DHSSL Cryptographic Module

Software Module Version No:1.0.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.0
Date: October 31, 2019

Prepared By:

Zhejiang Dahua Technology Co., Ltd.
No.1199, Bin'an Road, Binjiang District, Hangzhou, China
www.dahuasecurity.com

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose by Dahua

This non-proprietary Security Policy for the DHSSL cryptographic module by Zhejiang Dahua Technology Co., Ltd. describes how the module meets the security requirements of FIPS 140-2 and how to run the module in a secure FIPS 140-2 mode.

This document was prepared as part of the Level 1 FIPS 140-2 validation of the module. The following table lists the module's FIPS 140-2 security level for each section.

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

Table 1 - FIPS 140-2 Section Security Levels

1.2 Background

Federal Information Processing Standards Publication (FIPS PUB) 140-2 – *Security Requirements for Cryptographic Modules* details the requirements for cryptographic modules. More information on the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP), the FIPS 140-2 validation process, and a list of validated cryptographic modules can be found on the CMVP website:

http://csrc.nist.gov/groups/STM/cmvp/index.html

More information about Dahua products can be found on the Dahua website:

http://www.dahuasecurity.com

1.3 Document Organization

This non-proprietary Security Policy is part of the DHSSL cryptographic module FIPS 140-2 submission package. Other documentation in the submission package includes:

- Product documentation
- Vendor evidence documents
- Finite state model
- Additional supporting documents

The DHSSL cryptographic module is also referred to in this document as the cryptographic module, or the module.

2 Module Overview

The module is a set of software libraries, whose purpose is to provide cryptographic algorithm services (such as encryption and decryption services), as well as Transport Layer Security protocol (TLS) v1.0, v1.1, v1.2, certificate management, asymmetric key generation, random number generation and so on. The module provides API interface for application calling.

The module can act as a TLS server or TLS client, and interacts with other peers via the TLS protocol.

The module has been tested on the following platforms.

Tested Platform	Processor	Operating System
DHI-NVR5832-4KS2	ARM Cortex-A17+ARM Cortex-A7 (ARMv7 32bit)	Linux-3.10.0(32bit)
DHI-NVR5416-16P-4KS2E	ARM Cortex-A17+ARM Cortex-A7 (ARMv7 32bit)	Linux-3.10.0(32bit)
DHI-NVR5432-16P-4KS2E	ARM Cortex-A17+ARM Cortex-A7 (ARMv7 32bit)	Linux-3.10.0(32bit)
DHI-NVR5832-I	ARM Cortex-A73+ARM Cortex-A53 (ARMv8 64bit)	Linux-4.9.37(64bit)
DH-IPC-HFW5442EP-ZE	ARM Cortex-A7(ARMv7 32bit)	Linux-4.9.37(32bit)
DH-SD5A445XA-HNR	ARM Cortex-A7(ARMv7 32bit)	Linux-4.9.37(32bit)
DH-SD49225XA-HNR	ARM Cortex-A7(ARMv7 32bit)	Linux-4.9.37(32bit)
DH-IPC-HFW7442HP-Z	ARM Cortex-A53(ARMv8 32bit)	Linux-4.9.37(32bit)

Table 2 - Tested Platforms[1]

In table 2, DHI-NVR5832-4KS2, DHI-NVR5416-16P-4KS2E, DHI-NVR5432-16P-4KS2E and DHI-NVR5832-I belong to Network Video Recorder(NVR). DH-IPC-HFW5442EP-ZE, DH-SD5A445XA-HNR, DH-SD49225XA-HNR and DH-IPC-HFW7442HP-Z belong to Network Camera.

Zhejiang Dahua Technology Co., Ltd affirms that the module runs correctly on the following network camera and NVR models:

[1] the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment

- Network Cameras: DH-SD5A***XA-HNR, DH-SD49***XA-HNR, IPC-H*5** and IPC-H*7** model names (* characters vary depending on model).
- NVRs: DHI-NVR5***-4KS*, DHI-NVR5***-*P-4KS*E and DHI-NVR5***-I* (* characters vary depending on model).

All of the above vendor affirmed devices have the same processor and operating system as the ones tested in table.

2.1 Cryptographic Module Specification

The cryptographic module is installed into embedded device. The cryptographic module embodiment is classified as multi-chip standalone. The physical and logical boundaries of the module is depicted in the figure below.

