IBM Corporation

IBM® Crypto for C

FIPS 140-3 Non-Proprietary Security Policy

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

Table of Contents

1 General 5
1.1 Overview 5
1.2 Security Levels 5
1.3 Additional Information 5
2 Cryptographic Module Specification 6
2.1 Description 6
2.2 Tested and Vendor Affirmed Module Version and Identification 8
2.3 Excluded Components 10
2.4 Modes of Operation 10
2.5 Algorithms 11
2.6 Security Function Implementations 16
2.7 Algorithm Specific Information 20
2.7.1 AES-GCM 20
2.7.2 AES-XTS 20
2.7.3 PBKDF2 20
2.7.4 Diffie-Hellman and EC Diffie-Hellman 21
2.7.5 Key Wrapping 21
2.7.6 Key Agreement 21
2.7.7 Legacy Algorithms 21
2.8 RBG and Entropy 21
2.9 Key Generation 22
2.10 Key Establishment 22
3 Cryptographic Module Interfaces 24
3.1 Ports and Interfaces 24
4 Roles, Services, and Authentication 25
4.1 Authentication Methods 25
4.2 Roles 25
4.3 Approved Services 25
4.4 Non-Approved Services 33
4.5 External Software/Firmware Loaded 35
5 Software/Firmware Security 36
5.1 Integrity Techniques 36
5.2 Initiate on Demand 36
6 Operational Environment 37
6.1 Operational Environment Type and Requirements 37

6.2 Configuration Settings and Restrictions 37
7 Physical Security....38
8 Non-Invasive Security 39
9 Sensitive Security Parameters Management....40
9.1 Storage Areas....40
9.2 SSP Input-Output Methods 40
9.3 SSP Zeroization Methods 40
9.4 SSPs....41
10 Self-Tests 51
10.1 Pre-Operational Self-Tests 51
10.2 Conditional Self-Tests 51
10.3 Periodic Self-Test Information 55
10.4 Error States 57
10.5 Operator Initiation of Self-Tests 57
11 Life-Cycle Assurance....58
11.1 Installation, Initialization, and Startup Procedures 58
11.2 Administrator Guidance 58
11.3 Non-Administrator Guidance 58
11.4 End of Life 58
12 Mitigation of Other Attacks....59
Appendix A. Glossary and Abbreviations 60
Appendix B. References 61

List of Tables

Table 1: Security Levels 5
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) 9
Table 3: Tested Operational Environments - Software, Firmware, Hybrid 10
Table 4: Modes List and Description 10
Table 5: Approved Algorithms 15
Table 6: Vendor-Affirmed Algorithms 15
Table 7: Non-Approved, Not Allowed Algorithms 16
Table 8: Security Function Implementations 19
Table 9: Entropy Sources 22
Table 10: Ports and Interfaces 24
Table 11: Roles 25
Table 12: Approved Services 33
Table 13: Non-Approved Services 34
Table 14: Storage Areas 40
Table 15: SSP Input-Output Methods 40
Table 16: SSP Zeroization Methods 41
Table 17: SSP Table 1 47
Table 18: SSP Table 2 50
Table 19: Pre-Operational Self-Tests 51
Table 20: Conditional Self-Tests 55
Table 21: Pre-Operational Periodic Information 55
Table 22: Conditional Periodic Information 56
Table 23: Error States 57

List of Figures

Figure 1: Block Diagram 7

1 General

1.1 Overview

This document is a non-proprietary FIPS 140-3 Security Policy for version 8.8.1.0 of the IBM® Crypto for C (ICC) cryptographic module. It contains the security 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 an overall security level 1 module.
This non-proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

1.2 Security Levels

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	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

1.3 Additional Information

In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The IBM® Crypto for C cryptographic module (hereafter referred to as "the module" or "ICC") is defined as a software module in a multi-chip standalone embodiment. The module is a software library which provides a C language application program interface (API) for use by other applications that require cryptographic functionality.

The module is implemented in the C programming language and 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 module provides allows these applications to access cryptographic functions using an Application Programming Interface (API) provided through an ICC import library. The module uses OpenSSL as a base for most of the cryptographic functionality.

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.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

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.

Tested Operational Environment's Physical Perimeter (TOEPP):

The tested operating environment's physical perimeter is the general-purpose computer on which the module is running.

.

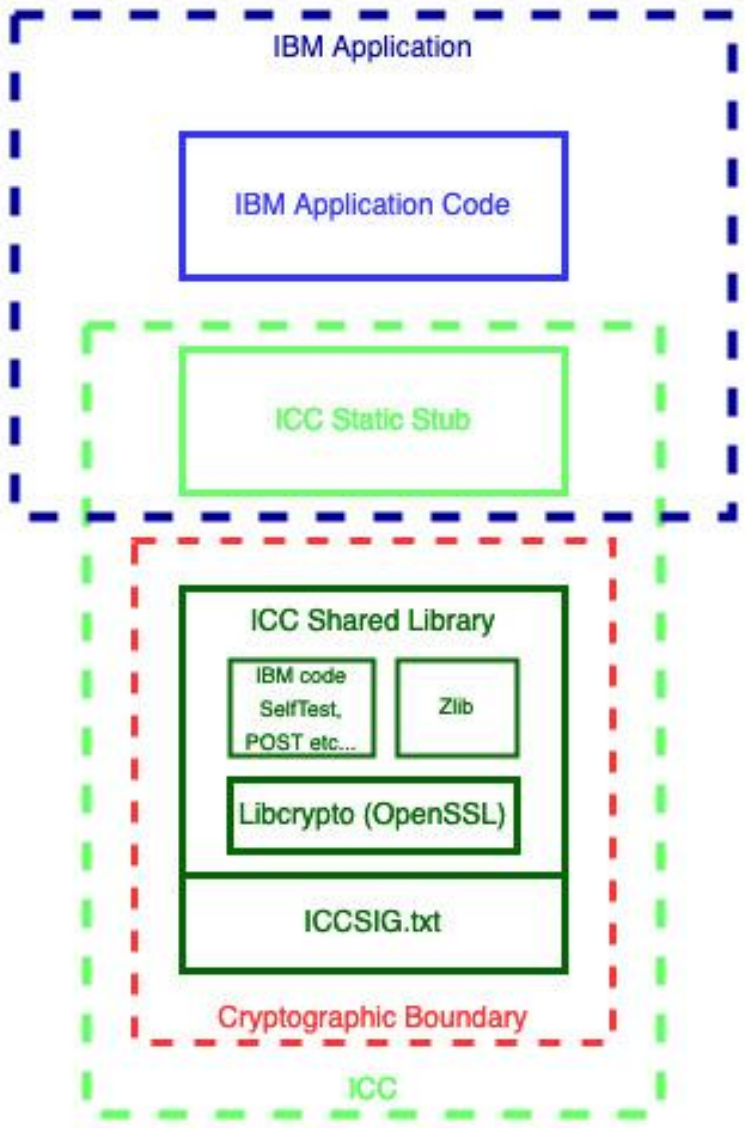

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libicclib84.dll on Windows Server 2019 running on Lenovo ThinkSystem SR630 with Intel® Xeon® Gold 5217	8.8.1.0	N/A	RSA Signature Verification
libicclib084.so on Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian) running on Lenovo ThinkSystem SR630 with Intel® Xeon® Gold 5217	8.8.1.0	N/A	RSA Signature Verification
libicclib084.so on Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian) running on IBM PowerVM 3.1 on IBM Power System S914 (9009-41A) with IBM POWER9	8.8.1.0	N/A	RSA Signature Verification
libicclib084.so on Red Hat Linux Enterprise Server 8.4 64-bit (Big Endian) running on IBM PowerVM 3.1 on IBM Power System S914 (9009-41A) with IBM POWER9	8.8.1.0	N/A	RSA Signature Verification
libicclib084.so on IBM AIX 7.2 64-bit (Big Endian) running on IBM PowerVM 3.1 on IBM Power System S914 (9009-41A) with IBM POWER9	8.8.1.0	N/A	RSA Signature Verification
libicclib084.so on zLinux Red Hat Linux Enterprise Server 8.6 64-bit (Big Endian) running on IBM z/VM	8.8.1.0	N/A	RSA Signature Verification

