HikSSL

Cryptographic Module

version 1.0.0

FIPS 140-2 Non-Proprietary Security Policy

Version 1.3

Last update: 2018-06-26

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

Copyrights and Trademarks

Linux is a registered trademark of Linus Torvalds.

HikVision is a registered trademark of Hangzhou Hikvision Digital Technology Co., Ltd.

1. Cryptographic Module Specification

This document is the non-proprietary FIPS 140-2 Security Policy for version 1.0.0 of the HikSSL Cryptographic Module. It contains the security rules under which the module must be operated and describes how this module meets the requirements as specified in FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 module.

The following sections describe the cryptographic module and how it conforms to the FIPS 140-2 specification in each of the required areas.

1.1. Module Overview

The HikSSL Cryptographic Module (hereafter referred to as "the module") is a set of software libraries implementing the Transport Layer Security (TLS) protocol v1.0, v1.1 and v1.2, as well as general purpose cryptographic algorithms. The module provides cryptographic services to applications running in the user space of the underlying Linux operating system through a C language Application Program Interface (API). The module does not use any Processor Algorithm Acceleration (PAA), but uses specific assembler code implementations for the ARM processors provided by the OpenSSL code, which optimize and increase performance. The module can act as a TLS server or TLS client, and interacts with other entities via the TLS network protocol.

The module is implemented as a set of shared libraries; as shown in the diagram below, the shared library files and the integrity check files used to verify the module's integrity constitute the logical cryptographic boundary.

The software block diagram in Figure 1 shows the module, its interfaces with the operational environment and the delimitation of its logical boundary.

![HIKVISION]
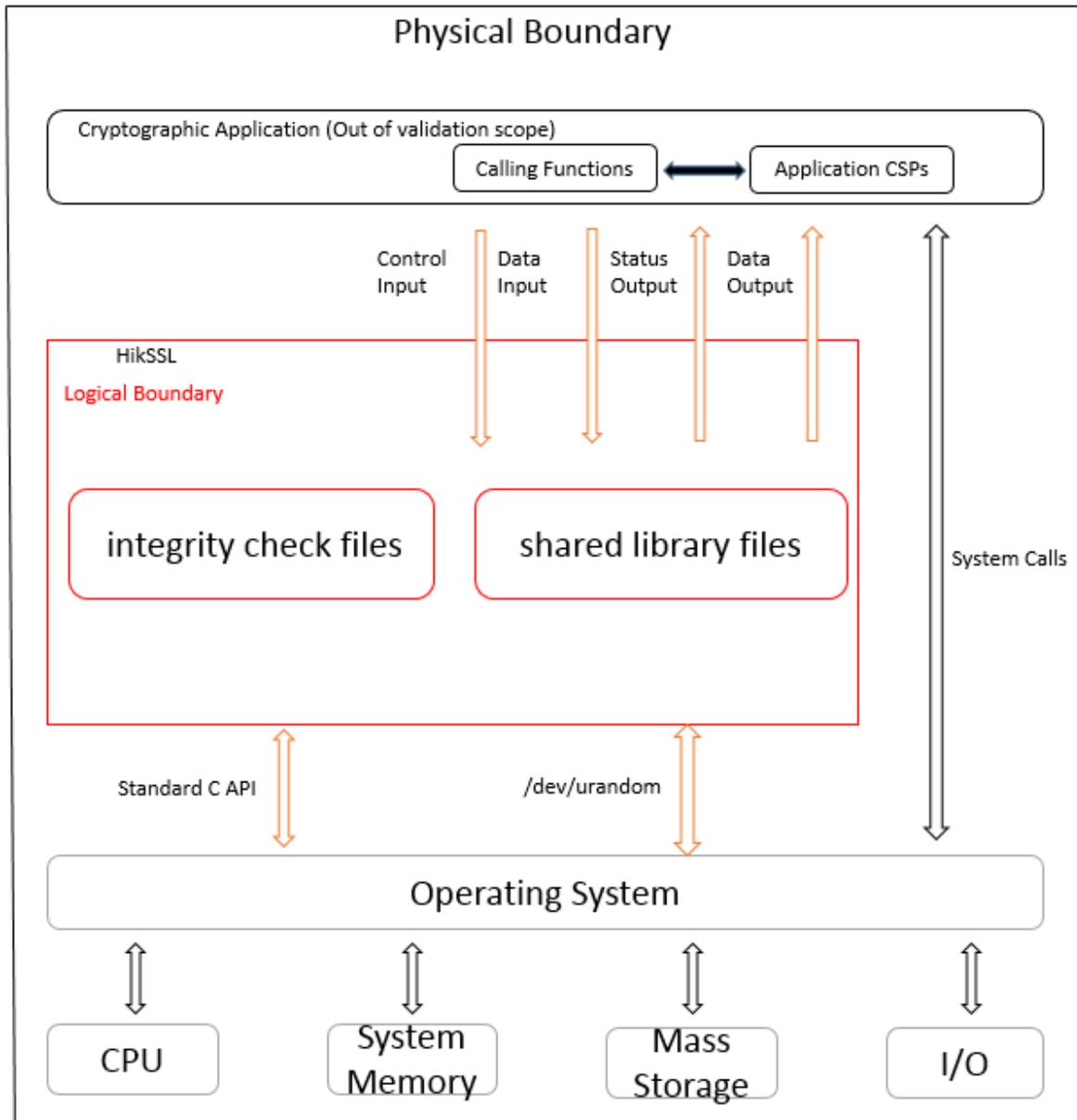
Figure 1 - Software Block Diagram

