codeEncrypt

Version 4.5.2

FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.0

July 23, 2021

Prepared for:

Prepared by:

Code Corporation
434 West Ascension Way
Suite 300
Murray, UT 84123
www.codecorp.com
+1 801.495.2200

wolfSSL Inc.
10016 Edmonds Way
Suite C-300
Edmonds, WA 98020
wolfSSL.com
+1 425.245.8247

KeyPair Consulting Inc.
846 Higuera Street
Suite 2
San Luis Obispo, CA
93401
keypair.us
+1 805.316.5024

Table of Contents

List of Tables

1 Introduction

This document defines the Security Policy for the Code Corporation codeEncrypt (Software Version 4.5.2) module, hereafter denoted the Module. The Module is a cryptography software library, designated as a multi-chip standalone embodiment in [140] terminology. The Module is intended for use by US and Canadian Federal agencies and other markets that require FIPS 140-2 validated cryptographic functionality.

The Module meets FIPS 140-2 overall Level 1 requirements, with security levels as follows:

Table 1 - Security Level of Security Requirements

Security Requirement	Security 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
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

The Module does not implement attack mitigations outside the scope of [140], hence [140] Section 4.11 *Mitigation of Other Attacks* is not applicable per [140IG] G.3 *Partial Validations and Not Applicable Areas of FIPS 140-2*. [140] Section 4.5 *Physical Security* is not applicable, as permitted by [140IG] 1.16, *Software Module* and [140IG] G.3.

The Module conforms to [140IG] D.11 *References to the Support of Industry Protocols*: while it provides [56A] conformant schemes and API entry points oriented to TLS usage, the Module does not provide a full implementation of TLS. The TLS protocol has not been reviewed or tested by the CAVP and CMVP.

The Module design corresponds to the Module security rules. Security rules enforced by the Module are described in the appropriate context of this document.

2 Operational Environment

Operational testing was performed for the following Operating Environments:

Table 2 – Tested Operating Environments

	Operating System	Processor	Platform	Version Tested
1	CodeOS v1.4	CodeCorp CT8200 (ARM FA626TE)	Series CR2700 Code Reader(s)	4.5.2

The Module conforms to [140IG] 6.1 *Single Operator Mode and Concurrent Operators*. The tested environments place user processes into segregated spaces. A process is logically removed from all other processes by the hardware and Operating System. Since the Module exists inside the process space of the application this environment implicitly satisfies requirement for a single user mode.

The Module conforms to [140IG] 1.21 *Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI).* The Intel Processor AES-NI functions are identified by [140IG] 1.21 as a known PAA.

3 Cryptographic Boundary and Logical Interfaces

Figure 1 depicts the Module operational environment, with the logical boundary highlighted in red inclusive of all Module entry points (API calls), conformant with [140IG] 14.3 *Logical Diagram for Software, Firmware and Hybrid Modules*.

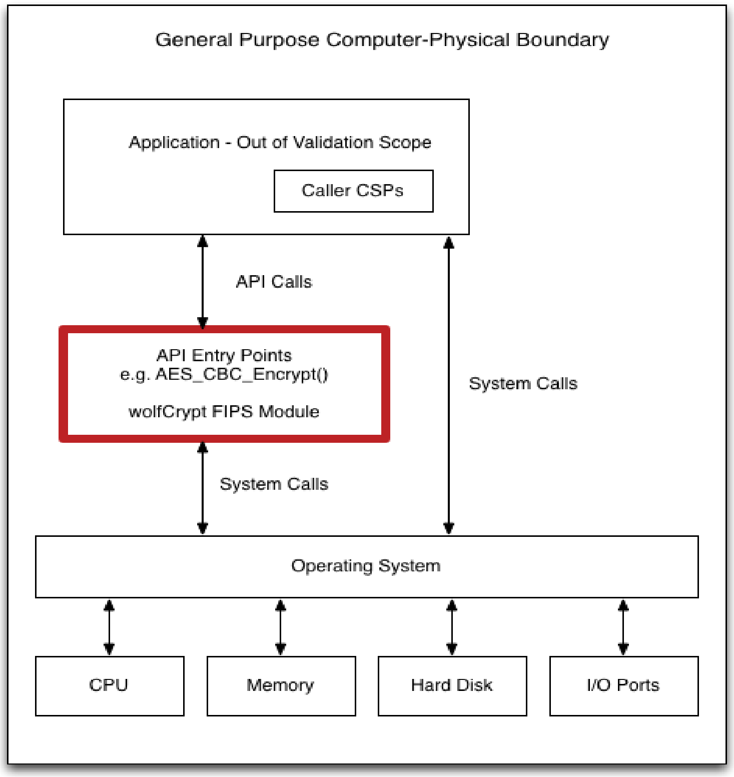

Figure 1 – Module Block Diagram

The Module conforms to [140IG] 1.16 *Software Module*:

- The physical cryptographic boundary is the general-purpose computer which wholly contains the Module and operating system.
- The logical cryptographic boundary is the set of object files corresponding to the source code files listed in Table 14.
- All components are defined per AS01.08; no components are excluded from [140] requirements.
- The Module does not map any interfaces to physical ports. Table 3 defines the Module’s [140] logical interfaces.
- The power-up approved integrity test is performed over all components of the logical boundary.
- Updates to the Module are provided as a complete replacement in accordance with [140IG] 9.7 *Software/Firmware Load Test*.

Table 3 – Ports and Interfaces

Description	Logical Interface Type
API entry point	Control in
API function parameters	Data in
API return value	Status out
API function parameters	Data out

4 Approved and Allowed Cryptographic Functionality

The Module implements the FIPS Approved and allowed cryptographic functions listed in the table below. VA in the Cert column indicates Vendor Affirmed. Strength citations use [57P1] notation.

Table 4 – Approved Cryptographic Functions

Cert	Algorithm	Mode	Key Lengths, Curves or Moduli (in bits)	Use
A902	AES [197]	CBC [38A]	Key sizes: 128, 192, 256	Encryption, Decryption
		CTR [38A]	Key sizes: 128, 192, 256	Encryption, Decryption
		CCM [38C]	Key sizes: 128, 192, 256 Tag len: 32*, 48*, 64, 80*, 96*, 112*, 128	Authenticated Encryption, Authenticated Decryption, Message Authentication
		CMAC [38C]	Key sizes: 128, 192, 256	Generation, Verification
		ECB [38A]	Key sizes: 128, 192, 256	Encryption, Decryption
		GMAC	Key sizes: 128, 192, 256	Message Authentication
		GCM [38D]	Key sizes: 128, 192, 256 Tag len: 96, 104, 112, 120, 128	Authenticated Encryption, Authenticated Decryption, Message Authentication
A902	DRBG [90A]	Hash_DRBG	SHA-256	Random Bit Generation, no prediction resistance
A902	DSA [186]		L = 2048 N = 256	FFC Key Generation for KAS
A902	ECDSA [186]		P-192 (Signature and Key Verification only), P-224, P-256, P-384, P-521 SHA-1**, SHA-224, SHA-256, SHA-384, and SHA-512 SHA3-224†, SHA3-256†, SHA3-384†, and SHA3-512†	ECC Key Generation, Public Key Validation, Signature Generation, Signature Verification
A902	HMAC [198]		SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 SHA3-224, SHA3-256, SHA3-384, and SHA3-512	Generation, Verification, Message Authentication
A902	CVL (KAS) [56A]	FFC	FC L = 2048 N = 256	Key Agreement primitives

Cert	Algorithm	Mode	Key Lengths, Curves or Moduli (in bits)	Use
		ECC	P-256, P-384, P-521	
		ECC CDH	P-224, P-256, P-384, P-521	
A902	CVL (RSADP) [56B] ***		k = 2048	Key transport primitive RSADP
A902	RSA [186]	PKCS v1.5 and PSS	1024 (verification only), 2048, 3072, 4096 SHA-1 (verification only) SHA-224, SHA-256, SHA-384, and SHA-512 SHA3-224†, SHA3-256†, SHA3-384†, and SHA3-512†	Key Generation, Signature Generation, Signature Verification
A902	SHA-3 [202]		SHA3-224, SHA3-256, SHA3-384, SHA3-512	Message Digest Generation
A902	SHS [180]		SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Message Digest Generation
A902	Triple-DES [67]	TCBC [38A]	Key size: 192	Encryption, Decryption

* Was only tested on CAVP certificate A902.

** CAVP testing permits testing with SHA-1; see Section 5, item 3.c below for conditions.

† Vendor-affirmed when used with SHA-3 (CAVP testing does not offer testing with SHA-3 variants).

