ALBANESE LAB

SOFTWARE ENGINEERING

EDGE Toolkit

Multi-purpose cross-platform hybrid cryptography tool for symmetric and asymmetric encryption, cipher-based message authentication code (CMAC/PMAC), recursive hash digest, hash-based message authentication code (HMAC), HMAC-based key derivation function (HKDF), Password-based key derivation function (PBKDF2/Scrypt), shared key agreement (ECDH/VKO/X25519), digital signature (RSA/ECDSA/EdDSA/GOST) and TLS 1.3 and TLCP for small or embedded systems.

Fully OpenSSL/LibreSSL/RHash/Mcrypt compliant

Command-line Integrated Security Suite

Asymmetric

- **Public key algorithms:**

Algorithm	256	512	ECDH	ECDSA	Encryption	TLS
ECDSA	O	O	O	O	O	O
Ed25519	O	-	O	O	-	O
GOST2012	O	O	O	O	-	O
RSA	-	-	-	O	O	O
SM2	O	-	O	O	O	O

- **Supported ParamSets:**

Curves Supported	A	B	C	D
GOST R 34.10-2012 256	O	O	O	O
GOST R 34.10-2012 512	O	O	O	-

Symmetric

- **Stream ciphers:**

Cipher	Key Size	IV	Modes of Operation
Chacha20	256	96/19	AEAD Stream Cipher
HC-128	128	128	XOR Stream
HC-256	256	256	XOR Stream
KCipher-2	128	128	XOR Stream
Rabbit	128	64	XOR Stream
RC4	40/128	-	XOR Stream
Salsa20	256	64/19	XOR Stream
Skein512	256	256	XOR Stream
ZUC-128	128	128	MAC + XOR Stream
ZUC-256	256	184	MAC + XOR Stream

- **128-bit block ciphers:**

Cipher	Block Size	Key Size	Modes of Operation
AES	128	128/192/256	All modes supported
Anubis	128	128	All modes supported
ARIA	128	128/192/256	All modes supported
Camellia	128	128/192/256	All modes supported
Kuznechik	128	256	All modes supported
LEA	128	128/192/256	All modes supported
SEED	128	128	All modes supported
Serpent	128	128/192/256	All modes supported
SM4	128	128	All modes supported
Twofish	128	128/192/256	All modes supported

- **64-bit block ciphers:**

Cipher	Block Size	Key Size	Modes of Operation
DES	64	64	CBC, CFB-8, CTR, OFB
3DES	64	192	CBC, CFB-8, CTR, OFB
Blowfish	64	128	CBC, CFB-8, CTR, OFB
CAST5	64	128	CBC, CFB-8, CTR, OFB
GOST89	64	256	MGM, CFB-8, CTR, OFB
HIGHT	64	128	CBC, CFB-8, CTR, OFB
IDEA	64	128	CBC, CFB-8, CTR, OFB
Magma	64	256	MGM, CFB-8, CTR, OFB
MISTY1	64	128	CBC, CFB-8, CTR, OFB
RC2	64	128	CBC, CFB-8, CTR, OFB
RC5	64	128	CBC, CFB-8, CTR, OFB

- **Modes of Operation:**

Mode	Mode of Operation Name	Blocks	Keys Sizes
EAX	Encrypt-Authenticate-Translate	128	128/192/256
GCM	Galois/Counter Mode (AEAD)	128	128/192/256

OCB1	Offset Codebook v1 (AEAD)	128	128/192/256
OCB3	Offset Codebook v3 (AEAD)	128	128/192/256
MGM	Multilinear Galois Mode (AEAD)	64/128	Any
CBC	Cipher-Block Chaining	All	Any
CFB	Cipher Feedback Mode	All	Any
CFB-8	Cipher Feedback Mode 8-bit	All	Any
CTR	Counter Mode (default)	All	Any
ECB	Eletronic Codebook Mode	All	Any
IGE	Infinite Garble Extension	All	Any
OFB	Output Feedback Mode	All	Any