The module is implemented as a set of shared libraries. The cryptographic logical boundary consists of all shared libraries and the integrity check files used for integrity tests. The following table enumerates the files that comprise each module variant.

Filename	Purpose
libssl.so.1.0.0	Shared library for the TLS protocol implementation.
libcrypto.so.1.0.0	Shared library for cryptographic algorithm implementations.

Filename	Purpose
.libssl.so.1.0.0.hmac	Integrity check HMAC value for the libssl shared library.
.libcrypto.so.1.0.0.hmac	Integrity check HMAC value for the libcrypto shared library.

Table 1 - Cryptographic Module Components

The module is aimed to run on Network Video Recorder (NVR) and Network Camera devices including ARMv7 processors running a Linux operating system. Applications, the cryptographic module itself, and the underlying operating system run within the target hardware platform. The physical enclosure of the hardware platform constitutes the physical boundary of the module.

1.2. FIPS 140-2 Validation

For the purpose of the FIPS 140-2 validation, the module is a software-only, multi-chip standalone cryptographic module validated at overall Security Level 1. The table below shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard.

	FIPS 140-2 Section	Security 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
Overall Level		1

Table 2 - Security Levels

The module has been tested on the platforms shown below.

Test Platform	Processor	Operating System
NVR model number DS-9632NI-I8	ARM Cortex-A17 (ARMv7 32-bit)	Linux version 3.10.0_hi3536, 32-bit (custom)
NVR model number DS-7716NI-I4/16P	ARM Cortex-A17 (ARMv7 32-bit)	Linux version 3.10.0_hi3536, 32-bit (custom)
NVR model number DS-7732NI-I4/16P	ARM Cortex-A17 (ARMv7 32-bit)	Linux version 3.10.0_hi3536, 32-bit (custom)
Network Camera model number DS-2CD2742FWD-IZS	ARM Cortex-A9 (ARMv7 32-bit)	Linux Hikvision version 3.10.73+, 32-bit (custom)

Table 3 - Tested Platforms

The module does not run on a full-fledged Linux distribution. The vendor trimmed down and customized the operating system to fit it for the resource-constrained devices within which the module runs, while keeping the Linux kernel intact.

