EF Johnson Technologies - Kenwood Cryptographic Library

FIPS 140-3 Non-Proprietary Security Policy

Author: John Tooker

Software Version: 4.0

Date: 2/14/2024

Table of Contents

1 General

This document is a non-proprietary FIPS 140-3 Security Policy for the EF Johnson Technologies' Kenwood Cryptographic Library (KCL) version 4.0. The KCL is a Level 1 software cryptographic module. Table 1 describes the security level of each section in this document.

Table 1: Security Levels

Section No.	Section Title	Security Level
1	General	Level 1
2	Cryptographic Module Specification	Level 1
3	Cryptographic Module Interfaces	Level 1
4	Roles, Services and Authentication	Level 1
5	Software/Firmware Security	Level 1
6	Operational Environment	Level 1
7	Physical Security	N/A
8	Non-invasive Security	N/A
9	Sensitive Security Parameter Management	Level 1
10	Self-Tests	Level 1
11	Life-Cycle Assurance	Level 1
12	Mitigation of Other Attacks	N/A

2 Cryptographic Module Specification

The KCL is a multi-chip standalone FIPS 140-3 module for use on a variety of platforms when a hardware module is unavailable. It provides access to basic cryptographic algorithms with no long-term key storage within the library itself. It is a dynamically linked C++ library compiled for various consumer grade operating environments, see Table 2 below.

Figure 1 diagrams the KCL's relationship with an application that may use it.

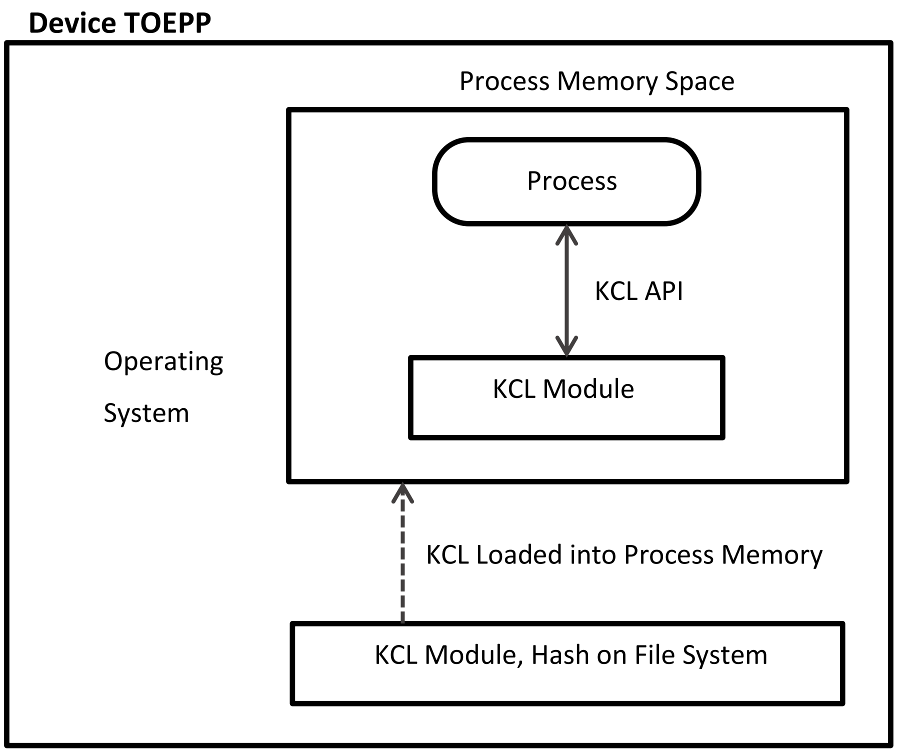

Figure 1: A diagram showing the KCL loaded from disk into process memory for use

The module itself consists of a single library file to be dynamically loaded by a process into its memory space. The library file has a companion hash file to verify its integrity when loaded by a process.

Table 2 and Table 3 list the operating environments in which the KCL was tested or affirmed.