- **Message Digest Algorithms:**

Message Digest	128	160	192	256	512	MAC
BLAKE-2B	-	-	-	O	O	O
BLAKE-2S	O	-	-	O	-	O
Chaskey	O	-	-	-	-	O
Cubehash	-	-	-	-	O	-
GOST94 CryptoPro	-	-	-	O	-	-
GrÃ¸stl	-	-	-	O	-	-
JH	-	-	-	O	-	-
Legacy Keccak	-	-	-	O	O	-
LSH	-	-	-	O	O	-
MD4 [Obsolete]	O	-	-	-	-	-
MD5 [Obsolete]	O	-	-	-	-	-
Poly1305	O	-	-	-	-	O
RIPEMD	O	O	-	O	-	-
SHA1 [Obsolete]	-	O	-	-	-	-
SHA2 (default)	-	-	-	O	O	-
SHA3	-	-	-	O	O	-
SipHash	O	-	-	-	-	O
Skein	-	-	-	O	O	O
SM3	-	-	-	O	-	-
Streebog	-	-	-	O	O	-
Tiger	-	-	O	-	-	-
Whirlpool	-	-	-	-	O	-
Xoodyak	-	-	-	O	-	O
ZUC-256 Zu Chongzhi	O	-	-	-	-	O

  - MAC refers to keyed hash function, like HMAC.

- **Experimental:**

Cipher	Key	IV	Modes of Operation
Xoodyak	128	128	AEAD Permutation Cipher
Ascon 1.2	128	128	AEAD Stream Cipher
Grain128a	128	40-96	AEAD Stream Cipher

AEAD

Authenticated encryption (AE) and authenticated encryption with associated data (AEAD) are forms of encryption which simultaneously assure the confidentiality and authenticity of data. Provides both authenticated encryption (confidentiality and authentication) and the ability to check the integrity and authentication of additional authenticated data (AAD) that is sent in the clear.

GOST (GOvernment STandard of Russian Federation)

GOST refers to a set of technical standards maintained by the Euro-Asian Council for Standardization, Metrology and Certification (EASC), a regional standards organization operating under the auspices of the Commonwealth of Independent States (CIS).

Key sizes

- **Bit-length Equivalence**

Symmetric Key Size	RSA and DSA Key Size	ECC Key Size
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

IKM (input key material value)

Keying material is in general to include things like shared Diffie-Hellman secrets (which are not suitable as symmetric keys), which have more structure than normal keys.

ShangMi (SM) National secret SM2/SM3/SM4 algorithms

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.

XOR

XOR (Exclusive OR) is a logical operator that works on bits. Letâ€™s denote it by ^. If the two bits it takes as input are the same, the result is 0, otherwise it is 1. This implements an exclusive or operation, i.e. exactly one argument has to be 1 for the final result to be 1. We can show this using a truth table:

- **exclusive or**

x	y	x^y
0	0	0
0	1	1
1	0	1
1	1	0

Features

- Asymmetric Encryption
- Symmetric Encryption + AEAD Modes
- Digital Signature
- Recursive Hash Digest + Check
- ECDH (Shared Key Agreement)
- CMAC (Cipher-based message authentication code)
- PMAC (Parallelizable message authentication code)
- HMAC (Hash-based message authentication code)
- HKDF (HMAC-based key derivation function)
- PBKDF2 (Password-based key derivation function)
- TLS (Transport Layer Security 1.2 and 1.3)
- TLCP (Transport Layer Cryptography Protocol)
- PKCS12 (Personal Information Exchange Syntax v1.1)
- X.509 CSRs and Certificates
- Hex string encoder/dump/decoder (xxd-like)
- Privacy-Enhanced Mail (PEM format)
- RandomArt (OpenSSH-like)