‡ Vendor affirmed when used with k=4096 for key generation and signature verification; the module supports k=4096 for all operations, but CAVP testing of k = 4096 is only available for signature generation.

*** RSADP with k=2048 is the only CAVP testable aspect of [56B] key transport and is listed in the Component Validation List (CVL). The vendor affirms conformance to [56B] for RSAEP and RSADP with other key sizes, since no CAVP test is available.

The module is capable of performing Key Derivation through Extraction-then-Expansion using HMAC-SHA-256 and HMAC-SHA-384. The vendor affirms conformance of this function to [56C], since no testing is available independent of a full key establishment scheme. This KDF is approved for use within an approved key establishment scheme but the CMVP does not currently provide CAVP component testing or vendor affirmation of this component.

As defined in [90A] Table 2, the SHA-256 Hash_DRBG requires 256 bits of entropy. If ported to an Intel operational environment, the Module provides the [140IG] 7.14 1(b) option to use a hardware NDRNG to supply all entropy necessary to instantiate Hash_DRBG. In this case, the Module collects 2048 bits of entropy input with 0.5 bits of entropy per one-bit sample, yielding 1024 bits of effective entropy. If the Intel NDRNG feature is not present or disabled during the build process, the caller is required to define a callback function to provide the required entropy. For this case, a caveat is required per [140IG] 7.14

Entropy Caveats; following the example of [140IG] G.13 *Instructions for Validation Information Formatting*:

> *When entropy is externally loaded, no assurance of the minimum strength of generated keys.*

The caller may optionally supply a nonce; if no nonce is supplied by the caller, a 1024-bit value obtained from the NDRNG is used for the nonce. The nonce and entropy input are provided to the [90A] hash derivation function as part of DRBG instantiation.

Seeds used for asymmetric key generation are the unmodified output of from the approved DRBG.

Table 5 – Allowed Functions

	Description / Usage
HW NDRNG	Hardware RNG (the Intel RDSEED function) when available.

5 Modes of Operation, Security Rules and Guidance

The Module supports a FIPS Approved mode of operation and a non-FIPS Approved mode of operation and conforms to [140IG] 1.2 and 1.19 *non-Approved Mode of Operation*. FIPS Approved algorithms are listed in Table 4.

The conditions for using the Module in the [140] Approved mode of operation are:

1. The Module is a cryptographic library and it is intended to be used with a calling application. The calling application is responsible for the usage of the primitives in the correct sequence.
2. The keys used by the Module for cryptographic purposes are determined by the calling application. The calling application is required to provide keys in accordance with [140D].
3. With the Module installed and configured in accordance with [UG] instructions, only the algorithms listed in Table 4 are available. The module is in the Approved mode if the following conditions for algorithm use are met.
    a. Adherence to [140IG] A.13 *SP 800-67rev1 Transition*. The calling process shall limit encryption with a Triple-DES key used in a recognized IETF protocol to 2^{20} 64-bit blocks of data. The calling application shall limit encryption with a Triple-DES key used in any other scenario to 2^{16} blocks of data.
    b. Adherence to [140IG] A.5 *Key/IV Pair Uniqueness Requirements from SP 800-38D*. The Module supports both internal IV generation (for use with the [56A] compliant KAS API entry points) and external IV generation (for TLS KAS usage). For internal IV generation, A.5 requires the calling application to use the internal hardware NDRNG to seed the Hash_DRBG. For external IV generation, the Module complies with A.5 1 (a), tested per option (ii) under A.5 **TLS protocol IV generation**.
    c. ECDSA and RSA signature generation must be used with a SHA-2 or SHA-3 hash function. In accordance with [131], use of SHA-1 for signature generation is disallowed, except where specifically allowed by NIST protocol-specific guidance. Otherwise, signature generation using SHA-1 for digital signature generation places the Module in the non-Approved mode.
    d. RSA signature generation and encryption primitives must use RSA keys with k = 2048, 3072 or 4096 bits or greater.
    e. The calling process shall adhere to all current [131A] algorithm usage restrictions.

6 Critical Security Parameters

All CSPs and public keys used by the Module are described in this section. The list of CSPs and public keys are arranged for consistency with Table 8 – *CSP and Public Key Access Rights within Services*, which is organized for reviewer convenience.