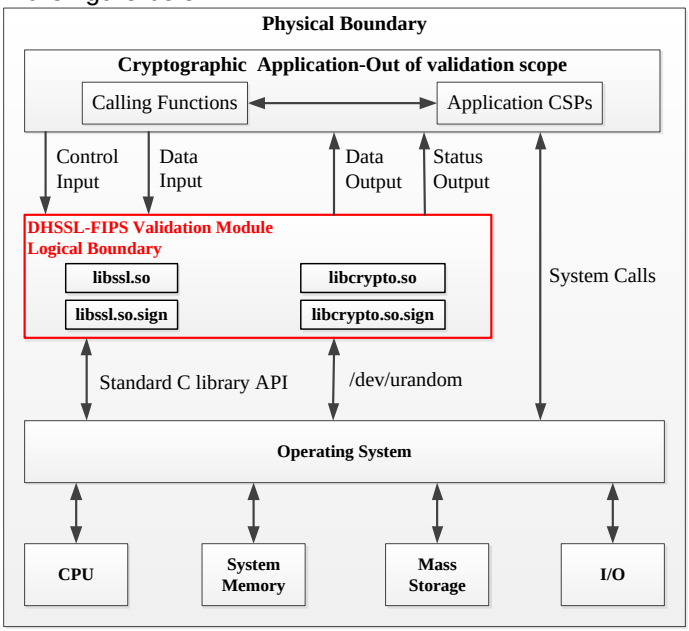

Figure 1 - Block Diagram

The logical boundary contains two shared files and two integrity checking files for integrity tests. The following table enumerates these files to illustrate their purpose.

Filename	Purpose
libssl.so	for the TLS protocol implementation.
libcrypto.so	for cryptographic algorithm implementations
libssl.so.sign	integrity check for the libssl.so shared library
libcrypto.so.sign	integrity check for the libcrypto.so shared library

Table 3 - Cryptographic Module Components

The module supports two modes of operation.

- In "FIPS mode" (the Approved mode of operation) only approved or allowed security functions with sufficient security strength can be used.
- In "Non-FIPS mode" (the non-Approved mode of operation) non-approved security functions can be used.

The module enters FIPS mode after the power-up self-tests have succeeded. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.

Critical security parameters used or stored in FIPS mode are not accessible in non-FIPS mode, and vice versa.

2.2 Cryptographic Module Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the API through which applications request services from the module, and the TLS internal state and protocol messages sent and received from the TCP/IP protocol. The following table provides a description of the logical interfaces:

FIPS 140-2 Interface	Logical Interface	Physical Interface
Data Input	• API input parameters for data • kernel I/O – network or files on filesystem • TLS protocol input messages	• Ethernet, Micro SD, RS-232, RS-485, USB, image sensor, Audio in
Data Output	• API output parameters for data • kernel I/O – network or files on filesystem • TLS protocol output messages	• Ethernet, Micro SD, RS-232, RS-485, eSATA, USB, relay output, Audio out
Control Input	• API function calls • API input parameters for control • TLS protocol internal state	• Ethernet, alarm input
Status Output	• API return codes • API output parameters for status • TLS protocol internal state provided in protocol messages	• Ethernet
Power	• N/A	• Power ports

Table 4 - Module Interface Mappings

2.3 Roles & Services

2.3.1 Roles

The module provides two operator roles: Crypto Officer and User.

The Crypto Officer can performs module installation and configuration, while the Users are the calling applications that utilize the module's cryptographic functions.

2.3.2 Services

The following table describes the services that the two operator roles can perform. In the Access column, Read and Execute mean the Critical Security Parameters(CSPs) or Keys are used by the API call to perform the service; and Write means the CSP is generated, modified or deleted by the API call.