Package or File Name	Software/ Firmware Version	Features	Integrity Test
7.2 on IBM z/15 (8561 T01) with IBM z15			
libicclib084.so on IBM z/OS 2.3 running on IBM z/VM 7.2 on IBM z/15 (8561 T01) with IBM z15	8.8.1.0	N/A	RSA Signature Verification

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian)	Lenovo ThinkSystem SR630	Intel® Xeon® Gold 5217	Yes		8.8.1.0
Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian)	Lenovo ThinkSystem SR630	Intel® Xeon® Gold 5217	No		8.8.1.0
Microsoft Windows Server 2019 64-bit	Lenovo ThinkSystem SR630	Intel® Xeon® Gold 5217	Yes		8.8.1.0
Microsoft Windows Server 2019 64-bit	Lenovo ThinkSystem SR630	Intel® Xeon® Gold 5217	No		8.8.1.0
Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian) on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	Yes		8.8.1.0
Red Hat Linux Enterprise Server 8.4 64-bit (Little Endian) on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	No		8.8.1.0
Red Hat Linux Enterprise Server 7.9 64-bit (Big Endian) on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	Yes		8.8.1.0
Red Hat Linux Enterprise Server 7.9 64-bit (Big	IBM Power System S914 (9009-41A)	IBM POWER9	No		8.8.1.0

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Endian) on IBM PowerVM 3.1					
IBM AIX 7.2 64-bit (Big Endian) on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	Yes		8.8.1.0
IBM AIX 7.2 64-bit (Big Endian) on IBM PowerVM 3.1	IBM Power System S914 (9009-41A)	IBM POWER9	No		8.8.1.0
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	Yes		8.8.1.0
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	No		8.8.1.0
IBM z/OS 2.3 on IBM z/VM 7.2	IBM z/15 (8561 T01)	IBM z15	Yes		8.8.1.0
IBM z/OS 2.3 on IBM z/VM 7.2	IBM z/15 (8561 T01)	IBM z15	No		8.8.1.0

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

2.3 Excluded Components

The module does not have any excluded components.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service
Non-approved mode	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service

Table 4: Modes List and Description

Mode Change Instructions and Status:

After the module passes the pre-operational self-test and cryptographic algorithm self-tests, the module will be in the operational state in the approved mode of operation. The module can be transitioned to the non-approved mode of operation by requesting one of the non-approved services listed in Section 4.4.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A2619, A2620	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104 Payload Length - Payload Length: 256 AAD Length - AAD Length: 0, 256, 65536	SP 800-38C
AES-CFB1	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A2619, A2620	-	SP 800-38B
AES-CTR	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - No Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A2619, A2620	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256 Tag Length - 128 IV Length - IV Length: 96, 1024 Payload Length - Payload Length: 1024, 8, 248 AAD Length - AAD Length: 1024, 8, 248, 0	SP 800-38D
AES-KW	A2619, A2620	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 128	SP 800-38F
AES-KWP	A2619, A2620	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 8-4096 Increment 8	SP 800-38F
AES-OFB	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-XTS Testing Revision 2.0	A2619, A2620	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 128 Tweak Mode - Hex Data Unit Length Matches Payload Length - Yes	SP 800-38E
Counter DRBG	A2619, A2620	Prediction Resistance - No, Yes Supports Reseed - No Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes Additional Input - Additional Input: 0 Entropy Input - Entropy Input: 128, Entropy Input: 192, Entropy Input: 256 Nonce - Nonce: 128, Nonce: 64 Personalization String Length - Personalization String Length: 0 Returned Bits - 1024, 512	SP 800-90A Rev. 1
DSA SigVer (FIPS186-4)	A2619, A2620	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A2619, A2620	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A2619, A2620	Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A2619, A2620	Component - No Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A2619, A2620	Component - No Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
Hash DRBG	A2619, A2620	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Entropy Input - Entropy Input: 256 Nonce - Nonce: 256 Personalization String Length - Personalization String Length: 0 Additional Input - Additional Input: 0, Additional Input: 0, 256 Returned Bits - 1024, 2048, 224, 384	SP 800-90A Rev. 1

Algorithm	CAVP Cert	Properties	Reference
HMAC DRBG	A2619, A2620	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Entropy Input - Entropy Input: 256 Nonce - Nonce: 256 Personalization String Length - Personalization String Length: 0 Additional Input - Additional Input: 0, Additional Input: 0, 256 Returned Bits - 1024, 2048, 224, 384	SP 800-90A Rev. 1
HMAC-SHA2-224	A2619, A2620	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A2619, A2620	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A2619, A2620	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A2619, A2620	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A2619, A2620	MAC - MAC: 224 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A2619, A2620	MAC - MAC: 256 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A2619, A2620	MAC - MAC: 384 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A2619, A2620	MAC - MAC: 512 Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A2619, A2620	Domain Parameter Generation Methods - P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A2619, A2620	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF Sp800-56Cr1	A2619, A2620	Fixed Info Pattern - uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	SP 800-56C Rev. 2

Algorithm	CAVP Cert	Properties	Reference
PBKDF	A2619, A2620	Iteration Count - Iteration Count: 10-1000 Increment 1 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Password Length - Password Length: 8-128 Increment 1 Salt Length - Salt Length: 128-4096 Increment 8 Key Data Length - Key Data Length: 128-4096 Increment 8	SP 800-132
RSA KeyGen (FIPS186-4)	A2619, A2620	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.3 Info Generated By Server - No Public Exponent Mode - Random Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A2619, A2620	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-256	FIPS 186-4
RSA SigVer (FIPS186-4)	A2619, A2620	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-256 Public Exponent Mode - Random	FIPS 186-4
Safe Primes Key Generation	A2619, A2620	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
Safe Primes Key Verification	A2619, A2620	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA2-224	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-224	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 202

Algorithm	CAVP Cert	Properties	Reference
SHA3-512	A2619, A2620	Message Length - Message Length: 0-65536 Increment 8	FIPS 202

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG	Key Type:Asymmetric	N/A	SP800-133rev2 section 4 example 1

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
DSA with any key sizes	Key Pair Generation, Domain Parameter Generation, 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, Signature Generation, Signature Verification
KBKDF	KBKDF Key Derivation. This algorithm has not been tested by the CAVP.
PBKDF with HMAC-SHA-1	PBKDF Key Derivation. The vendor considers SHA-1 as non-approved per SP 800-131A Rev.3.
RSA with keys smaller than 2048 bits	Key Pair Generation, Signature Generation, Signature Verification
RSA encryption and decryption with PKCS#1v1.5 and any key sizes	RSA Encapsulation, RSA Unencapsulation. PKCS#1v1.5 padding is non-approved.
KAS-FFC-SSC using non-safe prime parameters	Shared Secret Computation
KAS-ECC-SSC 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

Name	Use and Function
RC4	Symmetric Encryption; Symmetric Decryption
MD2	Message Digest
MD4	Message Digest
MD5	Message Digest
SHA-1	Message Digest. The vendor considers SHA-1 as non-approved per SP 800-131A Rev.3.
HMAC-MD5	Message Authentication Code (MAC)
HMAC-SHA1	Message Authentication Code (MAC). The vendor considers SHA-1 as non-approved per SP 800-131A Rev.3.
Hash-DRBG-SHA1	Random Number Generation. The vendor considers SHA-1 as non-approved per SP 800-131A Rev.3.
HMAC-DRBG-SHA1	Random Number Generation. The vendor considers SHA-1 as non-approved per SP 800-131A Rev.3.
MDC2	Message Digest
RIPEMD	Message Digest
chacha20	Symmetric Encryption; Symmetric Decryption
chacha20-poly1305	Authenticated Encryption; Authenticated Decryption