Table 6 - Critical Security Parameters (CSPs)

CSP	Description / Usage
DS_SGK	Private component of an RSA key pair (k = 2048, 3072 or 4096) * or ECC key pair (P-224, P-256, P-384, P-521)† for signature generation
GKP_Private	Private component of a general-purpose key pair generated by the Module, with use determined by the caller. May be RSA (k =2048, 3072 or 4096) * or ECC key pair (P-224, P-256, P-384, P-521)‡.
KAS_Private	Private component of an FFC (L = 2048 N = 256) † or ECC (P-256, P-384, P-521) key pair provided by the local participant, used for Diffie-Hellman shared secret generation.
KAS_SS	The Diffie-Hellman ([56A] Section 5.7.1.1 FFC DH or [56A] Section 5.7.1.2 ECC CDH) shared secret. FFC: 112-bit security strength†. ECC: (security strength between 128-bits and 256-bits)‡.
KD_DKM	Key Derivation derived keying material. The separation into specific keys is done outside the scope of the module but must be conformant to [56C].
KH_Key	Keyed Hash key. May be 128-bit, 192-bit or 256-bit for use with CMAC or GMAC; or 160-bit, 256-bit or 512-bit for use with HMAC.
KTS_KDK	Private component of an RSA key pair (k = 2048 or greater) used for RSA key transport*.
KTS_SS	The RSA key transport shared secret (112-bit and 128-bit security strength).
RBG_Seed	Entropy input (see the [140IG] 7.14 statement above) and nonce.
RBG_State	Hash_DRBG (SHA-256) state V (440-bit) and C (440-bit).
SC_EDK	AES (128-bit, 192-bit or 256-bit) or Triple-DES (192-bit 3-Key, 112-bit equivalent strength) key used for symmetric encryption (including AES authenticated encryption).

Table 7 - Public Keys

Public Key	Description / Usage
DS_SVK	Public component of an RSA key pair (k = 1024, 2048, 3072, or 4096)* or ECC key pair (P-224, P-256, P-384, or P-521)‡ for signature verification.
GKP_Public	Public component of a general-purpose key pair generated by the Module, with use determined by the caller. May be RSA (k =2048, 3072 or 4096)* or ECC key pair (P-224, P-256, P-384, P-521)‡.
KAS_Public	Public component of an FFC (L = 2048 N = 256) † or ECC (P-256, P-384, P-521) key pair received from the remote participant, used for Diffie-Hellman shared secret generation.
KTS_KEK	Public component of an RSA key pair (k = 2048 or greater) used for RSA key transport*.

* For RSA key pairs, equivalent strength is taken from [57P1] Table 2: k = 1024-bits (80-bits; signature verification only); k = 2048 (112-bits); k = 3072 (128-bits). [57P1] defines k as the size of modulus n.

† For DH key pairs, L = 2048 N = 256 is equivalent to 112-bits of security strength.

‡ For ECC key pairs, P-224, P-256, P-384, P-521 curves correspond to 112-bits, 128-bits, 192-bits and 256-bits of security strength, respectively.

7 Roles, Services, and Authentication

The Module supports two distinct operator roles, User and Cryptographic Officer (CO), and does not support multiple concurrent operators, a maintenance role or bypass capability. The cryptographic module does not provide an authentication or identification method of its own. The CO and the User roles are implicitly identified by the service requested.

All services implemented by the Module are listed in the tables below with a description of service CSP access. The calling application may use the wolfCrypt_GetStatus_fips() API to determine the current status of the Module. A return code of 0 means the Module is in a state without errors. Any other return code is the specific error state of the Module. See [UG] for additional information on the cryptographic services listed in this section. Keys are provided to the Module by the calling application; manual key entry is not supported. Data output is inhibited during self-tests, zeroization, and error states.

