IBM® Crypto for C version 8.8.1.0

FIPS 140-3 Non-Proprietary Security Policy

Document version: 1.1

Last update: 2024-07-18

Prepared by:
atsec information security corporation
4516 Seton Center Parkway, Suite 250
Austin, TX 78759
www.atsec.com

Table of Contents

List of Tables

1 General

This document is a non-proprietary FIPS 140-3 Security Policy for the IBM® Crypto for C (ICC) cryptographic module. It contains a specification of the rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for a security level 1 multi-chip standalone software module.

The table below shows the security level claimed for each of the twelve sections that comprise the FIPS 140-3 standard.

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic Module Specification	1
3	Cryptographic Module Interfaces	1
4	Roles, Services, and Authentication	1
5	Software/Firmware Security	1
6	Operational Environment	1
7	Physical Security	Not Applicable
8	Non-invasive Security	Not Applicable
9	Sensitive Security Parameter Management	1
10	Self-tests	1
11	Life-cycle Assurance	1
12	Mitigation of Other Attacks	Not Applicable

Table 1 - Security Levels

2 Cryptographic Module Specification

The IBM® Crypto for C cryptographic module is implemented in the C programming language. It is packaged as a dynamic (shared) library usable by applications written in a language that supports C language linking conventions (e.g., C, C++, Java, Assembler, etc.) for use on commercially available operating systems. The ICC allows these applications to access cryptographic functions using an Application Programming Interface (API) provided through an ICC import library and based on the API defined by the OpenSSL group.

The software provided to the customer consists of:

- **ICC shared library** (libicclib84.dll for Windows, libicclib084.so for the rest): shared library (executable code) containing proprietary code needed to meet FIPS and functional requirements not provided by OpenSSL (e.g., entropy source, DRBG, self-tests, startup/shutdown), the OpenSSL cryptographic library and the zlib used for entropy estimation. This shared library constitutes the cryptographic module.
- **ICCSIG.txt file**: contains the signature file used for integrity tests.

The cryptographic module takes advantage of the hardware cryptographic acceleration features supported by the testing platforms that are part of the operational environment, as shown in the following table.

The following table presents the operational environments on which version 8.8.1.0 of the cryptographic module was tested and validated. Each operational environment includes the hardware platform, the processor and the operating system. Each row of the table also includes the corresponding version of the module.

#	Operating System	Hardware Platform	Processor	Acceleration
1	Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian)	Lenovo ThinkSystem SR630	Intel® Xeon® Gold 5217	AES-NI
2	Microsoft Windows Server 2019 64-bit	Lenovo ThinkSystem SR630	Intel® Xeon® Gold 5217	AES-NI
3	Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian) on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	Power ISA
4	Red Hat Linux Enterprise Server 7.9 64-bit (Big Endian) on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	Power ISA
5	IBM AIX 7.2 64-bit (Big Endian) running on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	Power ISA
6	zLinux Red Hat Linux Enterprise Server 8.6 64-bit (Big Endian) on IBM z/VM 7.2	IBM z/15 (8561 T01)	IBM z15	CPACF
7	IBM z/OS 2.3 running on IBM z/VM 7.2	IBM z/15 (8561 T01)	IBM z15	CPACF

Table 2 - Tested Operational Environments

The module maintains its compliance on other operating systems, provided that:

- the operating system meets the operational environment requirements at the module's level of validation;

- the module does not require modification to run in the new environment.

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

The table below lists all approved algorithms of the module, including specific key strengths employed for approved services, and implemented modes of operation. Each algorithm specifies the CAVP certificate for each of the implementations (whether the module was set or unset to take advantage of the processor algorithm acceleration (PAA) or processor algorithm implementation (PAI) capabilities). The selection of the implementations with and without acceleration can be done using the ICC_CAPABILITY_MASK environment variable.

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
With PAA and PAI: A2619	AES [FIPS197] [SP800-38A]	AES-CBC	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38A]	AES-CBC	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38C]	AES-CCM	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38C]	AES-CCM	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38A]	AES-CFB1	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38A]	AES-CFB1	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38A]	AES-CFB128	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38A]	AES-CFB128	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38A]	AES-CFB8	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38A]	AES-CFB8	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
With PAA and PAI: A2619	AES [FIPS197] [SP800-38B]	AES-CMAC	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38B]	AES-CMAC	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	AES [FIPS197] [SP800-38A]	AES-CTR	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38A]	AES-CTR	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38A]	AES-ECB	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38A]	AES-ECB	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38D]	AES-GCM	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38D]	AES-GCM	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38A]	AES-OFB	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38A]	AES-OFB	128, 192 and 256 bits with 128, 192 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
With PAA and PAI: A2619	AES [FIPS197] [SP800-38E]	AES-XTS	128 and 256 bits with 128 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Without PAA and PAI: A2620	AES [FIPS197] [SP800-38E]	AES-XTS	128 and 256 bits with 128 and 256 bits of security strength	Symmetric encryption; Symmetric decryption
Vendor Affirmed	CKG [SP800-133rev2]	RSA key generation [FIPS-186-4]	2048, 3072 and 4096-bit keys with 112-149 bits of security strength	Key pair generation
		ECDSA key generation [FIPS-186-4]	P-224, P-256, P-384, P- 521 keys with 112-256 bits of security strength	

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
		Safe prime key generation [SP800-56Arev3]	2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of security strength	
With PAA and PAI: A2619	CTR_DRBG [SP800-90Arev1]	AES-128, AES-192, AES-256 with/without PR with DF	128, 192, 256-bit keys with 128, 192 and 256 bits of security strength	Random number generation
Without PAA and PAI: A2620	CTR_DRBG [SP800-90Arev1]	AES-128, AES-192, AES-256 with/without PR with DF	128, 192, 256-bit keys with 128, 192 and 256 bits of security strength	Random number generation
With PAA and PAI: A2619	DSA [FIPS186-4]	Signature Verification using SHA2-224 for N=224, SHA2-256 for N=256	L=2048, N=224; L=2048, N=256; L=3072, N=256; with 112 and 128 bits of security strength	Digital signature verification
Without PAA and PAI: A2620	DSA [FIPS186-4]	Signature Verification using SHA2-224 for N=224, SHA2-256 for N=256	L=2048, N=224; L=2048, N=256; L=3072, N=256; with 112 and 128 bits of security strength	Digital signature verification

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
With PAA and PAI: A2619	ECDSA [FIPS 186-4]	ECDSA KeyGen (B.4.2 Testing Candidates)	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Key pair generation
Without PAA and PAI: A2620	ECDSA [FIPS 186-4]	ECDSA KeyGen (B.4.2 Testing Candidates)	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Key pair generation

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
With PAA and PAI: A2619	ECDSA [FIPS 186-4]	Public Key Validation (PKV)	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Key pair validation
Without PAA and PAI: A2620	ECDSA [FIPS 186-4]	Public Key Validation (PKV)	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Key pair validation
With PAA and PAI: A2619	ECDSA [FIPS 186-4]	ECDSA SigGen using SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Digital signature generation
Without PAA and PAI: A2620	ECDSA [FIPS 186-4]	ECDSA SigGen using SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Digital signature generation
With PAA and PAI: A2619	ECDSA [FIPS 186-4]	ECDSA SigVer using SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Digital signature verification
Without PAA and PAI: A2620	ECDSA [FIPS 186-4]	ECDSA SigVer using SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 with 112, 128, 192 and 256 bits of security strength	Digital signature verification
N/A	ENT (NP) SP800-90B	ENT (NP)	N/A	Random number generation
With PAA and PAI: A2619	Hash_DRBG [SP800-90Arev1]	SHA2-224, SHA2-256, SHA2-384, SHA2-512 with/without PR	N/A	Random number generation
Without PAA and PAI: A2620	Hash_DRBG [SP800-90Arev1]	SHA2-224, SHA2-256, SHA2-384, SHA2-512 with/without PR	N/A	Random number generation