Table 2: Tested Operational Environments

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Android 10	Zebra TC21	Qualcomm Snapdragon™ 660 octa-core, 1.8 GHz	No
2	ST Microelectronics Linux v5.10-stm32mp-r1	VP8000	STM32MP151 ARM Cortex A7	No

Table 3: Vendor Affirmed Operational Environments

#	Operating System	Hardware Platform
1	Android 6.0	Nexus 5X with Qualcomm SDM630
2	Android 7.1	Nexus 5X with Qualcomm SDM630
3	Android 7.1	Sonim XP8 with Qualcomm Snapdragon 660
4	ST Microelectronics Linux v5.15-stm32mp-r2	VM8000 with STM32MP151 ARM Cortex A7
5	ST Microelectronics Linux v5.15-stm32mp-r2	VP8000 with STM32MP151 ARM Cortex A7

The overall security rating of this module is Level 1.

The module boundary consists of the device on which the KCL is installed with the binary file: “libkcl.so”. If multiple processes load the module, each will have its own, separate instance of the library in its own, separate memory. None of the data passed between the process and the KCL leaves the process’s memory space.

The KCL only operates in a single mode of operation. This unnamed mode is entered automatically when the library is loaded. If a failure occurs, the library enters a failure mode which cannot be exited except by unloading and reloading the library.

Other than the failure mode, the library does not operate in any degraded modes.

Table 4 lists the security functions provided by the KCL.

Table 4: Approved Algorithms

CAVP Cert.	Algorithm	Modes	Description & Key Sizes	Functions
A2280	AES SP 800-38A FIPS 197	ECB, CBC, OFB	128, 192, 256	Encryption, Decryption
A2280	AES Key Wrap/Unwrap SP 800-38F	KW	128, 192, 256	Wrap, Unwrap
A2280	DRBG SP 800-90A	SHA-512	N/A	DRBG
A2280	SHA-2 FIPS 180-4	SHA-512	N/A	Hash

No block diagram is required for understanding other than Figure 1, above.

The security design and rules of operation are as follows. The physical interface is provided by the hosting platform, generally consumer-grade hardware and operating systems with a keyboard, monitor, mouse, screen, and/or touch screen as well as network and USB ports. The logical interface of the KCL is provided through the library's Application Programming Interface (API). All data input, output, control and status are defined by this API. All data contained within the loaded library is erased when the error state is entered, or the library is unloaded. The DRBG must be seeded with a user-provided seed containing at least 384 bits of entropy as specified in scenario 2(b) of IG 9.3.A and there is *no assurance of the minimum strength of generated SSPs.* If fewer than 384 bits are supplied, the library will enter the error state.

The library initializes itself atomically on load, including all self-tests and integrity checks. The only requirement is a file containing the hash of loaded library's binary file be located next to it with the same name, but a `*.hash.*` extension. See Section 10 for a description of the self-tests that are run on load.

3 Cryptographic Module Interfaces

Table 5 lists the data that passes over the API of the KCL library categorized by logical interface. This encompasses all logical interfaces. There are no physical interfaces for the KCL itself.

Table 5: Ports and Interfaces

Logical Interface	Data that passes over this interface
Data Input	• DRBG Entropy & Seed • AES Key • AES Plaintext • AES Ciphertext • AES Key • SHA-512 Message
Data Output	• DRBG Random Bytes • AES Ciphertext • AES Plaintext • AES Key (for wrap/unwrapped only) • SHA-512 Hash
Control Input	• DRBG length
Status Output	• State Boolean • Library version
Power	N/A as this is a software module

The only channel of communication with the library is in-process function calls to the library from the process that loaded it. This channel's protection is enforced by the operating system and the process that loads the library and is not part of the library itself.

If the library goes into the error state, the output interface return values are not available as exceptions will be thrown.

4 Roles, Services and Authentication

The KCL modules supports the crypto-officer role only. There is no user or maintenance role. No authentication is required.