Table 7: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Symmetric Encryption	BC-UnAuth	Symmetric Encryption with AES		AES-CBC: (A2619, A2620) AES-CFB1: (A2619, A2620) AES-CFB128: (A2619, A2620) AES-CFB8: (A2619, A2620) AES-CTR: (A2619, A2620) AES-ECB: (A2619, A2620) AES-OFB: (A2619, A2620) AES-XTS Testing Revision 2.0: (A2619, A2620)
Symmetric Decryption	BC-UnAuth	Symmetric Decryption with AES		AES-CBC: (A2619, A2620) AES-CFB1: (A2619, A2620) AES-CFB128: (A2619, A2620) AES-CFB8:

Name	Type	Description	Properties	Algorithms
				(A2619, A2620) AES-CTR: (A2619, A2620) AES-ECB: (A2619, A2620) AES-OFB: (A2619, A2620) AES-XTS Testing Revision 2.0: (A2619, A2620)
Authenticated Encryption	BC-Auth	Authenticated Encryption with AES		AES-CCM: (A2619, A2620) AES-GCM: (A2619, A2620)
Authenticated Decryption	BC-Auth	Authenticated Decryption with AES		AES-CCM: (A2619, A2620) AES-GCM: (A2619, A2620)
Key Wrapping	BC-Auth	Key Wrapping (as a standalone service)		AES-KW: (A2619, A2620) AES-KWP: (A2619, A2620)
Key Unwrapping	BC-Auth	Key Unwrapping (as a standalone service)		AES-KW: (A2619, A2620) AES-KWP: (A2619, A2620)
Message Digest	SHA	Message Digest		SHA2-224: (A2619, A2620) SHA2-256: (A2619, A2620) SHA2-384: (A2619, A2620) SHA2-512: (A2619, A2620) SHA3-224: (A2619, A2620) SHA3-256: (A2619, A2620) SHA3-384: (A2619, A2620) SHA3-512: (A2619, A2620)
Message Authentication Code (MAC)	MAC	Message Authentication Code		AES-CMAC: (A2619, A2620) HMAC-SHA2-224: (A2619, A2620) HMAC-SHA2-

Name	Type	Description	Properties	Algorithms
				256: (A2619, A2620) HMAC-SHA2-384: (A2619, A2620) HMAC-SHA2-512: (A2619, A2620) HMAC-SHA3-224: (A2619, A2620) HMAC-SHA3-256: (A2619, A2620) HMAC-SHA3-384: (A2619, A2620) HMAC-SHA3-512: (A2619, A2620)
Random Number Generation	DRBG	Random number generation		Counter DRBG: (A2619, A2620) Hash DRBG: (A2619, A2620) HMAC DRBG: (A2619, A2620)
Key Pair Generation	AsymKeyPair-KeyGen CKG	Key pair generation using RSA, EC or Safe Primes		ECDSA KeyGen (FIPS186-4): (A2619, A2620) RSA KeyGen (FIPS186-4): (A2619, A2620) Safe Primes Key Generation: (A2619, A2620) CKG: () Key Type: Asymmetric
Key Pair Validation	AsymKeyPair-KeyVer	Key pair validation for EC and Safe Primes		ECDSA KeyVer (FIPS186-4): (A2619, A2620) Safe Primes Key Verification: (A2619, A2620)
Signature Generation	DigSig-SigGen	Digital signature generation for ECDSA and RSA		ECDSA SigGen (FIPS186-4): (A2619, A2620) RSA SigGen

Name	Type	Description	Properties	Algorithms
				(FIPS186-4): (A2619, A2620)
Signature Verification	DigSig-SigVer	Digital signature verification for DSA, ECDSA and RSA		ECDSA SigVer (FIPS186-4): (A2619, A2620) RSA SigVer (FIPS186-4): (A2619, A2620)
Signature Verification (legacy)	DigSig-SigVer	Digital signature verification for DSA, ECDSA and RSA	Publications:FIPS 140-3 IG C.M legacy algorithms DSA SigVer (FIPS186-4):L capabilities: 2048, 3072; N capabilities: 224, 256 Hashes:SHA-1 ECDSA SigVer (FIPS186-4):Key Size(Curve): P-224, P-256, P-384, P-521; Key Strength: from 112 to 256 bits RSA SigVer (FIPS186-4):Key Size: 1024, 2048, 3072, 4096 bits; Key Strength: from 80 to 150 bits	DSA SigVer (FIPS186-4): (A2619, A2620) ECDSA SigVer (FIPS186-4): (A2619, A2620) RSA SigVer (FIPS186-4): (A2619, A2620)
Shared Secret Computation	KAS-SSC	Diffie-Hellman and EC Diffie-Hellman share secret computation	Caveat:Key establishment methodology provides between 128 and 192 bits of security strength IG:D.F Scenario 2 (path 1)	KAS-ECC-SSC Sp800-56Ar3: (A2619, A2620) KAS-FFC-SSC Sp800-56Ar3: (A2619, A2620)
HKDF Key Derivation	KAS-56CKDF	KDA OneStep Key Derivation		KDA HKDF Sp800-56Cr1: (A2619, A2620)
PBKDF Key Derivation	PBKDF	Password-based Key Derivation		PBKDF: (A2619, A2620)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-GCM

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 SP 800-52 Rev. 2. 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.

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 cipher suites that explicitly select AES-GCM as the encryption/decryption cipher. The module supports acceptable AES-GCM cipher suites from Section 3.3.1 of SP 800-52 Rev. 2.

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.

2.7.2 AES-XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in SP 800-38E. The length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks (16MB of data). 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. As the module does not generate symmetric keys, the check is performed when keys are input via the service APIs.

The two AES keys shall be generated and/or established independently according to the rules for component symmetric keys from SP 800-133 Rev. 2, Section 6.3.

2.7.3 PBKDF2

The module provides password-based key derivation (PBKDF2), compliant with SP 800-132. The module supports option 1a from Section 5.4 of SP 800-132, in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK). In accordance to SP 800-132 and FIPS 140-3 IG D.N, the following requirements shall be met:

- Derived keys shall only be used in storage applications. The MK shall not be used for other purposes. The module accepts a minimum length of 112 bits for the MK or DPK.
- Password and passphrases, used as an input for the PBKDF2, shall not be used as cryptographic keys.
- 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 $1 / 36^{10} = 3.656\ 10^{-15}$. Combined with the minimum iteration count as described below, this provides an acceptable trade-off between user experience and security against brute-force attacks.

- A portion of the salt, with a length of at least 128 bits (this is verified by the module to determine the service is approved), shall be generated randomly using the SP 800-90A Rev. 1 DRBG.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The module enforces a minimum iteration count of 1000.

2.7.4 Diffie-Hellman and EC Diffie-Hellman

The module offers Diffie-Hellman and EC Diffie-Hellman shared secret computation services compliant to SP 800-56A Rev. 3 and meeting IG D.F scenario 2 path (1). 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].

2.7.5 Key Wrapping

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

2.7.6 Key Agreement

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

2.7.7 Legacy Algorithms

Algorithms designated as “Legacy” can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M. SHA-1 used in the context of ECDSA SigVer and RSA SigVer, is allowed for Legacy use only.

2.8 RBG and Entropy

At the moment of the Interim Certificate, there was no ESV certificate issued for this module.

ICC uses an 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.

This implementation maps to scenario 1.b of IG 9.3.A as it relies on an SP800-90B-validated entropy source that resides within its TOEPP.