Table 8 – Authorized Services available in FIPS mode

Service	Description	Role
Digital signature	Generate or verify ECDSA or RSA digital signatures.	User
Generate key pair	Generate asymmetric (ECDSA or RSA) key pairs.	User
Key agreement	Primitives used for DH key agreement on behalf of the application. The DH or EC DH keys are passed in by the calling application.	User
Key derivation	Derive keying material from a shared secret.	User
Keyed hash	Generate or verify data integrity with CMAC, GMAC or HMAC.	User
Key transport	Key transport primitives (RSAEP, RSADP) used to encrypt or decrypt keying material on behalf of the caller.	User
Message digest	Generate a message digest.	User
Random	Generate random bits using the DRBG.	User
Self-test	Run power-on self-test.	User
Show status	Provide Module status.	User
Symmetric cipher	Encrypt and decrypt data (including authenticated encrypt and decrypt).	User
Zeroize	Functions that destroy CSPs. FreeRng_fips destroys RNG CSPs. All other services automatically overwrite memory bound CSPs. Cleanup of the stack is the duty of the application. Restarting the general-purpose computer clears all CSPs in RAM.	CO

Table 9 – CSP and Public Key Access Rights within Services describes Module service access to CSPs/cryptographic keys. In each cell below, the following annotations indicate the type of access by the Module service:

- E (Execute): The service uses a CSP or public key provided by the calling application as positional parameter on the stack; the calling application owns the stack; the Module zeroizes all local copies of a CSP before returning.
- G (Generate): The Module generates or derives the cryptographic keys/CSPs internally. The Module does not retain copies of the key after call completion.
- I (Input): The Module receives the CSP on the stack.
- O (Output): The Module outputs a CSP/cryptographic key to the calling application through the logical interface. The Module does not output CSPs through a physical port.
- Z (Zeroize): The Module zeroizes the CSP.

Table 9 – CSP and Public Key Access Rights within Services

Services	DS_SGK	DS_SVK	GKP_Private	GKP_Public	KAS_Private	KAS_Public	KAS_SS	KD_DKM	KH_Key	KTS_KDK	KTS_KEK	KTS_SS	RBG_Seed	RBG State	SC_EDK
Digital signature	EI	EI	--	--	--	--	--	--	--	--	--	--	--	--	--
Generate key pair	--	--	GO	GO	--	--	--	--	--	--	--	--	--	--	--
Key agreement	--	--	--	--	EI	EI	GO	--	--	--	--	--	--	--	--
Key derivation	--	--	--	--	--	--	EI	GO	--	--	--	EI	--	--	--
Keyed hash	--	--	--	--	--	--	--	--	EI	--	--	--	--	--	--
Key transport	--	--	--	--	--	--	--	--	--	EI	EI	IO	--	--	--
Message digest	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Random	--	--	--	--	--	--	--	--	--	--	--	--	EI	EG	--
Self-test	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Show status	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Symmetric cipher	--	--	--	--	--	--	--	--	--	--	--	--	--	--	EI
Zeroize	Z	--	Z	--	Z	--	Z	--	Z	Z	--	Z	Z	Z	Z

Note that the caller provides the KAS_Private and KAS_Public keys for shared secret computation; the caller's exchange and assurance of public keys with the remote participant is outside the scope of the Module.

8 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. The Module provides a default entry point to automatically run the power on self-tests compliant with [140IG] 9.10 *Power-Up Tests for Software Module Libraries*. Power on self–tests are available on demand by reloading the Module.

On instantiation, the Module performs the self-tests described in Table 8. All KATs must complete 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 error state is persistent and no services are available. All attempts to use the Module's services result in the return of a non-zero error code, FIPS_NOT_ALLOWED_E (-197). To recover from an error state, reload the Module into memory.

Table 10 - Power-on Self-tests

Test Target	Description
Software Integrity	HMAC-SHA-256 with a 256-bit key.
AES	Separate encryption and decryption KATs, CBC mode, 128-bit key.
AES GCM	Separate authenticated encryption and decryption KATs, GCM mode, 128-bit key.
DRBG	KAT for the HASH_DRBG using SHA-256.
ECDSA	Performs an ECDSA PCT using the P-256 curve.
HMAC	HMAC-SHA-1 (160-bit key), HMAC-SHA-512 (512-bit key), and HMAC-SHA3-256 (256-bit key) KATs.
KAS ECC	[56A] Section 5.7.1.2 primitive "Z" computation KAT (per [140IG] 9.6), using P-256.
KAS FFC	[56A] Section 5.7.1.1 primitive "Z" computation KAT (per [140IG] 9.6), using L = 2048 N = 256.
RSA	Separate signature generation and signature verification KATs (k = 2048), inclusive of the embedded SHA-256, RSADP and RSAEP (KTS) self-tests.
Triple-DES	Separate encryption and decryption KATs, TCBC mode, 3-Key.