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA2-224	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA2-224	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA2-256	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA2-256	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA2-384	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA2-384	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA2-512	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA2-512	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA3-224	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA3-224	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA3-256	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA3-256	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA3-384	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA3-384	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC [FIPS 198-1]	SHA3-512	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
Without PAA and PAI: A2620	HMAC [FIPS 198-1]	SHA3-512	Keys of 112 bits or greater with 112-256 bits of security strength	Message authentication code generation; Message authentication code verification
With PAA and PAI: A2619	HMAC_DRBG [SP800-90Arev1]	SHA2-224, SHA2-256, SHA2-384, SHA2-512 with/without PR	N/A	Random number generation
Without PAA and PAI: A2620	HMAC_DRBG [SP800-90Arev1]	SHA2-224, SHA2-256, SHA2-384, SHA2-512 with/without PR	N/A	Random number generation
With PAA and PAI: A2619	KAS-ECC-SSC [SP800-56Arev3]	Scheme: Ephemeral Unified KAS Role: initiator, responder	Curves: P-224, P-256, P-384, P-521 with 112, 128, 192 and 256 bits of security strength	Shared secret computation
Without PAA and PAI: A2620	KAS-ECC-SSC [SP800-56Arev3]	Scheme: Ephemeral Unified KAS Role: initiator, responder	Curves: P-224, P-256, P-384, P-521 with 112, 128, 192 and 256 bits of security strength	Shared secret computation
With PAA and PAI: A2619	KAS-FFC-SSC [SP800-56Arev3]	Scheme: dhEphem KAS Role: initiator, responder	MODP-2048, MODP- 3072, MODP-4096, MODP-6144, MODP-8192, FFDHE-2048, FFDHE-3072, FFDHE-4096, FFDHE-6144, FFDHE-8192 with 112-200 bits of security strength	Shared secret computation
Without PAA and PAI: A2620	KAS-FFC-SSC [SP800-56Arev3]	Scheme: dhEphem KAS Role: initiator, responder	MODP-2048, MODP- 3072, MODP-4096, MODP-6144, MODP-8192, FFDHE-2048, FFDHE-3072, FFDHE-4096, FFDHE-6144, FFDHE-8192 with 112-200 bits of security strength	Shared secret computation

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
With PAA and PAI: A2619	KDA [SP800-56Crev1] [RFC 5869]	HKDF to support the TLS 1.3 PRF SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	Keys of 112 bits or greater with 112-256 bits of security strength	Key derivation
Without PAA and PAI: A2620	KDA [SP800-56Crev1] [RFC 5869]	HKDF to support the TLS 1.3 PRF SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	Keys of 112 bits or greater with 112-256 bits of security strength	Key derivation
With PAA and PAI: A2619	KTS [SP800-38F]	AES-KW	128, 192 and 256 bits with 112, 192 and 256 bits of security strength	Key wrapping; Key unwrapping
Without PAA and PAI: A2620	KTS [SP800-38F]	AES-KW	128, 192 and 256 bits with 112, 192 and 256 bits of security strength	Key wrapping; Key unwrapping
With PAA and PAI: A2619	KTS [SP800-38F]	AES-KWP	128, 192 and 256 bits with 112, 192 and 256 bits of security strength	Key wrapping; Key unwrapping
Without PAA and PAI: A2620	KTS [SP800-38F]	AES-KWP	128, 192 and 256 bits with 112, 192 and 256 bits of security strength	Key wrapping; Key unwrapping
With PAA and PAI: A2619	PBKDF [SP800-132]	HMAC with: SHA2-224, SHA2-256, SHA2-384, SHA2-512	N/A	Key derivation
Without PAA and PAI: A2620	PBKDF [SP800-132]	HMAC with: SHA2-224, SHA2-256, SHA2-384, SHA2-512	N/A	Key derivation
With PAA and PAI: A2619	RSA [FIPS186-4]	RSA KeyGen (B.3.3 Random Probable Primes)	2048, 3072, and 4096 bits with 112, 128 and 149 bits of security strength	Asymmetric key generation
Without PAA and PAI: A2620	RSA [FIPS186-4]	RSA KeyGen (B.3.3 Random Probable Primes)	2048, 3072, and 4096 bits with 112, 128 and 149 bits of security strength	Asymmetric key generation
With PAA and PAI: A2619	RSA [FIPS186-4]	PKCS#1v1.5 and PSS using SHA2-224, SHA2-256, SHA2-384, SHA2-512 X9.31 using SHA2-256, SHA2-384, SHA2-512	2048, 3072, and 4096 bits with 112, 128 and 149 bits of security strength	Digital signature generation
Without PAA and PAI: A2620	RSA [FIPS186-4]	PKCS#1v1.5 and PSS using SHA2-224, SHA2-256, SHA2-384, SHA2-512 X9.31 using SHA2-256, SHA2-384, SHA2-512	2048, 3072, and 4096 bits with 112, 128 and 149 bits of security strength	Digital signature generation
With PAA and PAI: A2619	RSA [FIPS186-4]	PKCS#1v1.5 and PSS using SHA2-224, SHA2-256, SHA2-384, SHA2-512 X9.31 using SHA2-256, SHA2-384, SHA2-512	2048, 3072, and 4096 bits 112, 128 and 149 bits of security strength	Digital signature verification

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
Without PAA and PAI: A2620	RSA [FIPS186-4]	PKCS#1v1.5 and PSS using SHA2-224, SHA2-256, SHA2-384, SHA2-512 X9.31 using SHA2-256, SHA2-384, SHA2-512	2048, 3072, and 4096 bits 112, 128 and 149 bits of security strength	Digital signature verification
With PAA and PAI: A2619	Safe Primes Key Generation [SP800-56Arev3]	Section 5.6.1.1.4 Testing Candidates	MODP-2048, MODP- 3072, MODP-4096, MODP-6144, MODP-8192, FFDHE-2048, FFDHE-3072, FFDHE-4096, FFDHE-6144, FFDHE-8192 with 112-200 bits of security strength	Key generation
Without PAA and PAI: A2620	Safe Primes Key Generation	Section 5.6.1.1.4 Testing Candidates	MODP-2048, MODP- 3072, MODP-4096, MODP-6144, MODP-8192, FFDHE-2048, FFDHE-3072, FFDHE-4096, FFDHE-6144, FFDHE-8192 with 112-200 bits of security strength	Key generation
With PAA and PAI: A2619	SHA-3 [FIPS 202]	SHA3-224	N/A	Message digest
Without PAA and PAI: A2620	SHA-3 [FIPS 202]	SHA3-224	N/A	Message digest
With PAA and PAI: A2619	SHA-3 [FIPS 202]	SHA3-256	N/A	Message digest
Without PAA and PAI: A2620	SHA-3 [FIPS 202]	SHA3-256	N/A	Message digest
With PAA and PAI: A2619	SHA-3 [FIPS 202]	SHA3-384	N/A	Message digest
Without PAA and PAI: A2620	SHA-3 [FIPS 202]	SHA3-384	N/A	Message digest
With PAA and PAI: A2619	SHA-3 [FIPS 202]	SHA3-512	N/A	Message digest
Without PAA and PAI: A2620	SHA-3 [FIPS 202]	SHA3-512	N/A	Message digest
With PAA and PAI: A2619	SHS [FIPS180-4]	SHA2-224	N/A	Message digest
Without PAA and PAI: A2620	SHS [FIPS180-4]	SHA2-224	N/A	Message digest
With PAA and PAI: A2619	SHS [FIPS180-4]	SHA2-256	N/A	Message digest
Without PAA and PAI: A2620	SHS [FIPS180-4]	SHA2-256	N/A	Message digest
With PAA and PAI: A2619	SHS [FIPS180-4]	SHA2-384	N/A	Message digest

CAVP Cert#	Algorithm / Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
Without PAA and PAI: A2620	SHS [FIPS180-4]	SHA2-384	N/A	Message digest
With PAA and PAI: A2619	SHS [FIPS180-4]	SHA2-512	N/A	Message digest
Without PAA and PAI: A2620	SHS [FIPS180-4]	SHA2-512	N/A	Message digest

Table 3 *- Approved Algorithms*

The Module contains no non-Approved but Allowed security functions, security claimed or otherwise.

