galois

LG OpenSSL Cryptographic Module
FIPS 140-2 Non-Proprietary
Security Policy

Version: 5.0
Date: December 13, 2016

CHANGE RECORD

Revision	Date	Author	Description of Change
1	January 7th, 2015	Adam Wick	Initial Revision
2	February 16th, 2016	Adam Wick	2016 Updates
3	August 11th, 2016	Adam Wick	Updates per CMVP Comments
4	November 30th, 2016	Adam Wick	Updates per CMVP Comments
5	December 13th, 2016	Adam Wick	Updates per CMVP Comments

Table of Contents

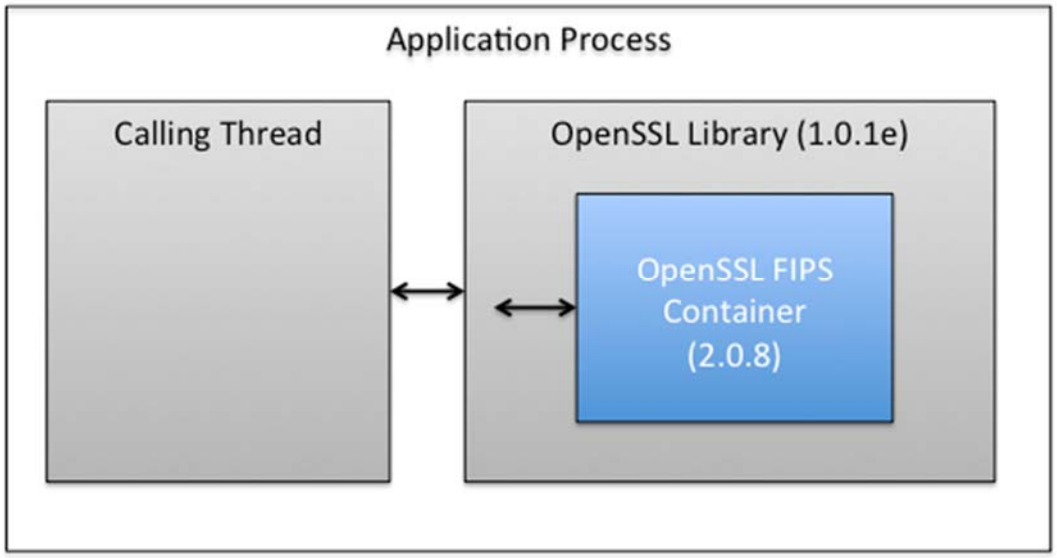
1. Module Description

This document is the non-proprietary security policy for the LG OpenSSL Cryptographic Module, hereafter referred to as the module.

The module is a software library providing a C-language application program interface (API) for use by other processes that require cryptographic functionality. The module is classified by FIPS 140-2 as a software module, multi-chip standalone module embodiment. The physical cryptographic boundary is the general purpose computer on which the module is installed. The logical cryptographic boundary of the module is the `fipscanister` object module, a single object module file named `fipscanister.o`.

The module performs no communications other than with the calling application (the process that invokes the module services). Please refer to Figure 1.

Figure 1: Logical Block Diagram



The FIPS 140-2 security levels for the module are as follows:

Table 1: Security Levels per FIPS 140-2 Area

Security Requirement Area	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1

Security Requirement Area	Security Level
EMI/EMC	1
Self Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A
Overall	**1**

The module has been tested in the following configuration:

Table 2: Tested Configuration

Module Software Version	Operating System	Processor	Devices
2.0.8	Android 5.0.1	Qualcomm Snapdragon 800 (32-bit and 64-bit)	LG G3 (Model VS985) LG G Flex 2 (Model LGLS996)

2. Ports and Interfaces

The physical ports of the module are the same as the computer system on which it is executing. The logical interface is a C-language application program interface (API).

Table 3: Logical Interfaces

Logical Interface Type	Description
Control Input	API entry point and corresponding stack parameters.
Data Input	API entry point's data input stack parameters.
Status Output	API entry point return values and status stack parameters.
Data Output	API entry point's data output stack parameters.

As a software module, control of the physical ports is outside module scope. However, when the module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The module is single-threaded and in error scenarios returns only an error value (no data output is returned).

3. Modes of Operation and Cryptographic Functionality

The module is designed to support both a FIPS approved mode of operation and a Non-FIPS approved mode of operation. During operation, the module can switch service by service between an Approved mode of operation and a non-Approved mode of operation, depending on the service being performed. Table 4 lists the approved algorithms. The module supports only NIST-defined curves for use with ECDSA and ECC CDH. Table 5 lists the non-approved but

allowed algorithms provided by the module. Table 6 lists the non-approved services implemented by the module. These algorithms shall not be used when operating in the FIPS Approved mode of operation.

Power-on self tests for the module (including the integrity check) are executed by the operating system loader at library load time, in a fashion identical to the one described in IG 9.10. The relevant self-test functionality is marked as a default entry point (DEP), and is invoked by the operating system before the execution of the application. Any failures that occur during execution of the self-test result in the module refusing to operate in FIPS Approved mode (see below).

Table 4: Approved algorithms implemented by the module. Note that the integrity check and self tests are performed at module load time, regardless of whether or not FIPS_mode_set() is ever invoked.

Function	Algorithm	Options	Cert
Random number generation	DRBG	Hash: SHA-1, 224, 256, 384, 512 HMAC: SHA-1, 224, 256, 384, 512 CTR: AES-128, AES-192, AES-256 with and without derivation function Prediction resistance supported for all variations	749
Encryption, decryption, and CMAC	Triple-DES	3-key ECB, CBC, CFB1, CFB8, CFB64, and OFB CMAC generate and verify (3-key)	1876
	AES	ECB: 256-bit; decrypt only CBC, CFB1, CFB8, CFB128, OFB: 128/192/256-bit; encrypt and decrypt CTR: 128/192/256-bit XTS: 128 and 256-bit; encrypt and decrypt* CCM: 128/192/256-bit GCM: 128/192/256-bit; encrypt and decrypt CMAC generate and verify: 128/192/256-bit * XTS is approved only for storage applications	3291
Message digest	SHA	1, 224, 256, 384, 512	2730
Keyed hash	HMAC	HMAC-SHA-1, 224, 256, 384, 512	2089
Asymmetric key generation and digital signature generation and verification	RSA	Key Gen (2048/3072) Sig GenPKCS1.5 (2048/3072 with all SHA-2 sizes) Sig VerPKCS1.5 (1024/2048/3072 with SHA-1 and all SHA-2 sizes)	1684
	DSA	Key Pair Gen (2048/3072) *using Approved DRBG* Sig Gen (2048/3072 with all SHA-2 sizes) *using Approved DRBG* Sig Ver (1024/2048/3072 with SHA-1 and all SHA-2 sizes)	944

Function	Algorithm	Options	Cert
	ECDSA	PKG: P-224, P-256, P-384, P-521 PKV: All P curves Sig Gen: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571; all SHA-2 sizes Sig Ver: P-192, P-224, P-256, P-384, P-521; SHA-1 and all SHA-2 sizes	638
Key agreement primitive	ECC CDH	All NIST defined B, K and P curves except sizes 163 and 192	468 (CVL)

The module is a cryptographic engine library, which can be used only in conjunction with additional software. Aside from the use of the NIST defined elliptic curves as trusted third party domain parameters, all other assurances are outside the scope of the module, and are the responsibility of the calling process.

Table 5: Unapproved algorithms allowed in the Approved mode.

Category	Algorithm	Description
Key agreement	ECC DH	Non-compliant (untested) DH scheme using elliptic curves, supporting all NIST defined B, K, and P curves except sizes 163 and 192 (provides between 112 and 256 bits of strength). Key agreement is a service provided for calling process use, but is not used to establish keys into the module.
Key encryption and decryption	RSA	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for compliance with SP800-56B, and no CSPs are established into or exported out of the module using these services. Uses 2048 or 3072 bit key size (provides 112 or 128 bits of encryption strength).

Table 6: Non-Approved Algorithms (Non-Approved Mode Only)

Category	Algorithm	Description
Random Number Generation	RNG	Non-compliant X9.31 generator, based on AES 128
Digital signature and asymmetric key generation	RSA	Non-compliant GenKey9.31, SigGen9.31, SigGen-PKCS1.5, SigGenPSS (1024/1536/4096 and larger key sizes with all SHA sizes, 2048/3072 with SHA-1)
	DSA	Non-compliant PQG Gen (not tested) Non-compliant Key Pair Gen, Sig Gen when using: – ANSI X9.31 RNG – 1024-bit key size – 2048 or 3072-bit key size with SHA-1

Category	Algorithm	Description
	ECDSA	Non-compliant PKG (curves P-192, K-163, B-163), SigGen (curves P-192 (SHA-1, 224, 256, 384, 512), P-224 (SHA-1), P-256 (SHA- 1), P-384 (SHA-1), P-521 (SHA-1), K-163 (SHA-1, 224, 256, 384, 512), K-233 (SHA-1), K-283 (SHA-1), K-409 (SHA-1), K-571 (SHA-1), B-163 (SHA-1, 224, 256, 384, 512), B-233 (SHA-1), B-283 (SHA-1), B-409 (SHA-1), B-571 (SHA-1)
Key agreement primitive	ECC CDH	Non-compliant B, K and P curves sizes 163 and 192
Key encryption and decryption	RSA	Used by the calling application for encryption or decryption of keys. No CSPs are established into or exported out of the module using these services. Uses 1024 bit key size.

3.1. Critical Security Parameters and Public Keys

All CSPs used by the module are described in this section. All access to these CSPs by module services are described in Section 3. The CSP names are generic, corresponding to API parameter data structures.

Table 7: CSPs

CSP Name	Description
RSA SGK	RSA (2048 and 3072 bits) signature generation key
RSA KDK	RSA (2048 and 3072 bits) key decryption (private key transport) key
DSA SGK	DSA (2048 / 3072) signature generation key
ECDSA SGK	ECDSA (P-224, P-256, P-384, P-521; K-233, K-283, K-409, K-571) signature generation key
EC DH Private	EC DH (P-224, P-256, P-384, P-521) private key agreement key
AES EDK	AES (128 / 192 / 256) encrypt / decrypt key
AES CMAC	AES (128 / 192 / 256) CMAC generate / verify key
AES GCM	AES (128 / 192 / 256) encrypt / decrypt / generate / verify key
AES XTS	AES (256 / 512) XTS encrypt / decrypt key
Triple-DES EDK	Triple-DES (3-Key) encrypt / decrypt key
Triple-DES CMAC	Triple-DES (3-Key) CMAC generate / verify key
HMAC Key	Keyed hash key (160 / 224 / 256 / 384 / 512)
Hash_DRBG CSPs	V (440 / 888 bits) and C (440 / 888 bits), entropy input (length dependent on security strength)
HMAC_DRBG CSPs	V(160/224/256/384/512bits) and Key(160/ 224 / 256 / 384 / 512 bits), entropy input (length dependent on security strength)

CSP Name	Description
CTR_DRBG CSPs	V (128 bits) and Key (AES 128 / 192 / 256), entropy input (length dependent on security strength)

All public keys are described in the following table.

Table 8: Public Keys

Key Name	Description
RSA SVK	RSA (1024, 2048 and 3072 bits) signature verification public key
RSA KEK	RSA (2048 and 3072 bits) key encryption (public key transport) key
DSA SVK	DSA (1024 / 2048 / 3072) signature verification key
ECDSA SVK	ECDSA (P-256 / P-384 / P-521) signature verification key
EC DH Public	EC DH (P-256 / P-384 / P-521) public key agreement key

For all CSPs and Public Keys:

Storage RAM, associated to entities by memory location. The module stores DRBG state values for the lifetime of the DRBG instance. The module uses CSPs passed in by the calling application on the stack. The module does not store any CSP persistently (beyond the lifetime of an API call), with the exception of DRBG state values used for the module's default key generation service.

Generation The module implements SP 800-90A compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys as shown in the table of CSPs above. The calling application is responsible for storage of generated keys returned by the module.

Entry All CSPs enter the module's logical boundary in plaintext as API parameters, associated by memory location. However, none cross the physical boundary.

Output The module does not output CSPs, other than as explicit results of key generation services. However, none cross the physical boundary.

Destruction Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module.

Private and secret keys as well as seeds and entropy input are provided to the module by the calling application, and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently. An application operating on behalf of an authorized Crypto-Officer or User has access to all key data generated during the operation of the module.

In the event module power is lost and restored the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed.

Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism. This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

The entropy loaded is assumed to be at least 112 bits. This is dependent upon the calling application and its selection of an entropy source that provides the minimum required amount of entropy based on the function called.

4. Roles, Authentication, and Services

The module implements the User and Crypto Officer roles, but does not require authentication for those roles. Roles are implicitly assumed based on the service selected.

- User Role (User): Loading the module and calling any of the API functions.
- Crypto Officer Role (CO): Installation of the module on the host computer system and calling of any API functions.

All services implemented by the module are listed below, along with a description of service CSP access.

Table 9: Authorized Services (Approved Mode)

Service	Role	Description
Initialize	CO	Module initialization (installation of the module on the host computer system). Does not access CSPs.
Self-Test	CO	Perform self tests (FIPS_selftest) Does not access CSPs.
Show Status	CO	Functions that provide module status information (Version, FIPS Mode) do not access CSPs.
Zeroize	CO	Functions that destroy CSPs: • `fips_rand_prng_reset`: destroys RNG CSPs. • `Fips_drbg_uninstantiate`: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRGB CSPs). All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application. (Zeroization of all CSPs can be accomplished by reset of the GPC.)
Random Number Generation	User	Used for random number generation: seed or reseed a DRBG instance, determine security strength of a DRBG instance, obtain random data. Uses and updates Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs.
Symmetric Key Generation	User	Used to generate AES and Triple-DES keys using the Approved DRBG.

Service	Role	Description
Asymmetric Key Generation	User	Used to generate DSA, ECDSA, RSA, and ECDH keys using mechanisms listed in Table 4 above.
Symmetric Encrypt / Decrypt	User	Used to encrypt or decrypt data.
Symmetric Digest	User	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, Triple-DES, CMAC.
Message Digest	User	Used to generate a SHA-1 or SHA2 message digest. Does not access CSPs.
Keyed Hash	User	Used to generate or verify data integrity with HMAC. Executes using HMAC key provided by calling process.
Key Transport[1]	User	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). Executes using RSA KDK or RSA KEK provided by the calling process.
Key Agreement	User	Used to perform key agreement primitives on behalf of the calling process. It does not establish keys into the module. Executes EC DH Private, EC DH Public, provided by the calling process.
Digital Signature	User	Used to generate or verify RSA, DSA, or ECDSA digital signatures using mechanisms listed in Table 4 above. Executes using RSA SGK, RSA SVK, DSA SGK, DSA SVK, ECDSA SGK, ECDSA SVK, passed in by the calling process.
Utility	User	Miscellaneous helper functions that do not access CSPs.

Table 10: CSP Access within Services

Service	CSPs														
	RSA SGK	RSA KDK	DSA SGK	ECDSA SGK	EC DH Private	AES EDK	AES CMAC	AES GCM	AES XTS	Triple-DES EDK	Triple-DES CMAC	HMAC Key	Hash_DRBG CSPs	HMAC_DRBG_CSPs	CTR_DRBG CSPs
Initialize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Self-Test	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Show Status	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Zeroize	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Random Number Generation	-	-	-	-	-	-	-	-	-	-	-	-	G	G	G
Symmetric Key Generation	-	-	-	-	-	G	G	G	G	G	G	-	-	-	-

[1] "Key transport" can refer either to moving keys in and out of the module or the use of keys by an external application. The latter definition is the one that applies to this module.

Service	CSPs														
	RSA SGK	RSA KDK	DSA SGK	ECDSA SGK	EC DH Private	AES EDK	AES CMAC	AES GCM	AES XTS	Triple-DES EDK	Triple-DES CMAC	HMAC Key	Hash_DRBG CSPs	HMAC_DRBG_CSPs	CTR_DRBG CSPs
Asymmetric Key Generation	G	G	G	G	G	-	-	-	-	-	-	-	-	-	-
Symmetric Encrypt / Decrypt	-	-	-	-	-	R,E	R,E	R,E	R,E	R,E	R,E	-	-	-	-
Symmetric Digest	-	-	-	-	-	-	R,E	-	-	-	R,E	-	-	-	-
Message Digest	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Keyed Hash	-	-	-	-	-	-	-	-	-	-	-	R,E	-	-	-
Key Transport	-	R,E	-	-	-	-	-	-	-	-	-	-	-	-	-
Key Agreement	-	-	-	-	R,E	-	-	-	-	-	-	-	-	-	-
Digital Signature	R,E	-	R,E	R,E	-	-	-	-	-	-	-	-	-	-	-
Utility	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- G = Generate: The module generates the CSP.
- R = Read: The module reads the CSP. The read access is typically performed before the module uses the CSP
- E = Execute: The module executes using the CSP.
- Z = Zeroize: The module zeroizes the CSP.

The following services are available in the non-Approved mode of operation:

Table 11: Services Restricted to Non-Approved Mode

Service	Role	Description
Random Number Generation	User	Used for random number generation using ANSI X9.31 RNG.
Symmetric Key Generation	User	Used to generate AES and Triple-DES keys using ANSI X9.31 RNG.
Asymmetric Key Generation	User	Used to generate DSA, ECDSA and RSA keys using generation mechanisms listed in Table 6 above.
Key Transport	User	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the Module). RSA Decryption Key is passed in by the calling process.
Key Agreement	User	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the Module) using curves defined Table 6 above. EC DH Private Key and EC DH Public Key are passed in by the calling process.

Service	Role	Description
Digital Signature	User	Used to generate or verify RSA, DSA or ECDSA digital signatures using key sizes, digest sizes or curves from Table 6 above. Keys are passed in by the calling process.

5. Self-Test

The module performs the self-tests listed below on invocation of Initialize or Self-test.

Table 12: Self-Tests

Algorithm/Function	Type	Test Attributes
Software Integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT each for SHA1, SHA224, SHA256, SHA384 and SHA512. This testing covers SHA POST requirements.
AES	KAT	Decrypt, ECB mode, 256 bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192 key length
AES GCM	KAT	Separate encrypt and decrypt, 256 key length
XTS-AES	KAT	128, 256 bit key sizes to support either the 256-bit key size (for XTS-AES-128) or the 512-bit key size (for XTS-AES-256)
AES CMAC	KAT	Generate and verify CBC mode; 128, 192, 256 key lengths
Triple-DES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key
Triple-DES CMAC	KAT	CMAC generate and verify, CBC mode, 3-Key
RSA	KAT	Sign and verify using 2048 bit key, SHA-256, PKCS#1
DSA	PCT	Sign and verify using 2048 bit key, SHA-384
DRBG	KAT	CTR_DRBG: AES, 256 bit with and without derivation function HASH_DRBG: SHA256 HMAC_DRBG: SHA256
ECDSA	PCT	Keygen, sign, verify using P-224 and SHA512.
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6

All power-up self-tests listed above are invoked by the operating system at library load time. Should any test fail, an internal flag is set to prevent the library from shifting into FIPS mode.

Applications may perform this shift to FIPS mode via the `FIPS_mode_set()`[2] function. A return value of "1" indicates that the self-tests passed and the module has successfully shifted into FIPS mode, while a return value of "0" indicates that the self-tests failed.

The power-up self-tests may also be performed on-demand by calling `FIPS_selftest()`, which returns a "1" for success and "0" for failure. Interpretation of this return code is the responsibility of the calling application.

The module also implements the following conditional tests:

Table 13: Conditional Tests

Algorithm	Test
DRBG	Tested as required by [SP800-90A, §11.3]
DRBG	FIPS 140-2 continuous test for stuck fault
DSA	Pairwise consistency test on each generation of a key pair
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair

In the event of a DRBG self-test failure the calling application must uninstantiate and re-instantiate the DRBG per requirements; this is not something the module can do itself.

Pairwise consistency tests are performed for both possible modes of use, e.g. Sign/Verify and Encrypt/Decrypt.

6. Operational Environment

The Android operating system segregates user processes into separate process spaces. Each process space is logically separated from all other processes by the operating system software and hardware. The module functions entirely within the process space of the calling application, and implicitly satisfies the FIPS 140-2 requirement for a single user mode of operation.

7. Security Rules

The Cryptographic Module was designed with the following security rules in mind:

1. The Module shall provide two distinct operator roles. These are the User role, and the Crypto-Officer role.
2. The cryptographic module does not provide any operator authentication.
3. The operator shall be capable of commanding the Module to perform the power-up self-test using recycling power or FIPS_selftest() function.
4. The Module is single-threaded and in error scenarios returns only an error value (no data output is returned).

[2] `FIPS_mode_set()` calls module function `FIPS_module_mode_set()`.

5. The Module does not support concurrent operators.
6. The Module does not output intermediate key generation values.
7. The cryptographic module is available to perform services only after successfully completing the power-up self-tests.
8. Security functions listed in Table 6 in this Security policy are not allowed for use in the FIPS Approved mode of operation. When these algorithms are used, the module is no longer operating in the FIPS Approved mode of operation. It is the responsibility of the consuming application to zeroize all keys and CSPs prior to and after utilizing these non-Approved algorithms. CSPs shall not be shared between the Approved and non-Approved modes of operation.

8. Mitigation of Other Attacks

The module is not designed to mitigate against attacks which are outside the scope of FIPS 140-2.