HMAC and RSA KATs include the embedded SHA self-tests per [140IG] 9.1, 9.2, and A.11.

Table 11 - Conditional Self-tests

Test Target	Description
DRBG	[90A] Section 11.3 Instantiate, Generate, Reseed health tests for SHA-256 Hash_DRBG.
NDRNG	CRNGT of 64 bit blocks on the output of the NDRNG when available.
ECC PCT	ECC Key Generation Pairwise Consistency Test, performed on ECC key pair generation.
FFC PCT	FFC Key Generation Pairwise Consistency Test, performed on FFC key pair generation.
RSA PCT	RSA Key Generation Pairwise Consistency Test, performed on RSA key pair generation.

PCTs are performed in accordance with [140IG] 9.9 *Pair-Wise Consistency Self-Test When Generating a Key Pair*. Per IG 9.8, the continuous RNG test is not required for [90A] compliant DRBG output.

9 References, Definitions and Source Files

Table 12 – References

Ref	Filename / Title / Description	Date
[140]	FIPS 140-2, *Security Requirements for Cryptographic Modules*	12/3/2002
[140IG]	FIPS 140-2 *Implementation Guidance*	3/27/2018
[180]	FIPS 180-4, *Secure Hash Standard (SHS)*	8/4/2015
[186]	FIPS 186-4, *Digital Signature Standard (DSS)*	7/19/2013
[197]	FIPS 197, *Advanced Encryption Standard (AES)*	7/19/2013
[198]	FIPS 198-1, *The Keyed Hash Message Authentication Code (HMAC)*	7/16/2008
[202]	FIPS 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*	8/4/2015
[38A]	SP 800-38A, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*	12/1/2001
[38C]	SP 800-38C, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*	7/20/2007
[38D]	SP 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*	11/28/2007
[56A]	SP 800-56A, Rev. 3, *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*	4/16/2018
[56B]	SP 800-56B, *Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography*	10/1/2014
[56C]	SP 800-56C, *Recommendation for Key-Derivation through Extraction-then-Expansion*	11/2011
[57P1]	SP 800-57 Part 1 Revision 4, *Recommendation for Key Management Part 1: General*	1/2016
[67]	SP 800-67 Rev. 2, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*	7/18/2017
[90A]	SP 800-90Ar1, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Revision 1*	6/24/2015
[131A]	SP 800-131A Rev. 1, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*	11/6/2015
[UG]	wolfCrypt FIPS User's Guide	

Table 13 - Acronyms and Definitions

Acronym	Definition
AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Programming Interface
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher-Block Chaining
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CSP	Critical Security Parameter
CTR	Counter-mode
CVL	Component Validation List
DES	Data Encryption Standard
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECC-CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
HMAC	Keyed-Hash Message Authentication Code
IG	Implementation Guidance
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KTS	Key Transport Scheme
LTS	Long Term Support
NDRNG	Non-deterministic Random Number Generator
NIST	National Institute of Standards and Technology
PAA	Processor Algorithm Accelerators
PCT	Pair-wise Consistency Test
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman Algorithm
RSADP	RSA Decryption Primitive
RSAEP	RSA Encryption Primitive
TCBC	TDEA Cipher-Block Chaining
TDEA	Triple Data Encryption Algorithm
TDES	Triple Data Encryption Standard
TLS	Transport Layer Security
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

The source code files listed in Table 14 create the corresponding object files that comprise the codeEncrypt module boundary on each supported operating environment; the extensions of the object file can differ across the environments.

Table 14 - Source Files

Source File Name	Description
aes.c	AES algorithm
aes_asm.s	AES assembler optimizations (Linux, AT&T style)
aes_asm.asm	AES assembler optimizations (Windows 10, Intel style)
cmac.c	CMAC algorithm
des3.c	TDES algorithm
dh.c	Diffie-Hellman
ecc.c	Elliptic curve cryptography
fips.c	FIPS entry point and API wrappers
fips_test.c	Power on Self Tests
hmac.c	HMAC algorithm
random.c	DRBG algorithm
rsa.c	RSA algorithm
sha.c	SHA algorithm
sha256.c	SHA-256 algorithm
sha3.c	SHA-3 algorithm
sha512.c	SHA-512 algorithm
wolfcrypt_first.c	First FIPS function and Read Only address
wolfcrypt_last.c	Last FIPS function and Read Only address