The table below lists Non-Approved security functions that are not Allowed in the Approved Mode of Operation.

Algorithm/Functions	Use/Function
DSA with any key sizes	Key pair generation, Domain parameter generation, Digital signature generation
DSA with keys generated with parameters L=512, N=160; L=1024, N=160	Signature verification
ECDSA with P-192, K-163, B-163 elliptic curves	Key pair generation, Key pair validation, Digital signature generation, Digital signature verification
KBKDF	KBKDF key derivation
PBKDF with HMAC using SHA-1	PBKDF key derivation
RSA with keys smaller than 2048 bits	Key generation, Digital signature generation, Digital signature verification
RSA encryption and decryption with any key sizes	RSA encapsulation, RSA unencapsulation
Diffie-Hellman with keys generated with domain parameters other than safe primes	Shared secret computation
EC Diffie-Hellman with P-192, K-163, B-163 elliptic curves	Shared secret computation
DES	Symmetric encryption, Symmetric decryption
Triple-DES	Symmetric encryption, Symmetric decryption
CAST	Symmetric encryption, Symmetric decryption
Camellia	Symmetric encryption, Symmetric decryption
Blowfish	Symmetric encryption, Symmetric decryption
RC2	Symmetric encryption, Symmetric decryption
RC4	Symmetric encryption, Symmetric decryption
MD2	Message digest
MD4	Message digest
MD5	Message digest

Algorithm/Functions	Use/Function
SHA-1	Message digest
HMAC-MD5	Message authentication code generation, Message authentication code verification
HMAC-SHA1	Message authentication code generation, Message authentication code verification
HMAC-DRBG-SHA1	Random number generation
Hash-DRBG-SHA1	Random number generation
MDC2	Message digest
RIPEMD	Message digest
ChaCha20	Symmetric encryption, Symmetric decryption
ChaCha20-Poly1305	Authenticated encryption, Authenticated decryption

Table 4 - Non-Approved Not Allowed in the Approved Mode of Operation

The relationship between ICC and IBM applications is shown in the following diagram. ICC comprises a static stub linked into the IBM application which binds the API functions with the shared library containing the cryptographic functionality.

(Figure 1) below depicts the following information:

- **IBM Application** - The IBM application using ICC. This contains the application code, and the ICC static stub.
- **IBM Application code** - The program using ICC to perform cryptographic functions.
- **ICC static stub**: static library (object code) that is linked into the customer's application and communicates with the Crypto Module. It includes the C headers (source code) containing the API prototypes and other definitions needed for linking the static library. Linked into the calling application to bind the API with the implementation of the cryptographic services in the shared library. This static library is not part of the cryptographic module.
- **ICC shared library** - This contains proprietary code needed to meet FIPS requirements and cryptographic services not provided by OpenSSL, a statically linked copy of zlib used by the entropy source for entropy estimation, and a statically linked copy of the OpenSSL cryptographic library.
- **The cryptographic boundary of the cryptographic module** consists of the ICC shared library bounded by the dashed red line in the figure. The signature used for the integrity check of the ICC during its initialization is contained in the file ICCSIG.txt. This file is considered within the cryptographic boundary.
- **The physical perimeter of the operational environment** is defined to be the enclosure of the computer that runs the ICC software.

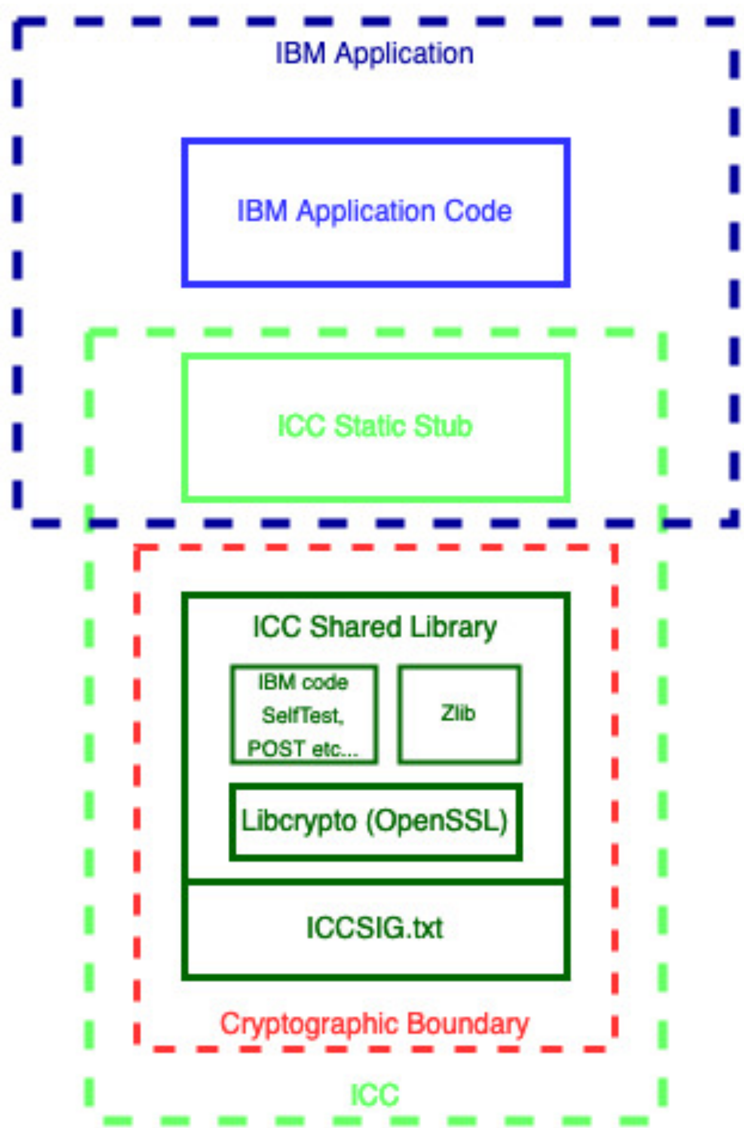

Figure 1 - Logical Block Diagram

3 Cryptographic Module Ports and Interfaces

The ICC meets the requirements of a multi-chip standalone module. Since the ICC is a software module, its interfaces are defined in terms of the API that it provides. These interfaces are described in the following table[1]: Note that because the module is a software only module, there are no physical ports.

Logical Interface	Data that passes over port/interface
Data Input	The input data parameters of those API functions that accept, as their arguments, data to be used or processed by the module.
Data Output	Data output to the caller after generated or otherwise processed by the API functions.
Control Input	The API functions used to control the operation of the module.
Status Output	Defined as the API function ICC_GetStatus that provides information about the status of the module, return codes, and error messages. The function may be called once the context of the module has been obtained.

Table 5 - Ports and Interfaces

[1]The module does not implement a control output interface.

4 Roles, services, and authentication

The ICC assumes the Crypto-Officer role only (there is no User or Maintenance Role). The module does not support operator identification or authentication. Only a single operator assuming the Crypto Officer role may operate the module at any particular moment in time as concurrent operation is not supported.

The module provides a service indicator that specifies, for a given service, whether the service is approved or non-approved. The module provides the ICC_SetValue() function with the ICC_FIPS_CALLBACK parameter to register a callback function using the following prototype:

```
void service_indicator_function(char *function, int nid, int status)
```

This function is invoked by the module whenever a service is requested, providing the service name (function), the algorithm (nid), and the service indicator (status). A status value of 1 means the service is approved, 0 means non-approved.

