FIPS 140-2 Non-Proprietary Security Policy

CryptoComply for .NET

Software Version 1.0.2

Document Version 1.0

January 10, 2024

SafeLogic Inc.
530 Lytton Ave, Suite 200
Palo Alto, CA 94301
www.safelogic.com

Overview

This document provides a non-proprietary FIPS 140-2 Security Policy for CryptoComply for .NET.

SafeLogic's CryptoComply for .NET is designed to provide FIPS 140-2 validated cryptographic functionality and is available for licensing. For more information, visit www.safelogic.com/cryptocomply.

Table of Contents

Overview **2**

1 Introduction **5**

1.1 About FIPS 140 ..... *5*

1.2 About this Document *5*

1.3 External Resources *5*

1.4 Notices *5*

2 CryptoComply for .NET **6**

2.1 Cryptographic Module Specification *6*

2.1.1 Validation Level Detail 6

2.1.2 Modes of Operation 7

2.1.3 Approved Cryptographic Algorithms 8

2.1.4 Non-Approved But Allowed Cryptographic Algorithms 15

2.1.5 Non-Approved Mode of Operation 15

2.2 Critical Security Parameters and Public Keys *16*

2.2.1 Critical Security Parameters 16

2.2.2 Public Keys 17

2.3 Module Interfaces *19*

2.4 Roles, Services, and Authentication *20*

2.4.1 Assumption of Roles 20

2.4.2 Services 21

2.5 Physical Security *26*

2.6 Operational Environment *26*

2.6.1 Use of External RNG 27

2.7 Self-Tests *27*

2.7.1 Power-Up Self-Tests 27

2.7.2 Conditional Self-Tests 29

2.8 Mitigation of Other Attacks *29*

3 Security Rules and Guidance **31**

3.1 Basic Enforcement *31*

3.2 Basic Guidance *31*

3.3 Enforcement and Guidance for AES GCM IVs *31*

3.4 Enforcement and Guidance for Use of the Approved PBKDF *32*

3.5 Rules for Setting the N and the S String in cSHAKE *32*

3.6 Software Installation *32*

4 References and Acronyms **33**

4.1 References *33*

4.2 Acronyms *35*

List of Tables

Table 1 - Validation Level by FIPS 140-2 Section 6
Table 2 - FIPS Approved Algorithm Certificates 8
Table 3 - Approved Cryptographic Functions Implemented with Vendor Affirmation 15
Table 4 - Non-Approved But Allowed Cryptographic Algorithms 15
Table 5 - Non-Approved Cryptographic Functions for Use in non-Approved mode Only 15
Table 6 - Critical Security Parameters 16
Table 7 - Public Keys 18
Table 8 - Logical Interface / Physical Interface Mapping 20
Table 9 - Description of Roles 21
Table 10 - Module Services, Descriptions, and Roles 21
Table 11 - CSP Access Rights within Services 23
Table 12 - Tested Environments 26
Table 13 - Power-Up Self-Tests 27
Table 14 - Conditional Self-Tests 29
Table 15 – References 33
Table 16 - Acronyms and Terms 35

List of Figures

Figure 1 – Module Boundary and Interfaces Diagram 19

1 Introduction

1.1 About FIPS 140

Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) run the FIPS 140 program. The NVLAP accredits independent testing labs to perform FIPS 140 testing; the CMVP validates modules meeting FIPS 140 validation. *Validated* is the term given to a module that is documented and tested against the FIPS 140 criteria.

More information is available on the CMVP website at https://csrc.nist.gov/projects/cryptographic-module-validation-program.

1.2 About this Document

This non-proprietary Cryptographic Module Security Policy for CryptoComply for .NET from SafeLogic Inc. (“SafeLogic”) provides an overview of the product and a high-level description of how it meets the overall Level 1 security requirements of FIPS 140-2.

CryptoComply for .NET may also be referred to as the “module” in this document.

1.3 External Resources

The SafeLogic website (www.safelogic.com) contains information on SafeLogic services and products. The Cryptographic Module Validation Program website contains links to the FIPS 140-2 certificate and SafeLogic contact information.

1.4 Notices

2 CryptoComply for .NET

2.1 Cryptographic Module Specification

CryptoComply for .NET is a standards-based “Drop-in Compliance™” cryptographic engine for .NET runtime environments. The module delivers core cryptographic functions to mobile and server platforms and features robust algorithm support, including Suite B algorithms.

The module’s software version is 1.0.2. The module's logical cryptographic boundary is the Windows Dynamic Link Library (DLL) file (ccn-1.0.2.dll).

The module is a software module that relies on the physical characteristics of the host platform. The module’s physical cryptographic boundary is defined by the enclosure of the host platform, which is the General Purpose Device that the module is installed on. For the purposes of FIPS 140-2 validation, the module’s embodiment type is defined as multi-chip standalone.

All operations of the module occur via calls from host applications and their respective internal daemons/processes. As such there are no untrusted services calling the services of the module.

2.1.1 Validation Level Detail

The following table lists the module’s level of validation for each area in FIPS 140-2:

Table 1 - Validation Level by FIPS 140-2 Section

FIPS 140-2 Section Title	Validation Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
Electromagnetic Interference / Electromagnetic Compatibility	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	1

2.1.2 Modes of Operation

The module supports two modes of operation: FIPS Approved mode and non-Approved mode. The module will be in FIPS Approved mode when the appropriate factory is called. To verify that a module is in the FIPS Approved mode of operation, the user can call a FIPS status method (*CryptoServicesRegistrar.isInApprovedOnlyMode()*). If the module is configured to allow FIPS Approved mode and non-Approved mode operations, a call to *CryptoServicesRegistrar.setApprovedMode(true)* will switch the current thread of user control into FIPS Approved mode.

In FIPS Approved mode, the module will not provide non-Approved algorithms, therefore, exceptions will be called if the user tries to access non-Approved algorithms in the FIPS Approved mode.

