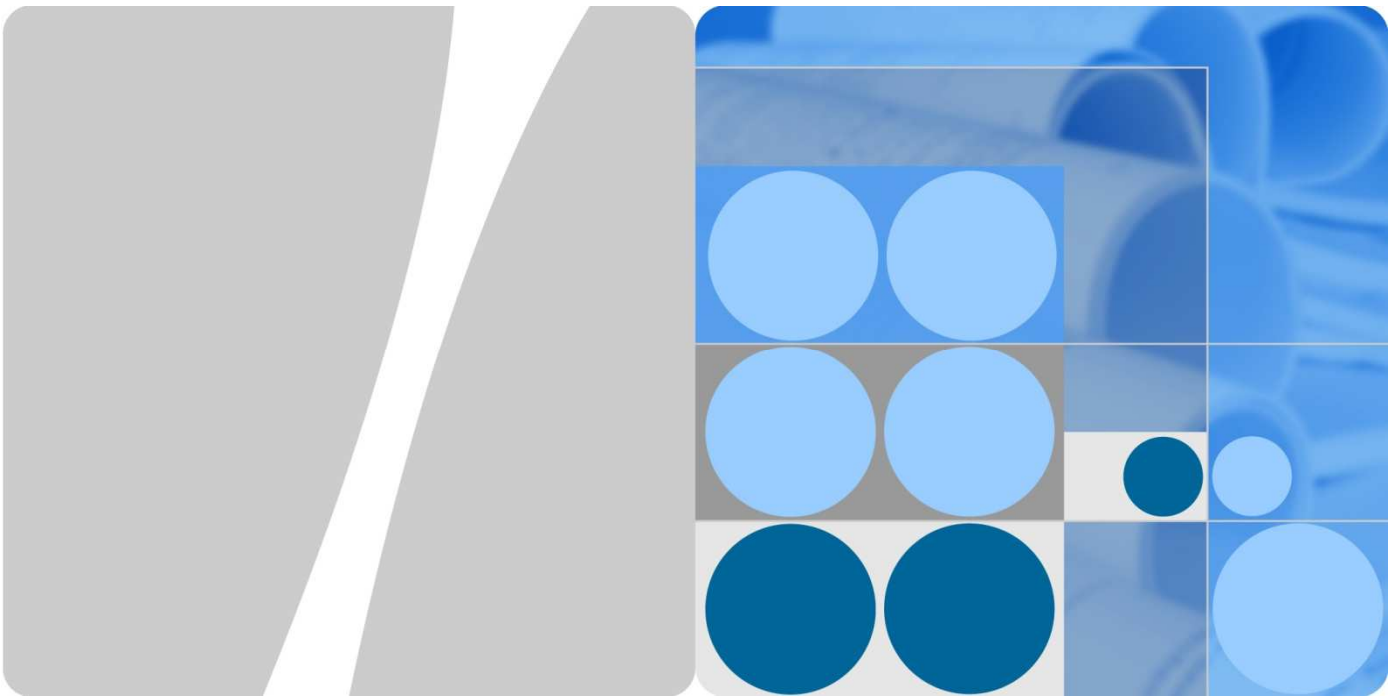

Huawei FIPS Cryptographic Library

Cryptography Library Security Policy
FIPS 140-2

This document provides a non-proprietary FIPS140-2 Security Policy
for the Huawei FIPS Cryptographic Library (HFCL) Version V300R003C22SPC805

Issue **1.9**

Date **2016-03-31**

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

Acknowledgements

Validation testing was performed by Epoche & Espri. For information on validation of the software contact:

Epoche & Espri

NVLAP Lab Code 200856-0

Avenida de los Pirineos, 7

Nave 9A

28703, San Sebastián de los Reyes (Madrid)

SPAIN

Contact: Mr. Miguel Banon Puente

E-Mail: mbp@epoche.es

TEL: +34 916588314

+34 916238772

Revision History

IUT Version	Revision	Date	Change Description
V300R003C22SPC805	1.9	2016-03-31	Refinements
V300R003C22SPC805	1.8	2016-01-26	Update for review comments
V300R003C22SPC804	1.7	2015-09-07	• Updated for review comments • Added Certificate No. for approved cryptographic algorithms.
V300R003C22 SPC803	1.6	2015-06-26	Updated content as per review comments
V300R003C22 SPC802	1.5	2015-06-04	Content updation
V300R003C22 SPC801	1.4	2015-01-20	Draft Release (Internal)

Contents

Figures

Tables

1 Introduction

1.1 Purpose and scope

This document is FIPS 140-2 non-proprietary security policy for the Huawei FIPS Cryptographic Library Module (HFCL) to meet FIPS 140-2 level 2 requirements.

This Security Policy details the secure operation of the **Huawei FIPS Cryptographic Library V300R003C22SPC805** developed by Huawei Technologies Co., Ltd. as required in Federal Information Processing Standards Publication 140-2 as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

It also serves as a technical reference describing the features and functions of the Cryptographic Library. The library is also referred to as the Cryptographic Module (CM) in this document.

For additional information on the cryptographic module, beyond the scope of this document, contact support@huawei.com.

1.2 Audience

This document is part of the documentation package required as part of the FIPS 140-2 validation process. It describes the **Huawei FIPS Cryptographic Library V300R003C22SPC805** module in relation to FIPS 140-2 requirements. The companion document, Huawei **FIPS Cryptography Developer Guide**, describes how to configure and operate with the cryptographic module (CM) in the FIPS 140-2 approved mode.

This document is intended for the following:

- FIPS validation team
- Cryptographic Module Validation Program (CMVP) team
- Architects, developers and testers

1.3 References

1.3.1 Companion documents of the CM

Developer Guide	Huawei FIPS Cryptography Developer Guide 8.0

1.3.2 FIPS 140-2 documents

FIPS 140-2	FIPS140-2 PUB FIPS 140-2 Security Requirements for cryptographic modules (and annexes)
FIPS 140-2 DTR	Derived Test Requirements for FIPS PUB 140-2, Security Requirements for Cryptographic Modules
NIST SP 800-131A	NIST Special Publication 800-131A. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
NIST SP 800-90A	NIST Special Publication 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators
NIST SP 800-57	NIST Special Publication 800-57 Recommendation for Key Management
FIPS PUB 186-4	Digital Signature Standard (DSS)

2 Cryptographic Module specification

2.1 Overview

The **Huawei FIPS Cryptographic Library** (HFCL) is software based cryptographic module, designed to achieve conformance as per FIPS 140-2 standard (security level 2).

HFCL is installed on a General Purpose multi-chip standalone device (DELL PowerEdge T110 II Intel Pentium w/ RHEL 5.3 evaluated at EAL4). The operating system meets the functional requirements specified in the "Controlled Access Protection Profile (CAPP)" defined in Annex B of the FIPS 140-2 and is evaluated at the CC evaluation assurance level EAL4.
(https://www.commoncriteriaportal.org/files/epfiles/st_vid10338-st.pdf).

The HFCL library is programmed in C language and the source code is compiled using the **gcc 4.1.2** compiler. HFCL is a static library and associated object file that contains the module is **libfipscrypto.o.**

Purpose of this module is to provide FIPS approved (and not approved but allowed) cryptographic functions to consuming applications via the Application Programming Interface (API).

2.2 Indicators

After completion of the required self-tests and initialization successfully, the module operates only in FIPS approved mode of operation. The utility functions and approved cryptographic services without CSPs are available. The HFCL module provides authenticated cryptographic services only when an approved role of Crypto Officer or User is logged in. The login is permitted only after the power on self test and initialization are successfully completed. The `FIPS_CRYPT_module_mode_set` function is used to login / logout / switch the Crypto Officer / User role to enable or disable the authenticated cryptography services.

The `FIPS_CRYPT_module_mode_set` function is used as follows:

```
FIPS_CRYPT_module_mode_set(iOnOff, authenticationData)
```

where `iOnOff` =**1** to enable the authenticated cryptography services or **0** to disable the authenticated cryptography services. And `authenticationData` contains the authentication password for the role.

The authenticated cryptography services are enabled only if the login is successful.

The indicator to show if a user is logged in to enable the authenticated cryptography services operation is provided by the `FIPS_CRYPT_module_mode` function as follows:

```
iOnOff=FIPS_CRYPT_module_mode()
```

where `iOnOff` =**1** if user is logged in/authenticated cryptography services are enabled or **0** if no user is logged in/authenticated cryptography services are not enabled.