The module does not identify nor authenticate any user (in any role) that is accessing the module. The Crypto Officer role is implicitly assumed by the services that are requested. The available services are as follows:

Role	Service	Input	Output
Crypto Officer	Symmetric encryption	Plaintext, key	Ciphertext
Crypto Officer	Symmetric decryption	Ciphertext, key	Plaintext
Crypto Officer	Authenticated encryption	Plaintext, key	Ciphertext, authentication tag
Crypto Officer	Authenticated decryption	Ciphertext, key, authentication tag	Plaintext, authentication result (true/false)
Crypto Officer	Key Pair Generation	Key size	Private key, Public key
Crypto Officer	Key Pair Validation	Private key, Public key	Validation result (true/false)
Crypto Officer	Signature generation	Message, hash algorithm, private key,	Signature
Crypto Officer	Signature verification	Message, hash algorithm, public key, signature	Verification result (true/false)
Crypto Officer	Key wrapping	Key wrapping key, key to be wrapped	Wrapped key
Crypto Officer	Key unwrapping	Key wrapping key, wrapped key	Key
Crypto Officer	Shared Secret Computation	Private key, public key from peer	Shared secret
Crypto Officer	Diffie-Hellman key generation	Domain parameters	Diffie-Hellman private key, Diffie-Hellman public key
Crypto Officer	Message digest generation	Message	Message digest
Crypto Officer	Message authentication code generation	Message, key	Message authentication code
Crypto Officer	Message Authentication Code verification	Message, key, message authentication code	Verification result (pass/fail)
Crypto Officer	PBKDF key derivation	Password/passphrase	PBKDF derived key
Crypto Officer	HKDF key derivation	Shared secret	HKDF derived key
Crypto Officer	KBKDF key derivation	Key	KBKDF derived key

Crypto Officer	Random number generation	Number of bits	Random numbers
Crypto Officer	DSA domain parameter generation	Key size	Domain parameters
Crypto Officer	DSA domain parameter verification	Domain parameters	Verification result (true/false)
Crypto Officer	Key encapsulation	Key to be encapsulated, key encapsulating key,	Encapsulated key
Crypto Officer	Key unencapsulation	Encapsulated key, key encapsulating key,	Key
Crypto Officer	Zeroization	Context containing SSPs	none
Crypto Officer	On-Demand Self-test	None	Result of self-test (pass/fail)
Crypto Officer	On-Demand Integrity Test	None	Result of test (pass/fail)
Crypto Officer	Get Status	None	Return codes and/or log messages
Crypto Officer	Module installation and configuration	API invocation	Operational/Error status
Crypto Officer	Show Version	None	Name and version information

Table 6 – Roles and Services

The table below lists all approved services that can be used in the approved mode of operation. The abbreviations of the access rights to keys and SSPs have the following interpretation:

G = **Generate**: The module generates or derives the SSP.

R = **Read**: The SSP is read from the module (e.g. the SSP is output).

W = **Write**: The SSP is updated, imported, or written to the module.

E = **Execute**: The module uses the SSP in performing a cryptographic operation.

Z = **Zeroise**: The module zeroises the SSP.

N/A: The service does not access any SSP during its operation.

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Symmetric encryption	Perform AES encryption	AES-CBC, AES-CFB1, AES-CFB128, AES-CFB8, AES-CTR, AES-ECB, AES-OFB, AES,XTS	AES key	CO	W, E	status =1
Symmetric decryption	Perform AES decryption	AES-CBC, AES-CFB1, AES-CFB128, AES-CFB8, AES-CTR, AES-ECB, AES-OFB, AES,XTS	AES key	CO	W, E	status =1
Authenticated encryption	Perform authenticated AES encryption	AES-CCM, AES-GCM	AES key	CO	W, E	status =1

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Authenticated decryption	Perform authenticated AES decryption	AES-CCM, AES-GCM	AES key	CO	W, E	status =1
DSA signature verification	Verify DSA signatures	DSA	DSA public key	CO	W, E	status =1
ECDSA key pair generation	Generate ECDSA key pairs	ECDSA, DRBG	Module-generated ECDSA private key, Module-generated ECDSA public key	CO	G, R, E	status =1
ECDSA key pair validation	Validate ECDSA key pairs	ECDSA, DRBG	ECDSA private key, ECDSA public key	CO	W, E	status =1
ECDSA signature generation	Sign using ECDSA	ECDSA, DRBG, SHS	ECDSA private key	CO	W, E	status =1
ECDSA signature verification	Verify ECDSA signatures	ECDSA, SHS	ECDSA public key	CO	W, E	status =1
RSA key pair generation	Generate RSA key pairs	RSA, DRBG	Module-generated RSA private key, Module-generated RSA public key	CO	G, R, E	status =1
RSA signature generation	Sign using RSA	RSA, SHS	RSA private key	CO	W, E	status =1
RSA signature verification	Verify RSA signatures	RSA, SHS	RSA public key	CO	W, E	status =1
Key wrapping	Perform AES-based key wrapping	AES-KW, AES-KWP	AES key	CO	W, E	status =1
Key unwrapping	Perform AES-based key unwrapping	AES-KW, AES-KWP	AES key	CO	W, E	status =1
Diffie-Hellman shared secret computation	Perform Diffie-Hellman shared secret computation	KAS FFC SSC	Diffie-Hellman private key	CO	W, E	status =1
			Diffie-Hellman public key from peer		W, E	
			Diffie-Hellman Shared Secret		G, R, E	
EC Diffie-Hellman shared secret computation	Perform Elliptic Curve Diffie-Hellman shared secret computation	KAS ECC SSC	EC Diffie-Hellman private key	CO	W, E	status =1
			EC Diffie-Hellman public key from peer		W, E	
			EC Diffie-Hellman shared secret		G, R, E	

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Diffie-Hellman key pair generation using safe primes	Perform Diffie-Hellman key generation with safe primes	Safe Primes key generation	Module-generated Diffie-Hellman private key, Module-generated Diffie-Hellman public key	CO	G, R, E	status =1
Message digest generation	Compute SHA hashes	SHA2-224, SHA2-256, SHA2-384, SHA2-512	None	CO	N/A	status =1
		SHA3-224, SHA3-256, SHA3-384, SHA3-512	None	CO	N/A	status =1
Message authentication code (MAC) generation	Compute hash-based message authentication	SHA2-224, SHA2-256, SHA2-384, SHA2-512	HMAC key	CO	W, E	status =1
		SHA3-224, SHA3-256, SHA3-384, SHA3-512				
	Compute AES-based message authentication	CMAC with AES	AES key			
Message authentication code (MAC) verification	Verify hash-based message authentication	SHA2-224, SHA2-256, SHA2-384, SHA2-512	HMAC key	CO	W, E	status =1
		SHA3-224, SHA3-256, SHA3-384, SHA3-512				
	Verify AES-based hash-based message authentication	CMAC with AES	AES key			
HKDF key derivation	Key derivation for TLSv1.3 pseudorandom function (PRF)	HKDF	Diffie-Hellman shared secret or EC Diffie-Hellman shared secret	CO	W, E	status =1
			HKDF derived key,		G, R, E	
PBKDF key derivation	Perform password-based key derivation	PBKDF, HMAC with SHA2-224, SHA2-256, SHA2-384, SHA2-512	PBKDF derived key	CO	G, R, E	status =1
			PBKDF password		W, E	
Random number generation	Generate random bitstrings	CTR_DRBG	Entropy Input	CO	W, E	status =1
			DRBG seed		G, E	
			DRBG internal state (V, Key)		G, E	
		Hash_DRBG, HMAC_DRBG	Entropy input		W, E	
			DRBG seed		G, E	
			DRBG internal state (V, C)		G, E	
Get status	Return module status	N/A	None	CO	N/A	status =1

