic Signature Scheme |
| **RNG** | Random Number Generator |
| **RSA** | Rivest, Shamir, Addleman |
| **SHA** | Secure Hash Algorithm |
| **SHS** | Secure Hash Standard |

1. References

|  |  |
| --- | --- |
| **FIPS140-3** | **FIPS PUB 140-3 - Security Requirements For Cryptographic Modules**  March 2019  https://doi.org/10.6028/NIST.FIPS.140-3 |
| **FIPS140-3\_IG** | **Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program**  September 2020  <https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements> |
| **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 2.1 February 2003 <http://www.ietf.org/rfc/rfc3447.txt> |
| **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> |
| **SP800-38B** | **NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 <http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf> |
| **SP800-38C** | **NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf> |
| **SP800-38D** | **NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**  November 2007  <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf> |
| **SP800-38E** | **NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 <http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf> |
| **SP800-38F** | **NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf> |
| **SP800-38G** | **NIST Special Publication 800-38G - Recommendation for Block Cipher Modes of Operation: Methods for Format - Preserving Encryption** March 2016 <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38G.pdf> |
| **SP800-56Ar3** | **NIST Special Publication 800-56A Revision 2 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** May 2013 <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.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> |
| **SP800-90B** | **(Second DRAFT) NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2016 <http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf> |
| **SP800-131A** | **NIST Special Publication 800-131A Revision 1- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** November 2015 <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf> |
| **SP800-133** | **NIST Special Publication 800-133 - Recommendation for Cryptographic**  **Key Generation** December 2012 <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133.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> |
| **SP800-140B** | **NIST Special Publication 800-140B - CMVP Security Policy Requirements**  March 2020  https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf |

1. “#” can be 128 or 256, that corresponds to the respective key size used for GCM. [↑](#footnote-ref-1)
2. The key establishment methodology provides between 128 and 256 bits of encryption strength [↑](#footnote-ref-2)