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
NIST SP800-90B compliant ENT (NP)	Non-Physical	See Tested Operational Environment Table	256 bits	128 bits	HMAC SHA2-256

Table 9: Entropy Sources

2.9 Key Generation

The module provides the following key generation methods:

- Safe primes key pair generation: compliant with SP 800-56A Rev. 3. The method described in Section 5.6.1.1.4 of SP 800-56A Rev. 3 (“Testing Candidates”) is used.
- RSA key pair generation: compliant with FIPS 186-4. The method described in Appendix A.1.6 of FIPS 186-4 (“Probable Primes with Conditions Based on Auxiliary Probable Primes”) is used.
- ECC (ECDH and ECDSA) key pair generation: compliant with FIPS 186-4. The method described in Appendix A.2.2 of FIPS 186-4 (“Rejection Sampling”) is used.

To obtain the random values used in asymmetric key pair generation, the module implements Cryptographic Key Generation (CKG, vendor affirmed), compliant with SP 800-133 Rev. 2 Section 4 without the use of V (in accordance with additional comment 2 of IG D.H).

Additionally, the module implements the following key derivation methods:

- PBKDF2 compliant with option 1a of SP 800-132. This implementation shall only be used to derive keys for use in storage applications.
- HKDF Key Derivation compliant with SP 800-135 Rev. 1 and SP 800-56C Rev. 2, using two-step key derivation, extraction and expansion procedure.

2.10 Key Establishment

The module provides Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) shared secret computation compliant with SP 800-56A Rev. 3, in accordance with Scenario 2 (1) of FIPS 140-3 IG D.F.

For Diffie-Hellman, the module supports the following safe prime groups:

For use in the IKE protocol (RFC 3526):

- MODP-2048
- MODP-3072
- MODP-4096
- MODP-6144
- MODP-8192

For use in the TLS protocol (RFC 7919):

- ffdhe2048
- ffdhe3072
- ffdhe4096
- ffdhe6144
- ffdhe8192

For Elliptic Curve Diffie-Hellman, the module supports the NIST-defined P-224, P-256, P-384, and P-521 curves.

According to SP 800-56A Rev. 3 and FIPS 140-3 IG D.B, the key sizes of DH and ECDH shared secret computation provide 112-200 and 112-256 bits of security strength respectively in the approved mode of operation.

The module also offers authenticated encryption and decryption as a service using AES-KW and AES-KWP. These algorithms can be used to wrap SSPs with a security strength of 128, 192, or 256 bits, depending on the wrapping key size.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	The input data parameters of those API functions that accept, as their arguments, data to be used or processed by the module.
N/A	Data Output	Data output to the caller after generated or otherwise processed by the API functions.
N/A	Control Input	The API functions used to control the operation of the module.
N/A	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 10: Ports and Interfaces

The module does not implement a control output interface.

All data output via data output interface is inhibited when the module is performing the pre-operational self-test or zeroization or when the module enters error state.

4 Roles, Services, and Authentication

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:

4.1 Authentication Methods

The module does not implement authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	Crypto Officer	None

Table 11: Roles

The Crypto Officer role is implicitly and always assumed by the operator of the module. The module does not support concurrent operators.

4.3 Approved Services

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 described indicator is a shared indicator for both approved and non-approved security services following Example 3 of FIPS 140-3 IG 2.4.C.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric Encryption	Perform AES encryption	status = 1	Plaintext, AES key, IV	Ciphertext	Symmetric Encryption	Crypto Officer - AES key: W,E
Symmetric Decryption	Perform AES decryption	status = 1	Ciphertext, AES key, IV	Plaintext	Symmetric Decryption	Crypto Officer - AES key: W,E
Authenticated Encryption	Perform authenticated AES encryption	status = 1	Plaintext, AES key, IV	Ciphertext, MAC tag	Authenticated Encryption	Crypto Officer - AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Authenticated Decryption	Perform authenticated AES decryption	status = 1	Ciphertext, AES key, MAC tag	Plaintext or fail	Authenticated Decryption	Crypto Officer - AES key: W,E
Key Wrapping	Perform AES-based key wrapping	status = 1	Key to be wrapped, AES key wrapping key	Wrapped key	Key Wrapping	Crypto Officer - AES key: W,E
Key Unwrapping	Perform AES-based key unwrapping	status = 1	Key to be unwrapped, AES key wrapping key	Unwrapped key or failure	Key Unwrapping	Crypto Officer - AES key: W,E
Message Digest	Compute SHA hashes	status = 1	Message	Message Digest	Message Digest	Crypto Officer
Message Authentication Code (HMAC)	Compute HMAC	status = 1	Message, HMAC key	MAC	Message Authentication Code (MAC)	Crypto Officer - HMAC key: W,E
Message Authentication Code (CMAC)	Compute an AES-based CMAC	status = 1	Message, AES key	MAC	Message Authentication Code (MAC)	Crypto Officer - AES key: W,E
Random Number Generation	Generate random bitstrings	status = 1	Output length	Random bytes	Random Number Generation	Crypto Officer - Entropy input: W,E,Z - CTR_DRBG internal state (V, Key): G,E - Hash_DRBG internal state (V, C): G,E - HMAC_DRBG internal state (V, C): G,E - CTR_DRBG seed: G,E,Z -

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Hash_DRBG seed: G,E,Z - HMAC_DRBG seed: G,E,Z
Diffie-Hellman Key Generation using safe primes	Generate Diffie-Hellman key pair using safe primes	status = 1	Safe prime	Diffie-Hellman Key Pair	Key Pair Generation	Crypto Officer - Module-generated Diffie-Hellman private key: G,R - Module-generated Diffie-Hellman public key: G,R - Intermediate key generation value: G,E,Z
EC Key Pair Generation	Generate Elliptic Curve key pairs	status = 1	EC Domain Parameters	EC Key Pair	Key Pair Generation	Crypto Officer - Module-generated EC public key: G,R - Module-generated EC private key: G,R - Intermediate key generation value: G,E,Z
RSA Key Pair Generation	Generate RSA key pairs	status = 1	Modulus size	RSA Key Pair	Key Pair Generation	Crypto Officer - Module-generated RSA public key: G,R - Module-

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						generated RSA private key: G,R - Intermediate key generation value: G,E,Z
EC Key Pair Validation	Verify Elliptic Curve key pairs	status = 1	Private key, Public key	Validation result (pass/fail)	Key Pair Validation	Crypto Officer - EC private key: W,E - EC public key: W,E - Diffie-Hellman private key: W,E - Diffie-Hellman public key: W,E
Diffie-Hellman Key Pair Validation	Verify Diffie-Hellman key pairs	status = 1	Private key, Public key	Validation result (pass/fail)	Key Pair Validation	Crypto Officer - Diffie-Hellman private key: W,E - Diffie-Hellman public key: W,E
ECDSA Signature Generation	Sign using ECDSA	status = 1	Message, EC private key	Signature	Signature Generation	Crypto Officer - EC private key: W,E
RSA Signature Generation	Sign using RSA	status = 1	Message, RSA private key	Signature	Signature Generation	Crypto Officer - RSA private key: W,E
DSA Signature Verification	Verify DSA signatures	status = 1	Message, DSA public key, Signature	Pass or Fail	Signature Verification (legacy)	Crypto Officer - DSA public key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
ECDSA Signature Verification	Verify ECDSA signatures	status = 1	Message, EC public key, Signature	Pass or Fail	Signature Verification Signature Verification (legacy)	Crypto Officer - EC public key: W,E
RSA Signature Verification	Verify RSA signatures	status = 1	Message, RSA public key, Signature	Pass or Fail	Signature Verification Signature Verification (legacy)	Crypto Officer - RSA public key: W,E
Diffie-Hellman Shared Secret Computation	Perform Diffie-Hellman shared secret computation	status = 1	Private key, received public key	Shared secret	Shared Secret Computation	Crypto Officer - Diffie-Hellman private key: W,E - Diffie-Hellman public key: W,E - Diffie-Hellman shared secret: G,R
EC Diffie-Hellman Shared Secret Computation	Perform EC Diffie-Hellman shared secret computation	status = 1	Private key, received public key	Shared secret	Shared Secret Computation	Crypto Officer - EC private key: W,E - EC public key: W,E - EC Diffie-Hellman shared secret: G,R
HKDF Key Derivation	Key derivation for TLSv1.3 pseudorandom function (PRF)	status = 1	Diffie-Hellman or EC Diffie-Hellman shared secret	HKDF derived key	HKDF Key Derivation	Crypto Officer - Diffie-Hellman shared secret: W,E - EC Diffie-Hellman shared secret: W,E - HKDF derived key: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
PBKDF Key Derivation	Password-based key derivation	status = 1	Password	derived key	PBKDF Key Derivation	Crypto Officer - Password: W,E - PBKDF derived key: G,R
Show Status	Return module status	status = 1	N/A	status output	None	Crypto Officer
Show module and version info	Return module name and versioning information	status = 1	N/A	module and version information	None	Crypto Officer
Self-tests	Perform pre-operational, and conditional self-tests	status = 1	N/A	pass or fail results	Symmetric Encryption Symmetric Decryption Authenticated Encryption Authenticated Decryption Key Wrapping Key Unwrapping Message Digest Message Authentication Code (MAC) Random Number Generation Key Pair Generation Key Pair Validation Signature Generation Signature Verification Shared Secret Computatio	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					n HKDF Key Derivation PBKDF Key Derivation	
Zeroization	Zeroize SSPs	status = 1	length of context to zeroize and address of context to be zeroized	N/A	None	Crypto Officer - AES key: Z - HMAC key: Z - Module-generated EC private key: Z - Module-generated EC public key: Z - EC private key: Z - EC public key: Z - Module-generated RSA private key: Z - Module-generated RSA public key: Z - RSA private key: Z - RSA public key: Z - DSA public key: Z - Entropy input: Z - Hash_DRBG seed: Z - HMAC_DRBG seed: Z - CTR_DRBG seed: Z -

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Hash_DRBG internal state (V, C): Z - HMAC_DRBG internal state (V, C): Z - CTR_DRBG internal state (V, Key): Z - Module-generated Diffie-Hellman private key: Z - Module-generated Diffie-Hellman public key: Z - Diffie-Hellman private key: Z - Diffie-Hellman public key: Z - Diffie-Hellman shared secret: Z - EC Diffie-Hellman shared secret: Z - Password: Z - PBKDF derived key: Z - HKDF derived key: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Module installation and configuration	Configure module for approved mode of operation	status = 1	N/A	N/A	None	Crypto Officer

Table 12: Approved Services

4.4 Non-Approved Services

The following table describes the non-approved services. For these services, the service indicator returns a value of 0. See Section 4.3 for a description of how to invoke the service indicator in the calling application.

Name	Description	Algorithms	Role
Symmetric Encryption	Symmetric Encryption with non-approved algorithms	DES Triple-DES CAST Camellia Blowfish RC2 RC4 chacha20	CO
Symmetric Decryption	Symmetric Decryption with non-approved algorithms	DES Triple-DES CAST Camellia Blowfish RC2 RC4 chacha20	CO
Authenticated Encryption	Authenticated Encryption with non-approved algorithms	chacha20-poly1305	CO
Authenticated Decryption	Authenticated Decryption with non-approved algorithms	chacha20-poly1305	CO
Message Digest	Message Digest with non-approved algorithms	MD2 MD4 MD5 MDC2 RIPEMD SHA-1	CO
Message Authentication Code (MAC)	Message Authentication Code generation with non-approved algorithms	HMAC-MD5 HMAC-SHA1	CO
Random Number Generation	Random Number Generation with non-approved algorithms	Hash-DRBG-SHA1 HMAC-DRBG-SHA1	CO
DSA Domain Parameter Generation	Domain Parameter Generation with non-approved algorithms	DSA with any key sizes	CO

Name	Description	Algorithms	Role
DSA Key Pair Generation	DSA Key Pair Generation	DSA with any key sizes	CO
DSA Signature Generation	DSA Signature Generation	DSA with any key sizes	CO
DSA Signature Verification	DSA Signature Verification with non-approved key sizes	DSA with keys generated with parameters L=512, N=160; L=1024, N=160	CO
EC Key Pair Generation	Key Pair Generation with non-approved elliptic curves	ECDSA with P-192, K-163, B-163 elliptic curves	CO
EC Key Pair Validation	Key Pair Validation with non-approved elliptic curves	ECDSA with P-192, K-163, B-163 elliptic curves	CO
ECDSA Signature Generation	ECDSA Signature Generation with non-approved elliptic curves	ECDSA with P-192, K-163, B-163 elliptic curves	CO
ECDSA Signature Verification	ECDSA Signature Verification with non-approved elliptic curves	ECDSA with P-192, K-163, B-163 elliptic curves	CO
RSA Key Pair Generation	RSA Key Pair Generation with non-approved key sizes	RSA with keys smaller than 2048 bits	CO
RSA Signature Generation	RSA Signature Generation with non-approved key sizes	RSA with keys smaller than 2048 bits	CO
RSA Signature Verification	RSA Signature Verification with non-approved key sizes	RSA with keys smaller than 2048 bits	CO
Key Encapsulation	RSA encryption with PKCS#1v1.5 and any key sizes	RSA encryption and decryption with PKCS#1v1.5 and any key sizes	CO
Key Unencapsulation	RSA decryption with PKCS#1v1.5 and any key sizes	RSA encryption and decryption with PKCS#1v1.5 and any key sizes	CO
Diffie-Hellman Key Pair Generation	Diffie-Hellman Key Pair Generation with domain parameters other than safe primes	KAS-FFC-SSC using non-safe prime parameters	CO
Diffie-Hellman Key Pair Validation	Diffie-Hellman Key Pair Validation with domain parameters other than safe primes	KAS-FFC-SSC using non-safe prime parameters	CO
Diffie-Hellman Shared Secret Computation	Diffie-Hellman Shared Secret Computation with keys generated with domain parameters other than safe primes	KAS-FFC-SSC using non-safe prime parameters	CO
EC Shared Secret Computation	EC Diffie-Hellman with non-approved elliptic curves	KAS-ECC-SSC with P-192, K-163, B-163 elliptic curves	CO
KBKDF Key Derivation	KBKDF Key Derivation	KBKDF	CO
PBKDF Key Derivation	PBKDF Key Derivation	PBKDF with HMAC-SHA-1	CO

Table 13: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not support the loading of external software/firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by performing an RSA PKCS#1v1.5 signature verification using SHA2-256 and a 2048-bit modulus public key. The RSA public key is stored inside the shared library.

5.2 Initiate on Demand

Integrity tests are performed as part of the pre-operational Self-Tests, which are executed when the module is initialized. The integrity tests can also be requested on demand through the API function ICC_IntegrityCheck.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

Any SSPs contained within the module are protected by the process isolation and memory separation mechanisms, and only the module has control over these SSPs.

If properly installed, the operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data and uncontrolled access to the data of other processes is prevented.

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1.

Instrumentation tools like the ptrace system call, gdb and strace utilities as well as other tracing mechanisms available in Linux, or any other similar tool or utility available in the rest of the operating systems, shall not be used in the operational environment.

7 Physical Security

The module is comprised of software only, and this section is therefore not applicable.

8 Non-Invasive Security

This module does not implement any non-invasive security mechanism and therefore this section is not applicable.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Temporary storage for SSPs used by the module as part of service execution. The module does not perform persistent storage of SSPs	Dynamic

Table 14: Storage Areas

SSPs are provided to the module by the calling application and are destroyed when released by the appropriate API function calls. The module does not perform persistent storage of SSPs.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters	Operator calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	
API output parameters	Cryptographic module	Operator calling application (TOEPP)	Plaintext	Manual	Electronic	

Table 15: SSP Input-Output Methods

The module does not support the input or output of cryptographically protected SSPs.