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Show module info	Return module name and versioning information	N/A	None	CO	N/A	status =1
Self-tests	Perform pre-operational and cryptographic algorithm self-tests during power on.	AES, Diffie-Hellman, DSA, EC Diffie-Hellman, ECDSA, DRBG, HKDF, HMAC, RSA, SHS, PBKDF	None	CO	N/A	status =1
On-demand self-tests	Perform cryptographic algorithm self-tests on demand.	AES, Diffie-Hellman, DSA, EC Diffie-Hellman, ECDSA, DRBG, HKDF, HMAC, RSA, SHS, PBKDF	None	CO	N/A	status =1
On-demand integrity test	Perform module integrity test on demand.	RSA	None	CO	N/A	status =1
Zeroization	Zeroize SSPs	N/A	All SSPs	CO	Z	status =1
Module installation and configuration	Configure module for approved mode of operation	N/A	None	CO	N/A	status =1

Table 7 - Approved Services

The following table shows the services and algorithms not allowed in the approved of operation. Requesting these services will implicitly put the module in the non-approved mode of operation.

Service	Description	Algorithms Accessed	Role
Symmetric encryption	Compute the cipher for encryption	Triple-DES, Blowfish, Camellia, CAST, DES, RC2, RC4, ChaCha20	CO
Symmetric decryption	Compute the plaintext for decryption	Triple-DES, Blowfish, Camellia, CAST, DES, RC2, RC4, ChaCha20	CO
Authenticated encryption	Compute the cipher for encryption	ChaCha20-Poly1305	CO
Authenticated decryption	Compute the plaintext for decryption	ChaCha20-Poly1305	CO
DSA parameter generation	Generate DSA parameters	DSA	CO
DSA key generation	Generate DSA key pairs	DSA	CO
DSA signature generation	Sign using DSA	DSA	CO
DSA signature verification	Verify DSA signatures	DSA with keys generated with L=512, N=160; L=1024, N=160	CO
ECDSA key pair generation	Generate ECDSA key pairs	ECDSA with P-192, K-163, B-163 curves	CO
ECDSA key pair validation	Validate ECDSA key pairs	ECDSA with P-192, K-163, B-163 curves	CO
ECDSA signature generation	Sign using ECDSA	ECDSA with P-192, K-163, B-163 curves	CO
ECDSA signature Verification	Verify using ECDSA	ECDSA with P-192, K-163, B-163 curves	CO
RSA key generation	Generate RSA key pairs	RSA with keys smaller than 2048 bits	CO
RSA signature generation	Sign using RSA	RSA with keys smaller than 2048 bits	CO

Service	Description	Algorithms Accessed	Role
RSA signature verification	Verify RSA signatures	RSA with keys smaller than 2048 bits	CO
Key encapsulation	Perform RSA encapsulation	RSA encryption	CO
Key unencapsulation	Perform RSA unencapsulation	RSA decryption	CO
Diffie-Hellman shared secret computation	Shared secret computation	KAS-FFC-SSC with parameters other than safe primes	CO
EC Diffie-Hellman shared secret computation	Shared secret computation	KAS-ECC-SSC with P-192, K-163, B-163 curves	CO
Message digest	Hashing algorithms	SHA-1, MD2, MD4, MD5, MDC2, RIPEMD	CO
Random Number Generation	Generate random bitstrings	Hash_DRBG or HMAC_DRBG using SHA-1	CO
Message authentication code (MAC) generation	Compute MAC	HMAC-MD5, HMAC-SHA-1	CO
Message authentication code (MAC) verification	Verify MAC	HMAC-MD5, HMAC-SHA-1	CO
Key Derivation Functions (KDF)	Key derivation	KBKDF, PBKDF with SHA-1	CO

Table 8 - Non-Approved Services

5 Software/Firmware security

5.1 Integrity Techniques

The services provided by the Module to a User are effectively delivered using appropriate API calls. When a client process attempts to load an instance of the Module into memory, the Module runs an integrity test and the cryptographic algorithm self-tests. If all the tests pass successfully, the Module makes a transition to the "Operational" state, where the API calls can be used by the client to obtain desired cryptographic services. Otherwise, the Module enters to “Error” state and returns an error to the calling application. When the Module is in “Error” state, no services are available, and all of data input and data output except the status information are inhibited.

The module uses an integrity test which uses a 2048-bit CAVP-validated RSA signature verification (PKCS#1v1.5) and SHA2-256 hashing. This RSA public key is stored inside the shared library.

5.2 On-Demand Integrity Test

Integrity tests are performed as part of the Pre-Operational Self-Tests. They are automatically executed at power-on. Integrity tests can also be requested on demand through the API function ICC_IntegrityCheck.

6 Operational Environment

6.1 Applicability

The IBM® Crypto for C operates in a modifiable operational environment per FIPS 140-3 level 1 specifications. It is part of a commercially available general-purpose operating system executing on the hardware specified in section 2.

6.2 Requirements

The following operational rules must be followed by any user of the cryptographic module:

1. Since the ICC runs on a general-purpose processor all main data paths of the computer system will contain cryptographic material. The following items need to apply relative to where the ICC will execute:
    - Virtual (paged) memory must be secure (local disk or a secure network)
    - The disk drive where ICC is installed must be in a secure environment.

The above rules must be always upheld in order to ensure continued system security and FIPS 140-3 approved mode compliance after initial setup of the validated configuration. If the module is removed from the above environment, it is assumed not to be operational in the validated mode until such time as it has been returned to the above environment and re-initialized by the user to the validated condition.

7 Physical Security

The FIPS 140-3 physical security requirements do not apply to the IBM® Crypto for C, since it is a software module.

8 Non-invasive Security

Currently, the non-invasive security is not required by FIPS 140-3 (see NIST SP 800-140F). The requirements of this area are not applicable to the module.

9 Sensitive Security Parameter Management

The following table summarizes the keys and Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module:

Key/SSP Name /Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
AES key	128, 192, 256 bits	AES-CBC, AES-CCM, AES-CFB1, AES-CFB128, AES-CFB8, AES-CMAC, AES-CTR, AES-GCM, AES-KW, AES-KWP, AES- OFB, AES-XTS, CTR-DRBG A2619, A2620	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Symmetric encryption, Symmetric decryption, Authenticated encryption, Authenticated decryption, Message authenticated code (MAC) generation, Message authenticated code (MAC) verification. **Related SSPs:** None
HMAC key	112 to 256 bits	HMAC A2619, A2620	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A	N/A		Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Message authenticated code (MAC) generation, Message authenticated code (MAC) verification. **Related SSPs:** None
Module-generated ECDSA private key	112, 128, 192, 256 bits	ECDSA A2619, A2620	B.4.2 Testing Candidates Generated using the testing candidates method specified in FIPS 186-4; random values are obtained from the SP800-90Arev1 DRBG	**Import:** N/A. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Key generation **Related SSPs:** Module-generated ECDSA public key
Module-generated ECDSA public key	112, 128, 192, 256 bits	ECDSA A2619, A2620						**Use:** Key generation **Related SSPs:** Module-generated ECDSA private key
ECDSA private key	112, 128, 192, 256 bits	ECDSA A2619, A2620	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is	**Use:** Key pair validation, Digital signature generation **Related SSPs:** ECDSA

