



Kony Solutions Cryptographic Library

FIPS 140-2 Level 1 Security Policy

Version: 1.0

Last Updated: May 29, 2013

Kony Solutions, Inc.
7380 West Sand Lake Road #390
Orlando, Florida 32819

Copyright © 2012-2013 Kony Solutions, Inc. All rights reserved. This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

Revision History

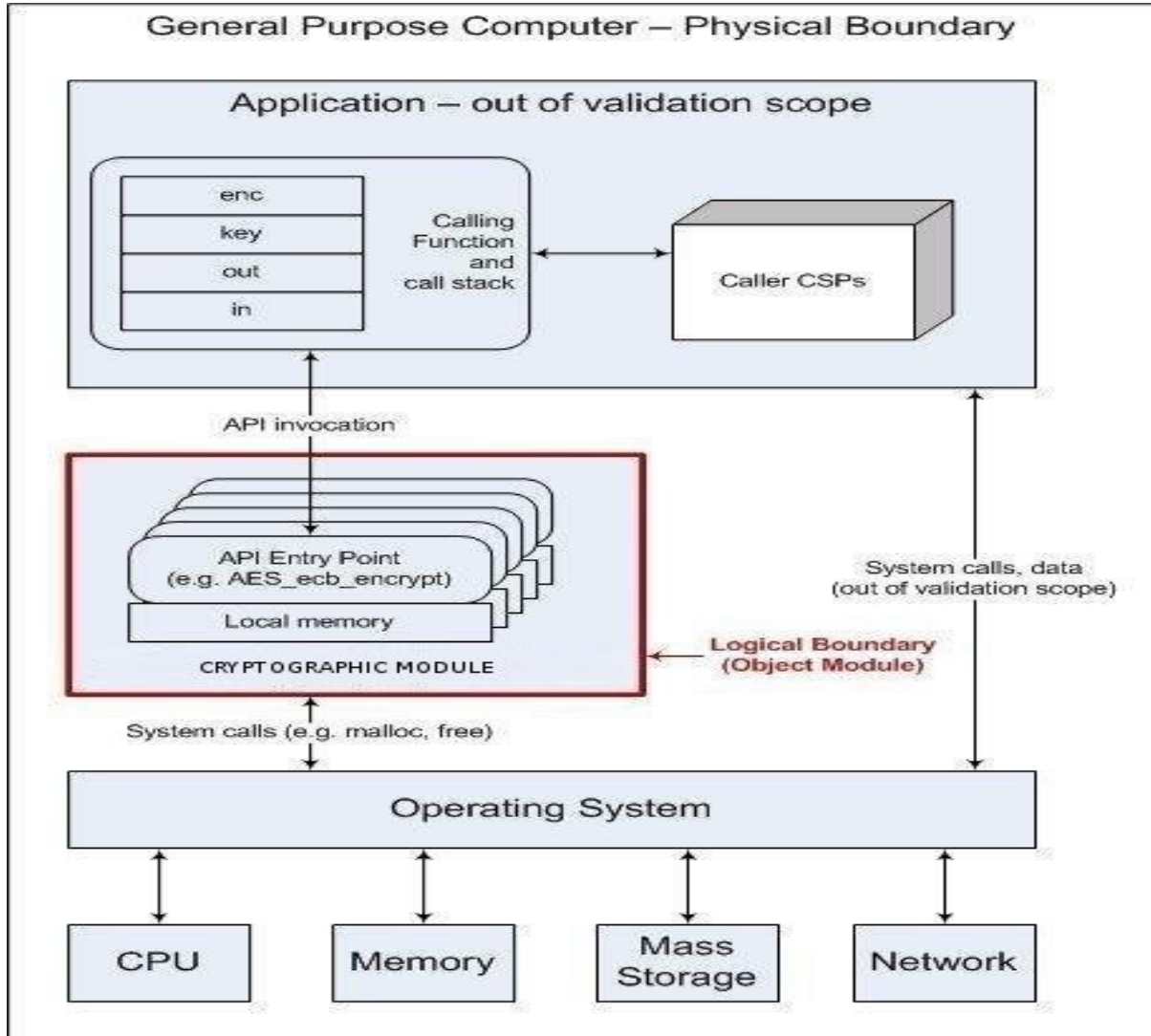
| Authors | Date | Version | Comment |
|----------------------|------------|---------|---------------|
| Kony Solutions, Inc. | 2013-05-29 | 1.0 | Initial draft |

Table of Contents

| | |
|--|----|
| Revision History..... | 2 |
| 1 Introduction..... | 4 |
| 2 Tested Configurations..... | 5 |
| 3 Ports and Interfaces | 5 |
| 4 Modes of Operation and Cryptographic Functionality | 6 |
| 4.1 Critical Security Parameters and Public Keys..... | 7 |
| 5 Roles, Authentication and Services | 9 |
| 6 Self-Test..... | 11 |
| 7 Operational Environment..... | 13 |
| 8 Mitigation of other attacks..... | 13 |

1 Introduction

This document comprises the non-proprietary FIPS 140-2 Security Policy for the Kony Solutions Cryptographic Library, hereafter referred to as the Module.