Hangzhou Hikvision Digital Technology Co., Ltd. affirms that the module runs correctly on the following network camera and NVR models:

- Network Cameras: model names starting with DS-2CD2.

- NVRs: DS-96xxNI-Ix and DS-77xxNI-Ix/xxP model names (x characters vary depending on model).

All of the above vendor affirmed devices have the same processor and operating system as the ones tested by the accredited Cryptographic Security Testing lab.

Per FIPS 140-2 IG G.5, the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys on the vendor affirmed platforms.

1.3. Modes of operation

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) only non-approved security functions can be used.

The module enters FIPS mode after power-up tests succeed. 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 used in non-FIPS mode, and vice versa.

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, and the TLS internal state and protocol messages sent and received from the underlying network protocol. The following table summarizes the four logical interfaces:

Logical Interface	Description
Data Input	API input parameters for data, kernel I/O – network or files on filesystem, TLS protocol input messages.
Data Output	API output parameters for data, kernel I/O – network or files on filesystem, TLS protocol output messages.
Control Input	API function calls, API input parameters for control, TLS protocol internal state.
Status Output	API return codes, API output parameters for status, TLS protocol internal state provided in protocol messages.

Table 4 - Ports and Interfaces

.

3. Roles, Services and Authentication

3.1. Roles

The module supports the following roles:

- **User role**: performs all services (in both FIPS mode and non-FIPS mode of operation), except module installation and configuration. This role is assumed by the calling application accessing the module.
- **Crypto Officer role**: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed depending on the service requested.

3.2. Services

The module provides services to calling applications that assume the user role, and human users assuming the Crypto Officer role. All services are shown in Table 5 and Table 6 and described in detail in the user documentation.

Table 5 shows the Approved services and the non-Approved but allowed services in FIPS mode of operation, the cryptographic algorithms supported for each service, the roles that can perform each service, and the public keys and Critical Security Parameters (CSPs) involved and how they are accessed. The details about the algorithms supported by the module are found in section 3.3.

Service	Algorithms	Role	Access	Keys/CSP
Cryptographic Library Services				
Symmetric encryption and decryption	AES	User	Read	AES key
	Triple-DES	User	Read	Triple-DES key
RSA key generation	RSA, DRBG	User	Create	RSA public/private key
RSA digital signature generation and verification	RSA	User	Read	RSA public/private key
DSA key generation	DSA, DRBG	User	Create	DSA public/private key
DSA domain parameter generation and verification	DSA	User	n/a	None
DSA digital signature generation and verification	DSA	User	Read	DSA public/private key
ECDSA key generation	ECDSA, DRBG	User	Create	ECDSA public/private key
ECDSA public key validation	ECDSA	User	n/a	ECDSA public key

Service	Algorithms	Role	Access	Keys/CSP
ECDSA digital signature generation and verification	ECDSA	User	Read	ECDSA public/private key
Message digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	User	n/a	None
Message authentication code (MAC)	HMAC	User	Read	HMAC key
	CMAC with AES	User	Read	AES key
	CMAC with Triple-DES	User	Read	Triple-DES key
Random number generation	DRBG	User	Read, Update	Entropy input string, Internal state
Key wrapping	AES KW	User	Read	AES key
Key encapsulation	RSA	User	Read	RSA public/private key
Diffie-Hellman Key Agreement	KAS FFC	User	Create, Read	Diffie-Hellman private components
EC Diffie-Hellman Key Agreement	KAS ECC, ECC CDH primitive	User	Create, Read	EC Diffie-Hellman public/private keys
Network Protocol Services				
Transport Layer Security (TLS) network protocol v1.0, v1.1 and v1.2	See Appendix A for the complete list of supported cipher suites	User	Read	AES key Triple-DES key HMAC Key Premaster secret Master secret Diffie-Hellman private components EC Diffie-Hellman public/private keys RSA, DSA or ECDSA public/private keys associated to an X.509 Certificate
TLS extensions	n/a	User	Read	RSA, DSA or ECDSA public/private keys associated to an X.509 Certificate