The `FIPS_CRYPT_module_failed` function indicates if the role authentication or a Self Test operation failed. It returns `SEC_TRUE` if the HFCL module is in error state else `SEC_FALSE`.

2.3 Module Specification

2.3.1 CM boundary

The physical cryptographic boundary of the module is the general purpose computer on which it is installed. The logical cryptographic boundary of the module is the **libfipscrypto** object module itself.

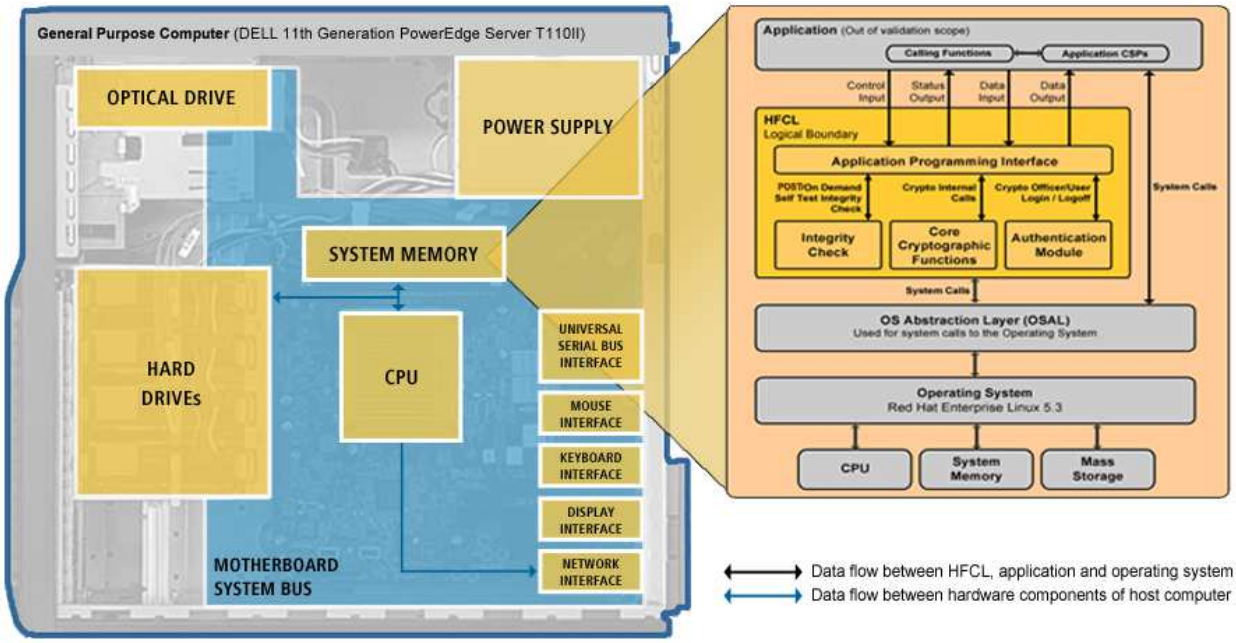

Figure 1 CM Boundary

Application Programming Interface: The interface, implemented in C language, which provides the consuming application access the services of the HFCL module.

Core Cryptographic Services: This is the cryptographic implementation provided by the Application Programming Interface.

Integrity Check: Internal module with in the cryptography module to implement the integrity check and to initiate the power on self test for the module.

Authentication Module: Internal module that provides role based authentication services to the Crypto Officer or User to enable the authenticated cryptography services.

Operating System Adaptation Layer (OSAL) Interface: This interface isolates the Cryptographic Module (CM) from the OS and provides interaction of the CM to the OS using abstracted APIs.

2.3.2 Approved cryptographic algorithms

The HFCL module supports the below approved security algorithms:

Algorithm Type	Algorithm	Standards	Use	Certificate#
Symmetric Key (AES)	AES – ECB (128/192/256) AES – CBC (128/192/256) AES – CFB (128/192/256) AES – CTR (128/192/256)	AES: FIPS 197, NIST SP 800-38A	Data encryption & decryption	3477
Symmetric Key (Triple-DES)	3-key Triple DES – ECB 3-key Triple DES – CBC 3-key Triple DES – CFB (8,64)	TRIPLE-DES: FIPS SP 800-67, FIPS SP 800-38A	Data encryption & decryption	1960
AEAD (Authentication Encryption with Associated Data)	AES – GCM (128/192/256) AES – GMAC (128/192/256)	NIST SP 800-38D	Data encryption & decryption	3477
Hashing	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	FIPS 180-4	Message Digest	2872
Keyed-Hash	HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	FIPS 198-1	MAC Calculation	2221
Cipher based MAC	AES – CMAC (128/192/256) 3-key Triple DES – CMAC	NIST SP 800-38B	CMAC Calculation	3477, 1960
Asymmetric Key	ECC CDH (KAS_ECC_DLC_Prim) Curves tested: P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571 KAS_ECC (scheme: EphemUnified) EB:P-224 EC:P-256 ED:384 EE: P-521	NIST SP 800-56A [§5.7.1.2]) [§6.1.2.2])	ECDH: Key Agreement	551, 552
	ECDSA Key Pair Gen, Sig Gen, Sig Ver PKG: CURVES(P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233	ECDSA: FIPS 186-4	ECDSA: Signature Generation &	707

	B-283 B-409 B-571 ExtraRandomBits) PKV: CURVES(ALL-P ALL-K ALL-B) SigGen: CURVES(P-224: (SHA-224, 256, 384, 512) P-256: (SHA-224, 256, 384, 512) P-384: (SHA-224, 256, 384, 512) P-521: (SHA-224, 256, 384, 512) K-233: (SHA-224, 256, 384, 512) K-283: (SHA-224, 256, 384, 512) K-409: (SHA-224, 256, 384, 512) K-571: (SHA-224, 256, 384, 512) B-233: (SHA-224, 256, 384, 512) B-283: (SHA-224, 256, 384, 512) B-409: (SHA-224, 256, 384, 512) B-571: (SHA-224, 256, 384, 512)) SigVer: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P-256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256, 384, 512) K-163: (SHA-1, 224, 256, 384, 512) K-233: (SHA-1, 224, 256, 384, 512) K-283: (SHA-1, 224, 256, 384, 512) K-409: (SHA-1, 224, 256, 384, 512) K-571: (SHA-1, 224, 256, 384, 512 B-163: (SHA-1, 224, 256, 384, 512) B-233: (SHA-1, 224, 256, 384, 512) B-283: (SHA-1, 224, 256, 384, 512) B-409: (SHA-1, 224, 256, 384, 512) B-571: (SHA-1, 224, 256, 384, 512))		Verification	
	DSA Key Gen, Sig Gen, Sig Ver Key Pair: [(2048,224) ; (2048,256) ; (3072,256)] SIG(gen)PARMS TESTED: [(2048,224) SHA(224 , 256 , 384 , 512); (2048,256) SHA(224 , 256 , 384 , 512); (3072,256) SHA(224 , 256 , 384 , 512);] SIG(ver)PARMS TESTED: [(1024,160) SHA(1 , 224 , 256 , 384 , 512); (2048,224) SHA(1 , 224 , 256 , 384 , 512); (2048,256) SHA(1 , 224 , 256 , 384 , 512); (3072,256) SHA(1 , 224 , 256 , 384 , 512)]	DSA: FIPS 186-4	DSA:Key generation Signature Generation & Verification	984
	RSA Key Gen, PKCS1 Sig Gen, PKCS1 Sig Ver KEY(gen): Fixed_e (10001) ; PGM(ProbRandom: (2048 , 3072) PPTT:(C.3) ALG[RSASSA-PKCS1_V1_5] SIG(gen) (2048 SHA(256 , 384 , 512)) (3072 SHA(256 , 384 , 512)) SIG(Ver) (1024 SHA(1 , 256 , 384 , 512	RSA: FIPS 186-4	RSA: Key generation Signature Generation & Verification	1785

	)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512))			
Deterministic Random Bit (DRBG)	Hash-DRBG HMAC-DRBG CTR-DRBG (AES) **Note**: For all DRBGs, the "supported security strengths" is just the highest supported security strength per [SP800-90] and [SP800-57]	NIST SP 800-90A	Random Bit generation	857

Table 1 - Approved Cryptographic Algorithms

2.3.3 Non-Approved cryptographic algorithms

The HFCL module supports the below non approved but allowed security algorithms.