Service	Algorithms	Keys/CSPs	Operator	Access
Symmetric encryption and decryption	AES	AES Key	User	Execute
	Triple-DES	Triple-DES Key	User	Execute
Asymmetric encryption and decryption	RSA	RSA public/private key	User	Execute
RSA key generation	RSA, DRBG	RSA public/private key	User	Read, Write, Execute
RSA digital signature generation and verification	RSA	RSA public/private key	User	Read, Execute
DSA key generation	DSA, DRBG	DSA public/private key	User	Read, Write, Execute
DSA domain parameter generation and verification	DSA	None	User	N/A
DSA digital signature generation and verification	DSA	DSA public/private key	User	Read, Execute
ECDSA key generation	ECDSA, DRBG	ECDSA public/private key	User	Write, Execute
ECDSA public key validation	ECDSA	ECDSA public key	User	N/A
ECDSA digital signature generation and verification	ECDSA	ECDSA public/private key	User	Read, Execute
Message digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	None	User	N/A
Message authentication code (MAC)	HMAC	HMAC key	User	Read, Execute
	CMAC with AES	AES key	User	Read, Execute
	CMAC with Triple-DES	Triple-DES key	User	Read, Execute
Random number generation	DRBG	Entropy input string, Internal state	User	Read, Write, Execute
Key wrapping	AES KW	AES key	User	Read, Execute
Key encapsulation	RSA	RSA public/private key	User	Read, Execute
Diffie-Hellman Key Agreement	KAS FFC	Diffie-Hellman private components	User	Read, Write, Execute
EC Diffie-Hellman Key Agreement	KAS ECC, ECC CDH primitive	EC Diffie-Hellman public/private keys	User	Read, Write, Execute
Transport Layer Security (TLS) network protocol v1.0, v1.1 and v1.2	See Appendix A for the complete list of supported cipher suites	AES key, Triple-DES key, HMAC Key, Premaster secret, Master secret, Diffie-Hellman private components, EC Diffie-Hellman public/private keys, RSA/DSA/ECDSA public/private keys associated to an X.509 Certificate	User	Read, Write, Execute
TLS extensions	N/A	RSA/DSA/ECDSA public/private keys associated to an X.509 Certificate	User	Read, Write, Execute
Certificates management	N/A	RSA/DSA/ECDSA public/private keys associated to an X.509 Certificate	User	Read, Write, Execute
Show status	N/A	None	User	N/A
Zeroization	N/A	All CSPs	User	Write
Self-Tests	AES, Triple-DES, SHS, HMAC, DSA, RSA, ECDSA, DRBG, Diffie-Hellman, EC Diffie-Hellman	None	User	N/A
Module installation	N/A	None	Crypto Officer	N/A
Module configuration	N/A	None	Crypto Officer	N/A

Table 5 - Services in FIPS mode of operation

The table below lists the services only available in Non-FIPS mode of operation

Service	Algorithms	Keys/CSP	Operator	Access
Symmetric encryption and decryption	RC5, RC4, DES, DES XCBCmode, Two-key Triple-DES	Symmetric keys	User	Execute
Asymmetric key generation	RSA, DSA, ECDSA using keys listed in Table 8	Public and private keys	User	Write,Execute
Digital signature generation	RSA, DSA, ECDSA using keys listed in Table 8	Public and private keys	User	Read,Execute
Message digest	MD2, MD4, MD5, MDC-2, RIPEMD160, Whirlpool	None	User	Execute
Message authentication code (MAC)	HMAC using keys listed in Table 8, CMAC with 2-key Triple-DES	HMAC and Triple-DES keys	User	Read,Execute
Key establishment using keys disallowed by[SP800-131A].	Diffie-Hellman, EC Diffie-Hellman, RSA encrypt/decrypt using keys listed in Table 8	Diffie-Hellman private Components, EC Diffie-Hellman public/private keys, RSA public and private keys	User	Read,Execute
[Transport Layer Security (TLS) network protocol v1.0, v1.1 and v1.2][2]	Using cipher suites not allowed by this security policy (see Appendix A for the allowed cipher suites)	AES key, Triple-DES key, HMAC Key, Premaster secret, Master secret, Diffie-Hellman private Components, EC Diffie-Hellman public/private keys, RSA/ECDSA/DSA public/private keys associated to an X.509 Certificate	User	Read,Write,Execute

Table 6 - Services in Non-FIPS mode of operation

[2] No part of the TLS protocol, other than the KDFs, have been tested by the CAVP and CMVP.

2.4 Authentication Mechanisms

For Security level 1, no authentication is required. The role of the user is implicitly assumed based on the service requested.

2.5 Physical Security

As a software-only module, physical security is outside the module's scope.

2.6 Operational Environment

The module operates in a modifiable operational environment per FIPS 140-2 Security Level 1 specifications. The module runs on Linux operating system executing on the hardware specified in table 2 of section 2.

The application that requests cryptographic services is considered the current single user of the cryptographic module. Concurrent operators are explicitly excluded.

2.7 Cryptographic Key Management

2.7.1 Algorithm Implementations

A list of FIPS-Approved algorithms implemented by the module can be found in Table 7.

