Security Policy (Non-Proprietary)

Communication Cryptographic Library (CCL)

Author: John Tooker

Software Version: 3.0

Document Version 3.6
June 21, 2018

Contents

1 Introduction

1.1 Scope

This document is a FIPS 140-2 Security Policy for the EFJohnson Technologies Communication Cryptographic Library™ (CCL), version 3.0 cryptographic module. The CCL is a Level 1 software cryptographic module. In terms of the FIPS 140-2 standard, the CCL is a multi-chip standalone FIPS 140-2 module.

The module's logical boundary consists of an Android device dynamically linked library and an Application Programming Interface to access the FIPS 140-2 security functions within the CCL library.

This library has been validated under FIPS 140-2 for the Android operating environment listed under the platform in section 1.3.

1.2 CCL Implementation

The CCL is implemented as a multi-chip standalone module meeting Level 1 requirements of the FIPS 140-2 standard. The CCL is a dynamically linked library implemented using the C programming language with an external Java interface. Application developers wishing to use the CCL can use the CCL's Application Programming Interface (API) to perform AES, ECDSA, HMAC, DRBG, SHA256 and SHA512 security related functions. It also includes non-validated legacy services to support DES encryption while operating in the Non-Approved mode of operation.

The services of the CCL module are accessible through the CCL's API. The product number and version are:

1. CCL version 3.0 ~ Product Number 039-5804-200 Rev 3.0

1.3 Cryptographic Boundary

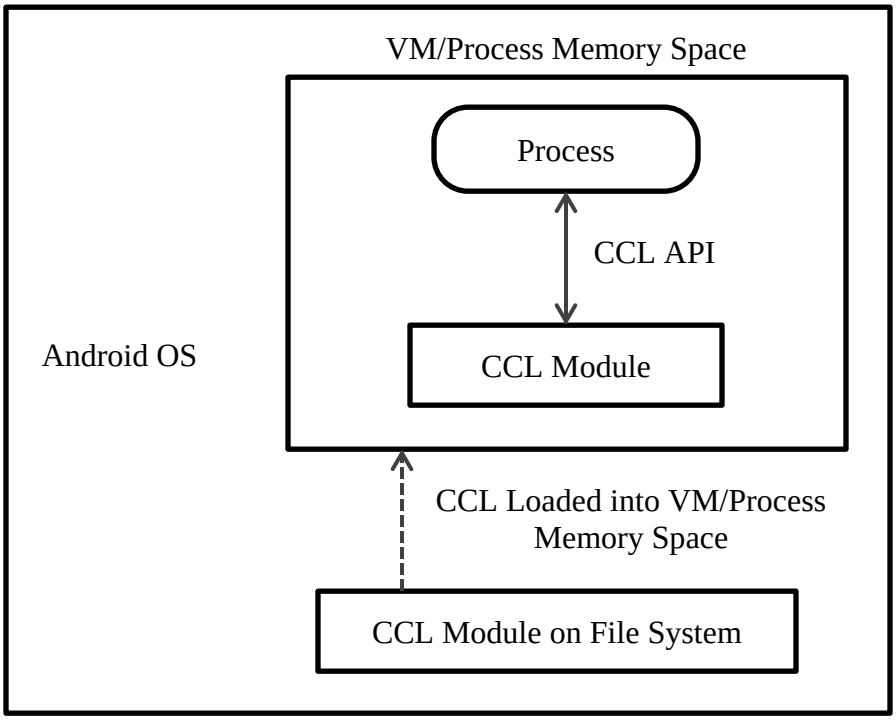
The physical cryptographic boundary of the CCL consists of the Android Device in which the CCL module is installed. The software logical boundary surrounds the CCL library (libFIPS_140_2.so).

The CCL module runs as a dynamically linked library under the Android operating system.