Service	Algorithms	Role	Access	Keys/CSP
Certificates management	n/a	User	Read	RSA, DSA or ECDSA public/private keys associated to an X.509 Certificate
Other FIPS-related Services				
Show status	n/a	User	n/a	None
Zeroization	n/a	User	Zeroize	All CSPs
Self-Tests	AES, Triple-DES, SHS, HMAC, DSA, RSA, ECDSA, DRBG, Diffie-Hellman, EC Diffie-Hellman	User	n/a	None
Module installation	n/a	Crypto Officer	n/a	None
Module configuration	n/a	Crypto Officer	n/a	None

Table 5 - Services in FIPS mode of operation

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

Service	Algorithms / Key sizes	Role	Access	Keys
Symmetric encryption and decryption	RC5, DES, DES XCBC mode, Two-key Triple-DES	User	Read	Symmetric keys.
Asymmetric key generation	RSA, DSA, ECDSA using keys listed in Table 9.	User	Create	Public and private keys.
Digital signature generation	RSA, DSA, ECDSA using keys listed in Table 9.	User	Read	Public and private keys.
Message digest	MD2, MD4, MD5, MDC-2, RIPEMD160, Whirlpool	User	n/a	None
Message authentication code (MAC)	HMAC using keys listed in Table 9. CMAC with 2-key Triple-DES.	User	Read	HMAC and Triple-DES keys
Key establishment using keys disallowed by [SP800-131A].	Diffie-Hellman, EC Diffie-Hellman, RSA encrypt / decrypt using keys listed in Table 9.	User	Create, Read	Diffie-Hellman private components EC Diffie-Hellman public/private keys RSA public and private keys.

Service	Algorithms / Key sizes	Role	Access	Keys
Transport Layer Security (TLS) network protocol v1.0, v1.1 and v1.2	Using cipher suites not allowed by this security policy (see Appendix A for the allowed cipher suites)	User	Create, Read	AES key Triple-DES key HMAC Key Premaster secret Master secret Diffie-Hellman private components EC Diffie-Hellman public/private keys RSA, ECDSA or DSA public/private keys associated to an X.509 Certificate

Table 6 - Services in non-FIPS mode of operation

3.3. Algorithms

The algorithms implemented in the module approved to be used in FIPS mode of operation are tested and validated by the CAVP. Notice that for the Transport Layer Security (TLS) protocol, no parts of this protocol implementation, other than the key derivation function (KDF), have been tested by the CAVP.

The module provides specific assembler implementations for ARMv7 processors in the AES core, SHA-1, SHA-256, SHA-512 and GHASH algorithms, and C language generic implementations for the rest of the algorithm.

The following table shows the cryptographic algorithms that are approved in FIPS mode of operation, including the CAVP certificates for different implementations, the algorithm name, supported standards, available modes and key sizes, and usage. Notice that some information included in a single column (e.g. CAVP certificates, algorithm name, standard) may be applicable to several rows.

CAVP Cert#	Algorithm	Standard	Mode / Method	Key size	Use
#5259 #5262	AES	[FIPS197] [SP800-38A]	ECB, CBC, OFB, CFB1, CFB8, CFB128, CTR	128, 192 and 256 bits	Data Encryption and Decryption
		[FIPS197] [SP800-38B]	CMAC	128, 192 and 256 bits	MAC Generation and Verification
		[FIPS197] [SP800-38C]	CCM	128, 192 and 256 bits	Data Encryption and Decryption
		[FIPS197] [SP800-38D]	GCM	128, 192 and 256 bits	Data Encryption and Decryption