2.1.3 Approved Cryptographic Algorithms

2.1.3.1 CAVP Tested Approved Algorithms

The module's cryptographic algorithm implementations have received the following certificate numbers from the Cryptographic Algorithm Validation Program (CAVP).

Table 2 - FIPS Approved Algorithm Certificates

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
A1905	AES	FIPS 197 SP 800-38A Addendum to SP 800-38A (2010)	CBC, CBC-CS1, CBC-CS2, CBC-CS3, ECB, CFB8, CFB128, CTR, OFB	128, 192, 256	Encryption, Decryption
A1905	AES CCM	SP 800-38C	CCM	128, 192, 256	Authenticated Encryption, Decryption
A1905	AES CMAC	SP 800-38B	CMAC	128, 192, 256	Authenticated Encryption, Decryption
A1905	AES-FF1	SP 800-38G	FF1	128, 192, 256	Format Preserving Encryption, Decryption
A1905	AES GCM/GMAC[1]	SP 800-38D	GCM/GMAC	128, 192, 256	Authenticated Encryption, Decryption
A1905	AES KW, KWP	SP 800-38F	KW, KWP	128, 192, 256	Key Wrapping
A1905	CVL: KDF, Existing Application-Specific[2]	SP 800-135	TLS v1.0/1.1 KDF, TLS 1.2 KDF, X9.63 KDF	Various (See #A1905 for details)	KDF Services

[1] GCM encryption with an internally generated IV as outlined in Section 8.2.2 of NIST SP 800-38D; see Section 8.3 concerning external IVs. See Security Policy section 3.3 concerning external IVs. IV generation is compliant with IG A.5.
[2] These protocols have not been reviewed or tested by the CAVP and CMVP.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
A1905	DRBG	SP 800-90A	Hash DRBG HMAC DRBG CTR DRBG	112, 128, 192, 256 (SHA-1, SHA-2, 3-Key Triple DES, AES)	Random Bit Generation
A1905	DSA[3]	FIPS 186-4	Key Pair Generation, PQG Generation, PQG Verification, Signature Generation, Signature Verification	(1024, 160)[4] (2048, 224) (2048, 256) (3072, 256)	Digital Signature Services
A1905	ECDSA	FIPS 186-4	Key Generation, Signature Generation, Signature Verification, Public Key Validation, Signature Generation Component (CVL)	P-192[5], P-224, P-256, P-384, P- 521, K-163[6], K-233, K-283, K-409, K-571, B-163[7], B-233, B-283, B-409, B-571	Digital Signature Services

[3] DSA signature generation with SHA-1 is only for use with protocols
[4] Key size only used for Signature Verification
[5] In approved mode of operation, the use of this curve for anything other than verification is non-compliant.
[6] In approved mode of operation, the use of this curve for anything other than verification is non-compliant.
[7] In approved mode of operation, the use of this curve for anything other than verification is non-compliant.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
A1905	HMAC	FIPS 198-1	HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA-512/224, HMAC-SHA-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	Various (KS<BS, KS=BS, KS>BS)	HMAC Generation, HMAC Authentication
A1905	KAS[8]	SP 800-56Ar3	KAS-FFC: dhEphem, dhStatic KAS-ECC: ephemeralUnified, staticUnified	• FB, FC • ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 • MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 • P-224, P-256, P-384, P-521 • K-233, K-283, K-409, K-571 • B-233, B-283, B-409, B-571	Key Agreement For KAS-ECC, the key establishment methodology provides between 112 and 256 bits of encryption strength; anything less than 112 bits of encryption strength is non-compliant. For KAS-FFC, the key establishment methodology provides between 112 and 200 bits of encryption strength; anything less than 112 bits of encryption strength is non-compliant.

[8] Keys are not directly established into the module using KAS-ECC and KAS-FFC.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
A1905	KAS (KAS-SSC Cert. #A1905, CVL Cert. #A1905)[9]	SP 800-56Ar3 SP 800-135	SP 800-56Ar3 KAS-SSC with TLS v1.0/1.1 KDF, TLS 1.2 KDF or X9.63 KDF. Compliant to IG D.8 X1 Option 2, testing the shared secret and separately testing the key derivation function.		Key Agreement
A1905	KAS (KAS-SSC Cert. #A1905, KDA Cert. #A1905)[10]	SP 800-56Ar3 SP 800-56Cr2	SP 800-56Ar3 KAS-SSC with One Step KDA or HKDF KDA. Compliant to IG D.8 X1 Option 2, testing the shared secret and separately testing the key derivation function.		Key Agreement
A1905	KAS-SSC	SP 800-56Ar3	KAS-FFC: dhEphem, dhStatic KAS-ECC: ephemeralUnified, staticUnified	• FB, FC • ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 • MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 • P-224, P-256, P-384, P-521 • K-233, K-283, K-409, K-571 • B-233, B-283, B-409, B-571	Key Agreement – Shared Secret Computation

[9] Keys are not directly established into the module using KAS-ECC and KAS-FFC.
[10] Keys are not directly established into the module using KAS-ECC and KAS-FFC.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
A1905	KDA, One Step	SP 800-56Cr2	PRFs: • SHA-1 • SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 • SHA3-224, SHA3-256, SHA3-384, SHA3-512 • HMAC SHA-1 • HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512, HMAC SHA-512/224, HMAC SHA-512/256 • HMAC SHA3-224, HMAC SHA3-256, HMAC SHA3-384, HMAC SHA3-512 • KMAC-128, KMAC-256		Key Derivation
A1905	KDA, HKDF	SP 800-56Cr2	PRFs: • HMAC SHA-1 • HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512, HMAC SHA-512/224, HMAC SHA-512/256 • HMAC SHA3-224, HMAC SHA3-256, HMAC SHA3-384, HMAC SHA3-512		Key Derivation
A1905 (AES)	KTS: Key Wrapping Using AES[11]	SP 800-38F	AES KW, AES KWP	128, 192, 256	Key Transport For AES, the key establishment methodology provides between 128 and 256 bits of encryption strength