The module only supports SSP entry and output to and from a calling application running on the same operational environment. This corresponds to manual distribution, electronic entry/output ("CM Software to/from App via TOEPP Path") per FIPS 140-3 IG 9.5.A Table 1.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Free cipher handle	Zeroizes the SSPs contained within the cipher handle.	Memory occupied by SSPs is overwritten with zeroes and then it is released, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that the	By calling the cipher related zeroization API functions: ICC_BN_clear_free for big numbers, ICC_BN_CTX_free for low-level big number arithmetic functions, ICC_EVP_CIPHER_CTX_cleanup for AES keys, ICC_RSA_free for RSA keys, ICC_DSA_free for DSA keys, ICC_DH_free for Diffie-Hellman keys, ICC_EVP_PKEY_free for asymmetric keys, ICC_HMAC_CTX_free for HMAC keys, ICC_EC_KEY_free for EC keys,

Zeroization Method	Description	Rationale	Operator Initiation
		zeroization procedure succeeded.	ICC_CMAC_CTX_free for CMAC keys, ICC_AES_GCM_CTX_free for AES-GCM keys, ICC_RNG_CTX_free for DRBG SSPs.
Automatic	Automatically zeroized by the module when no longer needed	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	N/A
Module reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed.	By unloading and reloading the module

Table 16: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	AES key	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP			Symmetric Encryption Symmetric Decryption Authenticated Encryption Authenticated Decryption Key Wrapping Key Unwrapping Message Authentication Code (MAC)
HMAC key	HMAC key	112 to 524288 bits - 112 to 256 bits	Symmetric key - CSP			Message Authentication Code (MAC)
Module-generated	Module-generated	2048, 3072, 4096 bits	Public key - PSP	Key Pair Generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
RSA public key	RSA public key	- 112, 128, 149 bits				
Module-generated RSA private key	Module-generated RSA private key	2048, 3072, 4096 bits - 112, 128, 149 bits	Private key - CSP	Key Pair Generation		
RSA public key	RSA public key	2048, 3072, 4096 bits - 112, 128, 149 bits	Public key - PSP			Signature Verification
RSA private key	RSA private key	2048, 3072, 4096 bits - 112, 128, 149 bits	Private key - CSP			Signature Generation
Module-generated EC public key	Module-generated EC public key	P-224, P-256, P-384, P-521 bits - 112, 128, 192, 256 bits	Public key - PSP	Key Pair Generation		
Module-generated EC private key	Module-generated EC private key	P-224, P-256, P-384, P-521 bits - 112, 128, 192, 256 bits	Private key - CSP	Key Pair Generation		
EC public key	Elliptic Curve public key	P-224, P-256, P-384, P-521 bits - 112, 128, 192, 256 bits	Public key - PSP			Key Pair Validation Signature Verification Shared Secret Computation
EC private key	Elliptic Curve private key	P-224, P-256, P-384, P-521 bits - 112, 128,	Private key - CSP			Key Pair Validation Signature Generation Shared

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		192, 256 bits				Secret Computation
DSA public key	DSA public key	L: 2048, 3072; N: 224, 256 - 112, 128 bits	Public key - PSP			Signature Verification
Module-generated Diffie-Hellman private key	Diffie-Hellman private key generated by module	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192 - 112, 128, 152, 172, 200 bits	Private key - CSP	Key Pair Generation		
Module-generated Diffie-Hellman public key	Diffie-Hellman public key generated by module	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192,	Public key - PSP	Key Pair Generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		ffdhe8192 - 112, 128, 152, 172, 200 bits				
Diffie-Hellman private key	Diffie-Hellman private key input to module via API	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192 - 112, 128, 152, 172, 200 bits	Private key - CSP			Key Pair Validation Shared Secret Computation
Diffie-Hellman public key	Diffie-Hellman public key input to module via API	MODP-2048, ffdhe2048, MODP-3072, ffdhe3072, MODP-4096, ffdhe4096, MODP-6144, ffdhe6144, MODP-8192, ffdhe8192 - 112, 128, 152,	Public key - PSP			Key Pair Validation Shared Secret Computation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		172, 200 bits				
Diffie-Hellman shared secret	Shared secret generated by Diffie-Hellman shared secret computation	224-8912 bits - 112-256 bits	Shared Secret - CSP		Shared Secret Computation	Shared Secret Computation HKDF Key Derivation
EC Diffie-Hellman shared secret	Shared secret generated by EC Diffie-Hellman shared secret computation	224-521 bits - 112-256 bits	Shared Secret - CSP		Shared Secret Computation	Shared Secret Computation HKDF Key Derivation
Entropy input	Entropy input string used to seed the DRBG (IG D.L compliant)	128-384 bits - 128-256 bits	Entropy input - CSP			Random Number Generation
Hash_DRBG seed	DRBG seed derived from entropy input (IG D.L compliant)	440, 888 bits - 128, 256 bits	Seed - CSP	Random Number Generation		Random Number Generation
HMAC_DRBG seed	DRBG seed derived from entropy input (IG D.L compliant)	160, 256, 512 bits - 128, 256 bits	Seed - CSP	Random Number Generation		Random Number Generation
CTR_DRBG seed	DRBG seed derived from	256, 320, 384 bits - 128, 192, 256 bits	Seed - CSP	Random Number Generation		Random Number Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	entropy input (IG D.L compliant)					
Hash_DRBG internal state (V, C)	Internal state of DRBG (IG D.L compliant)	880, 1776 bits - 128, 256 bits	Internal State - CSP	Random Number Generation		Random Number Generation
HMAC_DRBG internal state (V, C)	Internal state of DRBG (IG D.L compliant)	320, 512, 1024 bits - 128, 256 bits	Internal State - CSP	Random Number Generation		Random Number Generation
CTR_DRBG internal state (V, Key)	Internal state of DRBG (IG D.L compliant)	256, 320, 384 bits - 128, 192, 256 bits	Internal State - CSP	Random Number Generation		Random Number Generation
PBKDF derived key	PBKDF derived key from Password	112-4096 bits - 112-256 bits	Symmetric key - CSP	PBKDF Key Derivation		
Password	Password or passphrase used by PBKDF to derive symmetric keys	8-128 characters - N/A	Password - CSP			PBKDF Key Derivation
HKDF derived key	HKDF derived key	112 - 256 bits - 112-256 bits	Symmetric key - CSP	HKDF Key Derivation		
Intermediate key generation value	Intermediate key generation value generated during key pair generation (SP 800-133 Rev. 2 Section 4, 5.1, and 5.2)	112-8192 bits - 112-256 bits	Intermediate value - CSP	Key Pair Generation		Key Pair Generation

Table 17: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters	RAM:Plaintext	From service invocation to service completion	Module reset Free cipher handle	
HMAC key	API input parameters	RAM:Plaintext	From service invocation to service completion	Module reset Free cipher handle	
Module-generated RSA public key	API output parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Module-generated RSA private key:Paired With
Module-generated RSA private key	API output parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Module-generated RSA public key:Paired With
RSA public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	RSA private key:Paired With
RSA private key	API input parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	RSA public key:Paired With
Module-generated EC public key	API output parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Module-generated EC private key:Paired With
Module-generated EC private key	API output parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Module-generated EC public key:Paired With
EC public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	EC private key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
EC private key	API input parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	EC public key:Paired With
DSA public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Module reset Free cipher handle	
Module-generated Diffie-Hellman private key	API output parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Module-generated Diffie-Hellman public key:Paired With Intermediate key generation value:Derived From
Module-generated Diffie-Hellman public key	API output parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Module-generated Diffie-Hellman private key:Paired With Intermediate key generation value:Derived From
Diffie-Hellman private key	API input parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Diffie-Hellman shared secret:Establishes
Diffie-Hellman public key	API input parameters	RAM:Plaintext	From service invocation to service completion	Free cipher handle Module reset	Diffie-Hellman shared secret:Establishes
Diffie-Hellman shared secret	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipher handle is freed	Free cipher handle Module reset	Diffie-Hellman private key:Established By Diffie-Hellman public key:Established By HKDF derived key:Derives