Algorithm	Standard	Mode/ Method	Key Sizes	Use	Validation Number
AES*	FIPS197, SP800-38A	ECB, CBC, OFB, CFB1, CFB8, CFB128, CTR	128, 192, 256 bits	Data Encryption and Decryption	C1636 C1639 C1640 C1641
	FIPS197, SP800-38B	CMAC	128, 192, 256 bits	MAC Generation and Verification	
	FIPS197, SP800-38C	CCM	128, 192, 256 bits	Data Encryption and Decryption	
	FIPS197, SP800-38D	GCM	128, 192, 256 bits	Data Encryption and Decryption	
	FIPS197, SP800-38D	GMAC	128, 192, 256 bits	Data Encryption and Decryption	
	FIPS197, SP800-38E	XTS	128, 256 bits	Data Encryption and Decryption	
	FIPS197, SP800-38F	KW	128, 192, 256 bits	Key Wrapping and Unwrapping	
DSA	FIPS186-4		L=2048, N=224; L=2048, N=256; L=3072, N=256	Key Pair Generation, Domain Parameter Generation	C1636 C1639 C1640 C1641
		SHA-224, SHA-256, SHA-384, SHA-512	L=2048, N=224; L=2048, N=256; L=3072, N=256	Signature Generation	

Algorithm	Standard	Mode/ Method	Key Sizes	Use	Validation Number
			L=1024, N=160; L=2048, N=224; L=2048, N=256; L=3072, N=256	Domain Parameter Verification	
		SHA1, SHA-224, SHA-256, SHA-384, SHA-512	L=1024, N=160; L=2048, N=224; L=2048, N=256; L=3072, N=256	Signature Verification	
DRBG	SP800-90A	Hash_DRBG SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 with/without PR	N/A	Random Number Generation	C1636 C1639 C1640 C1641
		HMAC_DRBG HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 with/without	N/A	Random Number Generation	
		CTR_DRBG AES128, AES192, AES256 with/without DF,	N/A	Random Number Generation	
ECDSA	FIPS186-4		P-256, P-384, P-521	Key Pair Generation	C1636 C1639 C1640 C1641
		SHA-224, SHA-256, SHA-384, SHA-512	P-256, P-384, P-521	Signature Generation	
			P-256, P-384, P-521	Public Key Verification	
		SHA1, SHA-224, SHA-256, SHA-384, SHA-512	P-256, P-384, P-521	Signature Verification	

CVL
Partial DH
	SP800-56A	FFC dhEphem scheme	p=2048, q=224;
			p=2048, q=256;
			p=3072, q=256
				Diffie-Hellman Key Agreement	C1636
					C1639
					C1640
					C1641

CVL Partial EC-DH	SP800-56A	ECC Ephemeral Unified scheme	P-256, P-384,P-521	EC Diffie-Hellman Key Agreement	C1636 C1639 C1640 C1641

Algorithm	Standard	Mode/ Method	Key Sizes	Use	Validation Number
CVL ECC CDH Primitive	SP800-56A		P-256, P-384,P-521	EC Diffie-Hellman Key Agreement	C1636 C1639 C1640 C1641
HMAC	FIPS198-1	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 bits or greater	Message Authentication Code	C1636 C1639 C1640 C1641
CVL TLS, TLS v1.0/1.1 TLS v1.2	SP800-135			Key Derivation	C1636 C1639 C1640 C1641
RSA	FIPS186-4	X9.31	2048, 3072bits	Key Pair Generation	C1636 C1639 C1640 C1641
		X9.31 with SHA-256, SHA-384, SHA-512	2048, 3072bits	Digital Signature Generation	
		PKCS#1v1.5 and PSS with SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072bits		
		X9.31 with SHA1, SHA-256, SHA-384, SHA-512	2048, 3072bits	Signature Verification	
		PKCS#1v1.5 and PSS with SHA1, SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072bits		
SHS	FIPS180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest	C1636 C1639 C1640 C1641
Triple-DES	SP800-67, SP800-38A	ECB, CBC, CFB1, CFB8, CFB64, OFB	192 bits	Data Encryption and Decryption	C1636 C1639 C1640 C1641
	SP800-67, SP800-38B	CMAC	192 bits	MAC Generation and Verification	

Table 7 - FIPS-Approved Algorithm Implementations

AES*- AES GCM meets te requirements of IG A.5 for IV generation for TLS.

The following table shows the cryptographic algorithms that are allowed in FIPS mode of operation, including the algorithm name and key sizes, any caveat applicable and the permitted usage.