[11] Keys are not established directly into the module using key unwrapping.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
A1905 (TDES)	KTS: Key Wrapping Using TDES[12]	SP 800-38F	TKW	3-key Triple-DES	Key Transport For Triple-DES, key establishment methodology provides 112 bits of encryption strength
A1905	KTS-RSA[13]	SP 800-56Br2	KTS-OAEP-basic	2048, 3072, 4096	Key Transport Key establishment methodology provides 112 or 128 bits of encryption strength
A1905	PBKDF	SP 800-132	PBKDF with Option 1a only.	HMAC-based KDF using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	Key Derivation
A1905	RSA[14]	FIPS 186-4 SP 800-56B Section 7.1.2 FIPS 186-2	Key Pair Generation: 2048, 3072, 4096 Signature Generation (ANSI X9.31, PKCS 1.5, and PKCSPSS): 2048, 3072, 4096 Signature Verification (ANSI X9.31, PKCS 1.5, and PKCSPSS): 1024, 2048, 3072, 4096 RSA Decryption Primitive Component (CVL) per SP 800-56B: 2048 Signature Verification (ANSI X9.31, PKCS 1.5, and PKCSPSS): 1024, 1536, 2048, 3072, 4096 bits		Digital Signature Services, Key Transport (per SP 800-56B)

[12] Keys are not established directly into the module using key unwrapping.
[13] Keys are not established directly into the module using key transport.
[14] Keys are not established directly into the module using key transport.

CAVP Cert.	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
A1905	SHA-3, SHAKE	FIPS 202	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256,	N/A	Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications
A1905	SHA-3 Derived Functions	SP 800-185	• cSHAKE-128, cSHAKE-256 • KMAC-128, KMAC-256 • TupleHash-128, TupleHash-256 • ParallelHash-128, ParallelHash-256		
A1905	SHS	FIPS 180-4	SHA-1[15], SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	N/A	Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications
A1905	Triple-DES	SP 800-67	TCBC, TCFB8, TCFB64, TECB, TOFB, CTR	2-key, 3-key[16]	Encryption, Decryption
A1905	Triple-DES CMAC	SP 800-38B	CMAC	2-key, 3-key[17]	Generation, Authentication
A1905	Triple-DES TKW	SP 800-38F	TKW	3-key[18]	Key Wrapping

[15] Only for verification.
[16] 2^{16} block limit is enforced by the module, 2-key encryption is disabled.
[17] 2^{16} block limit is enforced by the module. In FIPS Approved mode, the use of 2-key Triple-DES to generate MACs for anything other than verification purposes is non-compliant.
[18] 2^{16} block limit is enforced by the module.

2.1.3.2 Vendor Affirmed Approved Algorithms

The following Approved cryptographic algorithms were implemented with vendor affirmation.

Table 3 - Approved Cryptographic Functions Implemented with Vendor Affirmation

Algorithm	IG Reference	Use
CKG using output from DRBG[19]	Vendor Affirmed per IG D.12	[SP 800-133] Section 6.1 (Asymmetric from DRBG) Section 7.1 (Symmetric from DRBG) Using DRBG #A1905

2.1.4 Non-Approved But Allowed Cryptographic Algorithms

The module supports the following FIPS 140-2 non-Approved but allowed algorithms that may be used in the FIPS Approved mode of operation.

Table 4 - Non-Approved But Allowed Cryptographic Algorithms

Algorithm	Use
MD5 within TLS	[IG D.2, IG 1.23 example 2a]
RSA Key Wrapping, Non-SP 800-56B compliant[20]	[IG D.9] RSA may be used by a calling application as part of a key encapsulation scheme. Key sizes: >= 2048 bits Key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength.

2.1.5 Non-Approved Mode of Operation

The module supports a non-Approved mode of operation. The algorithms listed in this section are not to be used by the operator in the FIPS Approved mode of operation.

Table 5 - Non-Approved Cryptographic Functions for Use in non-Approved mode Only

Algorithm	Use
AES (non-compliant)	Encryption, Decryption
ARC4 (RC4)	Encryption, Decryption
Camellia	Encryption, Decryption
ChaCha	Encryption, Decryption
DSA (non-compliant[21])	Public Key Cryptography

[19] The resulting key or a generated seed is an unmodified output from a DRBG
[20] Keys are not established into the module using RSA
[21] Deterministic signature calculation, support for additional digests, and key sizes.

Algorithm	Use
ECDSA (non-compliant[22])	Public Key Cryptography
EdDSA	Public Key Cryptography
ElGamal	Public Key Cryptography
FF3-1	Encryption, Decryption
NewHope	Key Agreement
OpenSSL PBKDF (non-compliant)	Key Derivation
PKCS#12 PBKDF (non-compliant)	Key Derivation
Poly1305	Message Authentication
RSA (non-compliant[23])	Public Key Cryptography
SEED	Encryption, Decryption
Serpent	Encryption, Decryption
SPHINCS-256	Signature Scheme

2.2 Critical Security Parameters and Public Keys

2.2.1 Critical Security Parameters

The table below provides a complete list of Critical Security Parameters used within the module:

Table 6 - Critical Security Parameters

CSP	Description / Usage
AES Encryption Key	[FIPS 197, SP 800-38A, SP 800-38C, SP 800-38D, SP 800-38G, Addendum to SP 800-38A] AES (128/192/256) encrypt key[24]
AES Decryption Key	[FIPS 197, SP 800-38A, SP 800-38C, SP 800-38D, SP 800-38G, Addendum to SP 800-38A] AES (128/192/256) decrypt key
AES Authentication Key	[FIPS 197] AES (128/192/256) CMAC/GMAC key
AES Wrapping Key	[SP 800-38F] AES (128/192/256) key wrapping key
DH Agreement Key	[SP 800-56Ar3] Diffie-Hellman (224 - 512 bits) private key agreement key
DRBG (CTR AES)	V (128 bits) and AES key (128/192/256), entropy input (length dependent on security strength)
DRBG (CTR Triple-DES)	V (64 bits) and Triple-DES key (192 bits), entropy input (length dependent on security strength)

[22] Deterministic signature calculation, support for additional digests, and key sizes.
[23] Support for additional digests, signature formats, and key sizes.
[24] The AES GCM key is generated randomly per IG A.5, and the Initialization Vector (IV) is also generated randomly and at least 96 bits. In the event of power loss the AES-GCM key will be lost and the consuming application must ensure that new AES-GCM keys for encryption or decryption are re-distributed. Refer also to Security Policy section 3.3.

