CryptoVe User Guide v0.1

by Wei Zhang

<d5c5ceb0@gmail.com>

Content

[CryptoVe User Guide v0.1 1](#_Toc459992130)

[1. Introduction 3](#_Toc459992131)

[2. Overview 4](#_Toc459992132)

[3. Cipher 4](#_Toc459992133)

[3.1 DES 4](#_Toc459992134)

[3.1.1 ECB 4](#_Toc459992135)

[3.1.2 CBC 5](#_Toc459992136)

[3.2 DES3 6](#_Toc459992137)

[3.2.1 ECB 6](#_Toc459992138)

[3.2.2 CBC 7](#_Toc459992139)

[3.3 AES 8](#_Toc459992140)

[3.3.1 ECB 8](#_Toc459992141)

[3.3.2 CBC 8](#_Toc459992142)

[3.3.3 CFB 9](#_Toc459992143)

[3.3.4 OFB 10](#_Toc459992144)

[3.3.5 CTR 11](#_Toc459992145)

[3.3.6 XTS 12](#_Toc459992146)

[3.4 SM4 13](#_Toc459992147)

[3.4.1 ECB 13](#_Toc459992148)

[3.4.2 CBC 14](#_Toc459992149)

[4. Hash 15](#_Toc459992150)

[4.1 MD5 15](#_Toc459992151)

[4.2 SHA1 16](#_Toc459992152)

[4.3 SHA224 17](#_Toc459992153)

[4.4 SHA256 17](#_Toc459992154)

[4.5 SHA384 18](#_Toc459992155)

[4.6 SHA512 19](#_Toc459992156)

[4.7 SM3 20](#_Toc459992157)

[5. MAC 21](#_Toc459992158)

[5.1 CBC-MAC 21](#_Toc459992159)

[5.1.1 CMAC-AES 22](#_Toc459992160)

[5.1.2 CBC-MAC-AES 22](#_Toc459992161)

[5.1.3 XCBC-MAC-AES 23](#_Toc459992162)

[5.2 HMAC 24](#_Toc459992163)

[5.2.1 HMAC-MD5 24](#_Toc459992164)

[5.2.2 HMAC-SHA1 25](#_Toc459992165)

[5.2.3 HMAC-SHA224 25](#_Toc459992166)

[5.2.4 HMAC-SHA256 26](#_Toc459992167)

[5.2.5 HMAC-SHA384 27](#_Toc459992168)

[5.2.6 HMAC-SHA512 28](#_Toc459992169)

[6. PK 29](#_Toc459992170)

[6.1 BIG NUMBER 29](#_Toc459992171)

[6.1.1 addition 29](#_Toc459992172)

[6.1.2 subtraction 30](#_Toc459992173)

[6.1.3 multiplication 30](#_Toc459992174)

[6.1.4 division 31](#_Toc459992175)

[6.1.5 remainder 31](#_Toc459992176)

[6.1.6 compare 32](#_Toc459992177)

[6.1.7 or 32](#_Toc459992178)

[6.1.8 and 33](#_Toc459992179)

[6.1.9 not 33](#_Toc459992180)

[6.1.10 exclusive or 33](#_Toc459992181)

[6.1.11 shift 34](#_Toc459992182)

[6.1.12 modular addition 34](#_Toc459992183)

[6.1.13 modular subtraction 35](#_Toc459992184)

[6.1.14 modular multiplication 35](#_Toc459992185)

[6.1.15 modular inverse 36](#_Toc459992186)

[6.1.16 modular exponent 36](#_Toc459992187)

[6.1.17 gcd 36](#_Toc459992188)

[6.1.18 genprime 37](#_Toc459992189)

[6.1.19 isprime 37](#_Toc459992190)

[6.1.20 is a ecc point 38](#_Toc459992191)

[6.1.21 point addition 38](#_Toc459992192)

[6.1.22 point multiplication 39](#_Toc459992193)

[6.1.23 multiple point multiplication 39](#_Toc459992194)

[6.2 RSA 40](#_Toc459992195)

[6.2.1 keygen 40](#_Toc459992196)

[6.2.2 keygen with p and q 41](#_Toc459992197)

[6.2.3 encription & verify 41](#_Toc459992198)

[6.2.4 decryption & sign 42](#_Toc459992199)

[6.2.5 crt 42](#_Toc459992200)

[6.3 SM2 43](#_Toc459992201)

[6.3.1 hash z 43](#_Toc459992202)

[6.3.2 hash e 43](#_Toc459992203)

[6.3.3 keygen 44](#_Toc459992204)

[6.3.4 sign 44](#_Toc459992205)

[6.3.5 verify 44](#_Toc459992206)

[6.3.6 key exchange 45](#_Toc459992207)

[6.3.7 encryption 47](#_Toc459992208)

[6.3.8 decryption 47](#_Toc459992209)

[7. random number 48](#_Toc459992210)

[8. CRC 48](#_Toc459992211)

[8.1 crc16 48](#_Toc459992212)

[9. util 50](#_Toc459992213)

[9.1 dec2hex 50](#_Toc459992214)

[9.2 endian 50](#_Toc459992215)

[9.3 hex2bin 50](#_Toc459992216)

[9.4 bin2hex 50](#_Toc459992217)

# 1. Introduction

This document describes the CryptoVe TCL API.

For further support in using this document please contact me. (Wei Zhang, [d5c5ceb0@gmail.com](mailto:d5c5ceb0@gmail.com))

# 2. Overview

The CryptoVe is a TCL library for cryptographic. Varities of cryptographic algorithms are provided (e.g. , AES, DES, HASH, HMAC, RSA, ECC, SM2, SM3, SM4, CRC, among others).

# 3. Cipher

The data can be processed in one of two modes of operation:

• Integrated operation – Processes all data in a single function call. This flow is applicable when all data is available prior to the cryptographic operation.

• Block operation – Processes a subset of the data buffers, and is called multiple times in a sequence. This flow is applicable when the next data buffer becomes available only during/after processing of the current data buffer.

## 3.1 DES

DES is a block cipher, i.e. it processes data in multiples of block size (8 bytes), and the key size is 8 bytes too.