All applications running on the Android operating system run as processes in a virtual machine. Every process running on the operating system will execute in its own separate memory space. When a process which makes use of the CCL is executed, the operating system maps the CCL's library code into the process' memory space. Multiple applications can load, and make use of the CCL library at the same time. Each process will have its own copy of the CCL loaded into its own memory space. Any data passed between the process and the CCL is specific to that process and never leaves the process' memory space.

Below is the CCL's block diagram as described above.

Figure 1-1 CCL Block Diagram



The CCL module was tested on the following computing platform:

Platform 1

Hardware:	Nexus 5X Android Compatible Phone (GPC)
Processor:	Qualcomm Snapdragon 808
Operating System:	Android 6.0

The cryptographic module is also supported on any platform with the following operating environment for which operational testing was not performed:

Processor:	Qualcomm SDM630
Operating System:	Android 7.1

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

2 Intended FIPS 140-2 Security Levels

The CCL is validated to meet FIPS 140-2 security requirements for the levels shown in the Table 1.3-1. The overall module is validated for Security Level 1.

Table 1.3-1 CCL Security Levels

Area	**FIPS 140-2 Intended Security Level**
Cryptographic Module Specification	Level 1
Cryptographic Module Ports and Interfaces	Level 1
Roles, Services, and Authentication	Level 1
Finite State Model	Level 1
Physical Security	N/A
Operational Environment	Level 1
Cryptographic Key Management	Level 1
EMI/EMC	Level 1
Self-Tests	Level 1
Design Assurance	Level 1
Mitigation of Other Attacks	N/A

3 FIPS 140-2 Approved Mode of Operation

The module operator determines the Mode of Operation implicitly by which security function they invoke. The following approved security functions are available in the Approved Mode of Operation:

1. AES-128 OFB
2. AES-128 ECB
3. AES-128 CBC
4. AES-192 OFB
5. AES-192 ECB
6. AES-192 CBC
7. AES-256 OFB
8. AES-256 ECB
9. AES-256 CBC
10. SP800-38F AES Key Wrap (128, 192, 256)
11. SHA256
12. SHA512
13. ECDSA Signature Verification with Curves/Key size P-256
14. HMAC-SHA256
15. DRBG

4 FIPS 140-2 Non-Approved Mode of Operation

The following non-approved security functions are available in the Non-Approved Mode of Operation:

1. DES OFB
2. DES ECB
3. DES CBC

5 Cryptographic Functionality

Table 5.1-1 Approved and CAVP Validated Cryptographic Functions

Algorithm	Description	Cert #
AES	[FIPS 197, SP 800-38A] Functions: Encryption, Decryption Modes: ECB, CBC, OFB Key sizes: 128, 192, 256 bits	#3985
AES Key Wrap/Unwrap	[SP 800-38F] Functions: Wrap, Unwrap Modes: ECB Key sizes: 128, 192, 256 bits	#3985
DRBG	[SP 800-90A] Functions: Hash DRBG using SHA-512 Security Strengths: 256 bits	#1178
ECDSA	[FIPS 186-4] Functions: Signature Verification Curves/Key sizes: P-256 with SHA-256	#882
HMAC	[FIPS 198-1] Functions: Generation, Verification SHA sizes: SHA-256	#2601
SHA	[FIPS 180-4] Functions: Digital Signature Verification, non-Digital Signature Applications SHA sizes: SHA-256, SHA-512	#3290

6 Security Rules

The CCL meets the requirements of a multi-chip standalone module. Since the CCL is a software module, the module interfaces are defined in terms of the API performing FIPS 140-2 security functions.

6.1 Operating Environment

The CCL requires the Android 6 operating system. The operator must disable all non-administrator accounts and all network services in order to restrict the operating system to a single operator mode of operation.

The CCL was tested under the Android 6.0 operating system using an Android Compatible phone: Nexus 5X with a Qualcomm Snapdragon 808 processor.