CAVP Cert#	Algorithm	Standard	Mode / Method	Key size	Use
		[FIPS197] [SP800-38E]	XTS	128 and 256 bites	Data Encryption and Decryption
		[FIPS197] [SP800-38F]	KW	128, 192 and 256 bits	Key Wrapping and Unwrapping
#1362 #1363	DSA	[FIPS 186-4]		L=2048, N=224; L=2048, N=256; L=3072, N=256	Key Pair Generation. Domain Parameter Generation
			SHA-224, SHA-256, SHA-384, SHA-512	L=2048, N=224; L=2048, N=256; L=3072, N=256	Signature Generation
				L=1024, N=160; L=2048, N=224; L=2048, N=256; L=3072, N=256	Domain Parameter Verification
			SHA-1, 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
#2012 #2015	DRBG	[SP800-90A]	Hash_DRBG SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) with/without PR	n/a	Random Number Generation
			HMAC_DRBG HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 with/without PR	n/a	Random Number Generation

CAVP Cert#	Algorithm	Standard	Mode / Method	Key size	Use
			CTR_DRBG AES128, AES192, AES256 with/without DF, with/without PR	n/a	Random Number Generation
#1371 #1372	ECDSA	[FIPS186-4]		P-256, P-384, P-521	Key Pair Generation
			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
			SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	P-256, P-384, P-521	Signature Verification
CVL #1731 CVL #1733	Partial Diffie-Hellman	[SP800-56A]	FFC dhEphem scheme	p=2048, q=224; p=2048, q=256	Diffie-Hellman Key Agreement
CVL #1731 CVL #1733	Partial EC Diffie-Hellman	[SP800-56A]	ECC Ephemeral Unified scheme	P-256, P-384, P-521	EC Diffie-Hellman Key Agreement
CVL #1731 CVL #1733	ECC CDH Primitive	[SP800-56A]		P-256, P-384, P-521	EC Diffie Hellman Key Agreement
#3481 #3482	HMAC	[FIPS198-1]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 bits or greater	Message Authentication Code
CVL #1732 CVL #1734	KDF(PRF) in TLS v1.0/1.1 TLS v1.2	[SP800-135]			Key Derivation
#2812 #2813	RSA	[FIPS186-4]	X9.31	2048 and 3072 bits	Key Pair Generation

CAVP Cert#	Algorithm	Standard	Mode / Method	Key size	Use
			X9.31 with SHA-256, SHA-384, SHA-512	2048 and 3072 bits	Digital Signature Generation
			PKCS#1v1.5 and PSS with SHA-224, SHA-256, SHA-384, SHA-512	2048 and 3072 bits	
			X9.31 with SHA-1, SHA-256, SHA-384, SHA-512	1024, 2048, and 3072 bits	Signature Verification
			PKCS#1v1.5 and PSS with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	1024, 2048, and 3072 bits	
#4232 #4233	SHS	[FIPS180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest
#2661 #2662	Triple-DES	[SP800-67] [SP800-38A]	ECB, CBC, CFB1, CFB8, CFB64, OFB	192 bits	Data Encryption and Decryption
		[SP800-67] [SP800-38B]	CMAC	192 bits	MAC Generation and Verification

Table 7 - FIPS-Approved Cryptographic Algorithms

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 Encapsulation[1] with Encryption and Decryption Primitives and at least 2048-bit key size	Provides between 112 and 128 bits of encryption strength.	Key Establishment; allowed by IG D.9 in [FIPS140-2_IG].
Diffie-Hellman with at least 2048 bit key size (CVL certs. #1731 and #1733)	Provides between 112 and 128 bits of encryption strength.	Key Agreement; allowed by IG D.8 in [FIPS140-2_IG].
EC Diffie-Hellman with P-256, P-384, P-521 curves (CVL certs. #1731 and #1733)	Provides between 128 and 256 bits of encryption strength.	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-Allowed Cryptographic Algorithms

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.

---

[1] RSA key encapsulation and RSA key wrapping are terms used interchangeably.