### 3.1.1 ECB

Integrated operation:

des\_ecb\_process direction key messages

direction : “enc” for encryption; “dec” for decryption

key : DES key of 8 bytes

messages : data, multiples of 8 bytes

example:

set K bbe07cd54390e120

set M ff50228ae7c3799ad8eea7c053d6bb0d8814a9bf89a6b1b62b485adb8765f60b

des\_ecb\_process enc $K $M

#the result is 910e28d0543dbf02ba46a430f8e1a0148cd7764d66f8ea3b986244dfc2e0d9bb

Block operation:

des\_ecb\_init ctx direction key

des\_ecb\_update ctx messages

des\_ecb\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : DES key of 8 bytes

messages : data, multiples of 8 bytes

example:

set K bbe07cd54390e120

set M1 ff50228ae7c3799a

set M2 d8eea7c053d6bb0d8814a9bf89a6b1b6

set M3 2b485adb8765f60b

des\_ecb\_init ecb\_ctx enc $K

des\_ecb\_update ecb\_ctx $M1

des\_ecb\_update ecb\_ctx $M2

des\_ecb\_update ecb\_ctx $M3

des\_ecb\_done ecb\_ctx

#the result is 910e28d0543dbf02ba46a430f8e1a0148cd7764d66f8ea3b986244dfc2e0d9bb

### 3.1.2 CBC

Integrated operation:

des\_cbc\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : DES key of 8 bytes

iv : iv

messages : data, multiples of 8 bytes

set K 75168e70130b9f7b

set IV 6c2fb0fa6a39073c

set M ff50228ae7c3799ad8eea7c053d6bb0d8814a9bf89a6b1b62b485adb8765f60b

des\_cbc\_process enc $K $IV $M

#d6fdb31e2cdb5d27a68a90cc88e9d30230b2fe8f6f49f352ba694cbdcb01f8ef

Block operation:

des\_cbc\_init ctx direction key iv

des\_cbc\_update ctx messages

des\_cbc\_done ctx

set K 75168e70130b9f7b

set IV 6c2fb0fa6a39073c

set M1 ff50228ae7c3799a

set M2 d8eea7c053d6bb0d8814a9bf89a6b1b6

set M3 2b485adb8765f60b

des\_cbc\_init cbc\_ctx enc $K $IV

des\_cbc\_update cbc\_ctx $M1

des\_cbc\_update cbc\_ctx $M2

des\_cbc\_update cbc\_ctx $M3

des\_cbc\_done cbc\_ctx

#d6fdb31e2cdb5d27a68a90cc88e9d30230b2fe8f6f49f352ba694cbdcb01f8ef

## 3.2 DES3

### 3.2.1 ECB

Integrated operation:

des3\_ecb\_process direction key messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes or 24 bytes)

messages : data, multiples of 8 bytes

example:

set K1 6a0cb7b06ddf5aca

set K2 3e914852e5afc9a4

set M ff50228ae7c3799ad8eea7c053d6bb0d8814a9bf89a6b1b62b485adb8765f60b

des3\_ecb\_process enc ${K1}$K2 $M

#the result is bc80a40bd8d22961f16a2ecd5e4c3ed0afdbfd37b7ca18df6e82798c3574bcb9

Block operation:

des3\_ecb\_init ctx direction key

des3\_ecb\_update ctx messages

des3\_ecb\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes or 24 bytes)

messages : data, multiples of 8 bytes

example:

set K1 6a0cb7b06ddf5aca

set K2 3e914852e5afc9a4

set M1 ff50228ae7c3799a

set M2 d8eea7c053d6bb0d8814a9bf89a6b1b6

set M3 2b485adb8765f60b

des3\_ecb\_init ecb\_ctx enc ${K1}$K2

des3\_ecb\_update ecb\_ctx $M1

des3\_ecb\_update ecb\_ctx $M2

des3\_ecb\_update ecb\_ctx $M3

des3\_ecb\_done ecb\_ctx

#the result is bc80a40bd8d22961f16a2ecd5e4c3ed0afdbfd37b7ca18df6e82798c3574bcb9

### 3.2.2 CBC

Integrated operation:

des3\_cbc\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : key (16 bytes, or 24 bytes)

iv : iv

messages : data, multiples of 8 bytes

set K1 6a0cb7b06ddf5aca

set K2 3e914852e5afc9a4

set IV 6c2fb0fa6a39073c

set M ff50228ae7c3799ad8eea7c053d6bb0d8814a9bf89a6b1b62b485adb8765f60b

des3\_cbc\_process enc ${K1}${K2} $IV $M

#15e40b8bf0ab86caac5a6b9178a94ce41836e1b7e7d9bdd54f26a29ac581d0a0

Block operation:

des\_cbc\_init ctx direction key iv

des\_cbc\_update ctx messages

des\_cbc\_done ctx

set K1 6a0cb7b06ddf5aca

set K2 3e914852e5afc9a4

set IV 6c2fb0fa6a39073c

set M1 ff50228ae7c3799a

set M2 d8eea7c053d6bb0d8814a9bf89a6b1b6

set M3 2b485adb8765f60b

des3\_cbc\_init cbc\_ctx enc ${K1}${K2} $IV

des3\_cbc\_update cbc\_ctx $M1

des3\_cbc\_update cbc\_ctx $M2

des3\_cbc\_update cbc\_ctx $M3

des3\_cbc\_done cbc\_ctx

#15e40b8bf0ab86caac5a6b9178a94ce41836e1b7e7d9bdd54f26a29ac581d0a0

## 3.3 AES

### 3.3.1 ECB

Integrated operation:

aes\_ecb\_process direction key messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

messages : data, multiples of 16 bytes

example:

set K 16124AD88976E46BEA7B25F52F9DBDBD59C436C635B3CE9E61ED4C9857CB7D5B

set M 63CDB50D8D7439459DA8288231B154F5

aes\_ecb\_process enc $K $M

#the result is 83e7b2981c148a46afb34278c033623f

Block operation:

aes\_ecb\_init ctx direction key

aes\_ecb\_update ctx messages

aes\_ecb\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

messages : data, multiples of 16 bytes

example:

set K 16124AD88976E46BEA7B25F52F9DBDBD59C436C635B3CE9E61ED4C9857CB7D5B

set M 63CDB50D8D7439459DA8288231B154F5

aes\_ecb\_init ecb\_ctx enc $K $IV

aes\_ecb\_update ecb\_ctx $M

aes\_ecb\_done ecb\_ctx

#the result is 83e7b2981c148a46afb34278c033623f

### 3.3.2 CBC

Integrated operation:

aes\_cbc\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 16124AD88976E46BEA7B25F52F9DBDBD59C436C635B3CE9E61ED4C9857CB7D5B

set IV E6490D022F79FB8CB3EFEEAED923E6E2

set M 7C5295B0E79D37C9B67FA572CBE304FD538A1D169E24DF1F3B11A3AF5D1335531C322D700128160ED8A0BDE2A57FA3DD

aes\_cbc\_process enc $K $IV $M

#the result is 2c8e43749486350ef1178500325dc5f69a73eb2c0222815bff5b91d122849f4f60394e405c12603f3d70b8f10cd4af75