Algorithm	Caveat	Use
RSA Key Wrapping with key size between 2048 bits and 3072 bits	Key wrapping, key establishment methodology provides between 112 and 128 bits of encryption strength	Key Establishment; allowed by IG D.9 in [FIPS140-2_IG]
CVL Diffie-Hellman	Key agreement with key size between 2048 bits and 3072 bits, key establishment methodology provides between 112 and 256 bits of encryption strength	Key Agreement; allowed by IG D.8 in [FIPS140-2_IG]
CVL EC Diffie-Hellman	Provides between 128 and 256 bits of encryption strength with P-256,P-384, P-521 curves	Key Agreement; allowed by IG D.8 in [FIPS140-2_IG]
MD5		Pseudo-random function (PRF) in TLSv1.0 and TLSv1.1, allowed by [SP800-52]
NDRNG		The module obtains the entropy data from NDRNG to seed the DRBG

Table 8 - FIPS-Non-Approved But Allowed Algorithm Implementations

The table below shows the cryptographic algorithms implemented in the module that are not allowed in FIPS mode of operation, including the algorithm name and the reason for being forbidden. Using any of these algorithms will implicitly turn the module in Non-FIPS mode of operation.

Algorithm	Reason
RC5, DES, XCBC	Non FIPS-Approved algorithms
Two-key Triple-DES	Not allowed per [SP800-131A]
MD2, MD4, MD5, MDC2,RIPEMD160, Whirlpool	Non FIPS-Approved algorithms, except MD5 when used as the PRF for TLSv1.0 and TLSv1.1, per [SP800-52]
SHA-1	Not allowed to be used in Digital Signature Generation per [SP800-131A]
HMAC with key size less than 112 bits	Not allowed key size for Message Authentication Code per [SP800-131A]
RSA with key size less than 2048 bits	Not allowed key size for Key Pair generation, Digital Signature Generation, Key Encapsulation
RSA with key size less than 1024 bits	Not allowed key size for Digital Signature Verification per [SP800-131A]
DSA with key size equal or less than L=1024, N=160	Not allowed key size for Key Pair Generation, Domain Parameters Generation, Digital Signature
DSA with key size less than L=1024, N=160	Not allowed key size for Digital Signature Verification per [SP800-131A].
Diffie-Hellman with key size less than 2048 bits	Not allowed key size for Key Agreement per [SP800-131A]
SSLeay Deterministic Random Number Generator (PRNG)	Non FIPS-Approved algorithm

Table 9 - Non-Approved Algorithm Implementations

2.7.2 Key Management Overview[3]

The following table summarizes the keys and CSPs that are used by the cryptographic services implemented in the module.

Name	Key Sizes	Generation	Entry and Output
AES key	128, 192, 256 bits	Not Applicable. Keys are provided by the calling application, or generated during the Diffie-Hellman or EC Diffie-Hellman key agreement	The key is passed into the module via API input parameters in plaintext
Triple-DES key	192 bits		
HMAC key	112 bits or greater		
RSA public/private key	2048, 3072, 4096 bits	Key pairs are generated using FIPS 186-4 key generation method, and the random value used is generated using the SP800-90A DRBG	The key is passed into the module via API input parameters in plaintext, The key is passed out of the module via API output parameters in plaintext
DSA public/private key	L=1024, N=160; L=2048, N=224; L=2048, N=256; L=3072, N=256		
ECDSA public/private key	P-256, P-384, P-521		
Entropy input string	N/A	Obtained from NDRNG	N/A
DRBG internal state (V, C, Key)	N/A	During DRBG initialization	N/A
AES key	128, 256 bits	Generated internally by the module during the establishment of the TLS protocol	N/A
Triple-DES key	192 bits		
HMAC key	112 bits or greater		N/A
Premaster secret	N/A		The key can exit the module to calling application via TLS protocol by using RSA key transport
Master secret	N/A		N/A
Diffie-Hellman private components	N/A		N/A
EC Diffie-Hellman public/private key	P-256, P-384, P-521		N/A

[3] Only 2^16 64- bit block encryptions are allowed with the same key for Triple-Des

Name	Key Sizes	Generation	Entry and Output
RSA, ECDSA , DSA public/private key associated to an X.509 Certificate	RSA: 2048, 3072, 4096 bits ECDSA: P-256, P-384, P-521 DSA: L=1024, N=160; L=2048, N=224; L=2048, N=256; L=3072, N=256	N/A, X.509 certificates are provided by the calling application	The key is passed into the module via API input parameters, The certificate can exit the module to calling application via TLS protocol

Table 10 - Keys and Critical Security Parameters (CSPs)

2.7.3 Key Generation & Input

The module does not implement symmetric key generation, the symmetric key is input by the application as a parameter.