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
EC Diffie-Hellman shared secret	API input parameters API output parameters	RAM:Plaintext	From service invocation until cipher handle is freed	Free cipher handle Module reset	EC private key:Established By EC public key:Established By HKDF derived key:Derives
Entropy input		RAM:Plaintext	From generation until DRBG seed is created	Automatic Module reset	Hash_DRBG seed:Derives HMAC_DRBG seed:Derives CTR_DRBG seed:Derives
Hash_DRBG seed		RAM:Plaintext	From creation until the DRBG is instantiated or reseeded	Free cipher handle Automatic Module reset	Entropy input:Derived From Hash_DRBG internal state (V, C):Derives
HMAC_DRBG seed		RAM:Plaintext	Until the DRBG is instantiated or reseeded	Free cipher handle Automatic Module reset	Entropy input:Derived From HMAC_DRBG internal state (V, C):Derives
CTR_DRBG seed		RAM:Plaintext	Until the DRBG is instantiated or reseeded	Free cipher handle Automatic Module reset	Entropy input:Derived From CTR_DRBG internal state (V, Key):Derives
Hash_DRBG internal state (V, C)		RAM:Plaintext	From DRBG instantiation to DRBG termination	Free cipher handle Automatic Module reset	Hash_DRBG seed:Derived From
HMAC_DRBG internal state (V, C)		RAM:Plaintext	From DRBG instantiation to DRBG termination	Free cipher handle Automatic Module reset	HMAC_DRBG seed:Derived From
CTR_DRBG internal state (V, Key)		RAM:Plaintext	From DRBG instantiation to DRBG termination	Free cipher handle Automatic Module reset	CTR_DRBG seed:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
PBKDF derived key	API output parameters	RAM:Plaintext	From service invocation to service completion	Module reset Free cipher handle	Password:Derived From
Password	API input parameters	RAM:Plaintext	From service invocation to service completion	Module reset Free cipher handle	PBKDF derived key:derives
HKDF derived key	API output parameters	RAM:Plaintext	From service invocation to service completion	Module reset Free cipher handle	HMAC key:Derived From
Intermediate key generation value		RAM:Plaintext	From service invocation until cipher handle is freed	Automatic Module reset	Module-generated RSA private key:Generates Module-generated RSA public key:Generates Module-generated EC private key:Generates Module-generated EC public key:Generates Module-generated Diffie-Hellman private key:Generates Module-generated Diffie-Hellman public key:Generates

Table 18: SSP Table 2

10 Self-Tests

The module performs the pre-operational self-test and CASTs automatically when the module is loaded into memory. The pre-operational integrity test is only executed after all cryptographic algorithm self-tests (CASTs) executed successfully.

While the module is executing the pre-operational test and the CASTs, the module services are not available, and input and output are inhibited. The module is not available for use by the calling application until the pre-operational self-test and the CASTs are completed successfully. After the pre-operational test and the CASTs succeed, the module becomes operational. If any of the pre-operational test or any of the CASTs fail an error message is returned, and the module transitions to the error state.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
RSA SigVer (FIPS186-4) (A2619)	2048-bit key, SHA2-256	Signature Verification	SW/FW Integrity	Module successful execution	This RSA public key is stored inside the shared library.

Table 19: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC encrypt	256-bit key	KAT	CAST	Module becomes operational	Encryption	Test runs at power-on before first use of the algorithm
AES-CBC decrypt	256-bit key	KAT	CAST	Module becomes operational	Decryption	Test runs at power-on before first use of the algorithm
AES-GCM encrypt	128-bit key	KAT	CAST	Module becomes operational	Encryption	Test runs at power-on before the integrity test
AES-GCM decrypt	128-bit key	KAT	CAST	Module becomes operational	Decryption	Test runs at power-on before the integrity test
AES-CCM encrypt	128-bit key	KAT	CAST	Module becomes operational	Encryption	Test runs at power-on before the integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CCM decrypt	128-bit key	KAT	CAST	Module becomes operational	Decryption	Test runs at power-on before the integrity test
AES-XTS Testing Revision 2.0 encrypt	128-bit key	KAT	CAST	Module becomes operational	Encryption	Test runs at power-on before first use of the algorithm
AES-XTS Testing Revision 2.0 decrypt	128-bit key	KAT	CAST	Module becomes operational	Decryption	Test runs at power-on before first use of the algorithm
AES-KW Wrap	128-bit key	KAT	CAST	Module becomes operational	Key Wrapping	Test runs at power-on before the integrity test
AES-KW Unwrap	128-bit key	KAT	CAST	Module becomes operational	Key Unwrapping	Test runs at power-on before the integrity test
AES-KWP Wrap	128-bit key	KAT	CAST	Module becomes operational	Key Wrapping	Test runs at power-on before the integrity test
AES-KWP Unwrap	128-bit key	KAT	CAST	Module becomes operational	Key Unwrapping	Test runs at power-on before the integrity test
AES-CMAC	128-, 192-, 256-bit keys	KAT	CAST	Module becomes operational	Message Authentication Code (MAC)	Test runs at power-on before the integrity test
Counter DRBG	AES-128, AES-192, AES-256	KAT	CAST	Module becomes operational	Compliant with SP 800-90Ar1	Test runs at power-on before first use of the algorithm
HMAC DRBG	SHA2-224, SHA2-256, SHA2-384, SHA2-512	KAT	CAST	Module becomes operational	Random Number Generation	Test runs at power-on before first use of the algorithm
Hash DRBG	SHA2-224, SHA2-256, SHA2-384, SHA2-512	KAT	CAST	Module becomes operational	Random Number Generation	Test runs at power-on before first

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
						use of the algorithm
HMAC-SHA2-256	SHA2-256	KAT	CAST	Module becomes operational	Message Authentication Code (MAC)	Test runs at power-on before first use of the algorithm
HMAC-SHA2-512	SHA2-512	KAT	CAST	Module becomes operational	Message Authentication Code (MAC)	Test runs at power-on before first use of the algorithm
SHA3-512	N/A	KAT	CAST	Module becomes operational	Message Digest	Test runs at power-on before first use of the algorithm
PBKDF	SHA2-256	KAT	CAST	Module becomes operational	Password-based key derivation	Test runs at power-on before first use of the algorithm
KDA HKDF SP800-56Cr2	SHA2-256, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, with 256-bit secret	KAT	CAST	Module becomes operational	Shared secret key derivation	Test runs at power-on before first use of the algorithm
Diffie-Hellman Safe Primes Key Generation	N/A	PCT	PCT	Successful key pair generation	SP 800-56A Rev. 3 Section 5.6.2.1.4	Key Pair Generation
DSA SigVer (FIPS186-4)	L=2048, N=224 with SHA2-256	KAT	CAST	Module becomes operational	Signature Verification	Test runs at power-on before first use of the algorithm
ECDSA KeyGen (FIPS186-4)	SHA2-256	PCT	PCT	Successful key pair generation	Signature Generation and Verification	Key Pair Generation
ECDSA SigGen (FIPS186-4) (A6510)	P-384 and B233 curves with SHA2-256	KAT	CAST	Module becomes operational	Signature Generation	Test runs at power-on before first