Block operation:

aes\_cbc\_init ctx direction key iv

aes\_cbc\_update ctx messages

aes\_cbc\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 16124AD88976E46BEA7B25F52F9DBDBD59C436C635B3CE9E61ED4C9857CB7D5B

set IV E6490D022F79FB8CB3EFEEAED923E6E2

set M 7C5295B0E79D37C9B67FA572CBE304FD538A1D169E24DF1F3B11A3AF5D1335531C322D700128160ED8A0BDE2A57FA3DD

aes\_cbc\_init cbc\_ctx enc $K $IV

aes\_cbc\_update cbc\_ctx $M

aes\_cbc\_done cbc\_ctx

#the result is 2c8e43749486350ef1178500325dc5f69a73eb2c0222815bff5b91d122849f4f60394e405c12603f3d70b8f10cd4af75

### 3.3.3 CFB

Integrated operation:

aes\_cfb\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set IV 000102030405060708090a0b0c0d0e0f

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_cfb\_process enc $K $IV $M

#the result is 3b3fd92eb72dad20333449f8e83cfb4ac8a64537a0b3a93fcde3cdad9f1ce58b26751f67a3cbb140b1808cf187a4f4dfc04b05357c5d1c0eeac4c66f9ff7f2e6

Block operation:

aes\_cfb\_init ctx direction key iv

aes\_cfb\_update ctx messages

aes\_cfb\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set IV 000102030405060708090a0b0c0d0e0f

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_cfb\_init cfb\_ctx enc $K $IV

aes\_cfb\_update cfb\_ctx $M

aes\_cfb\_done cfb\_ctx

#the result is 3b3fd92eb72dad20333449f8e83cfb4ac8a64537a0b3a93fcde3cdad9f1ce58b26751f67a3cbb140b1808cf187a4f4dfc04b05357c5d1c0eeac4c66f9ff7f2e6

### 3.3.4 OFB

Integrated operation:

aes\_ofb\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set IV 000102030405060708090a0b0c0d0e0f

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_ofb\_process enc $K $IV $M

#the result is 3b3fd92eb72dad20333449f8e83cfb4a7789508d16918f03f53c52dac54ed8259740051e9c5fecf64344f7a82260edcc304c6528f659c77866a510d9c1d6ae5e

Block operation:

aes\_ofb\_init ctx direction key iv

aes\_ofb\_update ctx messages

aes\_ofb\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set IV 000102030405060708090a0b0c0d0e0f

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_ofb\_init ofb\_ctx enc $K $IV

aes\_ofb\_update ofb\_ctx $M

aes\_ofb\_done ofb\_ctx

#the result is 3b3fd92eb72dad20333449f8e83cfb4a7789508d16918f03f53c52dac54ed8259740051e9c5fecf64344f7a82260edcc304c6528f659c77866a510d9c1d6ae5e

### 3.3.5 CTR

Integrated operation:

aes\_ctr\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set IV f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_ctr\_process enc $K $IV $M

#the result is 874d6191b620e3261bef6864990db6ce9806f66b7970fdff8617187bb9fffdff5ae4df3edbd5d35e5b4f09020db03eab1e031dda2fbe03d1792170a0f3009cee

Block operation:

aes\_ctr\_init ctx direction key iv

aes\_ctr\_update ctx messages

aes\_ctr\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes, 24 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set IV f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_ctr\_init ctr\_ctx enc $K $IV

aes\_ctr\_update ctr\_ctx $M

aes\_ctr\_done ctr\_ctx

#the result is 874d6191b620e3261bef6864990db6ce9806f66b7970fdff8617187bb9fffdff5ae4df3edbd5d35e5b4f09020db03eab1e031dda2fbe03d1792170a0f3009cee

### 3.3.6 XTS

Integrated operation:

aes\_xts\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K1 fffefdfcfbfaf9f8f7f6f5f4f3f2f1f0

set K2 22222222222222222222222222222222

set IV 33333333330000000000000000000000

set M 4444444444444444444444444444444444444444444444444444444444444444

aes\_xts\_process enc ${K1}${K2} $IV $M

#the result is af85336b597afc1a900b2eb21ec949d292df4c047e0b21532186a5971a227a89

Block operation:

aes\_xts\_init ctx direction key iv

aes\_xts\_update ctx messages

aes\_xts\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes or 32 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K1 fffefdfcfbfaf9f8f7f6f5f4f3f2f1f0

set K2 22222222222222222222222222222222

set IV 33333333330000000000000000000000

set M 4444444444444444444444444444444444444444444444444444444444444444

aes\_xts\_init xts\_ctx enc ${K1}${K2} $IV

aes\_xts\_update xts\_ctx $M

aes\_xts\_done xts\_ctx

#the result is af85336b597afc1a900b2eb21ec949d292df4c047e0b21532186a5971a227a89

## 3.4 SM4

### 3.4.1 ECB

Integrated operation:

sm4\_ecb\_process direction key messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes)

messages : data, multiples of 16 bytes

example:

set K 0123456789abcdeffedcba9876543210

set M 0123456789abcdeffedcba9876543210

sm4\_ecb\_process enc $K $M

#the result is 681edf34d206965e86b3e94f536e4246

Block operation:

sm4\_ecb\_init ctx direction key

sm4\_ecb\_update ctx messages

sm4\_ecb\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes)

messages : data, multiples of 16 bytes

example:

set K 0123456789abcdeffedcba9876543210

set M 0123456789abcdeffedcba9876543210

sm4\_ecb\_init ecb\_ctx enc $K $IV

sm4\_ecb\_update ecb\_ctx $M

sm4\_ecb\_done ecb\_ctx

#the result is 681edf34d206965e86b3e94f536e4246

### 3.4.2 CBC

Integrated operation:

sm4\_cbc\_process direction key iv messages

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 0123456789abcdeffedcba9876543210

set IV 0123456789abcdeffedcba9876543210

set M 131817B709047289284435032A988A7872198601F6066B6B077A97059C5D6792A8F75FF136A71FAA7D745322D75277B4

sm4\_cbc\_process enc $K $IV $M

#the result is 51e58110754e112683f90c4fea97cfd6cf17ce2fbb6292ee9f549f7fbd589e161a352cc7a523c4cdeb00fdca4cf49c3d