CSP	Description / Usage
DRBG (Hash)	V (440/888 bits) and C (440/888 bits), entropy input (length dependent on security strength)
DRBG (HMAC)	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
DSA Signing Key	[FIPS 186-4] DSA (2048/3072 bits) signature generation key
EC Agreement Key	[SP 800-56Ar3] EC (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) private key agreement key
EC Signing Key	[FIPS 186-4] ECDSA (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) signature generation key
HMAC Authentication Key	[FIPS 198-1] Keyed-Hash key (SHA-1, SHA-2, SHA-3). Key size determined by security strength required (>= 112 bits)
PBKDF Secret	[SP 800-132] Secret value used in construction of Keyed-Hash key for the specified PRF
RSA Signing Key	[FIPS 186-4] RSA (>=2048 bits) signature generation key
RSA Key Transport Key	[SP 800-56Br2] RSA (>=2048 bits) key transport (decryption) key
TLS KDF Secret Value	[SP 800-135] Secret value used in construction of Keyed-Hash key for the specified TLS PRF
Triple-DES Encryption Key	[SP 800-67] Triple-DES (192 bits) encryption key
Triple-DES Decryption Key	[SP 800-67] Triple-DES (128/192 bits) decryption key
Triple-DES Authentication Key	[SP 800-67] Triple-DES (128/192 bits) CMAC key
Triple-DES Wrapping Key	[SP 800-38F] Triple-DES key wrapping (192 bits)/unwrapping key (128/192 bits)
X9.63 KDF Secret Value	[SP 800-135] Secret value used in construction of input for the specified X9.63 PRF
SP 800-56C-rev2 One-Step Derivation Function	[SP 800-56C-rev2] Secret value used in construction of key for underlying PRF.
SP 800-56C-rev2 Hash Derivation Function (HKDF)	[SP 800-56C-rev2] Secret value used in construction of key for underlying PRF.

2.2.2 Public Keys

The table below provides a complete list of the public keys used within the module:

Table 7 - Public Keys

Public Key	Description / Usage
DH Agreement Key	[SP 800-56Ar3] Diffie-Hellman (>=2048) public key agreement key (All SP 800-56A-rev3 parameter sets)
DSA Verification Key	[FIPS 186-4] DSA (1024/2048/3072) signature verification key
EC Agreement Key	[SP 800-56Ar3] EC (P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409 and B-571) public key agreement key
EC Verification Key	[FIPS 186-4] ECDSA (P-192, P-224, P-256, P-384, P-521, K-163, K-233, K-283, K-409, K-571, B-163, B-233, B-283, B-409 and B-571) signature verification key
RSA Key Transport Key	[SP 800-56Br2] RSA (2048 - 16384) key transport (encryption) key
RSA Verification Key	[FIPS 186-4] RSA (1024, 1536, >=2048) signature verification key

2.3 Module Interfaces

The figure below shows the module's physical and logical block diagram:

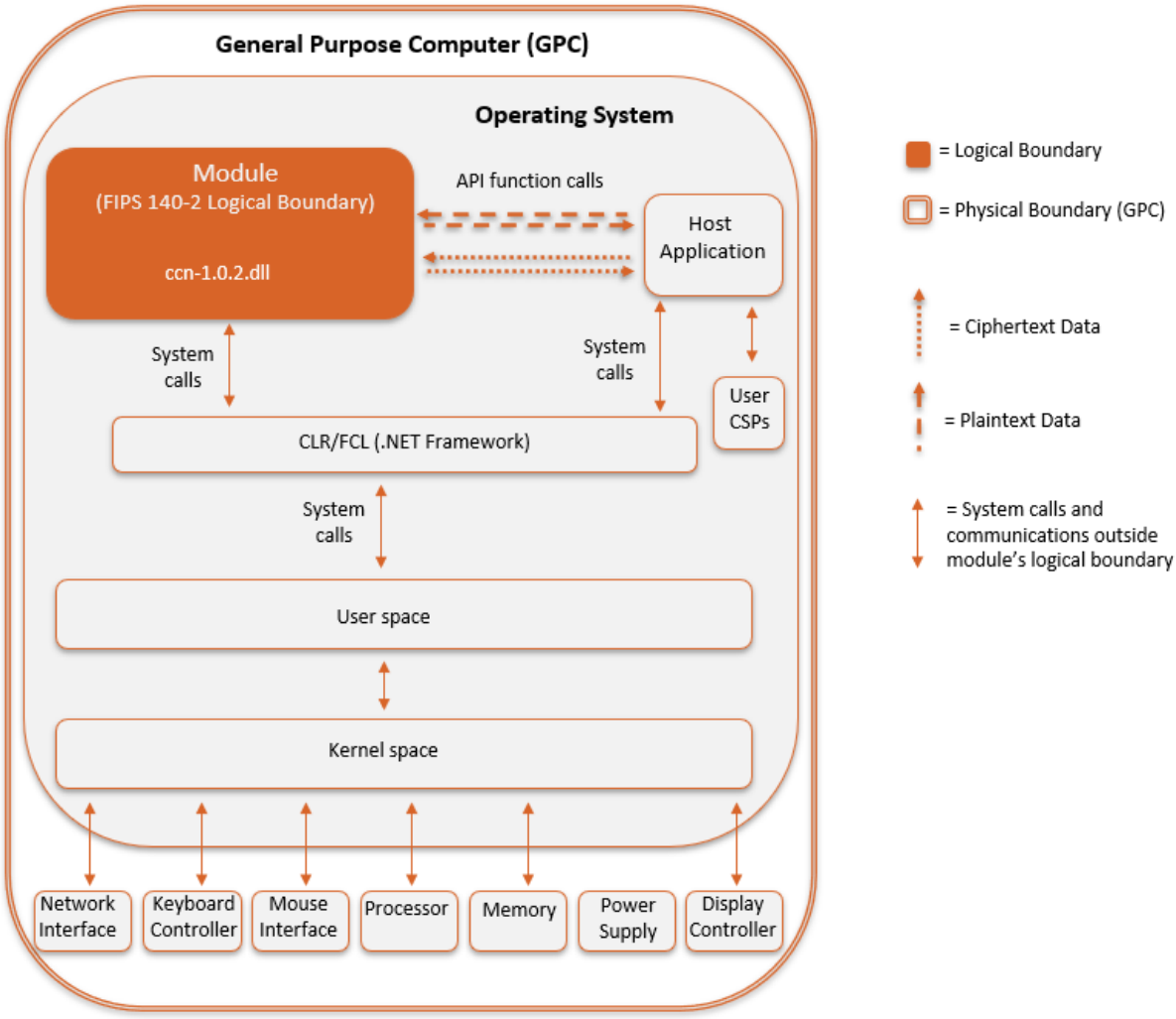

Figure 1 – Module Boundary and Interfaces Diagram

The module's physical boundary is the boundary of the General Purpose Computer (GPC) that the module is installed on, which includes a processor and memory. The interfaces (ports) for the physical boundary include the computer's network port, keyboard port, mouse port, power plug, and display. When operational, the module does not transmit any information across these physical ports because it is a software cryptographic module. Therefore, the module's interfaces are purely logical.

Figure 1 shows the logical relationship of the cryptographic module to the other software and hardware components of the GPC. The module classes are executed on the .NET Framework Common Language Runtime (CLR) using the classes of the Framework Class Library (FCL). The CLR is the interface to the computer's Operating System (OS), which is the interface to the various physical components of the

computer. The logical interface is provided through an Application Programming Interface (API) that a calling daemon can operate. The API itself defines the module's logical boundary, i.e. all access to the module is through this API. The API provides functions that may be called by an application (see Section 2.4 – Roles, Services, and Authentication for the list of available functions). The module distinguishes between logical interfaces by logically separating the information according to the defined API.

The API provided by the module is mapped onto the FIPS 140- 2 logical interfaces, which relate to the module's callable interface as follows:

Table 8 - Logical Interface / Physical Interface Mapping

FIPS 140-2 Interface	Logical Interface	Module Physical Interface
Data Input	API input parameters – plaintext and/or ciphertext data	Network Interface
Data Output	API output parameters and return values – plaintext and/or ciphertext data	Network Interface
Control Input	API method calls – method calls, or input parameters, that specify commands and/or control data used to control the operation of the module	Network Interface, Keyboard Interface, Mouse Interface
Status Output	API output parameters and return/error codes that provide status information used to indicate the state of the module	Display Controller, Network Interface
Power	None	Power Supply

When the module performs self-tests or is in an error state, the module prevents all output on the logical data output interface. Activities in the module are single-threaded, and when in an error state, the module does not return any output data, only an error value.

2.4 Roles, Services, and Authentication

2.4.1 Assumption of Roles

The module supports two distinct operator roles, which are the User and Crypto Officer (CO), as indicated in Table 9 - Description of Roles. The cryptographic module implicitly maps the two roles to the services. A user is considered the owner of the thread that instantiates the module and, therefore, only one concurrent user is allowed.

The module does not support a Maintenance role or bypass capability. The module leverages the CLR to allow multiple threads (concurrent operations), and the operating system and CLR manage separate memory for each thread. In addition, there is high level thread management implemented by the module. The module does not support authentication.

Table 9 - Description of Roles

Role	Role Description	Authentication Type
CO	Crypto Officer – Initialize the module	N/A – Authentication is not a requirement for FIPS 140 Level 1
User	User – Use of the complete API	N/A – Authentication is not a requirement for FIPS 140 Level 1

2.4.2 Services

All services implemented by the module are listed in Table 10 - Module Services, Descriptions. The second column provides a description of each service, and availability to the Crypto Officer and User is indicated in columns three and four, respectively. Table 11 - CSP Access Rights within Services describes all CSP usage by services. All services available to the User are also available to the Crypto Officer. Only one role may be active at a time and the module does not allow concurrent operators, although an operator may perform more than one task concurrently.

Authentication of the Crypto Officer and/or User is not supported by the module but is a task performed by the host environment.

Table 10 - Module Services, Descriptions, and Roles

Service	Description	CO	User
Initialize Module and Run Self-Tests on Demand	The CLR will call the static constructor for self-tests on module initialization.	X	
Show Status	A user can call *CryptoStatus.IsReady()* at any time to determine if the module is ready. *IsInApprovedOnlyMode()* can be called to determine the FIPS mode of operation.		X
Zeroize / Power-off	The module uses the CLR garbage collector on thread termination when objects are reclaimed.		X
Data Encryption	Used to encrypt data.		X
Data Decryption	Used to decrypt data.		X
MAC Calculation	Used to calculate data integrity codes with CMAC.		X
Signature Generation	Used to generate digital signatures (DSA, ECDSA, RSA).		X
Signature Verification	Used to verify digital signatures (DSA, ECDSA, RSA).		X
DRBG (SP 800-90A) output	Used for random number and IV key generation.		X
Key Generation – Based on DRBG (SP 800-90A)	Used for key generation.		
Message Hashing	Used to generate a SHA-1, SHA-2, or SHA-3 message digest, SHAKE output.		X
Keyed Message Hashing	Used to calculate data integrity codes with HMAC.		X
TLS Key Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a master secret in TLS from a pre-master secret and additional input.		X

Service	Description	CO	User
X9.63 Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
SP 800-56Cr2 One-Step Derivation Function	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
SP 800-56Cr2 Hash Derivation Function (HKDF)	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X
Password-Based Key Derivation Function (PBKDF)	(secret input) (outputs secret) Used to generate a key using an encoding of a password and an additional function such as a message hash.		X
Key Agreement Schemes	Used to calculate key agreement values (SP 800-56Ar3)		X
Key Wrapping/Transport	Used to encrypt a key value. (RSA, AES, Triple-DES)		X
Key Unwrapping	Used to decrypt a key value. (RSA, AES, Triple-DES)		X
NDRNG Callback	Gathers entropy in a passive manner from a user-provided function.		X
Utility	Miscellaneous utility functions, does not access CSPs.		X

Note: The module services are the same in the FIPS Approved and non-Approved modes of operation. The only difference is the function(s) used (Approved/allowed or non-Approved/non-allowed).

Services in the module are accessed via the public APIs of the DLL. The ability of a thread to invoke non-Approved services depends on whether it has been registered with the module as FIPS Approved mode only. In FIPS Approved only mode, no non-Approved services are accessible.

Table 11 - CSP Access Rights within Services defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as:

- **G** = Generate: The module generates the CSP.
- **R** = Read: The module reads the CSP. The read access is typically performed before the module uses the CSP.
- **E** = Execute: The module executes using the CSP.
- **W** = Write: The module writes the CSP. The write access is typically performed after a CSP is imported into the module, when the module generates a CSP, or when the module overwrites an existing CSP.
- **Z** = Zeroize: The module zeroizes the CSP.

Note: keys are not established directly into the module using derivation functions or unwrapping schemes.

Table 11 - CSP Access Rights within Services

Services	CSPs																						
	AES Encryption Key	AES Decryption Key	AES Authentication Key	AES Wrapping Key	DH Agreement Key	DRBG (CTR AES)	DRBG (CTR Triple-DES)	DRBG (Hash)	DRBG (HMAC)	DSA Signing Key	EC Agreement Key	EC Signing Key	HMAC Authentication Key	PBKDF Secret	RSA Signing Key	RSA Key Transport Key	SP 800-56C Concat. DF Secret	SP 800-56C HKDF Secret	TLS KDF Secret	Triple-DES Encryption Key	Triple-DES Decryption Key	Triple-DES Authentication Key	Triple-DES Wrapping Key	X9.63 KDF Secret Value
Initialize Module and Run Self-Tests on Demand																								
Show Status																								
Zeroize / Power-off	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Data Encryption	R																			R				
Data Decryption		R																			R			
MAC Calculation			R																			R		
Signature Generation										R		R			R									
Signature Verification										R		R			R									
DRBG (SP 800-90A) output	G	G	G	G	G	GR	GR	GR	GR	G	G	G	G		G	G				G	G	G	G	

Key Generation – Based on DRBG (SP 800-90A)						R	R	R	R															
Message Hashing																								
Keyed Message Hashing													R											
TLS Key Derivation Function																			R					
X9.63 Derivation Function					G						G				G									R
SP 800-56r2 One-Step Derivation Function					G						G				G		R							
SP 800-56r2 Hash Derivation Function (HKDF)					G						G				G			R						
PBKDF													GR	R										
Key Agreement Schemes	G	G	G	G	R						R		G			R				G	G	G	G	
Key Wrapping/ Transport				R									R			R							R	
Key Unwrapping				R									R			R							R	

NDRNG Callback						G	G	G	G															
Utility																								

2.5 Physical Security

The module is a software-only module and does not have physical security mechanisms.

2.6 Operational Environment

The module operates in a modifiable operational environment under the FIPS 140-2 definitions.

The module runs on a GPC running one of the operating systems specified in the approved operational environment list in this section. Each approved operating system manages processes and threads in a logically separated manner. The module's user is considered the calling application that instantiates the module within the process space of the CLR. When the Module is not otherwise configured, it will start by default in the non-FIPS-approved mode.

The module was tested on the following platforms:

Table 12 - Tested Environments

Operating System	.NET Framework Version	Hardware Platform	Processor (CPU)
Microsoft Windows 10 Professional (64-bit)	.NET 4.5.2 framework (CLR version 4)	Dell XPS 15 7590	Intel Core i7 9750H

FIPS 140-2 validation compliance is maintained for other compatible operating systems (in single user mode) where the module source code is unmodified, and the requirements outlined in NIST IG G.5 are met. No claim can be made as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate.

The module is vendor-affirmed to be FIPS 140-2 compliant when running one of the .NET Runtime environments on any of the following supported single-user operating systems for which operational testing and algorithm testing were not performed:

- Windows Vista with .NET 4.5.2
- Windows 7 with .NET 4.6.1
- Windows 8 with .NET 4.5.2
- Windows 8 with .NET 4.6.1
- Windows 8.1 with .NET 4.6.1
- Windows Server 2008 R2 SP1 with .NET 4.5.2
- Windows Server 2012 R2 with .NET 4.5.2
- Windows Server 2012 R2 with .NET 4.6.1

2.6.1 Use of External RNG

The module makes use of FipsSecureRandom to seed the DRBG. FipsSecureRandom has three builder methods used to control how entropy is provided. The method FromDefaultEntropy() shall not be used in the FIPS Approved mode of operation. In the FIPS Approved mode either FromEntropySource(SecureRandom) or FromEntropySource(IEntropySourceProvider) can be used. In either case, the user shall ensure an Approved entropy source is provided and will block, or fail, if it is unable to provide the amount of entropy requested.

The module's FipsSecureRandom() function will request entropy as appropriate to the security strength and seeding configuration for the DRBG that is using it. In FIPS Approved mode, the minimum amount of entropy that would be requested is 112 bits, with a larger minimum being set if the security strength of the operation requires it.

The module will wait until the FipsSecureRandom() returns the requested amount of entropy before seeding the DRBG.

2.7 Self-Tests

Each time the module is powered up, it tests that the cryptographic algorithms still operate correctly and that sensitive data has not been damaged. Power-up self-tests are available on demand by power cycling the module.

On power-up or reset, the module performs the self-tests that are described in Table 13 - Power-Up Self-Tests. All KATs must be completed successfully prior to any other use of cryptography by the module. If one of the KATs fails, the module enters the Self-Test Failure error state. The module will output a detailed error message when *CryptoStatus.isReady()* is called. The error state can only be cleared by reloading the module and calling *CryptoStatus.isReady()* again to confirm successful completion of the KATs.

2.7.1 Power-Up Self-Tests

Table 13 - Power-Up Self-Tests

Test Target	Description
Software Integrity Check	HMAC-SHA-512 (HMAC Cert. #A1905)
AES	KATs: Encryption, Decryption Modes: ECB Key sizes: 128 bits
AES CCM	KATs: Generation, Verification Key sizes: 128 bits
AES CMAC	KATs: Generation, Verification Key sizes: 128 bits
AES GCM/GMAC	KATs: Generation, Verification Key sizes: 128 bits

Test Target	Description
DRBG	KATs: HASH_DRBG, HMAC_DRBG, CTR_DRBG Security Strengths: 256 bits
DSA	KAT: Signature Generation, Signature Verification Key sizes: 2048 bits
ECDSA	KAT: Signature Generation, Signature Verification Curves/Key sizes: P-256, B-233
HKDF (SP 800-56Cr2)	KATs: Key derivation PRFs: HMAC-SHA-256
HMAC	KATs: Generation, Verification SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512
DH Agreement	KAT: Agreement Test (Diffie-Hellman computation) Parameter Sets/Key sizes: ffdhe2048
KAS: FFC	KATs: Per IG D.8 Scenario X1 – Primitive "Z" Computation Parameter Sets/Key sizes: ffdhe2048
KAS: ECC	KATs: Per IG D.8 Scenario X1 – Primitive "Z" Computation Parameter Sets/Key sizes: P-256, B-233
KDA (SP 800-56Cr2)	KATs: Key derivation Modes: One-Step PRFs: HMAC-SHA-256, SHA-256, KMAC-256
KDF, Existing Application-Specific (CVL)	• MD5 KAT performed to verify operation of MD5 digest used in TLS 1.0 KDF • TLS 1.0 SHA-1 KDF KAT performed to verify TLS 1.0 KDF, TLS 1.1/1.2 KDF • SHA-256-HMAC KAT performed to verify TLS 1.1/1.2 KDF • X9.63 SHA-256 KDF KAT performed to verify X9.63 KDF
PBKDF	KATs: Master key derivation PRFs: HMAC-SHA-256
RSA	KATs: Signature Generation, Signature Verification Key sizes: 2048 bits
RSA, Key Transport	KATs: SP 800-56Br2 specific KATs per IG D.4 Key sizes: 2048 bits
SHS	KATs: Output Verification SHA sizes: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3512
Triple-DES	KATs: Encryption, Decryption Modes: TECB Key sizes: 3-Key
Triple-DES CMAC	KATs: Generation, Verification Key sizes: 3-Key
XOF (Extendable-Output functions)	KATs: Output Verification XOFs: SHAKE128, SHAKE256

2.7.2 Conditional Self-Tests

The module implements the following conditional self-tests upon key generation, or random number generation (respectively):

Table 14 - Conditional Self-Tests

Test Target	Description
DRBG	DRBG Continuous Test performed when a random value is requested from the DRBG (all DRBGs).
DRBG Health Checks	Performed conditionally on DRBG (all DRBGs), per SP 800-90A Section 11.3.
DSA	DSA Pairwise Consistency Test performed on every DSA key pair generation.
ECDSA	ECDSA Pairwise Consistency Test performed on every EC key pair generation.
KAS: Pairwise consistency	DH Pairwise Consistency Test performed on every DH key pair generation (FFC and ECC)
KAS: SP 800-56A Assurances	Performed conditionally per SP 800-56Ar3 Sections 5.5.2 and/or 5.6.2. Required per IG 9.6 and IG D.8.
NDRNG	NDRNG Continuous Test performed when a random value is requested from the entropy source.
RSA	RSA Pairwise Consistency Test performed on every RSA key pair generation.

2.8 Mitigation of Other Attacks

The module implements basic protections to mitigate against timing-based attacks against its internal implementations. There are two countermeasures used.

The first countermeasure is Constant Time Comparisons, which protect the digest and integrity algorithms by strictly avoiding “fast fail” comparison of MACs, signatures, and digests so the time taken to compare a MAC, signature, or digest is constant regardless of whether the comparison passes or fails.

The second countermeasure is made up of Numeric Blinding and decryption/signing verification which both protect the RSA algorithm.

Numeric Blinding prevents timing attacks against RSA decryption and signing by providing a random input into the operation which is subsequently eliminated when the result is produced. The random input makes it impossible for a third party observing the private key operation to attempt a timing attack on the operation as they do not have knowledge of the random input and consequently the time taken for the operation tells them nothing about the private value of the RSA key.

Decryption/signing verification is carried out by calculating a primitive encryption or signature verification operation after a corresponding decryption or signing operation before the result of the decryption or signing operation is returned. The purpose of this is to protect against Lenstra's CRT attack by verifying the correctness of the private key calculations involved. Lenstra's CRT attack takes

advantage of undetected errors in the use of RSA private keys with CRT values and, if exploitable, can be used to discover the private value of the RSA key.

3 Security Rules and Guidance

3.1 Basic Enforcement

The module design corresponds to the module security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS 140-2 Level 1 module.

1. The module provides two distinct operator roles: User and Crypto Officer.
2. The module does not provide authentication.
3. The operator may command the module to perform the power up self-tests by cycling power or resetting the module.
4. Power-up self-tests do not require any operator action.
5. Data output is inhibited during key generation, self-tests, zeroization, and error states.
6. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
7. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
8. The module does not support concurrent operators.
9. The module does not have any external input/output devices used for entry/output of data.
10. The module does not enter or output plaintext CSPs from the module's physical boundary.
11. The module does not output intermediate key values.

3.2 Basic Guidance

Functionality in the module is provided via distinct classes that provide access to the FIPS Approved and non-Approved services provided by the module.

When the module is being used in FIPS Approved-only mode, classes providing implementations of algorithms which are not FIPS Approved, or allowed, are explicitly disabled.

3.3 Enforcement and Guidance for AES GCM IVs

The module supports two methods of AES GCM IV generation.

The first method of GCM IV generation is when AES GCM is used as part of TLS 1.2 cipher suites conformant to IG A.5 Scenario 1, RFC 5288 and SP 800-52 Section 3.3.1. The construction of the 64-bit nonce_explicit part of the IV is generated using the FipsNonceGenerator, where a monotonically increasing counter is used as the basis for the nonce. Rollover of the counter in the FipsNonceGenerator will result in an IllegalStateException indicating the FipsNonceGenerator is exhausted. Per IG A.5 (where used for TLS), rollover will terminate any TLS session in process using the current key and the exception can only be recovered from by using a new handshake and creating a new FipsNonceGenerator.

The GCM IV can also be generated randomly, per IG A.5, Scenario 2. The IV is constructed to be at least 96 bits. The module enforces the use of an approved DRBG in conformance with Section 8.2.2 of SP 800-38D.

Per IG A.5, Section 2.2.1 of this Security Policy also states that in the event module power is lost and restored the consuming application must ensure that any of its AES-GCM keys used for encryption or decryption are re-distributed.

3.4 Enforcement and Guidance for Use of the Approved PBKDF

In line with the requirements for SP 800-132, keys generated using the approved PBKDF must only be used for storage applications. Any other use of the approved PBKDF is non-compliant.

In FIPS Approved mode the module enforces that any password used must encode to at least 14 bytes (112 bits) and that the salt is at least 16 bytes (128 bits) long. The iteration count associated with the PBKDF should be as large as practical.

As the module is a general purpose software module, it is not possible to anticipate all the levels of use for the PBKDF, however a user of the module should also note that a password should at least contain enough entropy to be unguessable and also contain enough entropy to reflect the security strength required for the key being generated. In the event a password encoding is simply based on ASCII, a 14-byte password is unlikely to contain sufficient entropy for most purposes. Users are referred to Appendix A, “Security Considerations” in SP 800-132 for further information on password, salt, and iteration count selection.

3.5 Rules for Setting the N and the S String in cSHAKE

To customize the output of the cSHAKE function, the cSHAKE algorithm permits the operator to input strings for the Function-Name input (N) and the Customization String (S).

The Function-Name input (N) is reserved for values specified by NIST and should only be set to the appropriate NIST specified value. Any other use of N is non-conformant.

The Customization String (S) is available to allow users to customize the cSHAKE function as they wish. The length of S is limited to the available size of a byte array in the CLR running the module.

3.6 Software Installation

The module is provided directly to solution developers and is not available for direct download to the general public. Only the compiled module is provided to solution developers. The module and its host application are to be installed on an operating system specified in Section 2.6 or on an operating system where portability is maintained.

4 References and Acronyms

4.1 References

Table 15 – References

Abbreviation	Full Specification Name
ANSI X9.31	X9.31-1998, Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), September 9, 1998
FIPS 140-2	Security Requirements for Cryptographic modules, May 25, 2001
FIPS 180-4	Secure Hash Standard (SHS)
FIPS 186-2	Digital Signature Standard (DSS)
FIPS 186-4	Digital Signature Standard (DSS)
FIPS 197	Advanced Encryption Standard
FIPS 198-1	The Keyed-Hash Message Authentication Code (HMAC)
FIPS 202	SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions
IG	Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program
PKCS#1 v2.1	RSA Cryptography Standard
PKCS#5	Password-Based Cryptography Standard
SP 800-38A	Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode
SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
SP 800-38C	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
SP 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping
SP 800-38G	Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption
SP 800-56Ar3	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
SP 800-56Br2	Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography
SP 800-56Cr2	Recommendation for Key Derivation through Extraction-then- Expansion
SP 800-67	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
SP 800-89	Recommendation for Obtaining Assurances for Digital Signature Applications
SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators

Abbreviation	Full Specification Name
SP 800-131Ar2	Transitioning the Use of Cryptographic Algorithms and Key Lengths
SP 800-132	Recommendation for Password-Based Key Derivation
SP 800-133	Recommendation for Cryptographic Key Generation
SP 800-135	Recommendation for Existing Application–Specific Key Derivation Functions
SP 800-185	SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash

4.2 Acronyms

The following table defines acronyms found in this document:

Table 16 - Acronyms and Terms

Acronym	Term
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher-Block Chaining
CCM	Counter with CBC-MAC
CCCS	Canadian Centre for Cyber Security
CDH	Computational Diffie-Hellman
CFB	Cipher Feedback Mode
CLR	Common Language Runtime
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CPU	Central Processing Unit
CS	Ciphertext Stealing
CSP	Critical Security Parameter
CTR	Counter Mode
CVL	Component Validation List
DES	Data Encryption Standard
DH	Diffie-Hellman
DLL	Dynamic Link Library
DRAM	Dynamic Random Access Memory
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards Curve DSA using Ed25519, Ed448
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCL	Framework Class Library
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
GPC	General Purpose Computer
HMAC	(Keyed-) Hash Message Authentication Code
IG	Implementation Guidance
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test

Acronym	Term
KDF	Key Derivation Function
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
MD5	Message Digest algorithm MD5
N/A	Not Applicable
NDRNG	Non Deterministic Random Number Generator
OCB	Offset Codebook Mode
OFB	Output Feedback
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PKCS	Public-Key Cryptography Standards
PQG	Diffie-Hellman Parameters P, Q and G
PRF	Pseudorandom Function
RC	Rivest Cipher, Ron's Code
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
TCBC	TDEA Cipher-Block Chaining
TCFB	TDEA Cipher Feedback Mode
TDEA	Triple Data Encryption Algorithm
TDES	Triple Data Encryption Standard
TECB	TDEA Electronic Codebook
TOFB	TDEA Output Feedback
TLS	Transport Layer Security
USB	Universal Serial Bus
XOF	Extendable-Output Function