wolfSSL FIPS Ready

Documentation and Users Guide

March 27, 2019, version 4.0.0

---

wolfSSL FIPS Ready

FIPS認証の暗号ライブラリに将来必要になるかも知れないプロジェクトがあり、今のうちに準備だけしておきたい。 wolfSSL FIPS Readyはまさにそのような場合に必要なものです。このバージョンのwolfSSLは、wolfSSLソースツリーに含まれるFIPS対応のwolfCryptレイヤーのコードですので、FIPSオプションを有効にしてビルドすることができます。ただし、このままでは FIPS証明書を取得したことにはなりませんのでご注意ください。FIPSは認証(FIPS approved)されていませんが、FIPS準備完了 (FIPS Ready)になります。 FIPS Ready版には、FIPS向けソースコードをビルドに組み込み、FIPS向けに強化したしたデフォルトのエントリポイント、およびパワーオンセルフテスト(POST)が含まれています。認証が必要とする時が来れば、すみやかに動作環境をテストすることができ、すべてのコーディング作業が完了します。

FIPS Readyはオープンソースであり、デュアルライセンスです。 製品に組込みの際には商用サポートとライセンスの条件についてご相談ください。

スタンダード版との違い

wolfCrypt FIPS APIは、FIPS境界内にある認証されたすべてのアルゴリズム関数のラッパーを提供します。 FIPSラッパーは直接呼び出すことも、元々のAPI名で呼び出すこともできます。コンパイル時にAPI名はヘッダーによって入れ替えられるため、FIPSラッパーはどちらの方法

でも呼び出すことができます。 FIPSラッパー関数は、実際の関数を呼び出す前に、内部セルフテストのステータスをチェックします。

wolfCrypt FIPSコードには、メモリ内の実行可能ファイルを自動的にチェックするパワーオンセルフテスト（POST）が含まれています。実行可能ファイルでは、FIPS境界内のコードがメモリ内で隣接するように構成されています。 FIPSコードを使用するアプリケーションが起動するか、共有ライブラリがロードされると、ライブラリのデフォルトのエントリポイントが呼び出され、POSTが自動的に実行されます。これには、コア内メモリチェックと既知の回答テスト（KAT：Known Answer Test）の2つの主要な部分が含まれます。

コア内メモリテストが最初に実行されます。メモリ内のコードはHMAC-SHA-256でハッシュされます。ハッシュが一致した場合、テストは続行されます。それ以外の場合、FIPSモジュールはエラー状態になります。

次にKATが実行されます。 FIPS境界内の他のすべてのアルゴリズムは、同梱されたデータでテストされ、出力は事前に計算された既知の回答と比較されます。テスト値はすべてこの境界内にあり、コア内メモリテストでチェックされます。いくつかのテストには、たとえば署名と検証などのランダムなコンポーネントがあります。そのため、既知のデータに署名を付けてから、既定の鍵で検証します。鍵生成は同様の方法でテストされます。

FIPS対応としてwolfSSLをビルドする

ソースコードのコピーをディレクトリにおき、アーカイブを解凍します。ビルドは通常のwolfSSLに似ていますが、追加の手順が必要になります。

次の手順は、LinuxまたはmacOSを使用しており、wolfSSL FIPS ReadyのGPLv3ディストリビューションを使用して、共有ライブラリをシステムにインストールすることを前提としています。

1.ソースファイルアーカイブを解凍します。

$ tar xzvf wolfssl-4.0.0-gpl3-fips-ready.tar.gz

これにより、ソースがディレクトリwolfssl-4.0.0-gpl3-fips-readyに解凍されます。このディレクトリに移動します。商用リリースを受け取った場合は、GPL3を商用版に置き換えます。

2.ビルドを構成します。

$ ./configure --enable-fips = v2

このコマンドは、FIPS対応のwolfSSLをビルドするようにMakefileを構成します。

3.ライブラリを作成します。

$ make

これにより、すべてのソースがコンパイルされ、ライブラリがリンクされます。また、サンプルツールとテストツールもビルドします。

4.コア内メモリハッシュを更新します。

$ ./wolfcrypt/test/testwolfcrypt

このステップでは、コア内メモリテストのハッシュが計算され、更新する必要があります。wolfCryptテストは失敗し、次のメッセージが出力されます（ハッシュ値は一意になることに注意してください）。

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

ビルドに他のオプションを追加すると、ハッシュ値が変更される可能性があるため、この手順を繰り返す必要があります。また、アプリケーションを変更すると、アプリケーションを再コンパイルしたときにメモリ内でFIPS境界がシフトする場合があります。アプリケーションのみが更新された場合にハッシュがかわるのは、モジュールが影響を受けていることを示すものではなく、メモリ内の適切な位置にシフトされているだけです。これは、静的ライブラリとアプリケーションをコンパイルする場合にも起こることがあります。共有オブジェクトでは、この問題が発生しない傾向があります。

4.1 wolfcrypt/src/fips_test.cファイルを前項で説明したメッセージに基づいて編集し、ハッシュを更新します。配列verifyCoreの宣言まで下にスクロールして、前の手順のwolfCryptテストで提供された値で値を更新する必要があります。

4.2ライブラリを再度作成します。

5.ビルドをテストします。

$ make check

MakefileのcheckターゲットはwolfSSLとwolfCryptで提供するすべてのテストツールとスクリプトを実行します。すべて問題なければ、次の出力が表示されます。

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

6.ライブラリとヘッダーをインストールします。

$ make install

Makefileのインストールターゲットは、すべてのヘッダーとライブラリをシステムにインストールします。デフォルトでは、これは/usr/localディレクトリにあります。

この時点で、wolfSSL FIPS Readyはアプリケーションビルドで使用する準備ができているはずです。

wolfCrypt FIPS Ready APIドキュメント

以下は、wolfCrypt FIPS Ready APIの概要です。詳細については、wolfCrypt APIのドキュメントを参照してください。

FIPS認証されたセキュリティ関連関数のAPI

API	Description
Symmetric Encrypt/Decrypt Service	
`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.
`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_SetKey_fips`	Initializes **des3** object with **key**, **dir** indicates the direction while **iv** is optional. 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.
Keyed Hash Service	
`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.
`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.
Random Number Generation Service	
`InitRng_fips`	Initializes **RNG** object for use. 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 **outputSz** using 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.
Digital Signature Service	
`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.
`RsaPSS_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 **hash** with the mask generation function **mgf** and a salt length of **saltLen**. Returns 0 on success, < 0 on error.
`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 **hash** with 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 **hash** with 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 **hashType** and 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.
Message Digest Service	
`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.
`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_Final_fips`	Performs **sha3** (512-bit) Final, outputs digest to **hash**. Returns 0 on success, < 0 on error.
Key Generation Service	
`MakeRsaKey_fips`	Generates an RSA **key** with modulus length **size** and exponent **e** using the random number generator **rng**.
`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_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.
`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.
Key Agreement Service	
`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 **key** private **priv** of size **privSz** and peer's public key **otherPub** of size **pubSz**. Returns 0 on success, < 0 on error.
DSA Key Service	
`InitDhKey_fips`	Initializes DH **key** object for use. No return code.
`FreeDhKey_fips`	Releases DH **key** resources. No return code.
`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.
Key Transport Service	
`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.
`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 **hash** and **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.
Show status Service	
`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.

API Calls for Allowed Security Functions

API	Description
Message digest MD5 Service	
`wc_InitMd5`	Initializes **md5** object for use. Returns 0 on success, < 0 on error.
`wc_Md5Update`	Performs **md5** Update on input **data** of size **len**. Returns 0 on success, < 0 on error.
`wc_Md5Final`	Performs **md5** Final, outputs digest to **hash**. Returns 0 on success, < 0 on error.

以上