6.2 FIPS 140-2 Related Security Rules

1. The CCL has the following interfaces:

   The CCL's physical interfaces consist of those found on a typical Android device. These physical interfaces are the computer's hardware such as touchscreen, memory card, and USB ports. The CCL's logical interface is provided through the CCL's Application Programming Interface.

   - Data Input Interface: The Data Input Interface is the parameters of the API function calls defined as input.
   - Data Output Interface: The Data Output Interface is the parameters of the API function calls defined as output.
   - Control Input Interface: The touch screen or keyboard is the physical control input interface of the CCL module. Control inputs are the function calls.
   - Status Output Interface: The physical status output interface consists of the screen of the computer. The return values of the CCL APIs are the status interface for the CCL.
   - Power Interface: The Battery and Charging system of the computing device is the power port.

2. All data output via the CCL Data Output Interface is disabled when an error state exists.
3. The CCL supports a User role and a Crypto Officer role. The role is selected implicitly by the service that is invoked.
4. The CCL's DRBG must be seeded with a user provided seed containing at least 112 bits of entropy. *No assurance of the minimum strength of generated keys. The module generates cryptographic keys whose strengths are modified by available entropy*.

5. The CCL performs the self-tests specified below.

   Power-up Self-Tests (performed in accordance with IG 9.10):

   - Software Integrity Test
     All CCL software releases are digitally signed using the ECDSA algorithm at the EF Johnson Technologies facility. During self-tests, the CCL verifies the integrity of software library using the ECDSA (P-256) algorithm.
   - AES Algorithm Test
     The AES algorithm is tested for encrypt and decrypt with key sizes 128, 192 and 256 using a Known Answer Test (KAT) in the Electronic Code Book (ECB) mode of operation.
   - SHA-256 Algorithm Test
     The SHA-256 hash algorithm is tested using a KAT.
   - SHA-512 Algorithm Test
     The SHA-512 hash algorithm is tested using a KAT.
   - HMAC-SHA-256 Algorithm Test
     The HMAC-SHA-256 algorithm is tested using a KAT.
   - ECDSA Algorithm Test
     The ECDSA Algorithm test is an ECDSA signature verification test using a KAT.
   - Deterministic Random Bit Generator (DRBG) Test
     The DRBG is tested using KAT on load.

   Additional Self-Tests (to verify the proper functioning of the CCL module):

   - On-Demand Self-Testing
     The module exports an API function called ***Java_com_efjohnson_crypto_FIPS_1140_12_1JNI_initialize*** which can be used by the operator to initiate all the power-up self-tests of the CCL module.
   - Conditional Self-Testing
     - Continuous DRBG tests that compare a newly generated block of random data with the previously generated block of random data. If the two blocks are equal, the CCL enters an error state. In this error state, the CCL module will not perform any cryptographic operations.
     - DRBG health tests per SP 800-90A Section 11.3

6. If any of the power-up or conditional self-test routines fails, the CCL enters an error state, stops responding to commands, and outputs an error message specific to the cause of the error (e.g., “Failed to load: AES is not ok”).
7. The module does not perform any cryptographic functions while in an error state.

8. The CCL supports DES only for support of pre-existing legacy systems, and only while operating in the Non-Approved Mode of Operation.

7 Identification and Authentication Policy

The CCL does not support authentication for either the User or Crypto Officer roles.

8 Access Control Policy

8.1 Roles Supported

The CCL cryptographic module supports the User and Crypto-Officer role only. There is no Maintenance role in the CCL.

The User and Crypto-Officer roles are mutually exclusive and cannot both be active concurrently.

8.1.1 User Role

This role is implicitly assumed when an operator uses one of the User services of the CCL module (See Table 8.2-1).

8.1.2 Crypto-Officer Role

This role is implicitly assumed when an operator uses one of the Crypto-Officer services (See Table 8.2-1).

The CCL library will perform a Digital Signature verification of itself when the underlying operating system attempts to load the module into the address space of an application.

8.2 Services Provided

The services provided by the CCL are the actual security functions that will be made available to the User via API calls.

The CCL APIs made available to the user are the interfaces.

The table below lists all the security services and functions that are performed by the CCL. The operator using the CCL service is also listed in the table.

Table 8.2-1 CCL Services vs. Security Functions in Approved Mode

Service	Description	Security Function(s) Used	Key Type and Length	General Mode of Operation	Operator Using Service.
Generate Random Key	Uses the CCL's approved SP800-90A DRBG to generate a random key.	DRBG	256 bit AES 192 bit AES 128 bit AES	Key Generation	Crypto-Officer

Service	Description	Security Function(s) Used	Key Type and Length	General Mode of Operation	Operator Using Service.
Seed DRBG	Entropy input used to seed the DRBG	DRBG	Entropy Input – 200 byte Seed	Key Generation	Crypto-Officer
AES Encryption	Service provided by the CCL's approved AES algorithm to perform encryption.	AES	256 bit AES 192 bit AES 128 bit AES	Encryption	User
AES Decryption	Service provided by the CCL's approved AES algorithm to perform decryption.	AES	256 bit AES 192 bit AES 128 bit AES	Decryption	User
AES Key Wrapping Encryption	All keys that are stored outside of the CCL boundary can be stored in encrypted format using this service provided by the CCL's approved AES.	AES	256 bit AES 192 bit AES 128 bit AES	Encryption	User
AES Key Unwrapping Decryption	All keys that are stored outside of the CCL boundary can be retrieved and decrypted using this service provided by the CCL's approved AES.	AES	256 bit AES 192 bit AES 128 bit AES	Decryption	User
ECDSA Signature Verification	Service provided by the CCL's approved ECDSA algorithm to perform ECDSA Signature Verification.	ECDSA	P-256 ECDSA	ECDSA Signature verification	User
HMAC-SHA-256	Service provided by the CCL's approved HMAC-SHA-256 algorithm to perform a MAC given an encryption key.	HMAC-SHA-256	256 bit	MAC Generation	User
SHA-256	Service provided by the CCL's approved SHA-256 algorithm to generate a 256-bit message digest.	SHA-256	N/A	Hashing	User
SHA-512	Service provided by the CCL's approved SHA-512 algorithm to generate a 512-bit message digest.	SHA-512	N/A	Hashing	User

Service	Description	Security Function(s) Used	Key Type and Length	General Mode of Operation	Operator Using Service.
Self-Tests	Provides power up and continuous tests to verify the secure state and operation of the CCL. All cryptographic and security functions are tested using known answer tests. The CO initiates this service by power cycling or resetting the module, or by calling the initialize function (Java_com_efjohnson_crypto_FIPS_1140_12_1 JNI_initialize).	AES, ECDSA, HMAC, DRBG, SHA-256, SHA-512	Security Function Specific	Service Specific	Crypto-Officer
Show Status	Provides information on the CCL state, such as the Fatal Error State.	N/A	N/A	CCL operation State	User
Zeroize Keys	Zeroizes all CSPs present in the module.	N/A	N/A	Clear Keys	Crypto-Officer

Table 8.2-2 CCL Services vs. Security Functions in Non-Approved Mode

Service	Description	Security Function(s) Used	Key Type and Length	General Mode of Operation	Operator Using Service.
Generate Random Key	Uses the CCL's approved SP800-90A DRBG to generate a random key.	DRBG	56 bit DES	Key Generation	Crypto-Officer
DES Encryption	Service provided by the CCL's non-approved DES algorithm to perform encryption.	DES	56 bit DES	Encryption	User
DES Decryption	Service provided by the CCL's non-approved DES algorithm to perform decryption.	DES	56 bit DES	Decryption	User

8.3 Cryptographic Keys and CSPs supported

Table 8.3-1 CSP Description

CSP Identifier	Description
AES Encryption Key	A 256, 192, or 128 bit key used to encrypt and decrypt.
256 bit HMAC Key	A 256 bit key used with SHA-256 to authenticate messages.
DRBG Working State	The DRBG working state is used by the DRBG security function to generate random bits including those used for key generation.

Table 8.3-2 Public Key Description

CSP Identifier	Description
ECDSA Asymmetric Key	ECDSA P-256 public key used for signature verification

8.4 Access Rights within Services

An operator requiring a service within any role can read and/or write cryptographic keys and Critical Security Parameters (CSP) only through the invocation of the CCL module security service. Access to the module's security services is via the CCL's APIs.

The services within each role can only access the cryptographic keys and CSP that the service's API specifies.

The definition of Read/Write access for the CCL module is defined as follows:

- For Read/Write, the module both reads and writes to the internally stored cryptographic keys or CSPs.
- For Read access, the module will only Read the internally stored cryptographic keys or CSPs.
- For Delete access, the module will only delete a cryptographic key or CSPs.

Table 8.4-1 CCL Access Rights of CSPs

Service	Cryptographic Keys and CSPs	Type of Access (e.g. Read, Write, Delete)
Generate Random Key	AES Encryption Key, DRBG Working State	Read/Write
Seed DRBG	DRBG Working State	Read/Write
AES Encryption	AES Encryption Key	Read/Write
AES Decryption	AES Encryption Key	Read/Write
AES Key Wrapping Encryption	AES Encryption Key	Read/Write
AES Key Unwrapping Decryption	AES Encryption Key	Read/Write
ECDSA Signature Verification	ECDSA Asymmetric Key (public key)	Read/Write
HMAC-SHA-256	256 bit HMAC Key	Read/Write
SHA-256	None	NA
SHA-512	None	NA
Self-Tests	None	NA
Show Status	None	NA
Zeroize Keys	All CSPs	Delete

9 Key Management

The CCL module does not provide long-term key storage. All keys generated or processed by the CCL module reside in the application space in which the CCL library module has been loaded.

9.1 Key Generation

AES and DES keys (DES generation only available in the Non-Approved Mode) are generated using the DRBG.

9.2 Key Distribution and Storage

The CCL module does not provide long-term key storage or any protocol for key agreement or distribution.

Per IG 7.7, import or export of keys does not apply as all CSPs remain with the physical boundary of the module. All key used by the CCL module reside within the module's cryptographic boundary in the application space of the application which loaded the CCL module.

10 Physical Security Policy

The CCL is a cryptographic module that is implemented completely in software such that the physical security is provided solely by the host platform. Therefore, the Physical Security section of FIPS 140-2 is not applicable.

11 Mitigation of Other Attacks Policy

The CCL module is not designed for the mitigation of any attacks outside the scope of FIPS 140-2.

12 CCL API Functions

The CCL's API functions are specified in the *EFJ Communication Cryptographic Library (CCL) SDK Manual, version 3.0*.

13 References

The following standards and documents where used in the development of the CCL module.

1. FIPS 140-2: *Security Requirements for Cryptographic Modules*, May 25, 2001
2. SP 800-131A Revision 1: *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*, November 2015
3. SP 800-90A: *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, January 2012

4. SP 800-107 Revision 1: *Recommendation for Applications Using Approved Hash Algorithms,* August 2012
5. FIPS 186-4: *Digital Signature Standard*, July 2013
6. FIPS 198-1: *The Keyed-Hash Message Authentication Code*, July 2008

14 Acronym List

AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CCL	Communication Cryptographic Library
CFB	Cipher-Feedback
CSP	Critical Security Parameter
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
ECDSA	Elliptic Curve Digital Signature Algorithm
ECB	Electronic Codebook
FIPS	Federal Information Processing Standards
HMAC	Keyed-Hash Message Authentication Code
KAT	Known Answer Test
OFB	Output-Feedback
SHA-256	Secure Hash Algorithm-256
SHA-512	Secure Hash Algorithm-512