For generating RSA, DSA and ECDSA keys, the module implements asymmetric key generation services compliant with [FIPS186-4], and using a DRBG compliant with [SP800-90A].

2.7.4 Key Output

The module does not support manual key entry or intermediate key generation key output. The keys are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form. The module does not enter or output keys in plaintext format outside its physical boundary.

2.7.5 Storage

Symmetric keys, HMAC keys, public and private keys are provided to the module by the calling application via API input parameters, and are destroyed by the module when invoking the appropriate API function calls.

The module does not perform persistent storage of keys. The keys and CSPs are stored as plaintext in the RAM.

2.7.6 Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The application is responsible for calling the appropriate zeroization functions provided in the module's API, and documented in the API documentation. Also, calling the corresponding zeroization functions for TLS protocol sessions will zeroize the keys and CSPs stored in the TLS protocol internal state.

The zeroization functions overwrite the memory occupied by keys and CSPs with “zeros” and deallocate the memory with the regular memory deallocation operating system call.

2.7.7 Available Entropy

The entropy pool is 4096 bits. Every call to the pool for entropy is guaranteed to be successful due to the use of a non-blocking call. The minimum amount of entropy will be 384 bits as requested.

2.8 Electromagnetic Interference / Electromagnetic Compatibility

The test platforms listed in table 2 of section 2 have been tested and found to conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, FCC PART 15, Subpart B, Unintentional Radiators, Digital Devices, Class A. These devices are designed to provide reasonable protection against harmful interference when the devices are operated in a commercial environment. They shall be installed and used in accordance with the instruction manual.

2.9 Self Tests

2.9.1 Power-Up Self-Tests

The module performs power-up self-tests when the module is loaded into memory, without operator

intervention. The operator can perform power-up self-tests on demand by restarting the module. Power-up self-tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

While the module is executing the power-up self-tests, services are not available, and input and output are inhibited. The module is not available to be used by a calling application until the power-up self-tests have completed successfully.

If any power-up test fails, the module returns the error code and displays the specific error message associated with the returned error code, and then the module enters an error state. Subsequent calls to the module will fail, thus no further cryptographic operations are possible, and data output is inhibited. If the power-up tests complete successfully, the module will return 1 in the return code and will accept cryptographic operation service requests.

The module performs the following self-tests upon power up or on demand.

Algorithm	Self-Test
AES	✓ KAT[4] AES(ECB) with 128-bit key, encryption ✓ KAT AES(ECB) with 128-bit key, decryption ✓ KAT AES(CCM) with 192-bit key, encryption ✓ KAT AES(CCM) with 192-bit key, decryption ✓ KAT AES(GCM) with 256-bit key, encryption ✓ KAT AES(GCM) with 256-bit key, decryption ✓ KAT AES(CMAC) with 128-bit, 192-bit and 256-bit keys, MAC generation and verification ✓ KAT AES(XTS) with 128-bit and 256-bit keys, encryption ✓ KAT AES(XTS) with 128-bit and 256-bit keys, decryption
Triple-DES	✓ KAT Triple-DES (ECB) with 192-bit key, encryption ✓ KAT Triple-DES (ECB) with 192-bit key, decryption ✓ KAT Triple-DES with 192-bit key (CMAC), MAC generation and verification
SHS	✓ KAT SHA-1 ✓ KAT SHA-256 ✓ KAT SHA-512
HMAC	✓ KAT HMAC-SHA-1 ✓ KAT HMAC-SHA-224 ✓ KAT HMAC-SHA-256 ✓ KAT HMAC-SHA-384 ✓ KAT HMAC-SHA-512
DSA	✓ PCT[5] DSA with L=2048, N=256 and SHA-256
ECDSA	✓ PCT ECDSA with P-256 and SHA-256
RSA	✓ KAT RSA PKCS#1v1.5 signature generation and verification with 2048 bit key and using SHA-224,SHA-256, SHA-384, and SHA-512 ✓ KAT RSA PSS signature generation and verification with 2048-bit key and SHA-224, SHA-256, SHA-384, and SHA-512 ✓ KAT RSA with 2048 bit key, public-key encryption ✓ KAT RSA with 2048 bit key, private-key decryption
DRBG	✓ KAT Hash_DRBG using SHA-256 without PR ✓ KAT HMAC_DRBG using HMAC-SHA256 without PR ✓ KAT CTR_DRBG using AES-256, with DF and without DF
KAS ECC	✓ Primitive “Z” Computation KAT with P-256 curve
KAS FFC	✓ Primitive “Z” Computation KAT with 2048-bit key

[4] KAT: Known Answer Test
[5] PCT: Pairwise Consistency Test

Algorithm	Self-Test
Module Integrity	✓ HMAC-SHA 256

Table 11 - Power-On Self-Tests

2.9.2 Conditional Self Tests

The module performs the following conditional self-test.