Key/SSP Name /Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
				plaintext (P) format. **Export:** N/A			powered down	public key
ECDSA public key	112, 128, 192, 256 bits	ECDSA A2619, A2620						**Use:** Key pair validation, Digital signature verification **Related SSPs:** ECDSA private key
Module-generated RSA private key	112, 128, 149 bits	RSA A2619, A2620	Generated using the random probable primes method (B.3.3) specified in FIPS 186-4; random values are obtained from the SP800-90Arev1 DRBG.	**Import:** N/A **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Key generation **Related SSPs:** Module-generated RSA public key
Module-generated RSA public key	112, 128, 149 bits	RSA A2619, A2620						**Use:** Digital signature verification **Related SSPs:** Module-generated RSA private key
RSA private key	112, 128, 149 bits	RSA A2619, A2620	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Digital signature generation **Related SSPs:** RSA public key
RSA public key	112, 128, 149 bits	RSA A2619, A2620						**Use:** Digital signature verification **Related SSPs:** RSA private key
DSA public key	112, 128 bits	DSA signature verification, A2619, A2620	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Digital Signature Verification **Related SSPs:** None
Entropy Input IG D.L compliant	192 to 384 bits	CTR_DRBG, HMAC_DRBG, Hash_DRBG A2619, A2620	Obtained from the SP800-90B ENT (NP)	**Import:** N/A. **Export**: N/A. It remains within	N/A	RAM	Automatic zeroization when structure is deallocated or when the	**Use:** Random number generation, Key generation, Digital

Key/SSP Name /Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
				the cryptographic boundary.			system is powered down	signature generation **Related SSPs:** DRBG seed
DRBG seed IG D.L compliant	192 to 384 bits	CTR_DRBG, HMAC_DRBG, Hash_DRBG A2619, A2620	Derived from the entropy input as defined by SP800-90Arev1		N/A	RAM		**Use:** Random number generation, Key generation, Digital signature generation **Related SSPs:** Entropy Input, DRBG internal state
DRBG internal state (V, C) IG D.L compliant	128 to 256 bits	HMAC_DRBG, Hash_DRBG A2619, A2620	Computed as defined by SP800-90Arev1		N/A	RAM		**Use:** Random number generation, Key generation, Digital signature generation **Related SSPs:** DRBG seed
DRBG internal state (V, Key) IG D.L compliant	128 to 256 bits	CTR_DRBG A2619, A2620			N/A	RAM		**Use:** Random number generation, Key generation, Digital signature generation **Related SSPs:** DRBG seed
PBKDF derived key	112 to 256 bits	PBKDF A2619, A2620	Generated during the PBKDF compliant with [SP800-132]	**Import:** N/A. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	Automatic zeroization when the system is powered down	**Use:** PBKDF key derivation **Related SSPs:** PBKDF password
PBKDF password	N/A	PBKDF A2619, A2620	N/A	**Import:** CM from TOEPP Path. Passed to the module via API	N/A	RAM		**Use:** PBKDF key derivation **Related**

Key/SSP Name /Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
				parameters in plaintext (P) format. **Export:** N/A.				**SSPs:** PBKDF derived key
HKDF derived key	112 to 256 bits	HKDF A2619, A2620	Generated in accordance with SP800-56Crev1 Extraction and Expansion procedure, as referenced in SP800-135rev1	**Import:** N/A. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	Automatic zeroization when the system is powered down	**Use:** HKDF key derivation **Related SSPs:** None
Module-generated Diffie-Hellman private key	112 to 200 bits	KAS-FFC-SSC A2619, A2620	Generated using safe prime key generation method specified in SP800-56Arev3; random values are obtained from the SP800-90Arev1	**Import:** N/A. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Key generation **Related SSPs:** Module-generated Diffie-Hellman public key
Module-generated Diffie-Hellman public key					N/A	RAM		**Use:** Key generation **Related SSPs:** Module-generated Diffie-Hellman private key
Diffie-Hellman private key	112 to 200 bits	KAS-FFC-SSC A2619, A2620	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Diffie-Hellman shared secret computation **Related SSPs:** Diffie-Hellman shared secret
Diffie-Hellman public key from peer					N/A	RAM		**Use:** Diffie-Hellman shared secret computation **Related SSPs:** Diffie-Hellman shared secret
Diffie-Hellman shared secret	112 to 200 bits	KAS-FFC-SSC A2619, A2620	N/A	**Import:** N/A. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P)	Computed during the Diffie-Hellman shared secret computation per SP800-	RAM		**Use:** Diffie-Hellman shared secret computation **Related SSPs:** Diffie-Hellman private key,

Key/SSP Name /Type	Strength	Security Function and Cert. Number	Generation	Import/Export	Establishment	Storage	Zeroisation	Use & related keys
				format.	56Arev3.			Diffie-Hellman public key from peer
Module-generated EC Diffie-Hellman private key	112 to 256 bits	KAS-ECC-SSC [A2619](), [A2620]()	Generated using the testing candidates method specified in SP800-56Arev3; random values are obtained from the SP800 90Arev1 DRBG.	**Import:** N/A. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** Key generation **Related SSPs:** Module-generated EC Diffie-Hellman public key
Module-generated EC Diffie-Hellman public key					N/A	RAM		**Use:** Key generation **Related SSPs:** Module-generated EC Diffie-Hellman private key
EC Diffie-Hellman private key	112 to 256 bits	EC Diffie Hellman shared secret computation, [A2619](), [A2620]()	N/A	**Import:** CM from TOEPP Path. Passed to the module via API parameters in plaintext (P) format. **Export:** N/A.	N/A	RAM	Automatic zeroization when structure is deallocated or when the system is powered down	**Use:** EC Diffie-Hellman shared secret computation **Related SSPs:** EC Diffie-Hellman shared secret
EC Diffie-Hellman public key from peer					N/A	RAM		**Use:** EC Diffie-Hellman shared secret computation **Related SSPs:** EC Diffie-Hellman shared secret
EC Diffie-Hellman shared secret	112 to 256 bits	EC Diffie Hellman shared secret computation, [A2619](), [A2620]()	N/A	**Import:** N/A. **Export:** CM to TOEPP Path. Passed from the module via API parameters in plaintext (P) format.	Computed during the EC Diffie-Hellman shared secret computation per SP800-56Arev3.	RAM		**Use:** EC Diffie-Hellman shared secret computation **Related SSPs:** EC Diffie-Hellman private key, EC Diffie-Hellman public key from peer

Table 9 – SSPs

9.1 Random Number Generation

ICC employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90Arev1] for the creation of asymmetric keys. In addition, the module provides a Random Number Generation service to calling applications.

The default algorithm is Hash_DRBG using SHA2-256 with no prediction resistance, but another algorithm from the Hash_DRBG, HMAC_DRBG and CTR_DRBG algorithms (see Table 3 for the complete list) can be also configured.

ICC uses the entropy source to seed the DRBG. The entropy source is a non-physical entropy source ENT (NP) that obtains noise from time jitter produced by the CPU and detected through the CPU high-resolution timer. ENT(NP) is compliant with [SP800-90B], and guarantees an entropy rate of 0.5 bits per bit.

The DRBG entropy input and nonce to form the seed are of the same length (64 bytes = 512 bits each) and obtained from separate and independent calls to the entropy source. Then, the DRBG is seeded during initialization with the entropy input and nonce containing 512 bits of entropy ((512 + 512) * 0.5 = 512), and with the entropy input containing 256 bits of entropy (512 * 0.5) during reseeding . Therefore, the DRBG supports 256 bits of effective security strength in its output.

Entropy Source	Minimum number of bits of entropy	Details
NIST SP800-90B compliant ENT (NP)	256	The seed is provided by the post-processed entropy data from non-physical noise source provided by CPU time jitter.

Table 10 - Non-Deterministic Random Number Generation Specification

9.2 SSPs Generation

The module generates Keys and SSPs in accordance with FIPS 140-3 IG D.H. The cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per section 4, example 1, and section 5.1 [SP800-133rev2], compliant with [FIPS186-4] and [SP800-56Arev3]. A seed used for key generation is a direct output from DRBG compliant with [SP800-90A]. The security strength of 256 bits of the DRBG is equal to the security strength of the maximum key size that can be generated by the module.