- **ECDH:** Non-compliant DH scheme using elliptic curve, supporting all curves with order bit length >= 224 bits and defined in FIPS 186-4 and ANSI X9.62 except 176, 191 (Optimal Normal Basis), 208, 239 (Optimal Normal Basis), 272, 304, 368. Key agreement is a service provided for calling process use, but is not used to establish keys into the module. This key agreement scheme is considered Non-Approved but allowed as per IG D.8 Scenario 4 and IG D.11 implementation 4.
- **RSA**: The RSA algorithm may be used by the calling application for not compliant key transport method using a key encapsulation scheme. No claim is made for NIST SP 800-56B compliance, and no CSPs are established into or exported out of the module using these services. Supports key length of 2048-3072 bits. This key encapsulation scheme is considered Non-Approved but allowed as per IG D.9 allowed methods for key transport. This implementation is an RSA-based key methodology that uses key lengths specified in NIST SP 800-131A. No key derivation function is used along with this scheme, hence compliance with IG D.11 is not applicable.
- **DH**: The DH algorithm may be used by calling application for key establishment. This DH version is a variant on PKCS#3 and not the X9.42 specification. No claim is made for NIST SP800-56A, and no CSPs are established into or exported out of the module using these services. This key agreement scheme is considered Non-Approved but allowed as per IG D.8 Scenario 4 and IG D.11 implementation 4.

2.3.4 Excluded Components

The library is a software only module and all other hardware, software and components are excluded from the scope.

2.3.5 Critical Security Related Information

CSP Keys	Algorithm/ Modes	Generation/ input	Output	Storage	Usage	Zeroization
Symmetric Keys	AES-ECB(128/192/256) encrypt / decrypt key AES-CBC(128/192/256) encrypt / decrypt key AES-CFB (8 & 128) (128/192/256) encrypt / decrypt key AES-CTR (128/192/256) encrypt / decrypt key AES (128/192/256) CMAC generate / verify key AES (128/192/256) GCM/GMAC encrypt / decrypt key TRIPLE-DES-ECB(3-Key) encrypt / decrypt key TRIPLE-DES-CBC(3-Key) encrypt / decrypt key TRIPLE-DES-CFB (3-Key) encrypt / decrypt key Triple DES (3-Key) CMAC generate / verify key	Generated externally or internally/ input is in plain text as API parameters. The calling application is responsible for storage of the generated keys returned by the module.	Output to the consuming application during key generation.	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 symmetric key CSP persistently (beyond the lifetime of an API call).	Encryption/Decryption	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.
Asymmetric Key Pair	RSA (2048 - 3072 bits) signature generation key RSA (2048 - 3072 bits) key decryption (private key transport) key DSA (2048, 3072 bits) signature generation key ECDSA (PKG: CURVES(P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571 ExtraRandomBits) PKV: CURVES(ALL-P ALL-K ALL-B) signature generation key EC DH (Curves: P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571) private key agreement key DH (2048 – 3072 bits) private key agreement key	Generated externally or internally using NIST SP 800-90 compliant DRBG services	Output to the Consuming application.	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 asymmetric key CSP persistently (beyond the lifetime of an API call)	Signature generation/ Verification Key Pair Generation	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.

CSP Keys	Algorithm/ Modes	Generation/ input	Output	Storage	Usage	Zeroization
Per-message secret number	DSA Signature Generation per-message secret number up to (224 / 256) bits for (2048, 3072) mod bits. ECDSA Signature Generation per-message secret number up to order bits length for all curves with order bit length >= 224 (P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571). RSA signature generation blinding random number (2048 - 3072 bits).	Generated itself by the module using NIST SP 800-90 compliant DRBG services	Never outputs from the module	Stored in RAM in plaintext	Used for digital signing process.	Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs
HMAC Keys	HMAC-SHA1, HMAC-SHA2 (224, 256,384, 512) keys	Generated externally or internally/ input is in plain text as API parameters. The calling application is responsible for storage of generated keys returned by the module.	Output to the consuming application during key generation	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 HMAC key CSP persistently (beyond the lifetime of an API call)	MAC generation	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.

CSP Keys	Algorithm/ Modes	Generation/ input	Output	Storage	Usage	Zeroization
HMAC Key for Module integrity check and Role Authentication	HMAC-SHA-256 key	Generated externally and integrated into the module during build.	Never outputs from module	Stored in intermediate file used in the generation of an application and RAM as plaintext. Loaded into the module from the intermediate file by the module constructor when the module is installed.	Software integrity test in Self test. Password hash generation during Role Authentication.	Erased from the intermediate file in the module constructor once the HMAC key is loaded into the module. HMAC key stored in module is erased from RAM soon after calling FIPS_CRYPT_global_csp_cleanup API from consuming application and then uninstalling the module.
DRBG Internal State	Hash_DRBG V (440/888 bits) Hash_DRBG C (440/888 bits) HMAC_DRBG V (160/224/256/384/512 bits) HMAC_DRBG Key (160/224/256/384/512 bits) CTR_DRBG V (128 bits) CTR_DRBG Key (AES 128/192/256)	Generated itself by the module.	Never outputs from the module.	Stored in RAM as plaintext. The module stores DRBG state values for the lifetime of the DRBG instance.	Random bit generation	Erased from RAM soon after calling FIPS_CRYPT_rand_cleanup() API from consuming application.

CSP Keys	Algorithm/ Modes	Generation/ input	Output	Storage	Usage	Zeroization
DRBG Seed	HASH_DRBG HMAC_DRBG CTR_DRBG Entropy input length depends on the security strength. According to the [NIST SP 800-90] and [NIST SP 800-57] the minimum number of bytes of entropy believed to have been loaded by the application for each algorithm configurations are: Hash DRBG: ALGID_SHA1 - 16 bytes ALGID_SHA224 - 24 bytes ALGID_SHA256 -32 bytes ALGID_SHA384 - 32 bytes ALGID_SHA512 - 32 bytes HMAC DRBG: ALGID_HMAC_SHA1- 16 bytes ALGID_HMAC_SHA224 - 24 bytes ALGID_HMAC_SHA256 - 32 bytes ALGID_HMAC_SHA384 - 32 bytes ALGID_HMAC_SHA512 - 32 bytes CTR DRBG: ALGID_AES128_CTR - 16 bytes ALGID_AES192_CTR - 24 bytes ALGID_AES256_CTR - 32 bytes	Generated externally by the application. Provided as input in plain text as API parameters.	Never outputs from module	Stored in RAM as plaintext. The module uses CSPs passed as inputs by the calling application on the stack.	Input for DRBG random number generation.	The DRBG seed, provided to the module by the calling application, are destroyed when released by the appropriate API function calls. The calling application is responsible for zeroization of the parameters passed in and out of the module.

CSP Keys	Algorithm/ Modes	Generation/ input	Output	Storage	Usage	Zeroization
Crypto Officer Role Password Authentication (16-128 characters)	Authentication module HMAC-SHA-256 digest	Generated externally, input in plaintext.	Never outputs from module	Stored in intermediate file used in the generation of an application and RAM as plaintext. Loaded into the module from the intermediate file by the module constructor when the module is installed.	Role authentication for Crypto Officer	Erased from the intermediate file in the module constructor once the HMAC digest is loaded into the module. Erased from RAM soon after calling FIPS_CRYPT_global_csp_cleanup API from consuming application. HMAC digest stored in module is erased uninstalling the module.

CSP Keys	Algorithm/ Modes	Generation/ input	Output	Storage	Usage	Zeroization
Crypto User Role Passoword Authentication (16-128 characters)	Authentication module HMAC-SHA-256 digest	Generated externally, input in plaintext.	Never outputs from module	Stored in intermediate file used in the generation of an application and RAM as plaintext. Loaded into the module from the intermediate file by the module constructor when the module is installed.	Role authentication for Crypto User	Erased from the intermediate file in the module constructor once the HMAC digest is loaded into the module. Erased from RAM soon after calling FIPS_CRYPT_global_csp_cleanup API from consuming application. HMAC digest stored in module is erased uninstalling the module.

Table 2 - Critical Security Related Information

2.3.6 Security level 2 requirements

The CM meets the FIPS 140-2 requirements for an overall level 2 validation. The following table summarizes the individual FIPS 140-2 requirements, as well as the level implemented by the module for each section.