Block operation:

sm4\_cbc\_init ctx direction key iv

aes\_cbc\_update ctx messages

aes\_cbc\_done ctx

ctx : the context of this session

direction : “enc” for encryption; “dec” for decryption

key : key(16 bytes)

iv : iv

messages : data, multiples of 16 bytes

example:

set K 0123456789abcdeffedcba9876543210

set IV 0123456789abcdeffedcba9876543210

set M 131817B709047289284435032A988A7872198601F6066B6B077A97059C5D6792A8F75FF136A71FAA7D745322D75277B4

sm4\_cbc\_init cbc\_ctx enc $K $IV

sm4\_cbc\_update cbc\_ctx $M

sm4\_cbc\_done cbc\_ctx

#the result is 51e58110754e112683f90c4fea97cfd6cf17ce2fbb6292ee9f549f7fbd589e161a352cc7a523c4cdeb00fdca4cf49c3d

# 4. Hash

The data can be processed in one of two modes of operation:

• Integrated operation – Processes all data in a single function call. This flow is applicable when all data is available prior to the cryptographic operation.

• Block operation – Processes a subset of the data buffers, and is called multiple times in a sequence. This flow is applicable when the next data buffer becomes available only during/after processing of the current data buffer.

## 4.1 MD5

Integrated operation:

md5\_process messages

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

md5\_process $message

#the result is b3eb9ac023b813857b895dd3cc74ec11

Block operation:

md5\_init ctx

md5\_update ctx messages

md5\_done ctx

ctx : the context of this session

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

md5\_init md5\_ctx

md5\_update md5\_ctx $message

md5\_done md5\_ctx

#the result is b3eb9ac023b813857b895dd3cc74ec11

## 4.2 SHA1

Integrated operation:

sha1\_process messages

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

sha1\_process $message

#the result is cae74849fc4ca9ae98ce22db01d0561beaa47bd6

Block operation:

sha1\_init ctx

sha1\_update ctx messages

sha1\_done ctx

ctx : the context of this session

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

sha1\_init sha1\_ctx

sha1\_update sha1\_ctx $message

sha1\_done sha1\_ctx

#the result is cae74849fc4ca9ae98ce22db01d0561beaa47bd6

## 4.3 SHA224

Integrated operation:

sha224\_process messages

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

sha224\_process $message

#the result is 2234ac071a938111cb1cb79e054b548a80206cb38e0a038a565a3a05

Block operation:

sha224\_init ctx

sha224\_update ctx messages

sha224\_done ctx

ctx : the context of this session

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

sha224\_init sha224\_ctx

sha224\_update sha224\_ctx $message

sha224\_done sha224\_ctx

#the result is 2234ac071a938111cb1cb79e054b548a80206cb38e0a038a565a3a05

## 4.4 SHA256

Integrated operation:

sha256\_process messages

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

sha256\_process $message

#the result is af9f0cb3809944ba914dd2d28721c6f03956911f4450e481cb18ff9f92efdc65

Block operation:

sha256\_init ctx

sha256\_update ctx messages

sha256\_done ctx

ctx : the context of this session

messages : data

example:

set message 6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a6162636465666768696a6b6c6d6e6f707172737475767778797a30313233343536373839

sha256\_init sha256\_ctx

sha256\_update sha256\_ctx $message

sha256\_done sha256\_ctx

#the result is af9f0cb3809944ba914dd2d28721c6f03956911f4450e481cb18ff9f92efdc65

## 4.5 SHA384

Integrated operation:

sha384\_process messages

messages : data

example:

set message 4142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c

sha384\_process $message

#the result is d31114cf0abc09647b4737df418ea09d692054f0a10048a05d765e30398409597e4f6d1d83bff919f2584bd15a138430

Block operation:

sha384\_init ctx

sha384\_update ctx messages

sha384\_done ctx

ctx : the context of this session

messages : data

example:

set message 4142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c

sha384\_init sha384\_ctx

sha384\_update sha384\_ctx $message

sha384\_done sha384\_ctx

#the result is d31114cf0abc09647b4737df418ea09d692054f0a10048a05d765e30398409597e4f6d1d83bff919f2584bd15a138430

## 4.6 SHA512

Integrated operation:

sha512\_process messages

messages : data

example:

set message 4142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c

sha512\_process $message

#the result is 9bc1660a879982f04edee7fefab921f1e6e5fc7078023a0dd251987b6fcdbe9e7521a73e652b3e1ba4eb683d3967e39e37d21b057645b411b71efd461d3594fb

Block operation:

sha512\_init ctx

sha512\_update ctx messages

sha512\_done ctx

ctx : the context of this session

messages : data

example:

set message 4142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c4d4e4f505152535455565758595a6162636465666768696a6b6c6d6e6f707172737475767778797a313233343536373839304142434445464748494a4b4c

sha512\_init sha512\_ctx

sha512\_update sha512\_ctx $message

sha512\_done sha512\_ctx

#the result is 9bc1660a879982f04edee7fefab921f1e6e5fc7078023a0dd251987b6fcdbe9e7521a73e652b3e1ba4eb683d3967e39e37d21b057645b411b71efd461d3594fb

## 4.7 SM3

Integrated operation:

sm3\_process messages

messages : data

example:

set message 1e

sm3\_process $message

#the result is 83cf37260488edcb9cee59f9777e0cc613d6c70e5a55cf4318fcde04815a08bd

Block operation:

sm3\_init ctx

sm3\_update ctx messages

sm3\_done ctx

ctx : the context of this session

messages : data

example:

set message 1e

sm3\_init sm3\_ctx

sm3\_update sm3\_ctx $message

sm3\_done sm3\_ctx

#the result is 83cf37260488edcb9cee59f9777e0cc613d6c70e5a55cf4318fcde04815a08bd

# 5. MAC

The data can be processed in one of two modes of operation:

• Integrated operation – Processes all data in a single function call. This flow is applicable when all data is available prior to the cryptographic operation.

• Block operation – Processes a subset of the data buffers, and is called multiple times in a sequence. This flow is applicable when the next data buffer becomes available only during/after processing of the current data buffer.