The key generation services for RSA, Diffie-Hellman and EC key pairs as well as the [SP 800-90A] DRBG have been tested under the CAVP with algorithm certificates found in Table 3.

The ICC provides the following key derivation services in the approved mode of operation:

- PBKDF Key Derivation
    - For PBKDF, the module implements a CAVP compliance tested key derivation function compliant to [SP800-132] and IG D.N. The service returns the key derived from the provided password to the caller.
- HKDF Key Derivation
    - The module provides [SP800-135rev1] compliant key derivation function in accordance with [SP800-56Crev1] two-step key derivation, extraction and expansion procedure. The derived keys provide between 112 and 256 bits of security strength.

9.3 SSPs Establishment

The ICC uses the following key establishment methodologies in the approved mode of operation:

- Diffie-Hellman (DH) shared secret computation
    - The module provides SP800-56Arev3 compliant key agreement schemes according to FIPS 140-3 IG D.F scenario 2 path (1) with DH shared secret computation. The shared secret computation provides between 112 and 200 bits of encryption strength.

- Elliptic Curve Diffie-Hellman (ECDH) shared secret computation
    - The module provides SP800-56Arev3 compliant key agreement schemes according to FIPS 140-3 IG D.F scenario 2 path (1) with ECDH shared secret computation. The shared secret computation provides between 112 and 256 bits of encryption strength.
- AES key wrapping
    - The module implements a Key Transport Scheme (KTS) using AES-KW and AES-KWP compliant to [SP800-38F] and IG D.G. The SSP establishment methodology provides between 128 and 256 bits of encryption strength.

9.4 SSPs Import/Export

Keys/SSPs are entered into and output from the ICC module in electronic form through the data input and output interface (i.e. API function parameters). The ICC module does not support manual key entry or intermediate key generation key output. The SSPs are provided to the module via API input parameters in the plaintext form and output via API output parameters in the plaintext form to and from the calling application.

9.5 SSPs Storage

The module does not provide any long-term key storage and no keys are ever stored on the hard disk.

9.6 SSPs Zeroization

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The calling application that is acting as the Crypto Officer is responsible for calling the appropriate functions provided in the module's API to zeroize the memory areas allocated by the module.

Key zeroization services for cipher contexts are performed via the following API functions.

- ICC_BN_clear_free(): clean up big numbers
- ICC_BN_CTX_free(): clean up memory used by low-level big number arithmetic functions
- ICC_EVP_MD_CTX_cleanup(): clears Message Digest context
- ICC_EVP_CIPHER_CTX_cleanup(): clean up symmetric cipher context
- ICC_RSA_free(): clean up RSA context
- ICC_DSA_free(): clean up DSA context
- ICC_DH_free(): clean up Diffie-Hellman context
- ICC_EVP_PKEY_free(): clean up asymmetric key contexts
- ICC_HMAC_CTX_free(): clean up HMAC context
- ICC_EC_KEY_free(): clean up ECDSA and ECDH contexts
- ICC_CMAC_CTX_free(): clean up CMAC context
- ICC_AES_GCM_CTX_free(): clean up AES-GCM context
- ICC_RNG_CTX_free(): clean up RNG context

The zeroization functions overwrite the memory occupied by SSPs with “zeros” and deallocate the memory with the regular memory deallocation operating system call. The completion of a zeroization routine(s) will indicate that a zeroization procedure succeeded.

10 Self-tests

The ICC module implements a number of self-tests to check proper functioning of the module. This includes pre-operational self-tests and conditional self-tests.

Pre-operational integrity test and Cryptographic Algorithm Self-Tests (CASTs) are automatically invoked by the module when the module is powered on from the default entry point (DEP) of the shared library..

When the module is performing self-tests, no API functions are available, and no data output is possible until the self-tests are successfully completed. After the pre-operational self-tests and CASTs are successfully completed, the module turns to approved mode of operation. Requesting any services from Table 8 will implicitly put the module in the non-approved mode of operation.

The module performs self-tests automatically when it is loaded. Self-tests can also be requested on demand through the API functions ICC_SelfTest() and ICC_IntegrityCheck().

Whenever the startup tests are initiated the module performs the following; if any of these tests fail, the module enters the error state:

10.1 Pre-operational Software Integrity Test

The module performs a pre-operational software integrity test automatically when the module is powered on, before the module transitions into the operational state. The integrity test is performed with a 2048-bit CAVP-validated RSA signature verification (PKCS#1v1.5) and SHA2-256 hashing. This RSA public key is stored inside the shared library.

Prior to the invocation of the integrity test, the module runs the conditional Cryptographic Algorithm Self-Test (CAST) for RSA (2048-bit keys with SHA2-256) which verifies the proper functioning of all algorithms used as part of the integrity test.

10.2 Conditional Self-Tests

The following sections describe the conditional tests supported by the IBM® Crypto for C.

10.2.1 Cryptographic Algorithm Self-Tests

The IBM® Crypto for C runs all Cryptographic Algorithm Self-Tests during power-up, and consequently before the first operational use of the cryptographic algorithms. These tests are detailed in the following table.

Cryptographic Algorithm	Notes
AES–CBC with 256 bits AES–GCM with 128 bits AES–CCM with 128 bits AES–XTS with 128 bits	Separate encryption / decryption KATs are performed
AES KW and KWP with 128 bits	Separate wrapping / unwrapping KATs are performed
SHA3-512, SHAKE-128	KATs
HMAC-SHA2-256 HMAC-SHA2-512	KAT
CMAC with AES	KAT
SHA2-256, SHA2-512	Covered by high level HMAC self-tests

Cryptographic Algorithm	Notes
RSA with 2048-bit keys and SHA2-256	Separate signature generation/ verification KAT are performed
ECDSA with curves P-384 and B-233 and using SHA2-256	Separate signature generation / verification KAT are performed
DSA with L=2048, N=224 and SHA2-256	Signature verification KAT
Hash_DRBG with SHA-224, SHA-256, SHA-384 and SHA-512 HMAC_DRBG with SHA-224, SHA-256, SHA-384 and SHA-512 CTR_DRBG with AES-128, AES-192 and AES-256	Each DRBG mode tested separately.
DRBG health tests	Health tests according to section 11.3 of [SP800-90Arev1]
HKDF using SHA2-256	KAT
PBKDF using SHA2-256	KAT
Diffie-Hellman “Z” computation with 2048-bit key	KAT
EC Diffie-Hellman “Z” computation with P-521 curve	KAT
Repetitive Counter Test (RCT)	Startup tests of the ENT(NP) entropy source. Performed on 1024 consecutive samples.
Adaptive Proportion Test (APT)	Startup tests of the ENT(NP) entropy source. Performed on 1024 consecutive samples.

Table 11 - Cryptographic Algorithm Self-Tests

10.2.2 Pairwise Consistency Test

The IBM® Crypto for C does generate asymmetric keys and performs all required pair-wise consistency tests. The consistency of the keys is tested by the calculation and verification of a digital signature. If the digital signature cannot be verified, the test fails. Pair-wise consistency tests are performed on the following algorithms:

- ECDSA signature generation and verification using SHA2-256.
- RSA signature generation and verification using SHA2-256.
- Diffie-Hellman according to section 5.6.2.1.4 of [SP800-56Arev3].

EC Diffie-Hellman is covered by ECDSA PCT as allowed by IG 10.3, additional comment 1.

10.2.3 Entropy Health Test

The ICC module performs health tests during the startup of the ENT(NP), and continuously during its operation, to detect intermittent and permanent failures in the noise source. The health tests implemented are the Repetitive Count Test (RCT) and Adaptive Proportion Test (APT), both compliant with the requirements of SP800-90B, and the minimum-entropy assessment test, which analyzes whether the noise source provides the expected entropy rate using the min-entropy calculation formula as specified in section 2.1 of SP800-90B.