Table 6: Roles, Service Commands, Input and Output

Role	Service	Input	Output
Crypto Officer	Generate Random Value	Length	Random bytes
Crypto Officer	Seed DRBG	DRBG Entropy & Seed	N/A
Crypto Officer	AES Encryption	AES Key, Plaintext	Ciphertext
Crypto Officer	AES Decryption	AES Key, Ciphertext	Plaintext
Crypto Officer	AES Key Wrap	AES Key, AES Key-to-be-wrapped	AES Key-to-be-wrapped
Crypto Officer	AES Key Unwrap	AES Key, AES Unwrapped Key	AES Unwrapped Key
Crypto Officer	SHA-512	Message	Hash
Crypto Officer	Zeroize	N/A	N/A
Crypto Officer	Get Status	N/A	Boolean indicating if the library is in the ok state
Crypto Officer	Show Module's Version Information	N/A	Library version
Crypto Officer	Perform Self-Test	N/A	N/A

There are no ways to bypass any of these capabilities. There is no self-initiated cryptographic output capability. No external software or firmware may be loaded into the library. When seeding the DRBG, at least 48 bytes of entropy must be provided; otherwise, it will enter the error state.

The list of security services and their approved security functions can be found in Table 7. Access rights legend:

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 API return code indicators, the successful completion of a service is an implicit indicator for the use of an approved service. If the service does not complete successfully, an exception is thrown, and the module enters an error state.

Table 7: Approved Services

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Generate Random Value	Use previously seeded DRBG	DRBG SHA-512	DRBG V and C State	Crypto Officer	R,W,E	API return code
Seed DRBG	Initialize the DRBG portion of the library	DRBG SHA-512	DRBG Entropy & Seed	Crypto Officer	W,E	API return code
AES Encryption	Encrypt plaintext with a key	AES	AES Key	Crypto Officer	W, E	API return code
AES Decryption	Decrypt ciphertext with a key	AES	AES Key	Crypto Officer	W, E	API return code
AES Key Wrap	Wrap a key with another	AES	AES Key, AES Key-to-be-wrapped	Crypto Officer	W, E, Z	API return code
AES Key Unwrap	Unwrap a key with another	AES	AES Key, AES Unwrapped Key	Crypto Officer	W, E, Z	API return code
SHA-512	Hash a message	SHA2-512	N/A	Crypto Officer	-	API return code
Zeroize	Clear the DRBG and any loaded AES key	AES, DRBG	DRBG V and C State, DRBG Entropy & Seed, AES Key	Crypto Officer	Z	API return code
Get Status	Return whether the library is in an OK state	N/A	N/A	Crypto Officer	-	API return code
Show Module's Version Information	Return library version	N/A	N/A	Crypto Officer	-	API return code
Perform Self-Tests	(Re)load the library to (re)perform self-tests	DRBG, SHA-512 AES	DRBG V and C State, DRBG Entropy & Seed, AES Key	Crypto Officer	R, W, E, Z	None (call Get Status service to observe results)

The installation of the module simply requires placing it and its corresponding hash file on the file system and loading the library in a compiled C++ program. There is no authentication mechanism. The module itself contains no data when not loaded (no data is persisted when the module unloads).

5 Software/Firmware Security

The KCL automatically performs an integrity check on itself when it is loaded. It looks for a file next to the library file that contains the SHA2-512 hash of the library file itself. It then computes the SHA2-512 hash of itself. If the two match, the integrity check passes. Otherwise, the library enters a failed state and any calls made to the library will not return normally but throw an exception.

The operator can initiate this integrity check on demand by loading the library.

The library comes in the form of a single binary file. The filename is `libkcl.so` on Linux (including Android).

This module is not open source.

6 Operational Environment

The KCL operates in a modifiable environment. The operating system is in charge of guarding the memory of the KCL while it is loaded. No special rules, settings or restrictions are needed of the operational environment. This is how Level 1 security is satisfied. The library stores no keys/SSPs when unloaded. The operating systems and tested platforms can be found in Table 2 above.

7 Physical Security

The KCL is implemented completely in software such that physical security is provided solely by the host platform. Therefore, the physical security section of FIPS 140-3 is not applicable.

8 Non-invasive Security

No steps to mitigate non-invasive attacks have been made.

9 Sensitive Security Parameter Management

The sensitive security parameters (SSPs) managed by the KCL are enumerated in Table 8.