Section	Security Component	Security Level
1	Cryptographic module specification	2
2	Cryptographic module ports and interfaces	2
3	Roles, services and authentication	2
4	Finite state model	2
5	Physical security	N/A

6	Operational environment	2
7	Cryptographic key management	2
8	EMI/EMC (FIPS 140-2)	2
9	Self-test	2
10	Design assurance	2
11	Mitigation of Other Attacks	2

3 Cryptographic Module ports and interfaces

3.1 Physical ports and logical interfaces specification

The Huawei FIPS Crypto Library (HFCL) is a software based module and its interfaces are logical, that is, they are defined in terms of the Application Program Interface (APIs).

In this case, the physical interfaces are out of scope, since the application using the module is within the same PC and the module does not provide API functionality through physical interfaces.

The HFCL has four logical interfaces in the form of Application Program Interface (API) which can be categorized into the following FIPS 140-2 defined interfaces:

FIPS 140-2	**Logical interface**
Data Input	Input parameters of the API Function call.
Data Output	Output parameters of API Function call.
Control input	Control parameters of the API Function call.
Status Output	Status parameters of the API Function call, Error messages, return codes, and status parameters.

Table 3 – HFCL Interfaces

3.2 Information flows

All information flows of HFCL are through APIs as explained below.

	Interface	Description	Example
1	Data input	All data (except control data entered via the control input interface) enters via the "data input" interface which is processed by the cryptographic module enter through the data input interface.	Plaintext, cipher text, keys and CSP's, message length, initialization vector, authentication data, personalization string, random data input, additional input, key size, key type parameters etc
2	Data output	All data (except status data output via the status output interface) exits via the "data output" interface, which is output from the cryptographic module. All data output via the data output interface is stopped when an error state exists and during self-tests.	On successful execution of service(s): Plaintext, cipher text, public / private key pair, message digest, MAC, generated random numbers and key parameters.
3	Control input	All input commands and control data is entered via the "control input" interface. These include, algorithm IDs used to set the type of encryption and decryption operation; the hash algorithm IDs used to set the digest operation; and MAC algorithm IDs used to set the MAC operation enter through the control input interface.	ulAlgType, for example: Set type of encryption/decryption `ALGID_AES256_CBC, ALGID_RSA` Hash Algorithm ID `ALGID_SHA224` MAC Algorithm ID `ALGID_HMAC_SHA224`
4	Status output	All output signals including error indicators and status messages displayed via the "status output" interface are used to indicate the status of a cryptographic module.	Successful return status `SEC_SUCCESS, SEC_TRUE` Unsuccessful return status `SEC_CRYPT_ERR_SELF_TEST_FAIL, SEC_ERROR, SEC_FALSE, SEC_ERR_INVALID_ARG, SEC_ERR_INVALID_ALGID, SEC_CRYPT_ERR_INVALID_CTX, and SEC_ERR_MALLOC_FAIL`

Table 4 – Information Flow

The entire APIs are described in the FIPS Cryptography Developer Guide along with logical interface classification (Data input, Data Output, Control Input and Status Output).

For example:

Parameters of FIPS_CRYPT_hmacInit API are described as follows:

Parameter Name	Input/Output	Description	Range/Default Value
ulAlgType	Input	(Control Input) The algorithm id for HMAC	N/A
pucKey	Input	(Data Input) Key for HMAC	N/A
ulKlen	Input	(Data Input) Key length	N/A
pCtx	Output	(Data Output) The context which will be filled	N/A

Return Value	Description	Value	Action
SEC_UINT32	(Status Output) If context is initialized	SEC_SUCCESS	N/A
SEC_UINT32	(Status Output) If algorithm undefined	SEC_ERR_INVALID_ALGID	n/A
SEC_UINT32	(Status Output) If the key length is <112 bits	SEC_ERR_INVALID_KEY_LEN	N/A
SEC_UINT32	(Status Output) If memory alloc for context failed	SEC_ERR_MALLOC_FAIL	N/A
SEC_UINT32	(Status Output) If arguments are missing	SEC_ERR_INVALID_ARG	N/A
SEC_UINT32	(Status Output) If the selftest or authetication is already failed	SEC_CRYPT_ERR_SELF_TEST_FAIL	N/A

4 Access control policy

4.1 Roles

The module implements the required authenticated roles of Crypto Officer and User when using the authenticated cryptography services. The module allows for a single user to be logged in at a time and does not allow concurrent operators. If the user is logged in and another login is attempted, the current user will be automatically logged off and the other user will be logged in.

The Crypto User or Crypto Officer role is assumed by passing the appropriate password to the `FIPS_CRYPT_module_mode_set` function when enabling the cryptography services.

SEC_INT FIPS_CRYPT_module_mode_set (SEC_INT iOnOff, const SEC_CHAR *pcAuth)

Incorrect authentication will result in an error state where in all operations are stopped requiring a re-initialization of the module. Publicly available services are those that are also available without role authentication.

Both roles have access to all the services provided by the module. The Crypto Officer role additionally is responsible for the module installation.

Role	Description
Crypto Officer (CO)	CM Installation including: • Control the access before and after the installation. • Management of physical access to the computer. • Security of the Cryptographic module, Integrity Test and Role Authentication code within the application. • Management of security features provided by the operating system. CM Initialization Self Test Show Status Login to enable cryptography services (involving CSPs) Logout to disable cryptography services (involving CSPs) Cryptographic functions using the APIs • Symmetric cipher • Asymmetric cipher • Hashing and MAC • Digital signature generation & verification

	• Pseudo-random number generation • Key-generation • Key / CSP zeroization • Key Exchange Utility
User	CM Initialization Self Test Show Status Login to enable cryptography services (involving CSPs) Logout to disable cryptography services (involving CSPs) Cryptographic functions using the APIs • Symmetric cipher • Asymmetric cipher • Hashing and MAC • Digital signature generation & verification • Pseudo-random number generation • Key-generation • Key / CSP zeroization • Key Exchange Utility
Public	CM Initialization Self Test Message Digest operations Show Status Other services (No CSPs involved) Utility

Table 5 – User Roles

4.2 Services

4.2.1 CM Services

The following services refer to all the operations or functions that are performed by the cryptographic module involving API functions. Service inputs consist of data and control inputs to the cryptographic module. Similarly service outputs consist of data and status output from the cryptographic module.

The following table lists the HFCL services available to the roles:

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				
Show Module Information and Status	HFCL module version `FIPS_CRYPT_module_version`	Yes	Yes	Yes	None	Module Version	No	NA
	Get status of the authenticated user role logged in. If a user is logged in, the authenticated cryptography services are available. If no user is logged in then authenticated cryptography services are not available. `FIPS_CRYPT_module_mode`	Yes	Yes	Yes	None	Returns **0** (zero) – No authenticated user logged in **1** (one) – Authenticated user is logged in	No	NA
	Get currently logged in user. `FIPS_CRYPT_get_user`	Yes	Yes	Yes	None	Returns **0** (zero) – No authenticated user logged in. **1** (one) – Crypto Officer is logged in. **2** (two) – User is logged in	No	NA
	`FIPS_CRYPT_module_failed`	Yes	Yes	Yes	None	If the selftest fails or authentication is failed then this function will return **1** (one) else **0** (zero).	No	NA

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				
Module Initialization	Initializes HFCL `FIPS_CRYPT_libraryInit` Not intended as a user operation. Internal locks required for library are automatically initialized after power on self tests are successful	Yes	Yes	Yes	None	`SEC_SUCCESS, SEC_ERR`	No	NA
Self tests	Run self tests of all FIPS supported algorithms. Also runs the software integrity check. `FIPS_CRYPT_selftest`	Yes	Yes	Yes	None	`SEC_TRUE, SEC_FALSE`	No	NA

	Self test of individual modules
	`*FIPS_CRYPT_selftest_hmac`
	`*FIPS_CRYPT_selftest_aes`
	`*FIPS_CRYPT_selftest_des`
	`*FIPS_CRYPT_selftest_sha1`
	`*FIPS_CRYPT_selftest_rsa`
	`*FIPS_CRYPT_selftest_drbg`
	`*FIPS_CRYPT_selftest_drbg_all`
	`*FIPS_CRYPT_selftest_ecdsa`
	`*FIPS_CRYPT_selftest_dsa`
	`*FIPS_CRYPT_selftest_ecdh`
	`*FIPS_CRYPT_selftest_cmac`
	`*FIPS_CRYPT_selftest_aes_gcm`
	`*FIPS_CRYPT_selftest_aes_ctr`
	`*FIPS_CRYPT_post_set_callback`
		Yes	Yes	Yes	None	`SEC_TRUE, SEC_FALSE`	No	NA