Algorithm	Test
DSA key generation	✓ PCT using SHA-256, signature generation and verification
ECDSA key generation	✓ PCT using SHA-256, signature generation and verification
RSA key generation	✓ PCT using SHA-256, signature generation and verification ✓ PCT for encryption and decryption
NDRNG	✓ Continuous Random Number Generator Test
AES-XTS	✓ Duplicate Key Test
DRBG	✓ Health Tests ✓ Continuous Random Number Generator Test

Table 12 - Conditional Self-Tests

2.10 Design Assurance

Configuration management for the module is provided by SVN source code management system which uniquely identifies each configuration item and the version of each configuration item.

Documentation version control is performed manually by updating the document data as well as the major and minor version numbers in order to uniquely identify each version of a document.

2.11 Mitigation of Other Attacks

The module does not claim to mitigate any attacks outside the requirements of FIPS 140-2.

3 Secure Operation

3.1 Initialization

The module is installed as part of a firmware upgrade in the device. To initialize the module the `FIPS_mode_set()` function is invoked. During initialization, the Power-On Self Test described in Section 2.9 Self Tests are run. If any one component of the self tests fails, subsequent invocation of any cryptographic function calls to the module will fail. `FIPS_mode_set()` initializes the module (`FIPS_mode` flag is `TRUE`) only after all tests have completed successfully.

3.2 Crypto Officer Guidance

The module is a software-only cryptographic module delivered as part of the update installed in the hardware device. The firmware includes the operating system, user applications and the module itself.

3.3 User Guidance

In order to run in FIPS Approved mode of operation, the module must be operated using the FIPS approved services, with their corresponding FIPS approved or FIPS allowed cryptographic algorithms indicated in this Security Policy (see section 2.3.2 Services). In addition, the key sizes indicated comply with [SP800-131A].

As explained in section 2.1, the module is provided as a set of shared libraries. Applications must link the module dynamically to run the module in FIPS approved mode.

The application can query whether the FIPS operation is active by calling `FIPS_mode()` and it can query whether an integrity check or KAT self test failed by calling `FIPS_selftest_failed()`.

3.4 AES GCM IV

AES GCM encryption and decryption is used in the context of the TLS protocol version 1.2. The module is compliant with [SP 800-52] and the mechanism for IV generation is compliant with [RFC5288]. The operations of one of the two parties involved in the TLS key establishment scheme are performed entirely within the cryptographic boundary of the module.

In case the module's power is lost and then restored, the key used for AES GCM encryption or decryption shall be re-distributed.

4 Acronyms

Acronym	Definition
DHSSL	Name Of Cryptographic Algorithm Module
AES	Advanced Encryption Standard
CA	Certificate Authority
CBC	Cipher Block Chaining
CDH	Co-factor Diffie-Hellman
CFB	Cipher Feedback
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CCCS	Canadian Centre for Cyber Security
CSP	Critical Security Parameter
CVS	Concurrent Versions System
DF	Derivation Function
DHSSL	The Cryptographic Algorithm Module
DRBG	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
ECDSA	Elliptical Curve Digitial Signature Algorithm
EFP	Environmental Failure Protection
EMI/EMC	Electromagnetic Interference / Electromagnetic Compatibility
FCC	Federal Communications Commission
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
HMAC	(Keyed-) Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KW	Key Wrapping
LED	Light Emitting Diode
NIST	National Institute of Standards and Technology
NRBG	Non-Deterministic Random Bit Generator
NVM	Non-Volatile Memory
OFB	Output Feedback
PR	Prediction Resistance
PRF	Pseudo Random Function
PRNG	Pseudo Random Number Generator
QVGA	Quarter Video Graphics Array
RC4, RC5	Rivest Cipher 4, Rivest Cipher 5
RIPEMD160	RIPE Message Digest
ROM	Read Only Memory
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm

TDES	Triple Data Encryption Standard
TLS	Transport Layer Security
USB	Universal Serial Bus
XCBC	eXtended Ciphertext Block Chaining

Table 13 - Acronym Definitions

Appendix A. TLS cipher suites

The module supports several cipher suites for the TLS protocol. Each cipher suite defines the key exchange algorithm, the bulk encryption algorithm (including the symmetric key size) and the MAC algorithm.

