wolfSSL FIPS Ready User Guide

2024-12-06

Contents

1 wolfSSL FIPS Ready

Do you have a project that will need a FIPS approved cryptographic library at a later date, but want to be ready for it now? wolfSSL FIPS Ready includes the cryptography layer from the public wolfSSL source tree, along with the FIPS tooling enabled for a FIPS Ready build. FIPS Ready allows application developers to assure that they have correctly integrated the necessary FIPS setup, and are correctly using the API and underlying protocols, providing for a seamless transition to the full FIPS certified wolfSSL library for an application. When the time comes, we will shepherd your operating environment through testing and add it to the wolfCrypt FIPS 140-3 certificate. In special situations, a new FIPS 140-3 certificate will be required, but we are ready to provide guidance through the testing and certification process. In both scenarios, FIPS Ready assures that your application will be fully functional with wolfCrypt as certified.

FIPS Ready is open source and dual-licensed. wolfSSL Inc distributes FIPS Ready wolfSSL with the GPLv3 license or can negotiate commercial licensing terms with support if needed beyond the proof of concept phase.

FIPS is a complicated topic. If you have questions after reviewing this document, then just contact us at facts -at- wolfssl *dot* calm. The email address is obfuscated for the spiders, but the calm is for the calm you'll feel when you are FIPS Ready.

1.1 What is Different from a non-FIPS build of wolfSSL?

The wolfCrypt FIPS API provides wrappers for all the approved algorithm functions that are within the FIPS boundary. The FIPS wrappers can be called directly, or you can keep calling the original APIs; at compile time the API is swapped by the headers so that the FIPS wrappers will be called either way. The FIPS wrapper functions check the status of the internal self-testing before calling the actual function. If the CAST for that specific algorithm has not been run at least one time previously it will run the first time the algorithm is used. Users now have the option to either pre-emptively test algorithms at start up if they wish to avoid the test running at a later time or they can just let the test run when the algorithm is used.

The wolfCrypt FIPS 140-3 ready code has a required power-on self-test (POST) that automatically checks the integrity of the executable in memory, this has not changed since 140-2, only the known answer tests (KATs) for algorithms not used by the POST are now conditional upon use. The executable is organized so the code in the FIPS boundary is contiguous in memory. When the application using the FIPS code starts up, or the shared library is loaded, the default entry point of the library is called, and the POST runs automatically. It has two major parts: the in-core memory check and the known answer tests (KAT) for the algo used by the POST (HMAC-SHA256).

The POST for HMAC-SHA256 is performed first followed by the in-core memory test. The code in memory is hashed with HMAC-SHA256. If the hash matches, the test progresses. Otherwise the FIPS module is placed in an error state and all calls into the module will fail. In this case, the FIPS source code must be updated with the correct integrity hash, and only after recompilation can the module successfully initialize.

All other algorithms in the FIPS boundary are tested with canned data the first time they are used or optionally whenever the developer wants them to run. The output is compared to pre-computed known answers. The test values are all inside the boundary and are checked at the time they are called. Several of the tests have a random component, for example a sign and verify, so a known piece of data is signed and then verified with a canned key. The key generation is tested in a similar fashion.

The FIPS boundary provides confirmation about multiple aspects of your application. The processing of private key access that will assure proper unlocking and relocking of keys according to the FIPS specification. The assurance that no FIPS-forbidden modes or key sizes are being used, along with proper entropy source setup. FIPS Ready also helps discover conflicts with outside sub-system inte-

gration your application needs, for example determining if other applications or devices on a network support your new FIPS Ready cipher suites for communication between each other.

2 Building wolfSSL as FIPS Ready

Once you have a copy of the source code unarchived into a directory, building is similar to normal wolfSSL, but with extra steps.

The following steps assume you are on a Linux or macOS box and are using the GPLv3 distribution of wolfSSL FIPS Ready to make a shared library to be installed into the system.

2.1 Unarchive the source

```
$ tar xzvf wolfssl-5.6.4-gplv3-fips-ready.tar.gz
```

This unarchives the source into the directory wolfssl-5.6.4-gplv3-fips-ready. Change into this directory. If you received a commercial release, replace gplv3 with commercial and .tar.gz with .7z and tar xzvf with 7z x -p found in the distribution email.

2.2 Configure the build.

```
$ ./configure --enable-fips=ready
```

This command configures the Makefile to build wolfSSL for FIPS Ready.

2.3 Make the library.

```
$ make
```

This compiles all the sources and links together the library. It also builds the example tools and testing tools.

2.4 Update the in-core memory hash.

```
$ ./fips-hash.sh
```

```
$ make # Re-build once the hash has been updated
```

This step is where the hash for the in-core memory test is calculated and will need to be updated. The wolfCrypt test should fail when called by the fips-hash.sh script and if you were to echo out the output the following message would be observed (NOTE the hash value will be unique):

```
in my Fips callback, ok = 0, err = -203
```

```
message = In Core Integrity check FIPS error
```

```
hash = 8D29242F610EAEA179605BB1A99974EBC72B0ECDB26B483B226A729F36FC82A2
```

```
In core integrity hash check failure, copy above hash
```

```
into verifyCore[] in fips_test.c and rebuild
```

Should you add other options to the build, this may change the hash value and this step will need to be repeated. Also modifications to the application may result in the fips boundary shifting in memory when the application is re-compiled. The hash changing when only the application is updated is not an indication of the module being effected, only shifted in place in memory. This is expected if compiling a static library and application. Shared objects tend to not experience this issue.

4.1 If one were to do the above without using the provided fips-hash.sh/span>script one could either edit the file wolfcrypt/src/fips_test.c and update the hash manually or use a configure like so:

```
$ ./configure --enable-fips=ready CFLAGS”=-DWOLFCRYPT_FIPS_CORE_HASH_VALUE=8
   ”D29242F610EAEA179605BB1A99974EBC72B0ECDB26B483B226A729F36FC82A2
```

4.2 Make the library again.

2.5 Test the build.

```
$ make check
```

The check target in the Makefile will run all the test tools and scripts we provide with wolfSSL and wolfCrypt. If everything is OK, you should see the following output:

```
PASS: scripts/resume.test

PASS: scripts/external.test

PASS: scripts/google.test

PASS: testsuite/testsuite.test

PASS: scripts/openssl.test

PASS: tests/unit.test

============================================================================

Testsuite summary for wolfssl 4.0.0

============================================================================

# TOTAL: 6

# PASS:  6

# SKIP:  0

# XFAIL: 0

# FAIL:  0

# XPASS: 0

# ERROR: 0

============================================================================
```

2.6 Install the library and headers.

```
$ make install
```

The install target in the Makefile will install all the headers and the library into your system By default, this is into the directory /usr/local.

At this point, wolfSSL FIPS Ready should be ready to be used in your application builds.

3 What has changed from the original FIPS 140-2 ready?

1. One now needs to call wc_SetSeed_Cb at the application level when running in FIPS mode.

a. +#ifdef WC_RNG_SEED_CB

b. • wc_SetSeed_Cb(wc_GenerateSeed);

c. +#endif

2. KEY Access Management

a. Users calling wolfSSL (SSL/TLS) APIs' do not need to worry about the KEY Access Management item, however for those calling crypto APIs please see next steps

b. Users invoking wolfcrypt (wc_XXX) APIs' directly that involve loading or using a private key must manage the key access at the application level. To be able to read in or use a private key the application must allow this by calling PRIVATE_KEY_UNLOCK(); prior to reading a key or using a key. When finished the application must** lock the key access again before terminating by calling PRIVATE_KEY_LOCK();

```
i.  The PRIVATE_KEY_UNLOCK and PRIVATE_KEY_LOCK can optionally be invoked
only once on startup and once on shutdown …or

ii.  If the application wishes to be very strict, these can be called
immediately before and after each call that involves a private key load
or use.

** “application must lock again before ”terminating - This is a
documentation requirement, this is not enforced at run-time by an error
or prevention from exiting. Failing to re-locking the key before exiting
makes the application “not FIPS ”compliant.
```

c. To support an application that can link to both a wolfSSL FIPS library version and a wolfSSL non-FIPS library version users can implement NO-OP versions for the non-FIPS cases like so:

```
#if !defined(PRIVATE_KEY_LOCK) && !defined(PRIVATE_KEY_UNLOCK)
    #define PRIVATE_KEY_LOCK() do {} while (0)
    #define PRIVATE_KEY_UNLOCK() do {} while (0)
#endif
```

4 wolfCrypt FIPS Ready API Documentation

The following is a summary of the wolfCrypt FIPS Ready API. Please see the wolfCrypt API documentation for more detail.

4.1 Map of API to FIPS 140-3 module services

4.1.1 Digital Signature Service

API Call	Description
InitRsaKey_fips	Initializes RSA key object for use with optional heap hint p. Returns 0 on success, < 0 on error.
FreeRsaKey_fips	Releases RSA key resources. Returns 0 on success, < 0 on error.
RsaSSL_Sign_fips	Performs RSA key Signing operation on input in of size inLen, outputting to out of size outLen using rng. Returns 0 on success, < 0 on error.
RsaSSL_VerifyInline_fips	Performs RSA key Verification without allocating temporary memory on input in of size inLen, writes to output out. Returns 0 on success, < 0 on error.
RsaSSL_Verify_fips	Performs RSA key Verification on input in of size inLen, writes to output out of size outLen. Returns 0 on success, < 0 on error.
SS_Sign_fips	Performs RSA key Signing operation with PSS padding on input in of size inLen, outputting to out of size outLen using rng. It uses the hash algorithm hash with the mask generation function mgf. Returns 0 on success, < 0 on error.
RsaPSS_SignEx_fips	Performs RSA key Signing operation with PSS padding on input in of size inLen, outputting to out of size outLen using rng. It uses the hash algorithm hash with the mask generation function mgf and a salt length of saltLen. Returns 0 on success, < 0 on error.
RsaPSS_VerifyInline_fips	Performs RSA key Verification without allocating temporary memory on input in of size inLen, writes to output out. It uses the hash algorithm hash with the mask generation function mgf. Returns 0 on success, < 0 on error.
RsaPSS_VerifyInlineEx_fips	Performs RSA key Verification on input in of size inLen, writes to output out of size outLen. It uses the hash algorithm hashwith the mask generation function mgf and a salt length of saltLen. Returns 0 on success, < 0 on error.

API Call	Description
RsaPSS_Verify_fips	Performs RSA key Verification on input in of size inLen, writes to output out of size outLen. It uses the hash algorithm hashwith the mask generation function mgf. Returns 0 on success, < 0 on error.
RsaPSS_VerifyEx_fips	Performs RSA key Verification on input in of size inLen, writes to output out of size outLen. It uses the hash algorithm hashwith the mask generation function mgf and a salt length of saltLen. Returns 0 on success, < 0 on error.
RsaPSS_CheckPadding_fips	Checks the padding after RSA key verification on input in of size inSz with signature sig of size sigSz using hash hashType. Returns 0 on success, < 0 on error.
RsaPSS_CheckPaddingEx_fips	Checks the padding after RSA key verification on input in of size inSz with signature sig of size sigSz using hash hashTypeand a salt length of saltLen. Returns 0 on success, < 0 on error.
RsaEncryptSize_fips	Retrieves RSA key Output Size. Returns key output size > 0 on success, < 0 on error.
wc_RsaPrivateKeyDecode	Decodes an Rsa Private key from a buffer input starting at index inOutIdx of size inSz. Returns 0 on success, < 0 on error.
wc_RsaPublicKeyDecode	Decodes an Rsa Public key from a buffer input starting at index inOutIdx of size inSz. Returns 0 on success, < 0 on error.
ecc_init_fips	Initializes ECC key object for use. Returns 0 on success, < 0 on error.
ecc_free_fips	Releases ECC key object resources. Returns 0 on success, < 0 on error.
ecc_import_x963_fips	Imports the ECC public key in ANSI X9.63 format from in of size inLen. Returns 0 on success, < 0 on error.
ecc_sign_hash_fips	Performs ECC key Signing operation on in of length inlen and output to out of length outlen using rng. Returns 0 on success, < 0 on error.
ecc_verify_hash_fips	Performs ECC key Verification of sig of size siglen, with hash of length hashlen. The signature verification result is returned in res. Returns 0 on success, < 0 on error.

API Call	Description

4.1.2 Generate Key Pair Service

API Call	Description
MakeRsaKey_fips	Generates an RSA key with modulus length size and exponent e using the random number generator rng. Returns 0 on success, < 0 on error.
CheckProbablePrime_fips	For a potential modulus of length nlen, check the candidate numbers pRaw of size pRawSz and qRaw of size qRawSz to see if they are probably prime. They should both have a GCD with the exponent eRaw of size eRawSz of 1. The prime candidates are checked with Miller-Rabin. The result is written to isPrime. Returns 0 on success, < 0 on error.
RsaExportKey_fips	Exports the RSA key as its components e of eSz, n of nSz, d of dSz, p of pSz, q of qSz. The sizes should be the sizes of the buffers, and are updated to the actual length of number. Returns 0 on success, < 0 on error.
ecc_make_key_fips	Performs the ECC Key Generation operation on key of size keysize using rng. Returns 0 on success, < 0 on error.
ecc_make_key_ex_fips	Performs the ECC Key Generation operation on key of size keysize with elliptic curve curve_id using rng. Returns 0 on success, < 0 on error.
ecc_export_x963_fips	Exports the ECC public key in ANSI X9.63 format to out of size outLen. Returns 0 on success, < 0 on error.
InitDhKey_fips	Initializes DH key object for use. Returns 0 on success, < 0 on error.
FreeDhKey_fips	Releases DH key resources. Returns 0 on success, < 0 on error.
DhSetKeyEx_fips	Sets the group parameters for the DH key from the unsigned binary inputs p of size pSz, q of size qSz, and g of size gSz. Returns 0 on success, < 0 on error.
DhGenerateKeyPair_fips	Generates the public part pub of size pubSz, private part priv of size privSz using rng for DH key. Returns 0 on success, < 0 on error.

API Call	Description
CheckRsaKey_fips	Performs a pair-wise key validation on key. Returns 0 on success, < 0 on error.
ecc_check_key_fips	Performs a pair-wise key validation on key. Returns 0 on success, < 0 on error.
DhCheckPubKeyEx_fips	Performs a mathematical key validation on the pub value of size pubSz using the domain parameters in key or against the prime value of size primeSz.
DhCheckPrivKeyEx_fips	Performs a mathematical key validation on the priv value of size privSz using the domain parameters in key or against the prime value of size primeSz.
DhCheckKeyPair_fips	Performs a pair-wise key validation between the pub value of size pubSz and the priv value of size privSz using domain parameters key. Returns 0 on success, < 0 on error.
HKDF_fips	Performs HMAC based Key Derivation Function using a hash of type and inKey of size inKeySz, with a salt of length saltSz and info of infoSz. The key is written to out of size outSz. Returns 0 on success, < 0 on error.

4.1.3 Key Agreement Service

API Call	Description
ecc_shared_secret_fips	Performs ECDHE Key Agreement operation with privKey and the peer's pubKey and storing the result in sharedSecret of length sharedSz. Returns 0 on success, < 0 on error.
DhAgree_fips	Creates the agreement agree of size agreeSz using DH keyprivate priv of size privSz and peer's public key otherPub of size pubSz. Returns 0 on success, < 0 on error.

4.1.4 Key Transport Service

API Call	Description
RsaPublicEncrypt_fips	Performs RSA key Public Encryption on input in of size inLen, writes to output out of size outLen using rng. Returns 0 on success, < 0 on error.

API Call	Description
RsaPublicEncryptEx_fips	Performs RSA key Public Encryption on input in of size inLen, writes to output out of size outLen using rng. It uses padding of type. If using PSS padding, it uses hash and mgf, with label of size labelSz. Returns 0 on success, < 0 on error.
RsaPrivateDecryptInline_fips	Performs RSA key Private Decryption without allocating temporary memory on input in of size inLen, writes to output out. Returns 0 on success, < 0 on error.
RsaPrivateDecryptInlineEx_fips	Performs RSA key Private Decryption without allocating temporary memory on input in of size inLen, writes to output out. It uses padding of type. If using PSS padding, it uses hashand mgf, with label of size labelSz. Returns 0 on success, < 0 on error.
RsaPrivateDecrypt_fips	Performs Rsa key Private Decryption on input in of size inLen, writes to output out of size outLen. Returns 0 on success, < 0 on error.
RsaPrivateDecryptEx_fips	Performs Rsa key Private Decryption on input in of size inLen, writes to output out of size outLen. It uses padding of type. If using PSS padding, it uses hash and mgf, with label of size labelSz. Returns 0 on success, < 0 on error.

4.1.5 Keyed Hash Service

API Call	Description
HmacSetKey_fips	Initializes hmac object with key of size keySz using the hash type. Returns 0 on success, < 0 on error.
HmacUpdate_fips	Performs hmac Update on input data of size len. Returns 0 on success, < 0 on error.
HmacFinal_fips	Performs hmac Final, outputs digest to hash. Returns 0 on success, < 0 on error.
Gmac_fips	Performs GMAC on input authIn of size authInSz and outputs authTag of size authTagSz. Uses key of length keySz and randomly generates an IV of length ivSz stored in iv using random number generator rng. GMAC Returns 0 on success, < 0 on error.

API Call	Description
GmacVerify_fips	Verifies GMAC authTag of length authTagSz on input authIn of size authInSz using the key of length keySz and the iv of length ivSz. Returns 0 on success, < 0 on error.
InitCmac_fips	Initializes cmac object with key of size keySz using the hash type. Returns 0 on success, < 0 on error.
CmacUpdate_fips	Performs cmac Update on input in of size inSz. Returns 0 on success, < 0 on error.
CmacFinal_fips	Performs cmac Final, outputs digest to out of size outSz, which is updated with the actual output size. Returns 0 on success, < 0 on error.

4.1.6 Message Digest Service

API Call	Description
InitSha_fips	Initializes sha object for use. Returns 0 on success, < 0 on error.
ShaUpdate_fips	Performs sha Update on input data of size len. Returns 0 on success, < 0 on error.
ShaFinal_fips	Performs sha Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha224_fips	Initializes sha224 object for use. Returns 0 on success, < 0 on error.
Sha224Update_fips	Performs sha224 Update on input data of size len. Returns 0 on success, < 0 on error.
Sha224Final_fips	Performs sha224 Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha256_fips	Initializes sha256 object for use. Returns 0 on success, < 0 on error.
Sha256Update_fips	Performs sha256 Update on input data of size len. Returns 0 on success, < 0 on error.
Sha256Final_fips	Performs sha256 Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha384_fips	Initializes sha384 object for use. Returns 0 on success, < 0 on error.

API Call	Description
Sha384Update_fips	Performs sha384 Update on input data of size len. Returns 0 on success, < 0 on error.
Sha384Final_fips	Performs sha384 Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha512_fips	Initializes sha512 object for use. Returns 0 on success, < 0 on error.
Sha512Update_fips	Performs sha512 Update on input data of size len. Returns 0 on success, < 0 on error.
Sha512Final_fips	Performs sha512 Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha3_224_fips	Initializes sha3 (224-bit) object for use. Returns 0 on success, < 0 on error.
Sha3_224_Update_fips	Performs sha3 (224-bit) Update on input data of size len. Returns 0 on success, < 0 on error.
Sha3_224_Final_fips	Performs sha3 (224-bit) Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha3_256_fips	Initializes sha3 (256-bit) object for use. Returns 0 on success, < 0 on error.
Sha3_256_Update_fips	Performs sha3 (256-bit) Update on input data of size len. Returns 0 on success, < 0 on error.
Sha3_256_Final_fips	Performs sha3 (256-bit) Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha3_384_fips	Initializes sha3 (384-bit) object for use. Returns 0 on success, < 0 on error.
Sha3_384_Update_fips	Performs sha3 (384-bit) Update on input data of size len. Returns 0 on success, < 0 on error.
Sha3_384_Final_fips	Performs sha3 (384-bit) Final, outputs digest to hash. Returns 0 on success, < 0 on error.
InitSha3_512_fips	Initializes sha3 (512-bit) object for use. Returns 0 on success, < 0 on error.
Sha3_512_Update_fips	Performs sha3 (512-bit) Update on input data of size len. Returns 0 on success, < 0 on error.
Sha3_512_Update_fips	Performs sha3 (512-bit) Update on input data of size len. Returns 0 on success, < 0 on error.

API Call	Description
Sha3_512_Final_fips	Performs sha3 (512-bit) Final, outputs digest to hash. Returns 0 on success, < 0 on error.

4.1.7 Random (Number Generation) Service

API Call	Description
InitRng_fips	Initializes RNG object for use. Returns 0 on success, < 0 on error.
InitRngNonce_fips	Initializes RNG object for use with a nonce of size nonceSz. Returns 0 on success, < 0 on error.
FreeRng_fips	Releases RNG resources and zeros out state. Returns 0 on success, < 0 on error. Also part of Zeroize Service.
RNG_GenerateBlock_fips	Retrieves block of RNG output for user into buf of size in bytes bufSz. Returns 0 on success, < 0 on error.
RNG_HealthTest_fips	When reseed is 0, tests the output of a temporary instance of an RNG against the expected output of size in bytes outputSzusing the seed buffer entropyA of size in bytes entropyASz, where entropyB and entropyBSz are ignored. When reseed is 1, the test also reseeds the temporary instance of the RNG with the seed buffer entropyB of size in bytes entropyBSz and then tests the RNG against the expected output of size in bytes outputSz. Returns 0 on success, < 0 on error.

4.1.8 Show Status Service

API Call	Description
wolfCrypt_GetStatus_fips	Returns 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.
wolfCrypt_GetVersion_fips	Returns a pointer to the null-terminated char string of the wolfCrypt library version.
wolfCrypt_GetCoreHash_fips	Returns a pointer to the null-terminated char string of the core hash in hex.

4.1.9 Symmetric Cipher Service

API Call	Description
AesSetKey_fips	Initializes aes object with userKey of length keylen, dir indicates the direction while iv is optional. Returns 0 on success, < 0 on error.
AesSetIV_fips	Initializes aes object with user iv. Returns 0 on success, < 0 on error.
AesCbcEncrypt_fips	Performs aes CBC Encryption on input in to output out of size sz. Returns 0 on success, < 0 on error.
AesCbcDecrypt_fips	Performs aes CBC Decryption on input in to output out of size sz. Returns 0 on success, < 0 on error.
AesEcbEncrypt_fips	Performs aes ECB Encrypt on input in to output out of size sz. Returns 0 on success, < 0 on error.
AesEcbDecrypt_fips	Performs aes ECB Encryption on input in to output out of size sz. Returns 0 on success, < 0 on error.
AesCtrEncrypt_fips	Performs aes CTR Encryption on input in to output out of size sz. Returns 0 on success, < 0 on error. This API also performs CTR Decryption.
AesGcmSetKey_fips	Initializes aes object with key of length len. Returns 0 on success, < 0 on error.
AesGcmSetExtIV_fips	Initializes aes object with an externally generated iv of length ivSz. Returns 0 on success, < 0 on error.
AesGcmSetIV_fips	Initializes aes object with an internally generated IV of length ivSz using ivFixed as the first ivFixedSz bytes and the remainder being random bytes from rng. Returns 0 on success, < 0 on error.
AesGcmEncrypt_fips	Performs aes GCM Encryption on input in to output out of size sz. The current IV is stored in buffer ivOut of length ivOutSz. The authentication tag is stored in buffer authTag of size authTagSz. authInSz bytes from authIn are calculated into the authentication tag. Returns 0 on success, < 0 on error.
AesGcmDecrypt_fips	Performs aes GCM Decryption on input in to output out of size sz using iv of size ivSz. The authTag of size authTagSz is checked using the input and the authInSz bytes of authIn. Returns 0 on success, < 0 on error.
AesCcmSetKey_fips	Initializes aes object with key of length keySz. Returns 0 on success, < 0 on error.

API Call	Description
AesCcmSetNonce_fips	Initializes aes object with an externally generated nonce of length nonceSz. Returns 0 on success, < 0 on error.
AesCcmEncrypt_fips	Performs aes CCM Encryption on input in to output out of size inSz. The current IV is stored in buffer nonce of length nonceSz. The authentication tag is stored in buffer authTag of size authTagSz. authInSz bytes from authIn are calculated into the authentication tag. Returns 0 on success, < 0 on error.
AesCcmDecrypt_fips	Performs aes CCM Decryption on input in to output out of size inSz using nonce of size nonceSz. The authTag of size authTagSz is checked using the input and the authInSz bytes of authIn. Returns 0 on success, < 0 on error.
Des3_SetIV_fips	Initializes des3 object with User iv. Returns 0 on success, < 0 on error.
Des3_CbcEncrypt_fips	Performs des3 Cbc Encryption on input in to output out of size sz. Returns 0 on success, < 0 on error.
Des3_CbcDecrypt_fips	Performs des3 Cbc Decryption on input in to output out of size sz. Returns 0 on success, < 0 on error.

4.1.10 Zeroize Service

API Call	Description
FreeRng_fips	Destroys RNG CSPs. All other services automatically overwrite memory bound CSPs. Returns 0 on success, < 0 on error. Cleanup of the stack is the duty of the application. Restarting the general-purpose computer clears all CSPs in RAM. API Calls for Allowed Security Functions