Usage

```
-algorithm string
      Public key algorithm: RSA, EC, Ed25519, GOST, SM2. (default "RSA")
-bits int
      Key length. (for keypair generation and symmetric encryption)
-cert string
      Certificate path. (default "Certificate.pem")
-check string
      Check hashsum file. ('-' for STDIN)
-cipher string
      Symmetric algorithm: aes, blowfish, magma or sm4. (default "aes")
-crypt string
      Bulk Encryption with Stream and Block ciphers. [enc|dec]
-digest
```

```
       Target file/wildcard to generate hashsum list. ('-' for STDIN)
 -hex string
       Encode binary string to hex format and vice-versa. [enc|dump|dec]
 -info string
       Additional info. (for HKDF command and AEAD bulk encryption)
 -ipport string
       Local Port/remote's side Public IP:Port.
 -iter int
       Iter. (for Password-based key derivation function) (default 1)
 -iv string
       Initialization Vector. (for symmetric encryption)
 -kdf string
       Key derivation function with given bit length. [pbkdf2|hkdf]
 -key string
       Asymmetric key, symmetric key or HMAC key, depending on operation.
 -mac string
       Compute Hash/Cipher-based message authentication code.
 -md string
       Hash algorithm: sha256, sha3-256 or whirlpool. (default "sha256")
 -mode string
       Mode of operation: GCM, MGM, CBC, CFB8, OCB, OFB. (default "CTR")
 -paramset string
       Elliptic curve ParamSet: A, B, C, D. (for GOST2012) (default "A")
 -pkey string
       Subcommands: keygen|certgen, sign|verify|derive, text|modulus.
 -private string
       Private key path. (for keypair generation) (default "Private.pem")
 -public string
       Public key path. (for keypair generation) (default "Public.pem")
 -pwd string
       Password. (for Private key PEM encryption)
 -rand int
       Generate random cryptographic key with given bit length.
 -recursive
       Process directories recursively. (for DIGEST command only)
 -root string
       Root CA Certificate path.
 -salt string
       Salt. (for HKDF and PBKDF2 commands)
 -signature string
       Input signature. (for VERIFY command and MAC verification)
 -tcp string
       Encrypted TCP/IP Transfer Protocol. [server|ip|client]</pre>
```

Examples

Asymmetric RSA keypair generation:

```
./edgetk -pkey keygen -bits 4096 [-pwd "pass"]
```

Parse keys info:

```
./edgetk -pkey [text|modulus] [-pwd "pass"] -key private.pem
./edgetk -pkey [text|modulus|randomart] -key public.pem
```

Digital signature:

```
./edgetk -pkey sign -key private.pem [-pwd "pass"] < file.ext > sign.txt
sign=$(cat sign.txt|awk '{print $2}')
./edgetk -pkey verify -key public.pem -signature $sign < file.ext
echo $?
```

Encryption/decryption with RSA algorithm:

```
./edgetk -pkey encrypt -key public.pem < plaintext.ext > ciphertext.ext
./edgetk -pkey decrypt -key private.pem < ciphertext.ext > plaintext.ext
```

Asymmetric EC keypair generation (256-bit):

```
./edgetk -pkey keygen -bits 256 -algorithm EC [-pwd "pass"]
```

EC Diffie-Hellman:

```
./edgetk -pkey derive -algorithm EC -key private.pem -public peerkey.pem
```

Generate Self Signed Certificate:

```
./edgetk -pkey certgen -key private.pem [-pwd "pass"] [-cert "output.crt"]
```

Generate Certificate Signing Request:

```
./edgetk -pkey req -key private.pem [-pwd "pass"] [-cert certificate.csr]
```

Sign CSR with CA Certificate:

```
./edgetk -pkey x509 -key private.pem -root cacert.pem -cert cert.csr >
cert.crt
```

Parse Certificate info:

```
./edgetk -pkey [text|modulus] -cert certificate.pem
```

TLS Layer (TCP/IP):

```
./edgetk -tcp ip > MyExternalIP.txt
./edgetk -tcp server -cert certificate.pem -key private.pem [-ipport "8081"]
./edgetk -tcp client -cert certificate.pem -key private.pem [-ipport "127.0.0.1:8081"]
```

Symmetric key generation (256-bit):

```
./edgetk -rand 256
```

Encryption/decryption with block cipher:

```
./edgetk -crypt enc -key $256bitkey < plaintext.ext > ciphertext.ext
./edgetk -crypt dec -key $256bitkey < ciphertext.ext > plaintext.ext
```

HMAC:

```
./edgetk -mac hmac -key "secret" < file.ext
./edgetk -mac hmac -key "secret" -signature $256bitmac < file.ext
echo $?
```

HKDF (HMAC-based key derivation function) (128-bit):

```
./edgetk -kdf hkdf -bits 128 -key "IKM" [-salt "salt"] [-info "AD"]
```

Hex Encoder/Decoder:

```
./edgetk -hex enc < file.ext > file.hex
./edgetk -hex dec < file.hex > file.ext
./edgetk -hex dump < file.ext
```

License

This project is licensed under the ISC License.

Copyright (c) 2022, Pedro F. Albanese pedroalbanese@hotmail.com

Copyright (c) 2020-2023 Pedro F. Albanese - ALBANESE Research Lab.