## 5.1 CBC-MAC

### 5.1.1 CMAC-AES

aes\_cmac\_process key messages

key : key(16 bytes, 24 bytes or 32 bytes)

messages : data

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411

aes\_cmac\_process $K $M

#the result is dfa66747de9ae63030ca32611497c827

Block operation:

aes\_cmac\_init ctx key

aes\_cmac\_update ctx messages

aes\_cmac\_done ctx

ctx : the context of this session

key : key(16 bytes, 24 bytes or 32 bytes)

messages : data

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411

aes\_cmac\_init cmac\_ctx $K

aes\_cmac\_update cmac\_ctx $M

aes\_cmac\_done cmac\_ctx

#the result is dfa66747de9ae63030ca32611497c827

### 5.1.2 CBC-MAC-AES

aes\_cbcmac\_process key messages

key : key(16 bytes, 24 bytes or 32 bytes)

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_cbcmac\_process $K $M

#the result is a7356e1207bb406639e5e5ceb9a9ed93

Block operation:

aes\_cbcmac\_init ctx key

aes\_cbcmac\_update ctx messages

aes\_cbcmac\_done ctx

ctx : the context of this session

key : key(16 bytes, 24 bytes or 32 bytes)

messages : data, multiples of 16 bytes

example:

set K 2b7e151628aed2a6abf7158809cf4f3c

set M 6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411e5fbc1191a0a52eff69f2445df4f9b17ad2b417be66c3710

aes\_cbcmac\_init cbcmac\_ctx $K

aes\_cbcmac\_update cbcmac\_ctx $M

aes\_cbcmac\_done cbcmac\_ctx

#the result is a7356e1207bb406639e5e5ceb9a9ed93

### 5.1.3 XCBC-MAC-AES

aes\_xcbcmac\_process key messages

key : key(16 bytes)

messages : data

example:

set K 000102030405060708090a0b0c0d0e0f

set M 000102

aes\_xcbcmac\_process $K $M

#the result is 5b376580ae2f19afe7219ceef172756f

Block operation:

aes\_xcbcmac\_init ctx key

aes\_xcbcmac\_update ctx messages

aes\_xcbcmac\_done ctx

ctx : the context of this session

key : key(16 bytes)

messages : data

example:

set K 000102030405060708090a0b0c0d0e0f

set M 000102

aes\_xcbcmac\_init xcbcmac\_ctx $K

aes\_xcbcmac\_update xcbcmac\_ctx $M

aes\_xcbcmac\_done xcbcmac\_ctx

#the result is 5b376580ae2f19afe7219ceef172756f

## 5.2 HMAC

### 5.2.1 HMAC-MD5

md5\_hmac\_process key messages

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

md5\_hmac\_process $K $M

#the result is 3ba82db082e75093203ff41d9f4f6d52

Block operation:

md5\_hmac\_init ctx key

md5\_hmac\_update ctx messages

md5\_hmac\_done ctx

ctx : the context of this session

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

md5\_hmac\_init hmac\_ctx $K

md5\_hmac\_update hmac\_ctx $M

md5\_hmac\_done hmac\_ctx

#the result is 3ba82db082e75093203ff41d9f4f6d52

### 5.2.2 HMAC-SHA1

sha1\_hmac\_process key messages

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha1\_hmac\_process $K $M

#the result is 0d232f31b745171451e97fe73e9c307d9d5555bd

Block operation:

sha1\_hmac\_init ctx key

sha1\_hmac\_update ctx messages

sha1\_hmac\_done ctx

ctx : the context of this session

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha1\_hmac\_init hmac\_ctx $K

sha1\_hmac\_update hmac\_ctx $M

sha1\_hmac\_done hmac\_ctx

#the result is 0d232f31b745171451e97fe73e9c307d9d5555bd

### 5.2.3 HMAC-SHA224

sha224\_hmac\_process key messages

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha224\_hmac\_process $K $M

#the result is 13e71b85772ec123d2614870072c2330f70bd2c8f7973d27b032825b

Block operation:

sha224\_hmac\_init ctx key

sha224\_hmac\_update ctx messages

sha224\_hmac\_done ctx

ctx : the context of this session

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha224\_hmac\_init hmac\_ctx $K

sha224\_hmac\_update hmac\_ctx $M

sha224\_hmac\_done hmac\_ctx

#the result is 13e71b85772ec123d2614870072c2330f70bd2c8f7973d27b032825b

### 5.2.4 HMAC-SHA256

sha256\_hmac\_process key messages

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha256\_hmac\_process $K $M

#the result is 3c0b856e96f74d7f74d1f8e8838456dadcfc85dc34403c7f0ddc168108c2ce13

Block operation:

sha256\_hmac\_init ctx key

sha256\_hmac\_update ctx messages

sha256\_hmac\_done ctx

ctx : the context of this session

key : key

messages : data

example:

set K 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha256\_hmac\_init hmac\_ctx $K

sha256\_hmac\_update hmac\_ctx $M

sha256\_hmac\_done hmac\_ctx

#the result is 3c0b856e96f74d7f74d1f8e8838456dadcfc85dc34403c7f0ddc168108c2ce13

### 5.2.5 HMAC-SHA384

sha384\_hmac\_process key messages

key : key

messages : data

example:

set K 41414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha384\_hmac\_process $K $M

#the result is e685fa4ea7efd11b2d583cd7f035fded03612316083df1e6659266d5514a0b7a9f4e9af505c567501e1a8bbca16435f9

Block operation:

sha384\_hmac\_init ctx key

sha384\_hmac\_update ctx messages

sha384\_hmac\_done ctx

ctx : the context of this session

key : key

messages : data

example:

set K 41414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha384\_hmac\_init hmac\_ctx $K

sha384\_hmac\_update hmac\_ctx $M

sha384\_hmac\_done hmac\_ctx

#the result is e685fa4ea7efd11b2d583cd7f035fded03612316083df1e6659266d5514a0b7a9f4e9af505c567501e1a8bbca16435f9

### 5.2.6 HMAC-SHA512

sha512\_hmac\_process key messages

key : key

messages : data

example:

set K 41414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha512\_hmac\_process $K $M

#the result is 24e74026d857f6f4b9ac8e645f0753755e4895235e71a7833ff2c7c29b2cd3c9b3494b02fea0e43a91a3a8cd1970734d4172f058309f099331929153facebff8

Block operation:

sha512\_hmac\_init ctx key

sha512\_hmac\_update ctx messages

sha512\_hmac\_done ctx

ctx : the context of this session

key : key

messages : data

example:

set K 41414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141

set M 54657374205573696e67204c6172676572205468616e20426c6f636b2d53697a65204b657920616e64204c6172676572205468616e204f6e6520426c6f636b2d53697a652044617461

sha512\_hmac\_init hmac\_ctx $K

sha512\_hmac\_update hmac\_ctx $M

sha512\_hmac\_done hmac\_ctx

#the result is 24e74026d857f6f4b9ac8e645f0753755e4895235e71a7833ff2c7c29b2cd3c9b3494b02fea0e43a91a3a8cd1970734d4172f058309f099331929153facebff8