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
						use of the algorithm
ECDSA SigVer (FIPS186-4)	P-384 and B-233 curves with SHA2-256	KAT	CAST	Module becomes operational	Signature Verification	Test runs at power-on before first use of the algorithm
RSA KeyGen (FIPS186-4)	SHA2-256	PCT	PCT	Module Successful key pair generation becomes operational	Signature Generation and Verification	Key Pair Generation
RSA SigGen (FIPS186-4)	2048-bit modulus with SHA2-256	KAT	CAST	Module becomes operational	Signature Generation	Test runs at power-on before first use of the algorithm
RSA SigVer (FIPS186-4)	2048-bit modulus with SHA2-256	KAT	CAST	Module becomes operational	Signature Verification	Test runs at power-on before first use of the algorithm
KAS-ECC-SSC Sp800-56Ar3 (A5986)	P-521 curve	KAT	CAST	Module becomes operational	EC Diffie-Hellman "Z" computation	Test runs at power-on before the integrity test
KAS-FFC-SSC Sp800-56Ar3 (A5986)	MODP-2048	KAT	CAST	Module becomes operational	Diffie-Hellman "Z" computation	Test runs at power-on before the integrity test
ESV-RCT Startup	Startup test with 1024 8-bit samples	fault-detection test	CAST	successful seeding of SP 800-90A DRBG	SP 800-90B 4.4.1 Repetition Count Test	upon seeding or reseeding SP 800-90A DRBG
ESV-RCT Continuous	Continuous test; Cutoff value = 41	fault-detection test	CAST	successful seeding of SP 800-90A DRBG	SP 800-90B 4.4.1 Repetition Count Test	upon seeding or reseeding SP 800-90A DRBG
ESV-APT Startup	Startup test with 1024 8-bit samples	fault-detection test	CAST	successful seeding of SP 800-90A DRBG	SP 800-90B 4.4.2 Adaptive Proportion Test	upon seeding or reseeding SP 800-90A DRBG

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ESV-APT Continuous	Continuous test; Cutoff value = 459	fault-detection test	CAST	successful seeding of SP 800-90A DRBG	SP 800-90B 4.4.2 Adaptive Proportion Test	upon seeding or reseeding SP 800-90A DRBG

Table 20: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-4) (A2619)	Signature Verification	SW/FW Integrity	Whenever module is powered on	Upon every power-on

Table 21: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC encrypt	KAT	CAST	On Demand	Manually
AES-CBC decrypt	KAT	CAST	On Demand	Manually
AES-GCM encrypt	KAT	CAST	On Demand	Manually
AES-GCM decrypt	KAT	CAST	On Demand	Manually
AES-CCM encrypt	KAT	CAST	On Demand	Manually
AES-CCM decrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 encrypt	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 decrypt	KAT	CAST	On Demand	Manually
AES-KW Wrap	KAT	CAST	On Demand	Manually
AES-KW Unwrap	KAT	CAST	On Demand	Manually
AES-KWP Wrap	KAT	CAST	On Demand	Manually
AES-KWP Unwrap	KAT	CAST	On Demand	Manually
AES-CMAC	KAT	CAST	On demand	Manually
Counter DRBG	KAT	CAST	On Demand	Manually
HMAC DRBG	KAT	CAST	On Demand	Manually
Hash DRBG	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256	KAT	CAST	On Demand	Manually
HMAC-SHA2-512	KAT	CAST	On Demand	Manually
SHA3-512	KAT	CAST	On Demand	Manually
PBKDF	KAT	CAST	On Demand	Manually
KDA HKDF SP800-56Cr2	KAT	CAST	On Demand	Manually
Diffie-Hellman Safe Primes Key Generation	PCT	PCT	On Demand	Manually
DSA SigVer (FIPS186-4)	KAT	CAST	On Demand	Manually
ECDSA KeyGen (FIPS186-4)	PCT	PCT	On Demand	Manually
ECDSA SigGen (FIPS186-4) (A6510)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-4)	KAT	CAST	On Demand	Manually
RSA KeyGen (FIPS186-4)	PCT	PCT	On Demand	Manually
RSA SigGen (FIPS186-4)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-4)	KAT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A5986)	KAT	CAST	On Demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A5986)	KAT	CAST	On Demand	Manually
ESV-RCT Startup	fault-detection test	CAST	On demand	Manually
ESV-RCT Continuous	fault-detection test	CAST	On demand	Manually
ESV-APT Startup	fault-detection test	CAST	On demand	Manually
ESV-APT Continuous	fault-detection test	CAST	On demand	Manually

Table 22: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Halt Error	Module is no longer operational. The data output is inhibited.	RSA SigVer CAST failure or RSA SigVer (PKCS #1v1.5) integrity test failure Failure of any of the CAST Failure of any of the PCTs	The module must be reinitialize	Integrity test failure: the module will not load. KAT failure: the module will not load. PCT failure: the module is aborted confirming it entered the error state. APT/RCT fault detection failure: the module is aborted confirming it entered the error state.

Table 23: Error States

If the module fails any of the self-tests, the module enters the error state. In the error state, 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.

10.5 Operator Initiation of Self-Tests

The operator can initiate the pre-operational self-tests and the cryptographic algorithm self-tests by unloading and subsequently re-initializing the module.

The operator can also initiate the pre-operational self-tests and the cryptographic algorithm self-tests by calling the API functions ICC_SelfTest() and ICC_IntegrityCheck().

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

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

1. 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'.
2. 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 Administrator Guidance

The Crypto Officer shall follow Section 11.1 of this Security Policy to verify that the module is installed correctly. The Crypto Officer shall follow this Security Policy to operate the module as a FIPS 140-3 validated module.

11.3 Non-Administrator Guidance

N/A

11.4 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory.

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
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
HMAC	Keyed-Hash Message Authentication Code
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PKCS	Public-Key Cryptography Standards
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SSP	Sensitive Security Parameter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix B. References

FIPS 140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS 140-3 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
FIPS 180-4	**Secure Hash Standard (SHS)** August 2015 https://doi.org/10.6028/NIST.FIPS.180-4
FIPS 186-4	**Digital Signature Standard (DSS)** July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS 198-1	**The Keyed-Hash Message Authentication Code (HMAC)** July 2008 https://doi.org/10.6028/NIST.FIPS.198-1
FIPS 202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://doi.org/10.6028/NIST.FIPS.202
RFC 3526	**More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)** May 2003 https://www.ietf.org/rfc/rfc3526.txt
RFC 5288	**AES Galois Counter Mode (GCM) Cipher Suites for TLS** August 2008 https://www.ietf.org/rfc/rfc5288.txt
RFC 7919	**Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)** August 2016 https://www.ietf.org/rfc/rfc7919.txt
SP 800-38A	**Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://doi.org/10.6028/NIST.SP.800-38A
SP 800-38B	**Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://doi.org/10.6028/NIST.SP.800-38B
SP 800-38C	**Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality**

	May 2004 https://doi.org/10.6028/NIST.SP.800-38C
SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://doi.org/10.6028/NIST.SP.800-38D
SP 800-38E	**Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality of Storage Devices** January 2010 https://doi.org/10.6028/NIST.SP.800-38E
SP 800-38F	**Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://doi.org/10.6028/NIST.SP.800-38F
SP 800-52 Rev. 2	**Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://doi.org/10.6028/NIST.SP.800-52r2
SP 800-56A Rev. 3	**Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP 800-56C Rev. 2	**Recommendation for Key-Derivation Methods in Key-Establishment Schemes** August 2020 https://doi.org/10.6028/NIST.SP.800-56Cr2
SP 800-90A Rev. 1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://doi.org/10.6028/NIST.SP.800-90Ar1
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B
SP 800-132	**Recommendation for Password-Based Key Derivation Part 1: Storage Applications** December 2010 https://doi.org/10.6028/NIST.SP.800-132
SP 800-133 Rev. 2	**Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP 800-135 Rev. 1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://doi.org/10.6028/NIST.SP.800-135r1