If the ICC module detects a permanent failure in any of the health tests, the module transitions to the error state and an error message is shown (“Insufficient entropy”).

10.3 Error Handling

When errors are detected (e.g., self-test failure) then all security related functions are disabled and no partial data is exposed through the data output interface. The only way to transition from the error state to an operational state is to reinitialize the cryptographic module (from an uninitialized state). The error state can be retrieved via the Show Status service.

11 Life-cycle assurance

11.1 Delivery and Operation

The following steps must be performed to install and initialize the module for operating in a FIPS 140-3 compliant manner:

1. The operating system must be configured to operate securely and to prevent remote login. This is accomplished by disabling all services (within the Administrative tools) that provide remote access (e.g., – ftp, telnet, ssh, and server) and disallowing multiple operators to log in at once.
2. Before the module initialization, the user has a choice to configure the default DRBG algorithm to use. This can be set using the environment variable 'ICC_RANDOM_GENERATOR'.
3. The module is initialized automatically when the shared library is loaded in the calling application process space. The module executes the pre-operational self tests (POST) and, if they are successful, the module enters the approved mode of operation. The calling application must include the following calling sequence to have access to the cryptographic services:
   - ***ICC_Init()*** creates the crypto module context.
   - ***ICC_Attach()*** binds the cryptographic functions with the API entry points.

11.2 Crypto Officer Guidance

It is the responsibility of the Crypto-Officer to configure the operating system to operate securely.

The services provided by the Module to a User are effectively delivered by using the appropriate API calls. When a client process attempts to load an instance of the Module into memory, the Module runs an integrity test and several of cryptographic functionality self-tests. If all the tests pass successfully, the Module makes a transition to the "Operational" state, where the API calls can be used by the client to obtain desired cryptographic services. Otherwise, the Module enters to "Error" state and returns an error to the calling application. When the Module is in "Error" state, no services are available, and all of data input and data output except the status information are inhibited.

The Crypto Officer shall consider the following requirements and restrictions when using the module:

1. 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 (16MB of data).
2. To meet the requirement in [FIPS140-3-IG] C.I, the module implements a check to ensure that the two AES keys used in the XTS-AES algorithm are not identical.
3. AES-GCM IV is constructed in compliance with IG C.H scenario 1. In case the module's power is lost and then restored, the keys used for the AES GCM encryption/decryption shall be re-distributed. The GCM is used in the context of TLS version 1.2. The mechanism for IV generation is compliant with RFC 5288 as described in Section 3.3.1 of SP800-52rev2. The design of the TLS protocol implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all its possible values.
4. The module also offers an AES-GCM implementation under the context of Scenario 5 of IG C.H. The protocol that provides this compliance is TLS 1.3, using the ciphersuites that

explicitly select AES-GCM as the encryption/decryption cipher. The module supports acceptable AES-GCM ciphersuites from Section 3.3.1 of SP800-52rev2.

The design of the TLS protocol implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all its possible values. In the event the module's power is lost and restored, the consuming application must ensure that new AES-GCM keys encryption or decryption under this scenario are established. TLS 1.3 provides session resumption, but the resumption procedure derives new AES-GCM encryption keys.

5. For PBKDF, the module implements a CAVP compliance tested key derivation function compliant to [SP800-132] and IG D.N. The service returns the key derived from the provided password to the caller.

   PBKDF is implemented to support the option 1a specified in section 5.4 of [SP800-132]. The keys derived from [SP800-132] map to section 4.1 of [SP800-133rev2] as indirect generation from DRBG.

   In accordance with [SP800-132], the following requirements shall be met:

   a. Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The length of the MK or Data Protection Key (DPK) shall be of 112 bits or more.
   b. A portion of the salt, with a length of at least 128 bits, shall be generated randomly using the SP800-90A DRBG,
   c. The iteration count shall be equal or greater than 1000, so as to make the key derivation computationally intensive.
   d. Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.
   e. The length of the password or passphrase shall be at least ten characters long, and may consist of lower-case, upper-case, numeric, or special characters. At a minimum length of ten characters, and assuming a worst case scenario where the password uses a combination of only lower case and numbers (36 symbols), the chance of randomly guessing this password is $1 / 36^{10} = 3.656\ 10^{-15}$.

6. For SHA-3 algorithms, the module implements HMAC with SHA3-224, SHA3-256, SHA3-384, SHA3-512. The CAVP certificates have been obtained for the HMAC and HKDF algorithms as well as for all the SHA-3 implementations. The CAVP certificates are listed in Table 3 in Section 2.
7. The module implements FIPS 186-4 RSA SigGen and SigVer. RSA SigGen is supported with key sizes of 2048, 3072, 4096 bits while RSA SigVer is supported with 1024, 2048, 3072, 4096 bits. All RSA key sizes have been CAVP tested with the certificates listed in Table 3 in Section 2.
8. For Diffie-Hellman or EC Diffie-Hellman shared secret computation, the module has to comply with the assurances found in Section 5.6.2 of [SP800-56Arev3] and IG D.F. The operator must obtain the ephemeral Diffie-Hellman or EC Diffie-Hellman key pairs on both ends either by using the approved key pair generation service provided by the module, or by using another FIPS-validated module. As part of the key pair generation service, the module internally performs the full key validation of the generated key pair. Similarly, the shared secret computation service internally performs the full public key validation of the peer public key, complying with Sections 5.6.2.2.1 and 5.6.2.2.2 of [SP800-56Arev3].

The module code is a component provided to IBM products, not a product on its own. Typically it is provided as part of IBM's SSL component and creates packaging with the OS specific install tools.

The module's End-of-Life/sanitization procedure can take one of two forms:

- OS uninstall which removes the package after checking that no currently installed package still depends on it.
  The module does not possess persistent storage of SSPs. The SSP value only exists in volatile memory and that value vanishes when the module is powered off. The procedure for secure sanitization of the module at the end of life is simply to power it off, which is the action of zeroization of the SSPs . As a result of this sanitization via power-off, the SSP is removed from the module, so that the module may either be distributed to other operators or disposed.

- Alternatively the package may be upgraded and replaced by a newer version.

12 Mitigation of other attacks

The cryptographic module is not designed to mitigate any specific attacks.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
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
ENT	NIST SP 800-90B compliant Entropy Source
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
FSM	Finite State Model
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Schema
KAT	Known Answer Test
KW	AES Key Wrap
KWP	AES Key Wrap with Padding
MAC	Message Authentication Code
NDF	No Derivation Function
NIST	National Institute of Science and Technology
OFB	Output Feedback
O/S	Operating System
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme

RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSH	Secure Shell
TDES	Triple-DES
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix B. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements for Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
SP 800-140x	**CMVP FIPS 140-3 Related Reference** https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-standards
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** September 2020 https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
FIPS140-3_MM	**CMVP FIPS 140-3 Management Manual** September 2020 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS-140-3-CMVP%20Management%20Manual.pdf
FIPS180-4	**Secure Hash Standard (SHS)** March 2012 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-4	**Digital Signature Standard (DSS)** July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS197	**Advanced Encryption Standard** November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography** Specifications Version 2.1 February 2003 https://www.ietf.org/rfc/rfc3447.txt
RFC3394	**Advanced Encryption Standard (AES) Key Wrap Algorithm** September 2002 https://www.ietf.org/rfc/rfc3394.txt
RFC5649	**Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm** September 2009 https://www.ietf.org/rfc/rfc5649.txt

SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://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 https://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 https://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 https://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 https://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-56Arev3	**Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography** April, 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP800-56Crev1	**Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.pdf
SP800-57rev5	**NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management Part 1: General** May 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
SP800-90Arev1	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-90B	**NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

SP800-108	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions (Revised)** October 2009 https://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
SP800-131Arev2	**Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP800-133rev2	**Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf