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

Version: 8.0
Date: July 8, 2016

CHANGE RECORD

Revision	Date	Author	Description of Change
1	January 7th, 2015	Adam Wick	Initial Revision
2	March 12th, 2015	Daniel M. Zimmerman	Minor Corrections
3	March 27th, 2015	Daniel M. Zimmerman	Updates to Integrity Check
4	June 4th, 2015	Adam Wick	Conversion to Word, Version updates
5	June 10th, 2015	Daniel M. Zimmerman	Updates to Unapproved Algorithms and Authentication
6	June 22, 2015	Galois	Added updates per report requirements; added verbiage explaining non-approved mode; moved ANSI X9.31 RNG to the non-approved mode.
7	September 3, 2015	Galois	Updated Key Generation; added Block Diagram; added algorithm certs for OE 2 and 3 (SW version 3.10.49)
8	May5, 2016	InfoGard	Updated Mode of Operation

1. Module Description

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

The module is a software library located within the operating system kernel providing a C-language application program interface (API) for use by user and kernel applications 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 (GPC) on which the module is installed. The logical cryptographic boundary of the module is the linux kernel (version 3.4.0 and 3.10.49).

The module performs no communications other than with the calling application (the process that invokes the module services).

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

Table 1: Module Security Level Specification

Security Component	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

Overall the module has security level 1. This has been tested in the following configurations:

Table 2: Platform Configurations

Kernel Version	Platform	Operating System	Processor
3.4.0	LG G3 Model VS985 4G	Android 5.0.1	Qualcomm Snapdragon 800 (32-bit mode)
3.10.49	LG G-Flex 2 Model LGLS996	Android 5.0.1	Qualcomm Snapdragon 800 (64-bit mode)
3.10.49	LG G4 Model VS986	Android 5.1	Qualcomm Snapdragon 800 (64-bit mode)

2. Cryptographic Module Boundary

2.1. Software Block Diagram

The figure below illustrates the relationship of the module to the kernel and GPC.

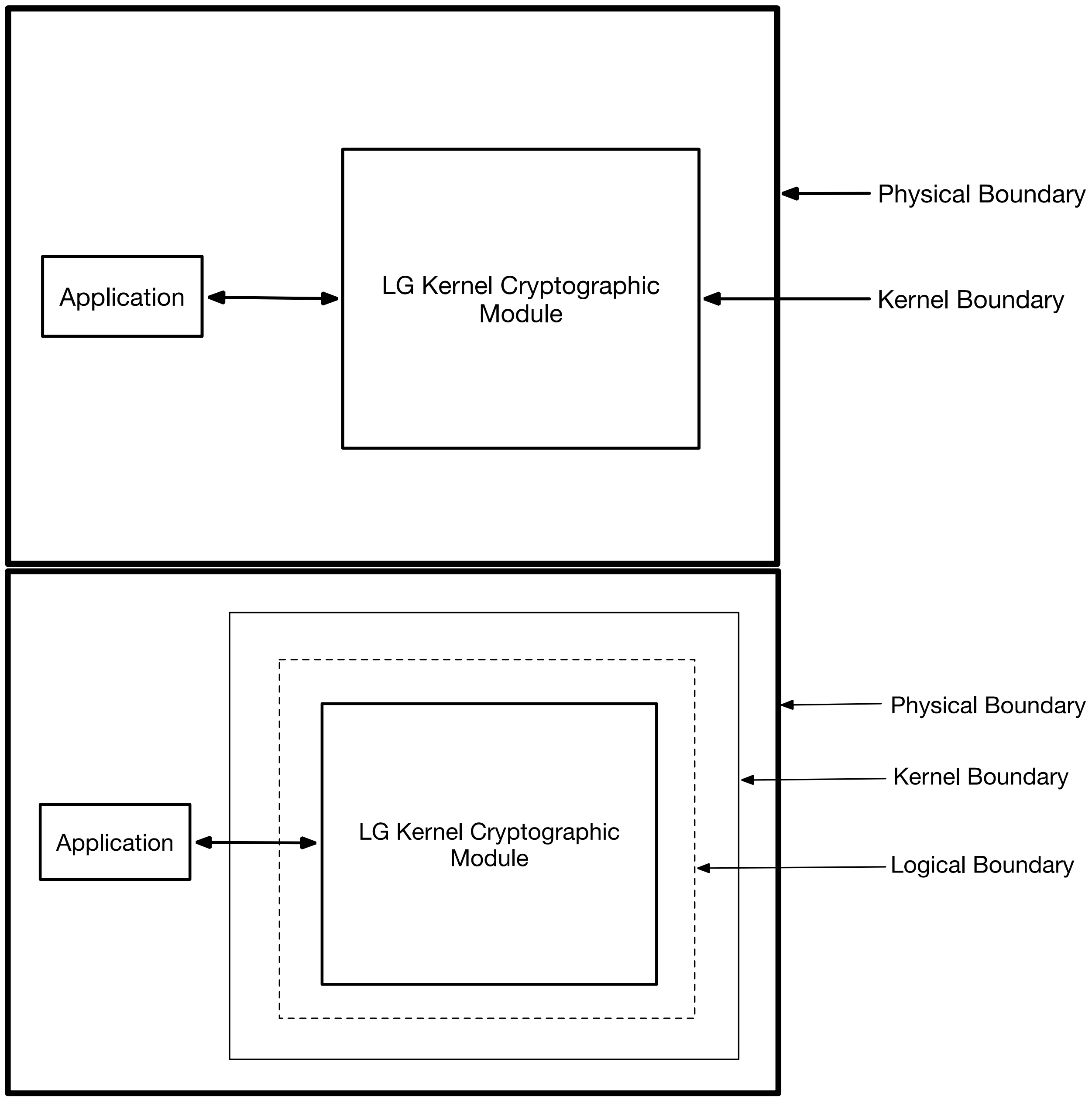

Figure 1: Software Block Diagram

3.4.0 kernel object files (all within the *crypto* folder).

ablkcipher.o	authenc.o	crypto_algapi.o	des_generic.o	md5.o	shash.o
aes_generic.o	authencesn.o	crypto_blkcipher.o	ecb.o	pcompress.o	tcrypt.o
ahash.o	blkcipher.o	crypto_hash.o	eseqiv.o	proc.o	testmgr.o
algapi.o	built-in.o	crypto_null.o	fips_integrity.o	rng.o	twofish_common.o
algboss.o	cbc.o	crypto_wq.o	fips.o	scatterwalk.o	twofish_generic.o
ansi_cprng.o	chainiv.o	crypto.o	gf128mul.o	seqiv.o	xcbc.o
api.o	cipher.o	cryptomgr.o	hmac.o	sha1_generic.o	xts.o
arc4.o	compress.o	ctr.o	krng.o	sha256_generic.o	
crc32c.o	deflate.o	md4.o	sha512_generic.o		

3.10.49 kernel object files (all within crypto folder):

ablk_helper.o	arc4.o	crc32c.o	ctr.o	krng.o	sha256_generic.o
ablkcipher.o	authenc.o	cryptd.o	deflate.o	md4.o	sha512_generic.o
aes_generic.o	authencesn.o	crypto_algapi.o	des_generic.o	md5.o	shash.o
ahash.o	blkcipher.o	crypto_blkcipher.o	ecb.o	pcompress.o	tcrypt.o
algapi.o	built-in.o	crypto_hash.o	eseqiv.o	proc.o	testmgr.o
algboss.o	cbc.o	crypto_null.o	fips_integrity.o	rng.o	twofish_common.o
ansi_cprng.o	chainiv.o	crypto_wq.o	fips.o	scatterwalk.o	twofish_generic.o
api.o	cipher.o	crypto.o	gf128mul.o	seqiv.o	xcbc.o
compress.o	cryptomgr.o	hmac.o	sha1_generic.o	xts.o	

3. 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: Ports and 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).

4. Modes of Operation and Cryptographic Functionality

The module supports both Approved and non-Approved modes of operation. While in the Approved mode of operation, only FIPS Approved and tested algorithms may be used.

In the Approved mode, the module provides the following approved algorithms:

Table 4: Approved Algorithms for Approved Mode of Operation

Algorithm	Description	Cert #
AES	[FIPS 197, SP 800-38A] Functions: Encryption, Decryption Modes: ECB, CBC, CTR Key sizes: 128, 192, 256 bits	3290 (3.4.0) 3443 (3.10.49)
Triple-DES	[SP 800-38C] Functions: Generation, Verification Modes: ECB, CBC, CTR Key sizes: 128, 192, 256 bits	1875 (3.4.0) 1940 (3.10.49)

Algorithm	Description	Cert #
HMAC	[FIPS 198-1] Functions: Generation, Verification SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	2088 (3.4.0)
		2192 (3.10.49)
SHA	[FIPS 180-4] Functions: Digital Signature Verification, non-Digital Signature Applications SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	2729 (3.4.0)
		2843 (3.10.49)

This module requires an initialization sequence to operate in FIPS mode. The kernel must be booted with the command line options "fips=1 fips_ic=(HMAC)", where (HMAC) is the expected HMAC-SHA-1 for the compressed kernel image (computed at kernel build time and used in the kernel integrity check). Use of the "fips=1" command line option triggers FIPS mode. Upon entering FIPS mode, the module boots and automatically begins the required power-up self-tests. Any failure of the self-tests causes the kernel to print a message (or series of messages) to the log and enter the FIPS error state. When the module is in the FIPS error state, it is unusable via any interface.

In addition to the above initialization sequence, the calling application is also responsible for ensuring that only Approved functions are utilized during Approved operation. The following non-Approved algorithms are available in the non-Approved mode of operation, and shall not be used in the Approved mode of operation:

- DES
- AES-CTR (non-compliant)
- AES-CCM (non-compliant)
- AES-CMAC (non-compliant)
- AES-XTS (non-compliant)
- AES-XCBC
- Triple-DES-CTR (non-compliant)
- Triple-DES-CMAC (non-compliant)
- 2 Key Triple-DES
- Twofish
- AEAD (AES GCM) (non-compliant)
- MD5
- MD4
- ARC4
- GHASH
- ANSI X9.31 RNG (non-compliant)

The module is a cryptographic engine library, used only in conjunction with additional software.

4.1. Critical Security Parameters

All CSPs used by the module are described in this section. All access to these CSPs by module services are described in Section 4. The CSP names are generic, corresponding to API parameter data structures.

Table 5: Critical Security Parameters and Descriptions

CSP Name	Description
AES EDK	AES (128 / 192 / 256bit) encrypt / decrypt key□
TDES EDK	Triple-DES (3-Key 168bit) encrypt / decrypt key□
HMAC Key	Keyed hash key (dependent on calling application, must be minimum 112 bits in length)

The module does not output intermediate key generation values.

For all CSPs:

Storage RAM, associated to entities by memory location. 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).

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. The calling application is responsible for parameters passed in and out of the module. Module power off is also a valid means of zeroizing all keys and CSPs.

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. 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 used during the operation of the module.

5. Roles and Services

The module meets all FIPS 140-2 level 1 requirements for Roles and Services. The Module does not allow concurrent operators.

Both roles have access to all of the services provided by the module.

- User Role (User): Access to user space usage
- Crypto Officer Role (CO): Access to kernel space usage

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

Table 6: Roles and Services in Approved Mode

Service	Role	Description
Initialize	User, CO	Module initialization. Does not access CSPs.
Self-Test	User, CO	Perform self-tests. Does not access CSPs.
Show Status	User, CO	Functions that provide module status information (FIPS Setting) do not access CSPs.
Zeroize	User, CO	All services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.
Symmetric Encrypt / Decrypt	User, CO	Used to encrypt or decrypt data. Executes using AES EDK, Triple-DES EDK, passed in by the calling process.
Message Digest	User, CO	Used to generate a SHA-1 or SHA2 message digest. Does not access CSPs.
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC. Executes using HMAC key and key size provided by calling process.
Utility	User, CO	Miscellaneous helper functions that do not access CSPs.

In addition to the services available in the approved mode, the module also supports the following services for the non-approved mode.

Table 7: Roles and Services in Non-Approved Mode

Service	Role	Description
Symmetric Encrypt / Decrypt	User, CO	Used to encrypt or decrypt data. Executes using DES, AES-CTR (non-compliant), AES-CCM (non-compliant), AES-CMAC (non-compliant), AES-XTS (non-compliant), AES-XCBC, Triple-DES-CTR (non-compliant), Triple-DES-CMAC (non-compliant), 2 Key Triple-DES, Twofish, AEAD (AES-GCM) (non-compliant), ARC4, passed in by the calling process.
Message Digest	User, CO	Used to generate a MD4, MD5, or GHASH hash.
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC using keys less than 112 bits.
Random String Generation	User, CO	Generates a random value using the ANSI X9.31 RNG.

6. Self-Tests

The module performs the self-tests listed below at boot time, as part of the process of kernel loading by the boot loader. The software integrity check is performed by the kernel after the algorithm self-tests; if the binary kernel image has been modified, the integrity check fails. This is described in more detail below. If any self-test or integrity test fails, the module enters an error state and becomes nonfunctional. The operator can re-run all power-up self-tests by power cycling the GPC thereby reloading the module and automatically initiating all self-tests. The kernel is assured to run self-tests without operator intervention.

A kernel proc file is set to indicate when the device is in FIPS mode or an error state. The process file `/proc/sys/crypto/fips_enabled` serves as a flag to indicate whether the device was started in FIPS mode, and the process file `/proc/sys/crypto/fips_error` serves as a flag to indicate whether the device is in the error state. The possible combinations of these flags have the following meanings:

Table 8: Combination Legend

`fips_enabled`	`fips_error`	Meaning
0	0	Device is in non-FIPS mode
0	1	Impossible Combination
1	0	Device is in FIPS mode (all tests passed)
1	1	Device is in FIPS error state

The self-test process is completely automatic when the appropriate kernel boot command line options (described above) are supplied; the boot command line options are embedded into the signed system image supplied by LG Electronics, and are not modifiable by the user of the mobile device.

Table 9: Self-Tests

Algorithm	Type	Test Attributes
Software Integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT each for SHA1, SHA224, SHA256, SHA384 and SHA512.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128 bit key length.
Triple-DES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key.

6.1. Integrity Check Details

At build time, an HMAC-SHA-1 is calculated on the binary compressed kernel image. This HMAC-SHA-1, as a 40-character hexadecimal string, is passed as a command line parameter to the kernel at boot time.

At boot time, the kernel copies its image to a second location in RAM. After all the algorithm self-tests are complete, the kernel integrity test routine does the following:

1. Use the kernel crypto API to perform an HMAC-SHA-1 on the copy of the kernel image in RAM.
2. Compare the resulting HMAC to the HMAC passed to the kernel as a command line parameter.
3. If the calculated and command line values do not match, enter the FIPS error state.

7. Security Rules

The following rules must be followed in order to be compliant with the requirements of FIPS 140-2:

1. A calling application shall not invoke crypto_alloc_rng(), crypto_alloc_ablkcipher(), crypto_alloc_aead(), crypto_alloc_ablkcipher(), crypto_alloc_cipher(), crypto_alloc_hash(), crypto_alloc_ahash(), or crypto_alloc_shash() with any of the following algorithms:

    DES: "des", "cbc(des)", "ctr(des)", "ecb(des)"
    AES-CTR (non-compliant): "ctr(aes)" or "rfc3686(ctr(aes))"
    AES-CCM (non-compliant): "ccm(aes)" or "rfc4309(ccm(aes))"
    AES-CMAC (non-compliant): "cmac(aes)"
    AES-XTS (non-compliant): "xts(aes)"
    AES-XCBC: "xcbc(aes)"
    Triple-DES-CTR (non-compliant): "ctr(des3_ede)"
    Triple-DES-CMAC (non-compliant): "cmac(des3_ede)"
    Twofish: "cbc(twofish)", "ctr(twofish)", "ecb(twofish)", "lrw(twofish)", "xts(twofish)" or any value that includes the substrings "cbc-twofish" or "ecb-twofish".
    AEAD (AES GCM) (non-compliant): "gcm(aes)", "rfc4543(gcm(aes))", or any value that includes the substring "gcm-aes".
    MD5: "md5" or "hmac(md5)"
    MD4: "md4"
    ARC4: "ecb(arc4)"

    GHASH:"ghash" or any value that includes the substring "ghash"

2. The length and security strength of the keys provided are the responsibility of the calling application. Keys provided that offer less than 112 bits of security strength are to be used only in the non-approved mode of operation.

8. Operational Environment

The FIPS 140-2 module covers the static kernel binary and therefore provides a number of non-security services to callers within kernel space as well as user space. The following documents provide a description of these services:

- Linux system call API man pages, provided in Chapter 2 of the Linux man pages available from git://github.com/mkerrisk/man-pages.git.
- Linux kernel internals including interfaces between kernel components, documented in Professional Linux Kernel Architecture, ISBN 978-0470343432.
- Linux kernel driver development documentation covering the kernel inter- faces available for device drivers, documented in Linux Device Drivers, 3rd Edition, ISBN 978-0596005900.

The module inherently operates only in a single-operator mode. The kernel performs tasks in both Kernel space and User space. To access the kernel, User space code performs systems calls to Kernel space, the operating system performs the needed tasks and returns execution back to User space. This separation ensures that only one “operator” can access the module at a time.

9. Mitigation of Other Attacks

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