# 6. PK

## 6.1 BIG NUMBER

### 6.1.1 addition

command:

add a b

prototype:

result = a + b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

add 01 0102

#the result is 0103

### 6.1.2 subtraction

command:

sub a b

prototype:

result = a – b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

sub 0403 0102

#the result is 0301

### 6.1.3 multiplication

command:

mul a b

prototype:

result = a \* b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

mul ff ff

#the result is FE01

### 6.1.4 division

command:

div a b

prototype:

result = a / b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

div 10 04

#the result is 04

### 6.1.5 remainder

command:

rem a b

prototype:

result = a % b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

rem 10 07

#the result is 02

### 6.1.6 compare

command:

cmp a b

prototype:

result = a compare with b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

return code:

00：a equal to b

01：a larger than b

02：a smaller than b

example:

cmp FFFFFFFFFF AFFFFFFFFF

#the result is 01

### 6.1.7 or

command:

orr a b

prototype:

result = a | b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

orr 0f0f0f0f f0f0f0f0

#the result is ffffffff

### 6.1.8 and

command:

and a b

prototype:

result = a & b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

and 0f0f0f0f f0f0f0f0

#the result is 00000000

### 6.1.9 not

command:

not a

prototype:

result = ~a

a : argument a in Hexadecimal

example:

not 0f0f0f0f

#the result is f0f0f0f0

### 6.1.10 exclusive or

command:

xor a b

prototype:

result = a ^ b

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

xor 0f0f0f0f 0f0f0f0f

#the result is 00000000

### 6.1.11 shift

command:

sft mode x n

prototype:

result = sftmode(x, n)

mode : R – shift right; L – shift left

x : argument a in Hexadecimal

n : shift n bits

example:

sft R FFFFFFFFFF 4

#the result is 0FFFFFFFFF

### 6.1.12 modular addition

command:

modadd a b n

prototype:

result = (a+b) % n

a : argument a in Hexadecimal

b : argument b in Hexadecimal

n : argument b in Hexadecimal

example:

modadd 07 09 0F

#the result is 01

### 6.1.13 modular subtraction

command:

modsub a b n

prototype:

result = (a+b) % n

a : argument a in Hexadecimal

b : argument b in Hexadecimal

n : argument b in Hexadecimal

example:

modsub 07 09 0F

#the result is 0D

### 6.1.14 modular multiplication

command:

modmul a b n

prototype:

result = (a\*b) % n

a : argument a in Hexadecimal

b : argument b in Hexadecimal

n : argument b in Hexadecimal

example:

modmul 07 09 0F

#the result is 03

### 6.1.15 modular inverse

command:

modinv a n

prototype:

result = a\*\*-1 % n

a : argument a in Hexadecimal

n : argument b in Hexadecimal

example:

modinv 07 0F

#the result is 0d

### 6.1.16 modular exponent

command:

modexp a b n

prototype:

result = (a\*\*b) % n

a : argument a in Hexadecimal

b : argument b in Hexadecimal

n : argument b in Hexadecimal

example:

modexp 08 07 0F

#the result is 02

### 6.1.17 gcd

command:

gcd a b

prototype:

result = gcd(a,b)

a : argument a in Hexadecimal

b : argument b in Hexadecimal

example:

gcd 10 08

#the result is 08

### 6.1.18 genprime

command:

genprime a

prototype:

result = genprime(a), and result > a

a : argument a in Hexadecimal

example:

genprime 09

#the result is 0B

### 6.1.19 isprime

command:

isprime a

prototype:

result = isprime(a)

01- prime，00-not prime

a : argument a in Hexadecimal

example:

isprime 0B

#the result is 01

### 6.1.20 is a ecc point

command:

ispoint (Px||Py) (p||a||b)

01- prime，00-not prime

Px||Py : the point (Px, Py) in curve

(p||a||b) : the parameters of curve

01 – is a point; 00 - not a point

example:

ispoint 32C4AE2C1F1981195F9904466A39C9948FE30BBFF2660BE1715A4589334C74C7BC3736A2F4F6779C59BDCEE36B692153D0A9877CC62A474002DF32E52139F0A0 FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFC28E9FA9E9D9F5E344D5A9E4BCF6509A7F39789F515AB8F92DDBCBD414D940E93

#the result is 01

### 6.1.21 point addition

command:

padd (Pax||Pay) (Pbx||Pby) (P||A||B)

prototype:

result = add(Pa, pb) ( may be Pa is equal to Pb)

Pax||Pay : the point (Pax, Pay) in curve

Pbx||Pby : the point (Pbx, Pby) in curve

(p||a||b) : the parameters of curve

example:

padd 32C4AE2C1F1981195F9904466A39C9948FE30BBFF2660BE1715A4589334C74C7BC3736A2F4F6779C59BDCEE36B692153D0A9877CC62A474002DF32E52139F0A0 CD459EA427E560E014F420F502055A20471AAE6B97CD5B66F01D87BAB250138B41DA65A7C7058F965EF911D6F5E45B536626DDE93E687C085EB506DC94BEDF79 FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFC28E9FA9E9D9F5E344D5A9E4BCF6509A7F39789F515AB8F92DDBCBD414D940E93

#the result is A2AC8E7C46DA9BCF17B4B7D41E9EB95F6B2746FCC162DF217523554D6599488A8DD6719817138AFCDF67FF521AAE15B8FB6D9A3F53982B3DC2C1E81F63EADFB3

### 6.1.22 point multiplication

command:

pmul k (Px||Py) (p||a||b)

prototype:

result = [k]\*P

k : scalar number

Px||Py : the point (Px, Py) in curve

(p||a||b) : the parameters of curve

example:

pmul D84DC07A8426395E0CE43AEA82DB9ACCF2568D0F2D63772D9897D1334D1F20C3 32C4AE2C1F1981195F9904466A39C9948FE30BBFF2660BE1715A4589334C74C7BC3736A2F4F6779C59BDCEE36B692153D0A9877CC62A474002DF32E52139F0A0 FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFC28E9FA9E9D9F5E344D5A9E4BCF6509A7F39789F515AB8F92DDBCBD414D940E93

#the result is CD459EA427E560E014F420F502055A20471AAE6B97CD5B66F01D87BAB250138B41DA65A7C7058F965EF911D6F5E45B536626DDE93E687C085EB506DC94BEDF79