Cipher Suite	Reference
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RFC2246
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	RFC2246
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	RFC2246
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	RFC2246
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	RFC2246
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	RFC2246
TLS_RSA_WITH_AES_128_CBC_SHA	RFC3268
TLS_DH_DSS_WITH_AES_128_CBC_SHA	RFC3268
TLS_DH_RSA_WITH_AES_128_CBC_SHA	RFC3268
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	RFC3268
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	RFC3268
TLS_DH_anon_WITH_AES_128_CBC_SHA	RFC3268
TLS_RSA_WITH_AES_256_CBC_SHA	RFC3268
TLS_DH_DSS_WITH_AES_256_CBC_SHA	RFC3268
TLS_DH_RSA_WITH_AES_256_CBC_SHA	RFC3268
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	RFC3268
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	RFC3268
TLS_DH_anon_WITH_AES_256_CBC_SHA	RFC3268
TLS_RSA_WITH_AES_128_CBC_SHA256	RFC5246
TLS_RSA_WITH_AES_256_CBC_SHA256	RFC5246
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	RFC5246
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	RFC5246
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	RFC5246
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	RFC5246
TLS_DH_DSS_WITH_AES_256_CBC_SHA256	RFC5246
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	RFC5246
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	RFC5246
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	RFC5246
TLS_DH_anon_WITH_AES_128_CBC_SHA256	RFC5246
TLS_DH_anon_WITH_AES_256_CBC_SHA256	RFC5246
TLS_PSK_WITH_3DES_EDE_CBC_SHA	RFC4279
TLS_PSK_WITH_AES_128_CBC_SHA	RFC4279
TLS_PSK_WITH_AES_256_CBC_SHA	RFC4279
TLS_RSA_WITH_AES_128_GCM_SHA256	RFC5288
TLS_RSA_WITH_AES_256_GCM_SHA384	RFC5288
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	RFC5288
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	RFC5288
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	RFC5288

Cipher Suite	Reference
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	RFC5288
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	RFC5288
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	RFC5288
TLS_DH_DSS_WITH_AES_128_GCM_SHA256	RFC5288
TLS_DH_DSS_WITH_AES_256_GCM_SHA384	RFC5288
TLS_DH_anon_WITH_AES_128_GCM_SHA256	RFC5288
TLS_DH_anon_WITH_AES_256_GCM_SHA384	RFC5288
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	RFC4492
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	RFC4492
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	RFC4492
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	RFC4492
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	RFC4492
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	RFC4492
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	RFC4492
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	RFC4492
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	RFC4492
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	RFC4492
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA	RFC4492
TLS_ECDH_anon_WITH_AES_128_CBC_SHA	RFC4492
TLS_ECDH_anon_WITH_AES_256_CBC_SHA	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	RFC5289
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	RFC5289
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	RFC5289
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	RFC5289
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	RFC5289

Appendix B. References

FIPS140-2	FIPS PUB 140-2 - Security Requirements For Cryptographic Modules May 2001 http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf
FIPS140-2_IG	Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program March 27, 2018 http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf
FIPS180-4	Secure Hash Standard (SHS) March 2012 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-4	Digital Signature Standard (DSS) July 2013 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS197	Advanced Encryption Standard November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	The Keyed Hash Message Authentication Code (HMAC) July 2008 http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
PKCS#1	Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 February 2003 http://www.ietf.org/rfc/rfc3447.txt
SP800-38A	NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-38B	NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication May 2005 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP800-38C	NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality May 2004 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP800-38D	NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP800-38E	NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices January 2010

http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf

SP800-38F NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping
December 2012
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

SP800-52 NIST Special Publication 800-52 Revision 1 - Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations
April 2014
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf

SP800-56A NIST Special Publication 800-56A - Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography(Revised)
March, 2007
http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf

SP800-57 NIST Special Publication 800-57 Part 1 Revision 4 – Recommendation for Key Management Part 1: General
January 2016
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

SP800-67 NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
January 2012
http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

SP800-90A NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators
June 2015
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-131A NIST Special Publication 800-131A Revision 1- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
November 2015
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf

SP800-133 NIST Special Publication 800-133 - Recommendation for Cryptographic Key Generation
December 2012
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133.pdf

SP800-135 NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions
December 2011
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf