AWS-LC Cryptographic Module

Module Version: AWS-LC FIPS 1.0.2

FIPS 140-3 Non-Proprietary Security Policy

Document version: 1.4

Last update: 2023-09-11

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

1 Table of Contents

Copyrights and Trademarks

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

1 General

This document is the non-proprietary FIPS 140-3 Security Policy for version AWS-LC FIPS 1.0.2 of the AWS-LC Cryptographic Module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module.

Table 1 describes the individual security areas of FIPS 140-3, as well as the security levels of those individual areas.

ISO/IEC 24759 Section 6. Subsections	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	1

Table 1: Security Levels.

1.1 This Security Policy Document

This Security Policy describes the features and design of the module named AWS-LC Cryptographic Module using the terminology contained in the FIPS 140-3 specification. The FIPS 140-3 Security Requirements for Cryptographic Module specifies the security requirements that will be satisfied by a cryptographic module utilized within a security system protecting sensitive but unclassified information. The NIST/CCCS Cryptographic Module Validation Program (CMVP) validates cryptographic module to FIPS 140-3. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

The Security Policy document is one document in a FIPS 140-3 Submission Package. In addition to this document, the Submission Package contains:

- The validation report prepared by the lab.
- The Entropy Assessment Report (EAR) if applicable.
- Other supporting documentation and additional references.

This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

1.2 How this Security Policy was Prepared

The vendor has provided the non-proprietary Security Policy of the cryptographic module, which was further consolidated into this document by atsec information security together with other vendor-supplied documentation. In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

2 Cryptographic Module Specification

2.1 Module Overview, Embodiment, Type

The AWS-LC 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).

2.2 Module Design, Components, Versions

The block diagram in Figure 1 shows the cryptographic boundary of the module, its interfaces with the operational environment and the flow of information between the module and operator (depicted through the arrows).

The module components consist of the `bcm.o` file in executable form and `fips_shared_support.c` file that holds the pre-computed integrity check value. They are all of version AWS-LC FIPS 1.0.2.

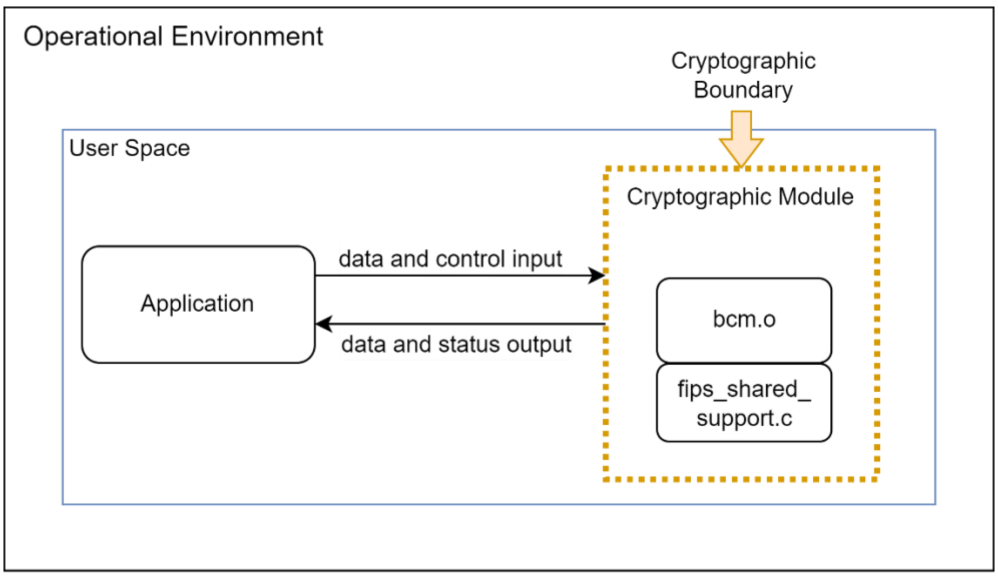

Figure 1: Block diagram depicting the cryptographic boundary and data flow between the module interfaces and operator.

2.2.1 Components Excluded from the Security Requirements

There are no components excluded from the security requirements.

2.3 Security Level

The module is validated according to FIPS 140-3 at overall security level 1. The security levels of individual areas are indicated in Table 1.

2.4 Tested Operational Environments

The module has been tested on the platforms indicated in Table 2, 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 (PAA)
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) (PAA)
4	Amazon Linux 2			

Table 2: Tested Operational Environments.

2.5 Vendor Affirmed Operational Environments

The vendor claims the platforms listed in Table 2-a to be vendor affirmed. The module functions the same way as it functions on the tested operational environments and provides the same services on the systems listed in Table 2-a.

Note: The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

#	Operating System	Hardware Platform
1	RHEL5	Amazon m4.4xlarge with Intel® Xeon® CPU E5-2686
2	Amazon Linux 2012	Amazon m4.4xlarge with Intel® Xeon® CPU E5-2686

Table 2-a: Vendor-Affirmed Operational Environments

2.6 Modes of Operation of the Module

When the module starts up successfully, after passing the pre-operational self-test and the cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default and can only be transitioned into the non-approved mode by calling one of the non-approved services listed in Table 9. Section 4 provides details on the service indicator implemented by the module. The service indicator identifies when an approved service is called.

2.7 Security Functions

2.7.1 Approved Algorithms

Table 3 lists the approved security functions (or cryptographic algorithms) of the module, including specific key lengths employed for approved services, and implemented modes or methods of operation of the algorithms.

CAVP Cert.	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use/Function
A2177 A2180 A2183 A2186 A2190 A2194	AES FIPS197, SP800-38A	CBC	128, 192, 256 bits with 128-256 bits of key strength	Encryption/Decryption *AES_cbc_encrypt*
	AES FIPS197, SP800-38C	CCM	128 bits with 128 bits of key strength	Encryption/Decryption *EVP_aead_aes_128_**
	AES FIPS197, SP800-38B	CMAC	128, 256 bits with 128 and 256 bits of key strength	Message Authentication Generation *EVP_aead_aes_**
	AES FIPS197, SP800-38A	CTR	128, 192, 256 bits with 128-256 bits of key strength	Encryption/Decryption *AES_ctr*_encrypt*
A2177 A2178 A2180 A2181 A2183 A2184 A2186 A2187 A2188 A2189 A2190 A2191 A2192 A2193 A2194 A2195 A2196 A2197		ECB	128, 192, 256 bits with 128-256 bits of key strength	Encryption/Decryption *AES_ecb_encrypt*

CAVP Cert.	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use/Function
A2178 A2181 A2184 A2187 A2188 A2189 A2191 A2192 A2193 A2195 A2196 A2197	AES FIPS197, SP800-38D	GCM with Internal IV Mode 8.2.2	128, 256 bits with 128 and 256 bits of key strength	Authenticated Encryption/Decryption *EVP_aead_aes_#[1]_gcm_randnonce*
A2178 A2181 A2184 A2187 A2188 A2189 A2191 A2192 A2193 A2195 A2196 A2197		GCM with external IV	128, 256 bits with 128 and 256 bits of key strength	Authenticated Encryption/Decryption *EVP_aead_aes_#[1]_gcm_tls12* *EVP_aead_aes_#[1]_gcm_tls13*

[1] Here, the “#” can be 128 or 256. This number corresponds to the respective key size used for GCM.

CAVP Cert.	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use/Function
A2178 A2181 A2184 A2187 A2188 A2189 A2191 A2192 A2193 A2195 A2196 A2197		GMAC	128, 192, 256 bits with 128-256 bits of key strength	Message Authentication Generation *EVP_aead_aes_**
A2177 A2180 A2183 A2186 A2190 A2194	AES FIPS197, SP800-38F	KW	128, 192, 256 bits with 128-256 bits of key strength	Key Wrapping/Unwrapping
A2177 A2180 A2183 A2186 A2190 A2194		KWP		
N/A	CKG IG D.H, SP800-133rev2 section 5.1	Vendor Affirmed	RSA: 2048, 3072, 4096 bits with 112, 128, 149 bits of key strength; EC: P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	Key Generation *RSA_generate_key_fips, EC_KEY_generate_key_fips EVP_PKEY_keygen*

CAVP Cert.	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use/Function
A2177 A2180 A2183 A2186 A2190 A2194	DRBG SP800-90Arev1	CTR_DRBG no DF, no PR	256 bit key with 256 bits of key strength	Random Number Generation
A2182 A2185 A2198 A2199 A2200	ECDSA FIPS 186-4	B.4.2 Testing Candidates	P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	Key Generation *EC_KEY_generate_key_fips or EVP_PKEY_keygen*
		N/A	P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	Key Verification
		SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	Signature Generation *EVP_DigestSign or* *EVP_DigestSignInit* *EVP_DigestSignUpdate* *EVP_DigestSignFinal*
		SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	Signature Verification *EVP_DigestVerify or* *EVP_DigestVerifyInit* *EVP_DigestVerifyUpdate* *EVP_DigestVerifyFinal*
N/A	ENT (NP) SP800-90B	CPU jitter source	N/A	Random Number Generation
A2182 A2185 A2198 A2199 A2200	HMAC FIPS198-1	HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-384, HMAC-SHA2-512	112 bits or greater with key strength of 112 bits or greater	Message Authentication Generation

CAVP Cert.	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use/Function
A2179 A2182 A2185 A2198 A2199 A2200		HMAC-SHA2-256		
A2182 A2185 A2198 A2199 A2200	KAS ECC SSC SP800-56ARev3	ECC Ephemeral Unified scheme	P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	Shared Secret Computation
CVL: A2182 A2185 A2198 A2199 A2200	KDF TLS SP800-135rev1	TLS 1.0/1.1/TLS 1.2: SHA2-256, SHA2-384, SHA2-512	N/A	Key Derivation
A2177 A2180 A2183 A2186 A2190 A2194	KTS SP800-38F	AES-KW, AES-KWP	128, 192, 256 bits with 128-256 bits of key strength	Key Wrapping/Unwrapping
A2182 A2185 A2198 A2199 A2200	RSA FIPS 186-4	B.3.3 Random Probable Primes	2048, 3072, 4096 bits with 112, 128, 149 bits of key strength	Key Generation *RSA_generate_key_fips or EVP_PKEY_keygen*
		PKCS#1v1.5 and PSS with SHA2-224, SHA2-256, SHA2-384, SHA2-512	2048, 3072, 4096 bits with 112, 128, 149 bits of key strength	Signature Generation *EVP_DigestSign or* *EVP_DigestSignInit* *EVP_DigestSignUpdate* *EVP_DigestSignFinal*

CAVP Cert.	Algorithm and Standard	Mode/Method	Description/Key Size(s)/Key Strength(s)	Use/Function
		PKCS#1v1.5 and PSS with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	2048, 3072, 4096 bits with 112, 128, 149 bits of key strength	Signature Verification *EVP_DigestVerify or* *EVP_DigestVerifyInit* *EVP_DigestVerifyUpdate* *EVP_DigestVerifyFinal*
A2182 A2185 A2198 A2199 A2200	SHA FIPS180-4	SHA-1, SHA2-224, SHA2-384, SHA2-512, SHA2-512/256	N/A	Message Digest
A2179 A2182 A2185 A2198 A2199 A2200		SHA2-256		

Table 3: Approved Algorithms.

Note: no parts of the TLS v1.0/1.1, v1.2 protocols, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.

2.7.2 Non-Approved Algorithms Allowed in the Approved Mode of Operation

The module does not implement non-approved algorithms that are allowed in the approved mode of operation.

2.7.3 Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed

Table 4 lists the non-approved algorithms that are allowed in the approved mode of operation with no security claimed. These algorithms are used by the approved services listed in Table 8.

Algorithm	Caveat	Use/Function
MD5	Allowed per IG 2.4.A	Message Digest used in TLS 1.0/1.1 KDF only

Table 4: Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed.

2.7.4 Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

Table 5 lists non-approved algorithms that are not allowed in the approved mode of operation. These algorithms are used by the non-approved services listed in Table 9.

Algorithm/Functions	Use/Function
DES	Encryption/Decryption
Triple-DES	Encryption/Decryption
AES with OFB or CFB modes	Encryption/Decryption
AES-GCM with 192 bits	Encryption/Decryption
AES using *aes_*_generic* function	Encryption/Decryption
AES GMAC using *aes_*_generic*	Message Authentication Generation
Diffie Hellman	Shared Secret Computation
MD4	Message Digest
MD5	Message Digest (outside of TLS)
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
TLS KDF using any SHA algorithms not listed in Table 3	Key Derivation

Table 5: Non-Approved Algorithms, Not Allowed in the Approved Mode of Operation.

2.8 Rules of Operation

The module initializes upon power-on. After the pre-operational self-tests are successfully concluded, the module automatically transitions to the operational state. In this state, the module awaits services requests from the operator.

3 Cryptographic Module Interfaces

As a Software module, the module interfaces are defined as Software or Firmware Module Interfaces (SMFI), and there are no physical ports. The interfaces are mapped to the API provided by the module, through which the operator can interact. The interfaces are listed in Table 6.

All data output via data output interface is inhibited when the module is performing pre-operational test or zeroization or when the module enters error state.

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.

Table 6: Ports and Interfaces. [2]

[2] The control output interface is omitted on purpose because the module does not implement it. The physical ports are not applicable because the module is software only.

4 Roles, Services, and Authentication

4.1 Roles

The module supports the Crypto Officer role only. This sole role is implicitly assumed by the operator of the module when performing a service.

Table 7 lists the roles supported by the module with corresponding services with input and output.

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 Unwrapping	Key unwrapping key, key to be unwrapped	Unwrapped key
	Key Wrapping	Key wrapping key, key to be wrapped	Wrapped key
	Message Authentication Generation	Message, HMAC key, AES key	Message authentication code
	Message Digest	Message	Digest of the message
	Random Number Generation	Number of bits	Random numbers
	Key Generation	Key size	Key pair
	Key Verification	Key to verify	Return codes and/or log messages
	Signature Generation	Message, hash algorithm, private key	Signature
	Signature Verification	Signature, hash algorithm, public key	Verification result
	Shared Secret Computation	Private key, public key from peer	Shared secret
	Key Derivation	PRF algorithm, TLS pre-master secret, TLS master secret	Derived keys
	Zeroization	Context containing SSPs	none
	On-Demand Self-test	Module reset	Result of self-test (pass/fail)
	On-Demand Integrity Test	None	Result of test (pass/fail)

	Show Status	None	Return codes and/or log messages
	Show Version	None	Name and version information

Table 7: Roles, Service Commands, Input and Output.

4.2 Authentication

The module does not support authentication.

4.3 Services

The module provides services to operators who assume the available role. All services are described in detail in the developer documentation.

The next subsections define the services that utilize approved and allowed security functions, and the services that utilize non-approved security functions in this module. For the respective tables, 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 = Zeroize: The module zeroizes the SSP.

For the role, CO indicates "Crypto Officer".

4.3.1 Service Indicator

The module implements a service indicator that indicates whether the invoked service is approved. The 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 i.e., API function for performing a service.

```
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);
```

Alternatively, all the above steps can be done by using a single call using the function `CALL_SERVICE_AND_CHECK_APPROVED(approved, func)`.

4.3.2 Approved Services

Table 8 lists the approved services that utilize approved and allowed security functions.

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to SSPs	Indicator
Encryption	Encryption	AES CBC, CTR, ECB listed in Table 3	AES Key	CO	W, E	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	AES Key	CO	W, E	
Authenticated Decryption	Authenticated Decryption					
Key wrapping	Encrypting a key	AES KW, KWP	AES key	CO	W, E	
Key unwrapping	Decrypting a key	AES KW, KWP	AES key	CO	W, E	
Message Authentication Generation	MAC computation	AES CMAC	AES Key HMAC Key	CO	W, E	
		AES GMAC				
		HMAC				
Message Digest	Generating message digest	SHA	N/A	CO	N/A	
Random Number Generation	Generating random numbers	CTR_DRBG, ENT (NP)	Entropy Input	CO	W, E	
			DRBG Seed, V, Key	CO	G, E	
Key Generation	Generating key pair	RSA listed in Table 3, CKG	RSA key pair	CO	W, E, G	
		ECDSA listed in Table 3, CKG	ECDSA key pair	CO		
Key Verification	Verifying the public key	ECDSA listed in Table 3	ECDSA Public key	CO	W, E	
Signature Generation	Generating signature	RSA, ECDSA listed in Table 3	RSA/ECDSA private key	CO	W, E	
Signature Verification	Verifying signature	RSA, ECDSA listed in Table 3	RSA/ECDSA public key	CO	W, E	
Shared Secret Computation	Calculating shared secret	KAS-ECC-SSC	EC key pair	CO	W, E	
			Shared secret	CO	G	
Key Derivation	Deriving TLS keys	TLS KDF 1.0/1.1/1.2	TLS pre-master secret	CO	W, E	
			TLS master secret	CO	W, E, G	
			TLS derived keys		G	
Zeroization	Zeroize PSP in volatile memory	None	All SSPs	CO	Z	

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to SSPs	Indicator
On-Demand Self-test	Initiate power-on self-tests by reset	AES, HMAC, SHA, DRBG, RSA, ECDSA, KAS ECC SSC, TLS KDF	N/A[3]	CO	N/A	
On-Demand Integrity Test	Initiate integrity test on-demand	HMAC-SHA2-256	N/A[3]	CO	N/A	
Show Status	Show status of the module state	N/A	N/A	CO	N/A	
Show Version	Show the version of the module using *awslc_version_string*	N/A	N/A	CO	N/A	

Table 8: Approved Services.

4.3.3 Non-Approved Services

Table 9 lists the non-approved services that utilize non-approved security functions.

Service	Description	Algorithms Accessed	Role	Indicator
Encryption	Encryption	AES, DES, Triple-DES listed in Table 5	CO	Return value 0 from the function `FIPS_service_indicator_check_approved()`
Decryption	Decryption			
Message Authentication Generation	MAC computation	AES GMAC listed in Table 5		
Message Digest	Generating message digest	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 shared secret	Diffie-Hellman		
Key Derivation	Deriving TLS keys	TLS KDF listed in Table 5		
Key Unwrapping	Decrypting a key	RSA		
Key Wrapping	Encrypting a key	RSA		

Table 9: Non-Approved Services.

[3] Keys for self-tests are not SSPs.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing a HMAC value calculated at run time on the bcm.o file, with the HMAC-SHA2-256 value stored in the module file `fips_shared_support.c` that was computed at build time.

5.2 On-Demand Integrity Test

The module provides on-demand integrity test. The integrity test is performed by the On-Demand Integrity Test service, which calls the `BORINGSSL_integrity_test` function. The integrity test is also performed as part of the Pre-Operational Self-Tests.

5.3 Executable Code

The module consists of executable code in the form of bcm.o file. The compilers and control parameters required to compile the code into an executable format are specified in Section 11.

6 Operational Environment

6.1 Applicability

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 module does not support concurrent operators. The module does not support software/firmware loading.

6.2 Requirements

The module should be compiled and installed as stated in section 11. The user should confirm that the module is installed correctly by following steps 4 and 5 listed in section 11.1.

6.3 Vendor Affirmation

The vendor claims the platforms listed in Table 2-a to be vendor affirmed, and the module functions the same way as it functions on the tested operational environments.

7 Physical Security

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

8 Non-Invasive Security

The module claims no non-invasive security techniques.

9 Sensitive Security Parameter Management

Table 10 summarizes the SSPs that are used by the cryptographic services implemented in the module.

Key/SSP Name/**Type**	**Strength**	**Security Function and Cert. Number[4]**	**Generation**	Import **/Export**	**Establishment**	**Storage**	**Zeroization**	**Use and related keys**
AES key	128 to 256 bits	A2177 A2178 A2180 A2181 A2183 A2184 A2186 A2187 A2188 A2189 A2190 A2191 A2192 A2193 A2194 A2195 A2196 A2197	N/A	**Import:** CM from TOEPP Path. Passed into the module via API input parameter in plaintext (P) format.	MD/EE	RAM	`OPENSSL_cleanse`, `EVP_AEAD_CTX_zero`	**Use:** Encryption, Decryption, Authenticated Encryption, Authenticated Decryption, Key wrapping, Key unwrapping, Message Authentication Generation
HMAC key	112 or greater	A2179 A2182 A2185 A2198 A2199 A2200	N/A		MD/EE	RAM	`HMAC_CTX_cleanup`	**Use:** Message Authentication Generation
DRBG Entropy Input	256	A2177 A2180 A2183 A2186 A2190 A2194	N/A	Obtained from the ENT (NP)	N/A	RAM	`CTR_DRBG_clear`	**Use:** Random Number Generation **Related SSPs:** DRBG Seed, V, Key
DRBG Seed, V, Key	256	A2177 A2180	Per SP800-90Arev1 DRBG	N/A	N/A	RAM	`CTR_DRBG_clear`	**Use:** Random

[4] See for the algorithm certificate numbers of each algorithm listed in this column.

Key/SSP Name/**Type**	**Strength**	**Security Function and Cert. Number**[4]	**Generation**	Import **/Export**	**Establishment**	**Storage**	**Zeroization**	**Use and related keys**
		A2183 A2186 A2190 A2194						Number Generation **Related SSPs:** DRBG Entropy Input
RSA key pair	112 to 150 bits	A2182 A2185 A2198 A2199 A2200	Per FIPS 186-4; random values generated using DRBG	**Import/Export:** CM to/from TOEPP Path. Passed into or out of the module via API input or output parameters in plaintext (P) format.	MD/EE	RAM	`RSA_free`	**Use:** Key Generation, Signature Generation, Signature Verification
ECDSA key pair	112 to 256 bits	A2182 A2185 A2198 A2199 A2200				RAM	`EC_GROUP_free`, `EC_POINT_free`, `EC_KEY_free`	**Use:** Key Generation, Key Verification, Signature Generation, Signature Verification
ECDH key pair		A2182 A2185 A2198 A2199 A2200				RAM		**Use:** Shared Secret Computation **Related SSPs:** Shared Secret
Shared secret		A2182 A2185 A2198 A2199 A2200	Per SP800-56Arev3			RAM	`OpenSSL_cleanse`	**Use:** Shared Secret Computation **Related SSPs:** ECDH key pair
TLS pre-master secret	112 to 256 bits	A2182 A2185 A2198 A2199 A2200	N/A	**Import:** CM to TOEPP Path. Passed into the module via API input parameters in plaintext (P) format.	MD/EE	RAM	`OPENSSL_cleanse`	**Use:** Key Derivation **Related SSPs:** TLS master secret
TLS master secret	384 bits	A2182 A2185 A2198 A2199 A2200	Generated using SP800-135rev1 TLS KDF	N/A	N/A	RAM		**Use:** Key Derivation **Related SSPs:** TLS pre-master secret

Key/SSP Name/**Type**	**Strength**	**Security Function and Cert. Number**[4]	**Generation**	Import **/Export**	**Establishment**	**Storage**	**Zeroization**	**Use and related keys**
TLS Derived key (AES/HMAC)	AES: 128 to 256 bits HMAC: 112 or greater	A2182 A2185 A2198 A2199 A2200		**Export:** CM from TOEPP Path. Passed out of the module via API output parameters in plaintext (P) format.	MD/EE	RAM		**Use:** Key Derivation **Related SSPs:** TLS master secret

Table 10: SSPs.

9.1 Random Bit Generator

The module provides an SP800-90Arev1-compliant Deterministic Random Bit Generator (DRBG) using CTR_DRBG mechanism with AES-256 for creation of key components of asymmetric keys, and random number generation. The module uses the entropy source specified in Table 11. This entropy source is located within the physical perimeter, but outside of the cryptographic boundary of the module.

Entropy Source	Minimum number of bits of entropy	Details
SP800-90B compliant ENT (NP)	256 bits of entropy in the 256-bit output	CPU Jitter entropy source with SHA-3 as the vetted conditioning component is located within the physical perimeter of the operational environment but outside the software module cryptographic boundary.

Table 11: Non-Deterministic Random Number Generation Specification.

9.2 SSP Generation

For generating RSA, ECDSA and EC Diffie-Hellman keys, the module implements asymmetric key generation services compliant with FIPS186-4 and using a DRBG compliant with SP800-90Arev1. The random value used in asymmetric key generation is obtained from the DRBG. In accordance with FIPS 140-3 IG D.H, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per section 5.1 of SP800-133rev2 (vendor affirmed) by obtaining a random bit string directly from an approved DRBG and that can support the required security strength requested by the caller (without any V, as described in Additional Comments 2 of IG D.H).

The module does not provide a dedicated service for generating symmetric key. However, symmetric keys can be derived using SP800-135rev1 for TLS KDF algorithm. This generation method maps to section 6.2 of SP800-133rev2.

9.3 SSP Entry and Output

The module does not support manual SSP entry or intermediate key generation output. The module does not support entry and output of SSPs beyond the physical perimeter of the operational environment. The SSPs are provided to the module via API input parameters in the plaintext form and output via API output parameters in the plaintext form to and from the calling application running on the same operational environment.

The output of plaintext CSPs requires two independent internal actions. Specially, the first action is creation of the cipher context to request the service and to hold the CSPs to be output from the module. The second action is to process the ‘Key Generation’ service request using the context created. Only after successful completion of this request, the generated CSP is output via the API output parameter.

9.4 SSP Establishment

The module provides EC Diffie-Hellman shared secret computation compliant with SP800-56Arev3, in accordance with scenario 2 (1) of IG D.F.

The module provides SP800-38F approved key transport methods according to IG D.G. The key transport method is provided using an AES-KW or AES-KWP key wrapping algorithm.

According to “Table 2: Comparable strengths” in SP800-57, the key sizes of AES and EC Diffie-Hellman provide the following security strengths:

- EC Diffie-Hellman shared secret computation provides between 112 and 256 bits of encryption strength.
- AES key wrapping provides between 128 and 256 bits of encryption strength.

Additionally, the module also supports key derivation using TLS 1.2 KDF compliant to SP800-135rev1.

9.5 SSP Storage

SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls. The module does not perform persistent storage of SSPs.

9.6 Zeroization

The zeroization is performed by the module overwriting zeroes or predefined values to the memory location occupied by the SSP and further deallocating that area. The calling application interacting with the module, is responsible for calling the appropriate destruction functions using the zeroization APIs listed in Table 10. The completion of a zeroization routine will indicate that a zeroization procedure succeeded.

10 Self-Tests

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

All the self-tests are listed in Table 12, with the respective condition under which those tests are performed. The software integrity test is performed after all conditional algorithm self-tests (CASTs) are performed.

The entropy source performs its required self-tests; those are not listed here, as the entropy source is not part of the cryptographic boundary of the module.

