ALBANESE LAB

SOFTWARE ENGINEERING

GMSM Toolkit

Multi purpose cross-platform cryptography tool for asymmetric/symmetric encryption, digital signature, cipher-based message authentication code (CMAC), hash digest, hash-based message authentication code (HMAC) and PBKDF2 function.

SM2/SM3/SM4 Chinese National Standard Algorithms

National secret SM2/SM3/SM4 algorithm library based on Go language. SM2 is a public key cryptographic algorithm based on elliptic curves, used for e.g. generation and verification of digital signatures; SM3, a hashing algorithm comparable to SHA-256; and SM4, a block cipher algorithm for symmetric cryptography comparable to AES-128. These standards are becoming widely used in Chinese commercial applications such as banking and telecommunications and are sometimes made mandatory for products procured by Chinese government agencies. SM4 is part of the ARMv8.4-A expansion to the ARM architecture.

- GM/T 0003-2012 - SM2 Public key algorithm 256-bit.
- GM/T 0004-2012 - SM3 Message digest algorithm. 256-bit hash value.
- GM/T 0002-2012 - SM4 Symmetric block cipher with 128-bit key.

Modes of operation:

- GCM: Galois/Counter Mode (AEAD)
- CTR: Counter Mode
- OFB: Output Feedback Mode

Cryptographic Functions:

- Asymmetric Encryption/Decryption
- Symmetric Encryption/Decryption
- Digital Signature (ECDSA)
- Shared Key Agreement (ECDH)
- Recusive Hash Digest + Check
- CMAC (Cipher-based message authentication code)
- HMAC (Hash-based message authentication code)
- PBKDF2 (Password-based key derivation function 2)
- TLS (Transport Layer Security)

Usage:

```
-bits int
      Bit-length. (for DERIVE, PBKDF2 and RAND) (default 128)
-check string
      Check hashsum file. (- for STDIN)
-crypt string
      Encrypt/Decrypt with SM4 symmetric block cipher.
-derive string
      Derive shared secret key (SM2-ECDH) 128-bit default.
-digest string
      Target file/wildcard to generate hashsum list. (- for STDIN)
-hex string
      Encode/Decode [e|d] binary string to hex format and vice-versa.
-iter int
      Iterations. (for PBKDF2 and SHRED commands) (default 1)
-key string
      Private/Public key, Secret key or Password.
-keygen
      Generate asymmetric EC-SM2 keypair.
-mac string
      Compute Cipher-based/Hash-based message authentication code.
-mode string
      Mode of operation: GCM, CTR or OFB. (default "GCM")
-pbkdf2
      Password-based key derivation function.
-pub string
      Remote's side public key/remote's side public IP/PEM BLOCK.
-rand
      Generate random cryptographic key.
-recursive
      Process directories recursively.
-salt string
      Salt. (for PBKDF2)
-shred string
      Files/Path/Wildcard to apply data sanitization method.
-sign
      Sign with PrivateKey.
-signature string
      Input signature. (for verification only)
-sm2dec
      Decrypt with asymmetric EC-SM2 Privatekey.
-sm2enc
      Encrypt with asymmetric EC-SM2 Publickey.
-tcp string
      Encrypted TCP/IP [dump|ip|send] Transfer Protocol.
-verbose
      Verbose mode. (for CHECK command)
```

```
 -verify
       Verify with PublicKey.
 -version
       Print version information.
```

Examples:

Asymmetric SM2 keypair generation:

```
./gmsmtk -keygen
```

Derive shared secret key (SM2-ECDH):

```
./gmsmtk -derive a -key $PrivateKeyB -pub $PublicKeyA [-salt RandA;RandB]
[-bits 64|128|256]
./gmsmtk -derive b -key $PrivateKeyA -pub $PublicKeyB [-salt RandA;RandB]
[-bits 64|128|256]
```

Derive shared secret key (ECDH Non-standard):

```
./gmsmtk -derive c -key $PrivateKey -pub $PublicKey [-bits 64|128|256]
```

Signature (SM2-ECDSA):

```
./gmsmtk -sign -key $PrivateKey < file.ext > sign.txt
sign=$(cat sign.txt)
./gmsmtk -verify -key $PublicKey -signature $sign < file.ext
echo $?
```

Asymmetric encryption/decryption with SM2 algorithm:

```
./gmsmtk -sm2enc -key $PublicKey < plaintext.ext > ciphertext.ext
./gmsmtk -sm2dec -key $PrivateKey < ciphertext.ext > plaintext.ext
```

Symmetric encryption/decryption with SM4 block cipher:

```
./gmsmtk -crypt enc -key $128bitkey < plaintext.ext > ciphertext.ext
./gmsmtk -crypt dec -key $128bitkey < ciphertext.ext > plaintext.ext
```

CMAC-SM4 (cipher-based message authentication code):

```
./gmsmtk -mac cmac -key $64bitkey < file.ext
```

SM3 hashsum (list):

```
./gmsmtk -digest "*.*" [-recursive]
```

SM3 hashsum (single):

```
./gmsmtk -digest - < file.ext
```

HMAC-SM3 (hash-based message authentication code):

```
./gmsmtk -mac hmac -key $128bitkey < file.ext
```

PBKDF2 (password-based key derivation function 2):

```
./gmsmtk -pbkdf2 -key "pass" -iter 10000 -salt "salt"
```

Note:

The PBKDF2 function can be combined with the CRYPT and HMAC commands:

```
./gmsmtk -crypt -pbkdf2 -key "pass" < plaintext.ext > ciphertext.ext
./gmsmtk -mac hmac -pbkdf2 -key "pass" -iter 10000 -salt "salt" < file.ext
```

Shred (Data sanitization method, 25 iterations):

Prevents data recovery using standard recovery tools.

```
./gmsmtk -shred "keypair.ini" -iter 25
```

Bin to Hex/Hex to Bin:

```
echo somestring|./gmsmtk -hex enc
echo hexstring|./gmsmtk -hex dec
```

TLS TCP/IP Layer Dump/Send:

```
./gmsmtk -tcp ip > PublicIP.txt
./gmsmtk -tcp dump [-pub "8081"] > Pubkey.txt
./gmsmtk -tcp send [-pub "127.0.0.1:8081"] < Pubkey.txt
```

Random Art (Public Key Fingerprint):

```
./gmsmtk -key $pubkey
```

License