Block Diagram

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 fipsanister 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).

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

| Security Requirement | Security Level |
|---|----------------|
| Cryptographic Module Specification | 1 |
| Cryptographic Module Ports and Interfaces | 1 |
| Roles, Services, and Authentication | 2 |
| Finite State Model | 1 |
| Physical Security | NA |
| Operational Environment | 1 |
| Cryptographic Key Management | 1 |
| EMI/EMC | 1 |
| Self-Tests | 1 |
| Design Assurance | 1 |
| Mitigation of Other Attacks | NA |

Table 1 – Security Level of Security Requirements

The Module’s software version for this validation is 2.0.

2 Tested Configurations

| | Operating System | Processor | Optimizations (Target) | Hardware Device |
|---|------------------|---------------------------|------------------------|-----------------|
| 1 | Android 2.2 | Qualcomm QSD 8250 (ARMv7) | none | HTC Desire |
| 2 | Android 2.2 | Qualcomm QSD 8250 (ARMv7) | NEON | HTC Desire |
| 3 | Android 3.0 | Ti OMAP 3621 (ARMv7) | none | Nook BNRV200 |
| 4 | Android 3.0 | Ti OMAP 3621 (ARMv7) | NEON | Nook BNRV200 |
| 5 | Android 4.0 | TI DM3730 (ARMv7) | none | Beagleboard-XM |
| 6 | Android 4.0 | TI DM3730 (ARMv7) | NEON | Beagleboard-XM |

Table 2 - Supported Platforms

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).

| Logical interface type | Description |
|------------------------|---|
| Control input | API entry point and corresponding stack parameters |
| Data input | API entry point data input stack parameters |
| Status output | API entry point return values and status stack parameters |
| Data output | API entry point data output stack parameters |

Table 3 - Logical interfaces

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 only a FIPS 140-2 Approved mode. Tables 4a and 4b list the Approved and Non-approved but Allowed algorithms, respectively.

| Function | Algorithm | Options | Cert # |
|---|---|--|--------|
| Random Number Generation; Symmetric key generation | [ANS X9.31] RNG | AES 128/192/256 | 1164 |
| | [SP 800-90] DRBG ¹ Prediction resistance supported for all variations | Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function Dual EC DRBG: P-256, P-384, P-521 | 290 |
| Encryption, Decryption and CMAC | [SP 800-67] | 3-Key TDES ECB, TCBC, TCFB, TOFB; CMAC generate and verify | 1464 |
| | [FIPS 197] AES | 128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CCM; GCM; CMAC generate and verify | 2338 |
| | [SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS | | |
| Message Digests | [FIPS 180-3] | SHA-1, SHA-2 (224, 256, 384, 512) | 2016 |
| Keyed Hash | [FIPS 198] HMAC | SHA-1, SHA-2 (224, 256, 384, 512) | 1448 |

¹ For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP800-90] and [SP800-57].

| | | | |
|---|-------------------------|--|------|
| Digital Signature and Asymmetric Key Generation | [FIPS 186-2] RSA | GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS, SigVer9.31, SigVerPKCS1.5, SigVerPSS (1024/1536/2048/3072/4096 with all SHA sizes) | 1204 |
| | [FIPS 186-2] DSA | PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024 with SHA-1 only) | 732 |
| | [FIPS 186-3] DSA | PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024/2048/3072 with all SHA sizes) | 732 |
| | [FIPS 186-2] ECDSA | Key Pair, PKV, SigGen, SigVer (all NIST defined B, K, and P curves with SHA-1 only) | 382 |
| | [FIPS 186-3] ECDSA | Key Pair, PKV, SigGen, SigVer (all NIST defined B, K and P curves with all SHA sizes) | 382 |
| ECC CDH (CVL) | [SP 800-56A] (§5.7.1.2) | All NIST defined B, K and P curves | 51 |

Table 4a – FIPS Approved Cryptographic Functions

The Module supports only NIST defined curves for use with ECDSA and ECC CDH.

| Category | Algorithm | Description |
|----------------------------|-----------|---|
| Key Agreement | EC DH | Non-compliant (untested) DH scheme using elliptic curve, supporting all NIST defined B, K and P curves. Key agreement is a service provided for calling process use, but is not used to establish keys into the Module. |
| Key Encryption, Decryption | RSA | The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services. |

Table 4b – Non-FIPS Approved But Allowed Cryptographic Functions

EC DH Key Agreement provides 80 to 256 bits of security strength. RSA Key Wrapping provides 80 to 256 bits of security strength.

The Module supports only a FIPS 140-2 Approved mode. The Module requires an initialization sequence (see IG 9.5): the calling application invokes `FIPS_mode_set()`², which returns a “1” for success and “0” for failure. If `FIPS_mode_set()` fails then all cryptographic services fail from then on. The application can test to see if FIPS mode has been successfully performed.

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 FIPS 186-3 assurances are outside the scope of the Module, and are the responsibility of the calling process.

4.1 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 4. The CSP names are generic, corresponding to API parameter data structures.

² The function call in the Module is `FIPS_module_mode_set()` which is typically used by an application via the `FIPS_mode_set()` wrapper function.

| CSP Name | Description |
|-------------------|--|
| RSA SGK | RSA (1024 to 16384 bits) signature generation key |
| RSA KDK | RSA (1024 to 16384 bits) key decryption (private key transport) key |
| DSA SGK | [FIPS 186-3] DSA (1024/2048/3072) signature generation key or [FIPS 186-2] DSA (1024) signature generation key |
| ECDSA SGK | ECDSA (All NIST defined B, K, and P curves) signature generation key |
| EC DH Private | EC DH (All NIST defined B, K, and P curves) private key agreement key. |
| AES EDK | AES (128/192/256) encrypt / decrypt key |
| AES CMAC | AES (128/192/256) CMAC generate / verify key |
| AES XTS | AES (256/512) XTS cipher key |
| TDES EDK | TDES (3-Key) encrypt / decrypt key |
| TDES CMAC | TDES (3-Key) CMAC generate / verify key |
| HMAC Key | Keyed hash key (160/224/256/384/512) |
| RNG CSPs | Seed (128 bit), AES 128/192/256 seed key and associated state variables for ANS X9.31 AES based RNG ³ |
| Hash_DRBG CSPs | V (440/880 bits) and C (440/880 bits), entropy input (length dependent on security strength) |
| HMAC_DRBG CSPs | V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength) |
| CTR_DRBG CSPs | V (128 bits) and Key (AES 128/192/256), entropy input (length dependent on security strength) |
| Dual_EC_DRBG CSPs | S (P-256, P-384, P-521), entropy input (length dependent on security strength) |
| CO-AD-Digest | Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication |
| User-AD-Digest | Pre-calculated HMAC-SHA-1 digest used for User role authentication |

Table 4.1a – Critical Security Parameters

Authentication data is loaded into the module during the module build process, performed by an authorized operator (Crypto Officer), and otherwise cannot be accessed.

The module does not output intermediate key generation values.

| CSP Name | Description |
|-----------------|--|
| RSA SVK | RSA (1024 to 16384 bits) signature verification public key |
| RSA KEK | RSA (1024 to 16384 bits) key encryption (public key transport) key |
| DSA SVK | [FIPS 186-3] DSA (1024/2048/3072) signature verification key or [FIPS 186-2] DSA (1024) signature verification key |
| ECDSA SVK | ECDSA (All NIST defined B, K and P curves) signature verification key |
| EC DH Public | EC DH (All NIST defined B, K and P curves) public key agreement key. |

Table 4.1b – Public Keys

³ There is an explicit test for equality of the seed and seed key inputs

For all CSPs and Public Keys:

Storage: RAM, associated to entities by memory location. The Module stores RNG and DRBG state values for the lifetime of the RNG or 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 RNG and DRBG state values used for the Modules' default key generation service.

Generation: The Module implements ANSI X9.31 compliant RNG and SP 800-90 compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys as shown in Table 4a. 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 authorized application as user (Crypto-Officer and 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: 128 bits for the [ANS X9.31] RNG mechanism, and as shown in [SP 800-90] Table 2 (Hash_DRBG, HMAC_DRBG), Table 3 (CTR_DRBG) and Table 4 (Dual_EC_DRBG). This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

5 Roles, Authentication and Services

The Module implements the required User and Crypto Officer roles and requires authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators. The User or Crypto Officer role is assumed by passing the appropriate password to

the `FIPS_module_mode_set()` function. The password values may be specified at build time and must have a minimum length of 16 characters. Any attempt to authenticate with an invalid password will result in an immediate and permanent failure condition rendering the Module unable to enter the FIPS mode of operation, even with subsequent use of a correct password.