### 6.1.23 multiple point multiplication

command:

mpmul k (Px||Py) k2 (P2x||P2y) (p||a||b)

prototype:

result = [k](Pax||Pay) + [k2](P2x||P2y)

k : scalar number

k2: scalar number

Px||Py : the point (Px, Py) in curve

P2x||P2y : the point (P2x, P2y) in curve

(p||a||b) : the parameters of curve

example:

mpmul

#the result is D6D2FD1E1950EAA66A5B6F91AE90ECDF2E839A3EEAA4F859F7B7561F3749630135AF9B85C8A3DD09521438D922B68F08E79819E4B11C1E0315D8134AB13D3139

## 6.2 RSA

### 6.2.1 keygen

command:

rsa\_keygen e nlen

e : public key

nlen : byte len of modulus n

return code:

{e n d p q dp dq qinv}

example:

rsa\_keygen 010001 64

#the result is

#010001 #89DEDAA0746C2259DD5767A20C6BE19C411A346F1C51A9F2F5BD32B3A679B08E52DF5DFACA79605A57C10861BC8D14B1B6879EEA3EC585389D2483D3FFB276BD #1C4479F4DE09F1BD1519BFA0C505357BB1096B68C098A9CCABBFBFEE7F9A81AD4E62F0A4FC1D4846B7D46CEEB527EFBB5986EC63CAAF2411A4F22682EFE1794D #9D37127EA963C39ECAD6377EC2F2CCFA213471737A943D4B766FBEA8B78E3BF3 #E08018465E942A97C4DEAF382C5D16718B2F220EAA60E5738AF6039E93859E8F #06C4691A64A633B4711CB974CE656F27512895BD97C82BBF44D0D88F209F91BB #616450A39E7E2AA82298F46CAB1A4B1540EB569C041FFB0A8FFB4000B840DC53 #82AD5A6EEA786CCA74F62C0646D38AA11E1AFEBA8E659F43C53C034EB538FC75

### 6.2.2 keygen with p and q

command:

rsa\_keygen\_pq e p q

e : public key

p : the first prime number

q : the second prime number

example:

set p 9D37127EA963C39ECAD6377EC2F2CCFA213471737A943D4B766FBEA8B78E3BF3

set q E08018465E942A97C4DEAF382C5D16718B2F220EAA60E5738AF6039E93859E8F

rsa\_keygen\_pq 010001 $p $q

#the result is

#010001 #89DEDAA0746C2259DD5767A20C6BE19C411A346F1C51A9F2F5BD32B3A679B08E52DF5DFACA79605A57C10861BC8D14B1B6879EEA3EC585389D2483D3FFB276BD #1C4479F4DE09F1BD1519BFA0C505357BB1096B68C098A9CCABBFBFEE7F9A81AD4E62F0A4FC1D4846B7D46CEEB527EFBB5986EC63CAAF2411A4F22682EFE1794D #9D37127EA963C39ECAD6377EC2F2CCFA213471737A943D4B766FBEA8B78E3BF3 #E08018465E942A97C4DEAF382C5D16718B2F220EAA60E5738AF6039E93859E8F #06C4691A64A633B4711CB974CE656F27512895BD97C82BBF44D0D88F209F91BB #616450A39E7E2AA82298F46CAB1A4B1540EB569C041FFB0A8FFB4000B840DC53 #82AD5A6EEA786CCA74F62C0646D38AA11E1AFEBA8E659F43C53C034EB538FC75

### 6.2.3 encription & verify

command:

rsa\_enc e n M

e : public key

n : modulus

M : messages

example:

set e 010001

set n AE98622E42C289499D28B88FAA3007C671111D448FF3B449D35D2C6999CE2C830BDAD6C38BA1AB356E227B3D626A5EB5716C1A1C319A00F7F2909FDBCA3CAECF

set M A6A1E0F26DDCE4D2512010256D3431F77EE9EA99091A7AB30AFB1DEC1B52DEBF0DF420AAB94CC9E50253309B308011246F511EEADD2A3230D6CDB14302502381

rsa\_enc $e $n $M

#the result is 4FF3492B6ED0670EBCF132879ECFDE3FC073AFE5DFC9D7BF21B1D25D59E3BE9FBE8C0637BAC4E2F1677524D45D718ADA67D80FF961A1DB5FD65867E9D3C04E37

### 6.2.4 decryption & sign

command:

rsa\_dec d n C

d : private key

n : modulus

C : ciphertext

example:

set d A6A1E0F26DDCE4D2512010256D3431F77EE9EA99091A7AB30AFB1DEC1B52DEBF0DF420AAB94CC9E50253309B308011246F511EEADD2A3230D6CDB14302502381

set n AE98622E42C289499D28B88FAA3007C671111D448FF3B449D35D2C6999CE2C830BDAD6C38BA1AB356E227B3D626A5EB5716C1A1C319A00F7F2909FDBCA3CAECF

set C 4FF3492B6ED0670EBCF132879ECFDE3FC073AFE5DFC9D7BF21B1D25D59E3BE9FBE8C0637BAC4E2F1677524D45D718ADA67D80FF961A1DB5FD65867E9D3C04E37

rsa\_dec $d $n $C

#the result is A6A1E0F26DDCE4D2512010256D3431F77EE9EA99091A7AB30AFB1DEC1B52DEBF0DF420AAB94CC9E50253309B308011246F511EEADD2A3230D6CDB14302502381

### 6.2.5 crt

command:

rsa\_crt p q dp dq qinv C

p : the first prime number

q : the second prime number

dp: d mod (p-1)

dq : d mod (q-1)

qinv : q\*\*-1 mod p

example:

set p E9CE7787806AD0699F9760DE166D34B41C149823759C82903B6188D8CD3B431F

set q BF2B12F6FB27E92DE31AE5CB3AA7957B4841335F59A057CE25C337E1F3D74E51

set dp B7DC5E50D762A80C37AD2246E399F35523B3EA44104C08BC8585D8C8ACF3CA7F

set dq 5CCFC2D35A3894430D01A0133D14E3C408DE6EDC9A1CF8C4431D3662630A6321

set qinv B059CF71886A2FB1F1C1C7D93C305B2A90A1FCFE0F75CD63919596432C5C3508

set C 4FF3492B6ED0670EBCF132879ECFDE3FC073AFE5DFC9D7BF21B1D25D59E3BE9FBE8C0637BAC4E2F1677524D45D718ADA67D80FF961A1DB5FD65867E9D3C04E37

rsa\_crt $p $q $dp $dq $qinv $C

#the result is A6A1E0F26DDCE4D2512010256D3431F77EE9EA99091A7AB30AFB1DEC1B52DEBF0DF420AAB94CC9E50253309B308011246F511EEADD2A3230D6CDB14302502381