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 per [SP800-131A].
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 Generation per [SP800-131A].
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 Cryptographic Algorithms

3.4. Operator Authentication

The module does not implement user authentication. The role of the user is implicitly assumed based on the service requested.

4. Physical Security

The module is comprised of software only and therefore this security policy does not make any claims on physical security.

5. Operational Environment

5.1. Applicability

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 section 1.2.

5.2. Policy

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

6. Cryptographic Key Management

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

Name	Generation	Entry and Output
AES keys	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 keys		
HMAC key		
RSA private key	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 private key		
ECDSA private key		
Entropy input string	Obtained from NDRNG	N/A
DRBG internal state (V, C, Key)	During DRBG initialization.	N/A
TLS network protocol		
AES key Triple-DES key	Generated internally by the module during the establishment of the TLS protocol.	N/A
HMAC key		N/A
Premaster secret		The key can exit the module via TLS protocol by using RSA key transport.
Master secret		N/A
Diffie-Hellman private components		N/A
EC Diffie-Hellman private key		N/A
RSA, ECDSA, or DSA private key associated to an X.509 Certificate	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 via TLS protocol.

Table 10 - Life cycle of Keys and Critical Security Parameters (CSP)

The following sections describe how keys and CSPs are managed during its life cycle.

6.1. Random Number Generation

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90A] for the creation of key components of asymmetric keys, and server and client random numbers for the TLS protocol. In addition, the module provides a Random Number Generation service to calling applications.

The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms. The DRBG is initialized during module initialization; the module loads by default the DRBG using the CTR_DRBG mechanism with AES-256 and derivation function without prediction resistance. A different DRBG mechanism can be chosen through an API function call.

For seeding the DRBG, the module uses a Non-Deterministic Random Number Generator (NDRNG). The NDRNG is implemented by the cryptographic module and therefore it is within its logical boundary. The NDRNG provides at least 256 bits of entropy to the DRBG during initialization (seed) and reseeding (reseed), sufficient for the security strength provided by the DRBG algorithm.

The NDRNG implements a continuous test on the output to ensure that consecutive random numbers do not repeat. The module performs DRBG health tests as defined in section 11.3 of [SP800-90A].

6.2. Key Generation

The module does not implement symmetric key generation.

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]. In accordance with [FIPS140-2_ IG] D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per [SP800-133] (vendor affirmed).

6.3. Key Agreement / Key Transport / Key Derivation

The module provides Diffie-Hellman and EC Diffie-Hellman key agreement schemes. These key agreement schemes are also used as part of the TLS protocol key exchange.

The module also provides key wrapping using the AES with KW mode and RSA key encapsulation using public key encryption and private key decryption primitives as part of the TLS protocol key exchange.

Table 7 and Table 8 specify the key sizes allowed in FIPS mode of operation. According to "Table 2: Comparable strengths" in [SP 800-57], the key sizes of RSA, Diffie-Hellman and EC Diffie-Hellman provide the following security strength:

- AES key wrapping provides between 128 and 256 bits of encryption strength.

- RSA key encapsulation provides between 112 and 128 bits of encryption strength.

- Diffie-Hellman key agreement provides between 112 and 128 bits of encryption strength.

- EC Diffie-Hellman key agreement provides between 128 and 256 bits of encryption strength.

The module supports key derivation for the TLS protocol. The module implements the pseudo-random functions (PRF) for TLSv1.0/1.1 and TLSv1.2 in accordance with [SP800-135].

6.4. Key Entry / 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.

6.5. Key / CSP 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. The only exception is the HMAC key used for integrity test, which is stored in the module and relies on the operating system for protection.

6.6. Key / CSP 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.

7. Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms listed in Table 3 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 (i.e., Business use). 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.

8. Self Tests

8.1. Power-Up Tests

The module performs power-up self-tests when the module is loaded into memory, without operator intervention. 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 the calling application until the power-up self-tests are completed successfully.