Enable Authenticated Cryptography Services / Login Crypto Officer or User	Login the Crypto Officer or User to enable the authenticated cryptography services. `FIPS_CRYPT_module_mode_set`	Yes	Yes	Yes	Flag to login and enable authenticated cryptography services or logout and exit authen	SEC_TRUE if Crypto officer/ user is logged in (authenticated cryptography services enabled), SEC_FALSE if Crypto	Crypto Officer/User Role Password Authentication	Read access to Crypto Officer/User Role Password Authentication

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				
					ticated cryptography services, Crypto Officer /User password digest.	officer/ user not logged in (authenticated cryptography services disabled),		
Key / CSP zeroization	Zeroization of the global DRBG context. `FIPS_CRYPT_rand_cleanup`	Yes	Yes	Yes	None	None	Yes	Read and write access to CSP which needs to be zeroized.
	Stack cleanup (Calling application is responsible) `FIPS_ipsi_cleanseData`	Yes	Yes	Yes	Pointer to the stack data and length to be cleaned up	None	Yes	
	Cleaning the HMAC key and digests used for authentication `FIPS_CRYPT_global_csp_cleanup`	Yes	Yes	Yes	None	None	Yes	

	Free memory used for key
	`FIPS_CRYPT_PKEY_free`
	`IPSI_EC_PARA_FREE (Macro)`
		Yes	Yes	Yes	Key to be freed	None	Yes	

Random number generation	`FIPS_CRYPT_rand_init` **Note**：`FIPS_CRYPT_rand_init` must be called immediately after `FIPS_CRYPT_module_mode_set`.	Yes	Yes	No	DRBG algorithm type, Personalization String, Entropy, Nonce, Additi	SEC_SUCCESS, else error.	DRBG Internal State DRBG Entropy Input	Read Write Uses and updates the DRBG CSPs

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				
					onal input, Reseed source.			

	Used for random numbers
	`FIPS_CRYPT_drbg_get_app_data`
	`FIPS_CRYPT_rand_status`
	`FIPS_CRYPT_rand_cleanup`
	`FIPS_CRYPT_rand_strength`
	`FIPS_CRYPT_rand_health_check`
		Yes	Yes	Yes	None	On success, the status output of the operation, else error.	DRBG Internal State	Read Write
								Uses and updates the DRBG CSPs

	`FIPS_CRYPT_rand_bytes`
	`FIPS_CRYPT_rand_seed`
	`FIPS_CRYPT_rand_add`
		Yes	Yes	No	Required number of DRBG random bytes, actual number of DRBG random bytes outputted.
					Seed Value and Seed Length
					Along with the
						SEC_SUCCESS, else error.
						Random number on success
							DRBG Internal State
							DRBG Entropy Input
								Read Write
								Uses and updates the DRBG CSPs

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				
					above the entropy strength of seed value is passed to the rand_add function.			
Asymmetric key generation	Used to generate DSA, ECDSA and RSA keys for Sign Generation and Verification. *FIPS_CRYPT_PKEY_new FIPS_CRYPT_genKeyPair FIPS_CRYPT_RSA_genKeyPair_cb *FIPS_CRYPT_PKEY_free *FIPS_ipsi_dh_new *FIPS_ipsi_dh_free FIPS_ipsi_dh_generate_key *FIPS_ipsi_dh_size *FIPS_ipsi_dh_check *FIPS_CRYPT_PKEY_bits FIPS_CRYPT_PKEY_copy FIPS_CRYPT_PKEY_copyPrivKey FIPS_CRYPT_PKEY_copyPubKey FIPS_CRYPT_PKEY_getAlgParams FIPS_CRYPT_PKEY_copyParam FIPS_CRYPT_PKEY_privORpub FIPS_CRYPT_PKEY_setAlgParams *FIPS_CRYPT_PKEY_size *FIPS_CRYPT_checkECKey *FIPS_CRYPT_createECParamsById_ex *FIPS_CRYPT_ecPointToOcts *FIPS_CRYPT_octsToECPoint *FIPS_CRYPT_cmpECPoint	Yes	Yes	No *APIs marked as utility are available for Public	Type of algorithm / callback function to be called after the specified input time out	SEC_SUCCESS, SEC_ERR or SEC_ERR_Detail. New key pair structure, Generated key pair.	Asymmetric Key Pair	Write Generates new key pair

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				

Symmetric Encryption and decryption	Used to encrypt and decrypt data (single or multiple blocks) using AES (128/192/256 ECB, CBC, CFB, GCM, CTR) and TRIPLE-DES (3-key ECB, CFB)
	`FIPS_CRYPT_encrypt`
	`FIPS_CRYPT_encryptInit`
	`FIPS_CRYPT_encryptUpdate`
	`FIPS_CRYPT_encryptFinal`
	`FIPS_CRYPT_decrypt`
	`FIPS_CRYPT_decryptInit`
	`FIPS_CRYPT_decryptUpdate`
	`FIPS_CRYPT_decryptFinal`
	`FIPS_CRYPT_aeadInitSession`
	`FIPS_CRYPT_aeadOp`
	`*FIPS_CRYPT_aeadRemoveSession`
	`FIPS_CRYPT_symInitSession`
	`FIPS_CRYPT_symOp`
	`*FIPS_CRYPT_symRemoveSession`
	`*FIPS_CRYPT_freeCtx`
	`*FIPS_CRYPT_SYM_blockSize`
	`*FIPS_CRYPT_SYM_ivLen`
	`*FIPS_CRYPT_SYM_keyLen`
	`*FIPS_CRYPT_SYM_mode`
	`*FIPS_CRYPT_SYM_padType`
	`FIPS_CRYPT_setFeedbackSize`
	`*FIPS_CRYPT_getFeedbackSize`
		Yes	Yes	No
				*APIs marked as utility are available for Public
					Control input that decides the algorithm to be used, Key, IV and Data to be encrypted / decrypted	Status of the operation and
						Cipher text / plain text
							Key used.	Read Write
								Read Write access to the key

MAC	Used to generate or verify data integrity using AES (128/192/256 GMAC, CMAC), TRIPLE-DES (3-key CMAC) and HMAC-SHA1, HMAC-SHA2 (224, 256,384, 512).
	`*FIPS_CRYPT_HMAC_size`
	`*FIPS_CRYPT_HMAC_hashType`
	`FIPS_CRYPT_hmac`
	`FIPS_CRYPT_hmacInit`
	`FIPS_CRYPT_hmacUpdate`
	`FIPS_CRYPT_hmacFinal`
	`*FIPS_CRYPT_CMAC_size`
		Yes	Yes	No
				*APIs marked as utility are available for Public
					Control input that decides the algorithm to be used, the input data, length of data	Status of the operation and MAC data	Key used	Read Write
								Read Write access to the key

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				
	FIPS_CRYPT_cmac FIPS_CRYPT_cmacInit FIPS_CRYPT_cmacUpdate FIPS_CRYPT_cmacFinal FIPS_CRYPT_aeadInitSession FIPS_CRYPT_aeadOp *FIPS_CRYPT_aeadRemoveSession							
Message digest	Used to generate SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 message digests. FIPS_CRYPT_digest FIPS_CRYPT_digestInit FIPS_CRYPT_digestUpdate FIPS_CRYPT_digestFinal FIPS_CRYPT_ctxdigestFinal FIPS_CRYPT_digestReset FIPS_CRYPT_digestFree FIPS_CRYPT_MD_size FIPS_CRYPT_MD_blockSize FIPS_CRYPT_HashCtxCopy FIPS_CRYPT_HASH_ALG	Yes	Yes	Yes	Control input that decides the algorithm to be used and data for which digest is to be calculated	Status of the operation and Message digest	No	NA
Key agreement	Used to perform key agreement primitives on behalf of the calling process (does not establish keys into the module). Computes DH, ECDH shared secret keys. FIPS_CRYPT_computeDHKey FIPS_ipsi_dh_compute_key FIPS_ipsi_dh_compute_key_padded FIPS_CRYPT_computeECDHKey_ex *FIPS_CRYPT_setDHSharedKeyPadScheme *FIPS_IPSI_CRYPT_dhConstTimeOff *FIPS_IPSI_CRYPT_dhConstTimeOn *FIPS_CRYPT_createDHParams	Yes	Yes	No *APIs marked as utility are available for Public	Own private key and Peer public key	Status of the operation and shared secret Keys.	DH Private Key agreement key, ECDH Private Key agreement key	Read Write Read Write access to the private keys.
Key transport	Used to encrypt or decrypt a key value on behalf of the calling process (does not establish keys into the module). Performs key	Yes	Yes	No	Peer public key or	Status of the operation	RSA key decryp	Read Write Read Write access to

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				

	(private) transport using RSA key decryption key.
	`FIPS_CRYPT_pubKeyEncrypt`
	`FIPS_CRYPT_privKeyDecrypt`
	`FIPS_CRYPT_privKeyEncrypt`
	`FIPS_CRYPT_pubKeyDecrypt`
					own private key	and encrypted/ decrypted private key.	tion key	the private keys.