Algorithm	Parameters	Condition for Test	Type	Test
HMAC-SHA2-256	HMAC key	Software integrity test on power up (load)	Pre-Operational Self-Test	MAC verification on software component
AES	128-bit AES key	Power up	Conditional Algorithm Self-Test	Encrypt KAT for CBC
				Decrypt KAT for CBC
				Encrypt KAT for GCM
				Decrypt KAT for GCM
SHS	None	Power up	Conditional Algorithm Self-Test	SHA-1, SHA2-256 and SHA2-512 KAT
DRBG	AES 256	Power up	Conditional Algorithm Self-Test	CTR_DRBG KAT AES 256
DRBG	N/A	Power up	Conditional Algorithm Self-Test	SP800-90Ar1 Section 11.3 Health Test
ECDSA	P-256 curve and SHA2-256	Power up	Conditional Algorithm Self-Test	Sign KAT
				Verify KAT
ECDSA	Respective curve and SHA2-256	Key generation	Conditional Pairwise consistency Test	Sign and verify PCT
KAS ECC SSC	P-256 curve	Power up	Conditional Algorithm Self-Test	Z computation
TLS KDF	SHA2-256	Power up	Conditional Algorithm Self-Test	TLS 1.2 KAT
RSA	2048 bit key and SHA2-256	Power up	Conditional Algorithm Self-Test	Sign KAT
				Verify KAT

Algorithm	Parameters	Condition for Test	Type	Test
RSA	SHA2-256 and respective keys	Key generation	Conditional Pairwise Consistency Test	Sign and verify PCT

Table 12: Self-Tests.

10.1 Pre-Operational Self-Tests

The module transitions to the operational state only after the pre-operational self-test is passed successfully. The pre-operational self-test is executed automatically after the automatic execution of the cryptographic algorithm self-tests.

The types of pre-operational self-tests are described in the next sub-section.

10.1.1 Pre-Operational Software Integrity Test

The integrity of the software component of the module is verified according to Section 5, using HMAC-SHA2-256. If the comparison verification fails, the module transitions to the error state (Section 10.4). The CAST for the integrity algorithm is performed before the integrity test itself.

10.1.2 Pre-Operational Bypass and Critical Functions Tests

The module does not implement pre-operational bypass or critical functions tests.

10.2 Conditional Self-Tests

10.2.1 Cryptographic Algorithm Self-Tests

The module performs self-tests on approved cryptographic algorithms supported in the approved mode of operation, using the tests shown in Table 12 (and indicated as CASTs) and using the provision of IG 10.3.A and IG 10.3.B for optimization of the number of self-tests. Data output through the data output interface is inhibited during the self-tests. The cryptographic algorithm self-tests are performed in the form of Known Answer Tests (KATs), in which the calculated output is compared with the expected known answer (that are hard-coded in the module). A failed match causes a failure of the self-test.

If any of these self-tests fails, the module transitions to error state and is aborted.

10.2.2 Conditional Pairwise Consistency Tests

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 (Table 12).

10.3 Periodic/On-Demand Self-Tests

On demand self-tests can be invoked by powering-off and reloading the module. This service performs the same pre-operational test that includes integrity test and cryptographic algorithm tests executed during power-up. The integrity test can also be performed on demand by calling the `BORINGSSL_integrity_test` function. During the execution of the on-demand self-tests, cryptographic services are not available, and no data output or input is possible.

10.4 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 and the abort function is called that raises the SIGABRT signal, causing the program termination such that module is no longer operational. In the error state, as the module is no longer operational the data output interface is inhibited. In order to recover from the Error state, the module needs to be rebooted.

Error State	Error Condition	Status Indicator
Error	Pre-operational test failure	Error message is output on the stderr and then the module is aborted.
	Conditional test failure	Error message is output in the error queue and then the module generates new key, If the PCT still does not pass, eventually the module will be aborted after 5 tries.

Table 13: Error States.

11 Life-Cycle Assurance

11.1 Delivery and Operation

The module bcm.o is distributed embedded into the shared library libcrypto.so which can be obtained building the source code at the following location. The set of files specified in the archive constitutes the complete set of source files of the validated module. There shall be no additions, deletions, or alterations of this set as used during module build.

https://github.com/awslabs/aws-lc/archive/refs/tags/AWS-LC-FIPS-1.0.2.zip

The downloaded zip file can be verified by issuing the "`sha256sum aws-lc-FIPS-1.0.2.zip`" command. The expected SHA2-256 digest value is:

dbd5fe8677a117c1b272a8b17b620177cac03355282adc25002263f0f9cc7cce

After the zip file is extracted, the instructions listed below will compile the module. The compilation instructions must be executed separately on platforms that have different processors and/or operating systems. Due to four possible combinations of OS/processor, the module count is four i.e., there are four separate binaries generated, one for each entry listed in Table 2.

1. Gather the following tools

   - 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 GCC:

```
printf "set(CMAKE_C_COMPILER \"gcc-7\")\nset(CMAKE_CXX_COMPILER \"g++-7\")\n" >
${HOME}/toolchain
```

3. Having the source code in the aws-lc-FIPS-1.0.2 folder, the following commands are used to compile the module:
   a. `cd aws-lc-FIPS-1.0.2`
   b. `mkdir build && cd build && cmake -GNinja -DCMAKE_TOOLCHAIN_FILE=${HOME}/toolchain -DFIPS=1 -DBUILD_SHARED_LIBS=1 -DCMAKE_BUILD_TYPE=Release ..`
   c. `ninja`
   d. `ninja run_tests`

4. Upon completion of the build process, the module's status can be verified by the command below. If the value obtained is "1" then the module i.e. the bcm.o has been installed and configured in to operate in approved mode.

```
./tool/bssl isfips
```

5. Lastly, the user can call the "show version" service using `awslc_version_string` function and the expected output is "AWS-LC FIPS 1.0.2" which is the module version. Additionally, the "AWS-LC FIPS" also acts as the module identifier. This will confirm that the module is in approved mode.

11.2 Crypto Officer Guidance

11.2.1 AES-GCM IV Generation

The module offers three AES GCM implementations. The GCM IV generation for these implementations complies respectively with IG C.H under Scenario 1, Scenario 2, and Scenario 5. The GCM shall only be used in the context of the AES-GCM encryption executing under each scenario, and using the referenced APIs explained next.

11.2.1.1 Scenario 1, TLS 1.2

For TLS 1.2, the module offers the GCM implementation via the functions *EVP_aead_aes_128_gcm_tls12()* and *EVP_aead_aes_256_gcm_tls12()*, and uses the context of Scenario 1 of IG C.H. The module is compliant with SP800-52rev2 and the mechanism for IV generation is compliant with RFC5288. The module supports acceptable AES-GCM ciphersuites from Section 3.3.1 of SP800-52rev2.

The module explicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values of 2^{64}-1 for a given session key. If this exhaustion condition is observed, the module returns an error indication to the calling application, which will then need to either abort the connection, or trigger a handshake to establish a new encryption key.

In the event the module's power is lost and restored, the consuming application must ensure that a new key for use with the AES-GCM key encryption or decryption under this scenario shall be established.

11.2.1.2 Scenario 2, Random IV

In this implementation, the module offers the interfaces *EVP_aead_aes_128_gcm_randnonce()* and *EVP_aead_aes_256_gcm_randnonce()* for compliance with Scenario 2 of IG C.H and SP800-38D Section 8.2.2. The AES-GCM IV is generated randomly internal to the module using module's approved DRGB. The DRBG seeds itself from the entropy source. The GCM IV is 96 bits in length. Per Section 9, this 96-bit IV contains 96 bits of entropy.

11.2.1.3 Scenario 5, TLS 1.3

For TLS 1.3, the module offers the AES-GCM implementation via the functions *EVP_aead_aes_128_gcm_tls13()* and *EVP_aead_aes_256_gcm_tls13()*, and uses the context of Scenario 5 of IG C.H. The protocol that provides this compliance is TLS 1.3, defined in RFC8446 of August 2018, using the ciphersuites that explicitly select AES-GCM as the encryption/decryption cipher (Appendix B.4 of RFC8446). The module supports acceptable AES-GCM ciphersuites from Section 3.3.1 of SP800-52rev2.

The module implements, within its boundary, an IV generation unit for TLS 1.3 that keeps control of the 64-bit counter value within the AES-GCM IV. If the exhaustion condition is observed, the module will return an error indication to the calling application, who will then need to either trigger a re-key of the session (i.e., a new key for AES-GCM), or terminate the connection.

In the event the module's power is lost and restored, the consuming application must ensure that new AES-GCM keys encryption or decryption under this scenario are established. TLS 1.3 provides session resumption, but the resumption procedure derives new AES-GCM encryption keys.

11.3 End of Life Procedure

When the module is at end of life, for the GitHub repo, the README will be modified to mark the library as deprecated. After a 6-month window, more restrictive branch permissions will be added such that only administrators can read from the FIPS branch.

The module does not possess persistent storage of SSPs. The SSP value only exists in volatile memory and that value vanishes when the module is powered off. So as a first step for the secure sanitization, the module needs to be powered off. Then for actual deprecation, the module will be upgraded to newer version that is approved. This upgrade process will uninstall/remove the old/terminated and provide a new replacement.

12 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 mechanism 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.

13 Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CAST	Cryptographic Algorithm Self-Test
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
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

14 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 **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

SP800-38F	**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
SP800-90Ar1	**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
SP800-131Ar1	**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
SP800-133rev2	**NIST Special Publication 800-133rev2 - Recommendation for Cryptographic** **Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP800-135r1	**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