If any power-up test fails, the module returns the error code listed in Table 13 and displays the specific error message associated with the returned error code, and then enters error state. The subsequent calls to the module will also fail - thus no further cryptographic operations are possible. If the power-up tests complete successfully, the module will return 1 in the return code and will accept cryptographic operation service requests.

8.1.1.   Integrity Tests

The integrity of the module is verified by comparing an HMAC-SHA-256 value calculated at run time with the HMAC value stored in the .hmac file that was computed at build time for each software component of the module. If the HMAC values do not match, the test fails and the module enters the error state.

8.1.2.   Cryptographic algorithm tests

The module performs self-tests on all FIPS-Approved cryptographic algorithms supported in the approved mode of operation, using the Known Answer Tests (KAT) and Pair-wise Consistency Tests (PCT) shown in the following table.

Algorithm	Test
AES	• KAT 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 key • 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)
SHS	• KAT SHA-1 • KAT SHA-256 • KAT SHA-512

Algorithm	Test
HMAC	• KAT HMAC-SHA-1 • KAT HMAC-SHA-224 • KAT HMAC-SHA-256 • KAT HMAC-SHA-384 • KAT HMAC-SHA-512
DSA	• PCT 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-1, SHA-224, SHA-256, SHA-384, and SHA-512 • KAT RSA PSS signature generation and verification with 2048-bit key and SHA-1, 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

Table 11- Self-Tests

For KATs, the module calculates the result and compares it with the known value. If the answer does not match the known answer, the KAT fails and the module enters the Error state. For PCTs, if the signature generation or verification fails, the module enters the Error state.

8.2. On-Demand self-tests

On-Demand self-tests can be invoked by powering-off and reloading the module, thus forcing the module to run the power-up self-tests.

8.3. Conditional Tests

The module performs conditional tests on the cryptographic algorithms using Pair-wise Consistency Tests (PCT) and Continuous Random Number Generator Test (CRNGT), as shown in the following table.

Algorithm	Test
DSA key generation	• PCT using SHA-256, signature generation and verification.

Algorithm	Test
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 test

Table 12 - Conditional Tests

Note: CRNGT on the SP800-90A DRBG is not required per IG 9.8 in [FIPS140-2_IG].

9. Guidance

9.1. Crypto Officer Guidance

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

For hardware devices that do not include the module, a firmware upgrade will be necessary. The firmware is signed by the publisher using an RSA private key. After the upgrade and during boot time, the new firmware signature and the publisher's certificate are verified by the device.

9.1.1. Prerequisites

The Crypto Officer will download the firmware published by Hangzhou Hikvision Digital Technology Co., Ltd. and will follow the documented procedures for upgrading the firmware.

9.1.2. Module installation

The module is installed as part of a firmware upgrade in the device. The Crypto Officer must follow the instructions provided in the device's User manual.

9.2. 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 provided in this Security Policy (see section 3.2 Services). In addition, key sizes must comply with [SP800-131A].

As explained in section 1.1, 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().

9.2.1. API Functions

Passing "0" to the FIPS_mode_set() API function is prohibited.

Replacement of the standard memory management functions (e.g. using the CRYPTO_set_mem_functions() API function) is prohibited.

9.2.2. TLS

The TLS protocol implementation provides both server and client sides. In order to operate in FIPS approved mode of operation, digital certificates and private keys used for server and client authentication shall comply with the restrictions of key size and message digest algorithms imposed by [SP800-131A]. In addition, as required also by [SP800-131A], Diffie-Hellman with keys smaller than 2048 bits must not be used.

For complying with the requirement of not allowing Diffie-Hellman key sizes smaller than 2048 bits, the Crypto Officer must ensure that:

- in case the module is used as a TLS server, the Diffie-Hellman parameters (dh argument) of the SSL_CTX_set_tmp_dh() API function must be 2048 bits or larger;

- in case the module is used as a TLS client, the TLS server must be configured to only offer Diffie-Hellman keys of 2048 bits or larger.