Digital Signature	Used to generate and verify RSA, DSA and ECDSA digital signatures.
	`FIPS_CRYPT_sign`
	`FIPS_CRYPT_signInit`
	`FIPS_CRYPT_signUpdate`
	`FIPS_CRYPT_signFinal`
	`FIPS_CRYPT_signData_ex`
	`FIPS_CRYPT_verify`
	`FIPS_CRYPT_verifyInit`
	`FIPS_CRYPT_verifyUpdate`
	`FIPS_CRYPT_verifyFinal`
	`FIPS_CRYPT_verifyData_ex`
	`*FIPS_CRYPT_PKEY_sign_size`
	`*FIPS_CRYPT_rsaBlindingAndConstTimeOff`
	`*FIPS_CRYPT_rsaBlindingAndConstTimeOn`
	`*FIPS_CRYPT_enableRsaSignWithPrvExp`
	`*FIPS_CRYPT_disableRsaSignWithPrvExp`
	`FIPS_CRYPT_signDec`
	`FIPS_CRYPT_signEnc`
	`FIPS_CRYPT_RSAPubKeyEncode`
	`FIPS_CRYPT_RSAPubKeyDecode`
		Yes	Yes	No
				*APIs marked as utility are available for Public
					Control input that decides the algorithm to be used, key (Private key to sign and public key to verify) and data.	Status of the operation and generated signature (For sign operation).	RSA, DSA and ECDSA signature generation key	Read Write
								Read Write access to the key.

Utility	Miscellaneous helper functions.
	`FIPS_ipsi_bn_new`
	`FIPS_ipsi_bn_free`
	`FIPS_ipsi_bn_bin2bn`
	`FIPS_ipsi_bn_bn2bin`
	`FIPS_ipsi_bn_num_bits`
	`FIPS_SEC_setLogCallback`
	`FIPS_SEC_cmpBigInt`
	`FIPS_SEC_cpyBigInt`
		Yes	Yes	Yes	Specific to each function	Status and output, if any, specific to each function	No	NA

Service	Description	Role			Input	Output	CSPs	Type of access to CSPs
		Crypto Officer	User	Public				

	`FIPS_SEC_cvtToBigInt`
	`FIPS_SEC_assignToBigInt`
	`FIPS_CRYPT_setEcpCallback`
	`FIPS_CRYPT_setBnCallback`							

Note:

- API functions marked with (*) indicate that it is a utility function and is available to Public (all users).
- The FIPS approved cryptographic services are only available to the authenticated roles of Crypto Officer and User.

Table 6 – CM Services

4.2.2 Service Bypass

The HFCL does not support service bypass.

4.3 Authentication

The HFCL module supports role based authentication to gain access to the cryptographic services.

4.3.1 Authenticated Roles

The roles and required authentication mechanism provided by the CM are detailed in the following table, where the role name, the type of authentication, the authentication data and the strength of the mechanism are provided:

Role	Type of authentication	Authentication data	Strength
Crypto-officer	Password based authentication. HMAC-SHA256 digest of the input password is computed and equated with the expected one.	Password	Password is required to be minimum of 16 characters and maximum of 128 in length. The password shall contain digits, alphabets and special characters.
Crypto-User	Password based authentication. HMAC-SHA256 digest of the input password is computed and equated with the expected one.	Password	Password is required to be minimum of 16 characters and maximum of 128 in length. The password shall contain digits, alphabets and special characters.

Table 7 – Authentication

Since minimum password length is 16 characters from 95 human readable characters (ASCII lower case, upper case, digits and non-alphanumeric characters). The probability of a random successful authentication attempt in one try is a maximum of (1/95)^16, or less than one in 1,000,000. The module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

The module does not support change role facility. An operator has to re-authenticate himself to change his role.

4.3.2 CM Access Control

The CM distinguishes between public services and authenticated role services. The authentication flow is as follows:

CM State	Authentication Required	Details
Power On	No	Power on state is invoked when the module is first loaded. The module automatically enters the Power On Self Test state.
Power On Self Tests	No	CM does not allow any services till the self tests are successfully completed. On successful completion, the module enters the Ready state. On any errors during self tests, module enters the Error state.
Ready State	No	Only public services such as self tests and utility functions are available. Cryptography services are limited to functions without CSPs and Non FIPS functions, for example, digest operations.
Authenticated Cryptography Services State	Yes	Authentication of an approved role (Crypto Officer or User) is required to access CSPs and FIPS algorithms. `FIPS_CRYPT_module_mode_set` is used to login, logout and switch a user role. Authentication data passed to FIPS_CRYPT_module_mode_set is the password for Crypto Officer or User. Function can be called for the first time to login a user role. Successful authentication will enable the authenticated cryptography services. Unsuccessful login will take the module to Error state. Function can be called to switch user roles by logging in a role when another role is already logged in. Successful authentication will switch (if different role) or relogin (if same role) and continue the authenticated cryptography services. Unsuccessful login will take the module to Error state. Function can be called to logout the current role. On logout the module will enter the Ready state. **Note**: In Error state, all CM services are stopped and CM must be reloaded by the application.

Table 8 – Authentication States

4.3.3 Authentication Indicators and Feedback

The module does not provide any feedback of the reasons when a failure in the authentication occurs. No output of authentication data is performed. Only the success or respective error code in case of failure is returned. The return value does not provide information that could be used to guess or determine the password.

- **On Successful Authentication**: On successful authentication, the CM will log the successful authentication event enter the authenticated cryptography services state. If a user role was already

logged in, then the module will log the authentication event and continue in the authenticated cryptography services state.

- **On unsuccessful Authentication:** On unsuccessful authentication, the CM will log the unsuccessful authentication event enter the error state. If a user role was already logged in, then the module will log the unsuccessful authentication and enter the error state.
- **On Log out**: On logging out, the CM will log the log off event enter the Ready state.

4.3.4 Invalid Authentication

Any attempt to authenticate with an invalid password will result in a standard and common error message. The module will immediately go to error state rendering the module unable to enter or continue any cryptographic operation. Subsequent authentication would require the module returning to power on/off state and performing the power on self tests initialization before trying authentication again. When module returns to the power off state, the previous authentication information handled by the login flag is disabled.

4.3.5 Protection of Password within the CM

The password values are specified at build time as digests and must have a minimum length of 16 characters. The authentication data is added to the Module during the installation process, by the Crypto Officer, and otherwise cannot be accessed. HMAC-SHA-256 digests of the passwords are integrated into the module at build time. It is the responsibility of the operator to secure the authentication data against unauthorized disclosure, modification and substitution of the authentication data.

5 Physical Security policy

The FIPS Cryptography library is a software based cryptographic module and thus does not claim any direct requirements for physical security.

6 CM operational environment tested

HFCL is capable of running in and tested for FIPS 140-2 Level 2 mode on a general purpose operational environment as follows:

- DELL PowerEdge T110 II Intel Pentium w/ RHEL 5.3 evaluated at EAL4
- The operating system meets the functional requirements specified in the "Controlled Access Protection Profile (CAPP)" defined in Annex B of the FIPS 140-2 and is evaluated at the CC evaluation assurance level EAL4 (https://www.commoncriteriaportal.org/files/epfiles/st_vid10338-st.pdf).

6.1 Operational Rules

The Huawei FIPS Cryptography Library (HFCL) module operates in a modifiable operational environment as per the FIPS 140-2 definition. The following rules must be adhered to for operating the module in a FIPS 140-2 compliant manner:

- The Crypto Officer shall install the cryptographic module correctly following the guidance.
- All host system components that can contain sensitive cryptographic data (main memory, system bus, disk storage) must be located within a secure environment.
- All passwords for the approved roles must be secured.
- To use the module, it must pass all self tests and be initialized before using it.
- When the application finishes using the module, the logged user must be logged out, the application stack must be cleaned calling to the FIPS_CRYPT_rand_cleanup and FIPS_CRYPT_global_csp_cleanup API functions.
- The Crypto Officer shall be adequately-trained and non-hostile.
- Administrative and Crypto Officer privileges must not be extended to unauthorized users/persons.
- The authenticated cryptography services are available only after an approved role is properly logged in.
- The application should be installed in a location protected by the host operating system security features.
- The application designer must be sure that the client application is designed correctly and does not corrupt the address space of the module.
- All Critical Security Parameters are verified as correct and are securely generated, stored, and destroyed.
- Secret or private keys that are input to or output from an application must be input or output within the cryptographic boundary of the module. Note that keys exchanged between the application and the HFCL are through the defined API in plaintext.

Note:

- The password must be at least 16 characters long and should adhere to the password complexity by using a combination of upper and lower case alphabets, digits and special characters.

6.2 Compatible platforms

The HFCL module is supported on and is tested in FIPS 140-2 Level 2 mode on the following Common Criteria-evaluated platform(s):

- DELL PowerEdge T110 II Intel Pentium w/ RHEL 5.3 evaluated at EAL4 (https://www.commoncriteriaportal.org/files/epfiles/st_vid10338-st.pdf).

7 Electromagnetic Interference/Electromagnetic Compatibility

The FIPS Cryptography library is installed on a DELL PowerEdge T110 II.

The DELL computer mentioned is a FCC–compliant computer which was calibrated to comply with FCC norms for class B digital devices. This is described in the technical guidance provided by the manufacturer.

EMI/EMC of Machines	FCC Rating
DELL PowerEdge T110 II	Class B

8 Mitigation of other attacks

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

9 Self-tests

9.1 Brief description

The HFCL performs tests at power on, on demand and on condition to check the health and integrity of the module and the cryptographic algorithms. These include:

- **Power on self test**
    - Integrity test
    - Cryptographic algorithm tests
- **Conditional tests**
    - Pair-wise consistency test
    - Continuous DRBG test
    - Continuous Entropy source test
    - DRBG health tests

9.2 Power on self test

The HFCL module performs self test automatically when loaded by using gcc provided attribute constructor. This ensures self test is called before any application function is executed. In addition, on demand self test can be performed by calling `FIPS_CRYPT_selftest()`.

On successful completion of the on demand tests the module returns to the service state of the operator role, who implemented the call.

When the power-on self tests are executed successfully, there is no output given by the module. Start of the Power-on self test and successful completion is logged in the system log.

To know that the execution of the power on selftest is started or finished, application can check the system log file "**/var/log/messages**". If the POST is passed, the log will print as

HFCL_<VERSION>[<PID>]: POST is started

HFCL_<VERSION> [<PID>]: FIPS Self-Test Passed

HFCL_<VERSION> [<PID>]: POST is successfully finished

When the on demand self tests are executed successfully, the function returns a SEC_TRUE value.

To know that the FIPS Module is in this state, application has to set the post *callback* by using the API FIPS_CRYPT_post_set_callback. When the self test is started then the *callback* is called with the parameters

callback (FIPS_CRYPTO_SELF_TEST_BEGIN, 0, 0, SEC_NULL);

and when the self-test ends then the *callback* is called with the following parameter

callback (FIPS_CRYPTO_SELF_TEST_END, SEC_TRUE, 0, NULL); ---> (in case of success)

With system log: HFCL_<VERSION>[<PID>]: FIPS Self-Test Passed

or

callback (FIPS_CRYPTO_SELF_TEST_END, SEC_FALSE, 0, NULL); ---> (in case of failure)

With system log: HFCL_<VERSION>[<PID>]: FIPS Self-Test Failed

When the self tests (power-on or on demand) are executed and a failure occurs, the module enters the error state and the data output interface is inhibited. While the error state exists, all cryptographic services are also inhibited.

API function FIPS_CRYPT_module_failed() can be used to know whether the library is in error state or not**.**

The module writes system log messages and enters the error state for the following error cases:

1. Power-on/On-demand self test failures, for example:

   HFCL_<VERSION> [<PID>]: FIPS Self-Test Failed

   HFCL_<VERSION> [<PID>]: FIPS_check_incore_fingerprint: Fingerprint does not match

2. Authentication failure, for example:

   HFCL_<VERSION> [<PID>]: Authentication failed, FIPS module is in error state

3. Conditional test failure, for example:

   HFCL_<VERSION> [<PID>]: DRBG PRNG Test Failed, FIPS module is in error state

   HFCL_<VERSION> [<PID>]: Conditional test failed, FIPS module is in error state

4. Health test failure, for example:

   HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Instantiation Failure

   HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Uninstantiation Failure

The module will go into the **error** state whenever a failure or error occurs that requires the module to be re-initialized. This is done by first uninitializing the module and then returning to the power on state. The DRBG is also uninstantiated.

9.2.1 Integrity test

The HFCL module uses the FIPS-Approved algorithm, HMAC based on SHA-256, for its integrity test. When HFCL is built, a HMAC fingerprint is precomputed and is embedded within **libfipscrypto.o**.

When the module is initialized by the consuming application, it generates the fingerprint again. This runtime generated fingerprint is compared with the precomputed fingerprint for verification. If the two fingerprint values are the same, then the integrity test is successful. If the integrity test fails, the module enters the error state and the services and data outputs are stopped.

9.2.2 Cryptographic algorithm test

There are tests performed to check the health and integrity of the approved algorithms used in cryptographic library The self tests can be called on demand by calling FIPS_CRYPT_selftest() to execute all self tests or by calling the individual test APIs listed in the table below:

Approved Algorithms	Tests
AES (ECB, CBC, CFB,CTR)	• AES-ECB KA Encryption Test for 128, 192 and 256 bit key length • AES-ECB KA Decryption Test for 128, 192 and 256 bit key length • AES-CBC KA Encryption Test for 128, 192 and 256 bit key length • AES-CBC KA Decryption Test for 128, 192 and 256 bit key length • AES-CFB KA Encryption Test for 128, 192 and 256 bit key length • AES-CFB KA Decryption Test for 128, 192 and 256 bit key length • AES-CTR KA Encryption Test for 128, 192 and 256 bit key length • AES-CTR KA Decryption Test for 128, 192 and 256 bit key length
AES – CMAC Triple DES – CMAC	• AES CMAC generate and verify for 128, 192 and 256 bit key lengths • Triple DES CMAC generate and verify (3-Key)
AES GCM	• AES-GCM KA Encryption Test for 128, 192 and 256 bit key lengths • AES-GCM KA Decryption Test for 128, 192 and 256 bit key lengths
AES CTR	• AES-CTR KA Encryption Test using 128, 192 and 256 bit key lengths • AES-CTR KA Decryption Test using 128, 192 and 256 bit key lengths
Triple DES (ECB, CBC, CFB)	• Triple DES-ECB KA Encryption Test (3-Key) • Triple DES-ECB KA Decryption Test (3-Key) • Triple DES-CBC KA Encryption Test (3-Key) • Triple DES-CBC KA Decryption Test (3-Key) • Triple DES-CFB KA Encryption Test (3-Key) • Triple DES-CFB KA Decryption Test (3-Key)
SHA-1	• SHA-1 Short message and Long message KA Hashing Test
HMAC SHA-1, HMAC-SHA-2 (224, 256, 384, 512)	• HMAC SHA-1 KA Hashing Test • HMAC SHA-2 KA Hashing Test (SHA224, SHA256, SHA384 and SHA512) Short message and Long message chosen for KA Hashing Test
ECC CDH (KAS)	• Shared secret calculation per NIST SP 800-56A §5.7.1.2, IG 9.6
ECDSA	• KA Sign and verify test using P-224 (SHA-256)

DSA	• KA Sign and verify test using 2048 and 3072 bit key (SHA-256)
RSA	• KA Sign and verify test using 2048 and 3072 bit key (SHA-256, PKCS#1)
Hash-DRBG HMAC-DRBG CTR-DRBG (AES)	For details about extended KAT and extensive error checking performed during DRBG health testing please refer to section 9.3.4. During a Power On Self Test (POST) an abbreviated KAT only is performed for one instance of each supported DRBG mechanism. This simply performs an instantiate and generate operation and checks that the output matches an expected value. • HASH_DRBG: SHA256 • HMAC_DRBG: HMAC-SHA256 • CTR_DRBG: AES (256 bit) with and without derivation function