Table 8: Sensitive Security Parameters (SSPs)

Key/ SSP Name	Strength	Security Funct. & Cert #	Gener-ation	Import/ Export	Establish-ment	Storage	Zeroization	Use
AES Key	128, 192 or 256	AES A2280	N/A	Plaintext import only	N/A	In memory only while module is loaded	Zeroize function invoked	AES Encrypt, Decrypt, Key Wrap/Unwrap
AES Key-to-be-wrapped	128, 192 or 256	AES A2280	N/A	Plaintext import only, cypher-text export only	N/A	In memory only during API call	Cleared on API call return	AES Wrap
AES Un-wrapped Key	128, 192 or 256	AES A2280	N/A	Cypher-text import only, Plaintext export only	N/A	In memory only during API call	Cleared on API call return	AES Unwrap
DRBG Entropy	256-bit	DRBG A2280	N/A	Import only	N/A	Not stored	N/A	Initialize DRBG V & C state
DRBG Seed	256-bit	DRBG A2280	N/A	N/A	N/A	In memory only while module is loaded	Zeroize function invoked	Initialize DRBG V & C State
DRBG V & C State	256-bit	DRBG A2280	V & C values	N/A	N/A	In memory only while module is loaded	Zeroize function invoked	Generate Keys, IVs

The approved random bit generator is a SHA-512 DRBG using certification A2280. The source of entropy is provided by the operator external to the KCL module. The state of the DRBG exists in volatile memory only and is cleared when the module is unloaded or zeroized.

An AES key is stored in volatile memory only and are cleared when the module is unloaded or zeroized. This AES key cannot be directly generated internal to the module, but the DRBG may be used to generate them. If the DRGB is used to generate an AES key, then the key must be the unmodified output of the DRBG. This AES key cannot be exported, only loaded for use.

The zeroize function is invoked using the API. When invoked, all keys and sensitive security parameters are cleared from memory. As the module is implemented in C++, cleared memory is released back to the OS, which takes responsibility for clearing the data so other processes cannot observe it (just as it protects that memory from being accessed by other processes while it is in use by the library). The module is always in approved mode.

Keys stored in memory are not protected within the module and rely on the operating system's process memory protection.

Unloading the library will zeroize all SSPs; this procedural zeroization method is under the control of the operator.

10 Self-Tests

The module runs self-tests automatically on load. The following tests are performed in the order listed in Table 9. Certain tests rely on and therefore test certain services.

Table 9: Self-Tests Performed by the KCL

On Load Order	Test Name	Category	Type	Services Covered	Key Size
1	AES	Conditional cryptographic algorithm test	Known Answer Test (KAT)	AES Key Wrap AES Encryption AES Key Unwrap AES Decryption	128-bit key
2	DRBG	Conditional cryptographic algorithm test	Known Answer Test (KAT)	Seed DRBG Generate Random Value SHA-512	
3	Integrity	Pre-operational software integrity test	SHA-512 of library file	-	
-	DRBG Request Limit	Conditional critical functions test	Continuous	Generate Random Value	

AES test performs known answer tests (KAT). The DRBG test is also a KAT. Finally, the integrity check is done by calculating the SHA-512 hash of the library itself on disk against a file with the expected hash. After each test, all temporary values are cleared. There is no condition where these tests are repeated once the module is loaded.

If a test fails, the module enters an error state. Subsequent function calls into the module will not return normally, rather, an exception is thrown.

The module operator cannot initiate self-tests other than (re)loading the module.

The DRBG continuously keeps track of requests and will put the library into the error state when the number of requests exceeds 2^{48}.

There is only a single error state, it is entered when a self-test fails or the reseed counter is triggered in the DRBG. The two status indicators are: 1) calling the Get Status service, which will return a Boolean indicating the error state, and 2) calling any other service will throw an exception if we are in the error state rather than returning a normal value.

11 Life-Cycle Assurance

The module's life cycle starts when it is loaded into memory by a process, and it ends when the module is unloaded. There are no maintenance requirements or administrator or non-administration guidance other than the module's API itself.

12 Mitigation of Other Attacks

The KCL is not designed for the mitigation of any attacks outside the scope of FIPS 140-3 Level 1.