Note: the TLS protocol lacks the support to negotiate the used Diffie-Hellman key sizes. Because of this, the TLS client implementation of the Module accepts Diffie-Hellman key sizes smaller than 2048 bits offered by the TLS server to ensure full support for all TLS protocol versions. Therefore, the calling application using the module must always set the DH parameters using the SSL_CTX_set_tmp_dh() API function in order to comply with [SP800-131A].

9.2.3. Random Number Generator

The RAND_cleanup() API function must not be used. Invoking this function will clean up the internal DRBG state. This call also replaces the DRBG instance with the non-FIPS approved SSLeay Deterministic Random Number Generator when using the RAND_* API functions.

9.2.4. 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.

9.2.5. AES XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E]. The length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks that is 16MB of data. To meet the requirement in [FIPS140-2_IG] A.9, the module implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

9.2.6. Triple-DES Keys

Data encryption using the same three-key Triple-DES key shall not exceed 2^{28} Triple-DES blocks (2GB of data), in accordance to [SP800-67] and IG A.13 in [FIPS140-2-IG].

9.2.7. Handling FIPS Related Errors

When the module fails any self-test or conditional test, the module returns an error code to indicate the error and enters the error state, in which any further cryptographic operation is not allowed and output is inhibited. The table below shows the error codes and the event that produce the error.

Error Code / Message	Error Event
FIPS_R_FINGERPRINT_DOES_NOT_MATCH (111)	The Integrity Test fails at power-up.
FIPS_R_SELFTEST_FAILED (134)	When any of the AES, Triple-DES, SHA-1, SHA-512 KATs fails at power-up.
FIPS_R_TEST_FAILURE (137)	When any of the RSA KATs, or the ECDSA or DSA PCTs fails at power-up.
FIPS_R_NOPR_TEST1_FAILURE (145)	When any of the DRBG KATs fails at power-up.

Error Code / Message	Error Event
FIPS_R_PAIRWISE_TEST_FAILED (127)	When the new generated RSA, DSA or ECDSA key pair fails the PCT during key generation.
FIPS_R_ENTROPY_SOURCE_STUCK (142)	When the CRNGT fails on the NDRNG output.
SSL_R_ONLY_TLS_ALLOWED_IN_FIPS_MODE (297)	When SSLv2.0 or SSL v3.0 protocols are used.
FIPS_R_FIPS_SELFTEST_FAILED (115)	When the module is in error state and any cryptographic operation is called
FIPS_R_SELFTEST_FAILED (134)	
FIPS_R_AES_XTS_WEAK_KEY (201)	When the AES key and tweak keys for XTS-AES are the same

Table 13 - Error Codes and Messages

These errors are reported through the regular ERR interface of the modules and can be queried by functions such as ERR_get_error(). See the API documentation for the function description.

When the module is in the error state and the application calls a crypto function of the module that cannot return an error in normal circumstances (void return functions), the error message: "FATAL FIPS SELFTEST FAILURE" is printed to stderr and the application is terminated with the abort() call. The only way to recover from this error is to restart the application. If the failure persists, the module must be reinstalled.

10. Mitigation of Other Attacks

There are no mitigations from other attacks.

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

Cipher Suite	Reference
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
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

Cipher Suite	Reference
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	RFC5289
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	RFC5289

Appendix B. Glossary and Abbreviations

AES	Advanced Encryption Standard
CAVP	Cryptographic Algorithm Validation Program
CAVS	Cryptographic Algorithm Validation System
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DF	Derivation Function
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Schema
KAT	Known Answer Test
KW	AES Key Wrap
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NDRNG	Non-Deterministic Random Number Generator
NVR	Network Video Recorder
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PCT	Pair-wise Consistency Test
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

XTS   XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix C.   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