Authentication data is loaded into the Module during the Module build process, performed by the Crypto Officer, and otherwise cannot be accessed.

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/256^{16}$, or less than $1/10^{38}$. The Module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

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

- 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.

| Service | Role | Description |
|--------------------------|----------|--|
| Initialize | User, CO | Module initialization, inclusive of all Table 9 tests (<code>FIPS_module_mode_set</code>). Does not access CSPs. |
| Self-test | User, CO | Perform all Table 9 tests (<code>FIPS_selftest</code>). Does not access CSPs. |
| Show status | User, CO | Functions that provide module status information: <ul style="list-style-type: none"> • Version (as unsigned long or const char *) • FIPS Mode (Boolean) Does not access CSPs. |
| Zeroize | User, CO | Functions that destroy CSPs: <ul style="list-style-type: none"> • <code>fips_rand_prng_reset</code>: destroys RNG CSPs. • <code>fips_drbg_uninstantiate</code>: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_DRBG CSPs.) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application. |
| Random number generation | User, CO | Used for random number and symmetric key generation. <ul style="list-style-type: none"> • Seed or reseed an RNG or DRBG instance • Determine security strength of an RNG or DRBG instance • Obtain random data Uses and updates RNG CSPs, Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_DRBG CSPs. |

| Service | Role | Description |
|----------------------------|----------|---|
| Asymmetric key generation | User, CO | Used to generate DSA, ECDSA and RSA keys: RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90 |
| Symmetric encrypt/decrypt | User, CO | Used to encrypt or decrypt data. Executes using AES EDK, TDES EDK (passed in by the calling process). |
| Symmetric digest | User, CO | Used to generate or verify data integrity with CMAC. Executes using AES CMAC, TDES, CMAC (passed in by the calling process). |
| Message digest | User, CO | Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs. |
| Keyed Hash | User, CO | Used to generate or verify data integrity with HMAC. Executes using HMAC Key (passed in by the calling process). |
| Key transport ⁴ | User, CO | 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, RSA KEK (passed in by the calling process). |
| Key agreement | User, CO | Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). Executes using EC DH Private, EC DH Public (passed in by the calling process). |
| Digital signature | User, CO | Used to generate or verify RSA, DSA or ECDSA digital signatures. Executes using RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process). |
| Utility | User, CO | Miscellaneous helper functions. Does not access CSPs. |

Table 5 - Services and CSP Access

6 Self-Test

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

| Algorithm | Type | Test Attributes |
|--------------------|------|---|
| Software integrity | KAT | HMAC-SHA1 |
| HMAC | KAT | One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512 Per IG 9.3, this testing covers SHA POST requirements. |
| AES | KAT | Separate encrypt and decrypt, ECB mode, 128 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 | Sign and verify CBC mode, 128, 192, 256 key lengths |

⁴ "Key transport" can refer to a) moving keys in and out of the module or b) the use of keys by an external application. The latter definition is the one that applies to this Module.

| Algorithm | Type | Test Attributes |
|-----------|------|--|
| TDES | KAT | Separate encrypt and decrypt, ECB mode, 3-Key |
| TDES 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 Dual_EC_DRBG: P-256 and SHA256 |
| ECDSA | PCT | Keygen, sign, verify using P-224, K-233 and SHA512. The K-233 self-test is not performed for operational environments that support prime curve only (see Table 2). |
| ECC CDH | KAT | Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6 |
| X9.31 RNG | KAT | 128, 192, 256 bit AES keys |

Table 6a - Power On Self Tests (KAT = Known answer test; PCT = Pairwise consistency test)

The `FIPS_mode_set()`⁵ function performs all power-up self-tests listed above with no operator intervention required, returning a “1” if all power-up self-tests succeed, and a “0” otherwise. If any component of the power-up self-test fails an internal flag is set to prevent subsequent invocation of any cryptographic function calls. The module will only enter the FIPS Approved mode if the module is reloaded and the call to `FIPS_mode_set()`⁵ succeeds.

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:

| Algorithm | Test |
|----------------|--|
| DRBG | Tested as required by [SP800-90] Section 11 |
| 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 |
| ANSI X9.31 RNG | Continuous test for stuck fault |

Table 6b - Conditional Tests

In the event of a DRBG self-test failure the calling application must unstantiate and re-instantiate the DRBG per the requirements of [SP 800-90]; this is not something the Module can do itself.

⁵ `FIPS_mode_set()` calls Module function `FIPS_module_mode_set()`

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

7 Operational Environment

The tested operating systems segregate 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.

8 Mitigation of other attacks

The Module does not claim any attack mitigation beyond FIPS 140-2 Level 1 requirements.