Table 9 – Cryptographic Algorithms Test

Critical Functions Test

The DRBG algorithms (HASH_DRBG, HMAC_DRBG and CTR-DRBG) implemented in module are critical functions for the secure operation of the module. Extensive health testing and error testing is implemented on each one (instantiate, generate and reseed) to ensure correct operation of DRBG. The details of health testing and error testing is given in the section 9.3.4 (DRBG health tests)

9.3 Conditional tests

9.3.1 Pair-wise consistency test

Algorithms	Condition	Tests
RSA	On each generation of a key pair	Pairwise consistency tests are performed for both possible modes of use - Sign/Verify and Encrypt/Decrypt. Use private key for signature generation and public key for signature verification Use public key for key encryption and private key for key decryption
DSA	On each generation of a key pair	Use private key for signature generation and public key for signature verification
ECDSA	On each generation of a key pair	Use private key for signature generation and public key for signature verification

Table 10 – Pair-wise Consistency Test

After key pair generation, if POST callback is set and application is intentionally failing the test then the module returns SEC_CRYPT_ERR_SELF_TEST_FAIL. Else during the key pair verification if the signature verification fails then also the module returns SEC_CRYPT_ERR_SELF_TEST_FAIL.

The possible error log messages due to the failure of pair-wise consistency test failure are:

- HFCL_<VERSION> [<PID>]: Conditional test failed, FIPS module is in error state

9.3.2 Continuous Random Number Generator Test for DRBG

The very first random number generated by the DRBG is only used for initiating the continuous DRBG test, which compares every newly generated number with the previously generated number.

The block length for different DRBG implementation is as follows:

Hash DRBG:

* ALGID_SHA1 => 20 bytes

* ALGID_SHA224 => 28 bytes

* ALGID_SHA256 => 32 bytes

* ALGID_SHA384 => 48 bytes

* ALGID_SHA512 => 64 bytes

HMAC DRBG:

* ALGID_HMAC_SHA1 => 20 bytes

* ALGID_HMAC_SHA224 => 28 bytes

* ALGID_HMAC_SHA256 => 32 bytes

* ALGID_HMAC_SHA384 => 48 bytes

* ALGID_HMAC_SHA512 => 64 bytes

CTR DRBG:

* ALGID_AES128_CTR => 16 bytes

* ALGID_AES192_CTR => 16 bytes

* ALGID_AES256_CTR => 16 bytes

The test fails if the new number is same as the previous number. If the Continuous DRBG test fails then HFCL module returns SEC_CRYPT_FIPS_DRBG_GENERATE_ERROR.

The possible error log messages due to the failure of continuous DRBG test failure are:

- HFCL_<VERSION> [<PID>]: DRBG PRNG Test Failed: Last Block Failure. FIPS module is in error state
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_generate: Generation failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_generate: Random bit generation failed

9.3.3 Continuous Entropy source test

The first block of the entropy source input is only used for initiating the continuous DRBG test, which compares each of the subsequent blocks with the previous blocks. The test fails if the new entropy block is same as the previous block.

Entropy source test is done at the time of entropy input by the application. HFCL module calls the get_entropy callback to get the entropy from application when reseed is needed or at the time of DRBG instantiate. If the entropy source test fails then HFCL module will return SEC_CRYPT_FIPS_ERR_DRBG_INSTANTIATE_RETRIEVING_ENTROPY.

The possible error log messages due to the failure of continuous entropy source test are:

- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_get_entropy: Entropy block length is 0 or 1
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_get_entropy: Entropy length returned is in-correct when the entropy is in blocks
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_get_entropy: Continuous PRNG test failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_instantiate: Get Entropy Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_instantiate: Error
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_rand_init_int : Error instantiating DRBG
- HFCL_<VERSION> [<PID>]: CRYPT_drbg_reseed: Get Entropy Failure
- HFCL_<VERSION> [<PID>]: CRYPT_drbg_reseed: Reseed failed. DRBG is in Error State
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_generate: Random bit generation failed

9.3.4 DRBG health tests

The health check on DRBG is performed in below mentioned cases and is considered a conditional self-test:

1. When a DRBG is first initialized (i.e. before instantiation) a health check is performed on a test instantiation using the same mechanism and parameters.
2. When DRBG is reseeded a health check is performed on a test instantiation. This effectively performs a superset of the requirements for testing the reseed function i.e. it tests all functions including the reseed function.
3. An internal counter determines the number of generate operations since the last health check. When this reaches a preset limit (`health_check_interval`) a health check is performed. This limit is set by default to 2^24 operations but it can be set to an alternative value by an application. The default `health_check_interval` value is same as the default reseed_interval which in turn is lower than the maximums specified in NIST SP 800-90.
4. When an application explicitly requests a health check with the call FIPS_CRYPT_rand_health_check ().

The DRBG health tests perform the tests as required by [NIST SP 800-90] Section 11. The integrity test during the power on self test ensures the integrity of the DRBG health tests subsequent to DRBG implementation validation testing.

Extended KAT for DRBG

During DRBG health check for the selected DRBG mechanism, extended Known Answer Test (KAT) of all functions is performed:

1. Instantiate DRBG.
2. Generate without prediction resistance and check output matches expected value.

3. Reseed
4. Perform second generate operation without prediction resistance and check output matches expected value.
5. Uninstantiate DRBG.
6. Instantiate DRBG.
7. Generate with prediction resistance and check output matches expected value
8. Perform second generate operation with prediction resistance and check output matches expected value.
9. Uninstantiate DRBG.

Extended Error DRBG Checking

During DRBG health check for the selected DRBG mechanism, extended error checking is performed: Invalid parameters are fed into all DRBG functions. For tests which make the DRBG in an error state, DRBG is uninstantiated and instantiated again to clear the error condition.

1. Test detection of too large personalisation string.
2. Test entropy source failure detection: i.e. returns no data.
3. Try to generate output from uninstantiated DRBG.
4. Test insufficient entropy.
5. Test too much entropy.
6. Test too small nonce.
7. Test too large nonce.
8. Instantiate with valid data and check generation is now OK.
9. Request too much data for one request.
10. Try too large additional input.
11. Check prediction resistance request fails if entropy source failure.
12. Set reseed counter to beyond interval. Generate output and check entropy has been requested for reseed.
13. Test explicit reseed operation with too large additional input.
14. Test explicit reseed with entropy failure.
15. Test explicit reseed with too much entropy.
16. Test explicit reseed with too little entropy.
17. Uninstantiate DRBG: check internal state is zeroed.

When an application explicitly requests health check with the call FIPS_CRYPT_rand_health_check, if POST callback is set and application is intentionally failing the test then the module returns SEC_ERR. Else during the health check if the KAT or Error Checking fails then also the module returns SEC_ERR.

When DRBG is first initiated, DRBG is reseeded and every `health_check_interval` call to the generate function, if the health test fails HFCL module returns SEC_CRYPT_ERR_DRBG_HEALTH_TEST_FAIL.

The possible error log messages due to the failure of DRBG health test are:

- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: Init Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: Instantiate Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: DRBG Generation Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: KAT Failure - Output Mismatch
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: Reseed Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: DRBG Generation with NO PR Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: KAT Failure - Output Mismatch (no PR)

- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: KAT Failure - Output Mismatch (with PR)
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_single_kat: KAT Failure - Output Mismatch 2 (with PR)
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Failure for Personalization string too large
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Failure for Entropy source test
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Failure for uninstantiated DRBG test
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Failure for uninstantiate DRBG
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Failure for insufficient entropy test
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Failure for insufficient entropy test for Uninstantiation
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Failure for too much entropy test for Uninstantiation
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Instantiation Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Uninstantiation Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Instantiation Failure - Too Large Nonce
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: UnInstantiation Failure - Too Large Nonce
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Too Much Data Request
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Too Large Additional Input
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Error for prediction resistance request fails if entropy source failure
- FIPS_CRYPT_drbg_error_check: Entropy Reseed failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Reset Counter Reset check failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Error
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Error Check prediction resistance request fails if entropy source failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Error - Generate output and check entropy has been requested for reseed
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Error - Reseed Counter reset
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Reseed
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: UnInstanitate
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Reseed Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Uninstantiate Failure

- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Reseed with too little entropy Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Zero setting Failure
- HFCL_<VERSION> [<PID>]: FIPS_CRYPT_drbg_error_check: Non Induced Error