## 6.3 SM2

### 6.3.1 hash z

command:

sm2\_getz ID Pubkey

ID : user ID

Pubkey : sm2 pubkey (x, y)

example:

set id 1E3C

set Pkx 26EA8A3930208EFD9132F71C510AAB57438B3DBC27D304E798ECCAF2A0EA74EB

set Pky 7500D9CFF30E631015C773728E8C2509380A22E1E742B6ABA09DCF857C42CCEA

sm2\_getz $id ${Pkx}${Pky}

#the result is a6ff86b98d0bf084d18a536e2d22816006dbc88eebd78902e3a5d19990fad9a9

### 6.3.2 hash e

command:

sm2\_gete Z M

Z: generated from sm2getz

M : messages

example:

set za F13C4E7998FD743DA0FD887540E803F85A4286241391CA6659CB168B572DE0B1

set M 6D65737361676520646967657374

sm2\_gete $za $M

#the result is fd780e1b1248655747c7842dba95b76be87d42037dd3af9b001ac8f4376c4090

### 6.3.3 keygen

command:

sm2\_keygen

return code:

{privatekey , publickey}

example:

sm2\_keygen

#the result is

#A95DBF5B0A09619CBCF588B7F975AA896E8F3C7BD1C81B0645CD438F0DA23BA2 #986850115D0D006D0FEFA2D68CA950831BFF91655228E072A6CF325922906DEA5DA27C4EE2DD4032C89EA865C00ADD62E668C2739D9A7FA524C7E2E436CAD2EF

### 6.3.4 sign

command:

sm2\_sig Random Prikey Ehash

Random : random number of 16 bytes

Prikey: private key of 16 bytes

Ehash : the digest of hash e

return code :

(R||S)

example:

set rand A8D1A8EB9ED870073D5A75E3D85BA56ED7E8A034F618DD19A5A13E36AA392032

set sk D84DC07A8426395E0CE43AEA82DB9ACCF2568D0F2D63772D9897D1334D1F20C3

set e 7C84316FC719431CA7921ACED955B407600C880F97D21826F438358051D0CB21

sm2\_sig $rand $sk $e

#the result is 53572E8EE06A2DC311ED8A6087E6A0E71C8C42E360B10B55983397964F44ECFF3538D6F877B83AB3C9E298BBA7459C9629B533281A5A823EAC601DE8CFF5A0CB

### 6.3.5 verify

command:

sm2\_ver Pubkey RS Ehash

Pubkey: public key of (Px, Py)

Ehash : the digest of hash e

RS : the signature of SM2

return code :

0- pass

else fail

example:

set pk CD459EA427E560E014F420F502055A20471AAE6B97CD5B66F01D87BAB250138B41DA65A7C7058F965EF911D6F5E45B536626DDE93E687C085EB506DC94BEDF79

set rs 53572E8EE06A2DC311ED8A6087E6A0E71C8C42E360B10B55983397964F44ECFF3538D6F877B83AB3C9E298BBA7459C9629B533281A5A823EAC601DE8CFF5A0CB

set e 7C84316FC719431CA7921ACED955B407600C880F97D21826F438358051D0CB21

sm2\_ver $pk $rs $e

#the result is 0

### 6.3.6 key exchange

command:

sm2\_kex Role OutKeyByteLen ZSelf ZSide KeySelf RKeySelf PubKeySide RPubKeySide

Role role 0-sender， 1-receiver

OutKeyByteLen the size of exchange key

ZSelf the hash z of A (use sm2\_getz)

ZSide the hash z of B(sm2\_getz)

KeySelf the keypair {prikey||pubkey} of A

RKeySelf the temp keypaire {rprikey ||rpubkey} of A (use sm2\_keygen)

PubKeySide the public key of B

RPubKeySide the temp public key of B

return code:

{exchange\_key, s1/sb, s2/sa}

example:

#set a new curve.

set sm2\_p 8542D69E4C044F18E8B92435BF6FF7DE457283915C45517D722EDB8B08F1DFC3

set sm2\_a 787968B4FA32C3FD2417842E73BBFEFF2F3C848B6831D7E0EC65228B3937E498

set sm2\_b 63E4C6D3B23B0C849CF84241484BFE48F61D59A5B16BA06E6E12D1DA27C5249A

set sm2\_n 8542D69E4C044F18E8B92435BF6FF7DD297720630485628D5AE74EE7C32E79B7

set sm2\_gx 421DEBD61B62EAB6746434EBC3CC315E32220B3BADD50BDC4C4E6C147FEDD43D

set sm2\_gy 0680512BCBB42C07D47349D2153B70C4E5D7FDFCBFA36EA1A85841B9E46E09A2

set sm2\_h 00000001

#==============================

#private key dA of A：

set dA 6FCBA2EF9AE0AB902BC3BDE3FF915D44BA4CC78F88E2F8E7F8996D3B8CCEEDEE

#public key PA of A：

set PA 3099093BF3C137D8FCBBCDF4A2AE50F3B0F216C3122D79425FE03A45DBFE16553DF79E8DAC1CF0ECBAA2F2B49D51A4B387F2EFAF482339086A27A8E05BAED98B

#private key dB of B：

set dB 5E35D7D3F3C54DBAC72E61819E730B019A84208CA3A35E4C2E353DFCCB2A3B53

#public key PB of B：

set PB 245493D446C38D8CC0F118374690E7DF633A8A4BFB3329B5ECE604B2B4F37F4353C0869F4B9E17773DE68FEC45E14904E0DEA45BF6CECF9918C85EA047C60A4C

#set digest ZA=H256(ENTLA||IDA||a||b||xG||yG||xA||yA)。

set ZA E4D1D0C3CA4C7F11BC8FF8CB3F4C02A78F108FA098E51A668487240F75E20F31

#set digest ZB=H256(ENTLB||IDB||a||b||xG||yG||xB||yB)。

set ZB 6B4B6D0E276691BD4A11BF72F4FB501AE309FDACB72FA6CC336E6656119ABD67

#temp private key rA of A：

set rdA 83A2C9C8B96E5AF70BD480B472409A9A327257F1EBB73F5B073354B248668563

#temp public key RA of A：

set rPA 6CB5633816F4DD560B1DEC458310CBCC6856C09505324A6D23150C408F162BF00D6FCF62F1036C0A1B6DACCF57399223A65F7D7BF2D9637E5BBBEB857961BF1A

#temp private key rB of B：

set rdB 33FE21940342161C55619C4A0C060293D543C80AF19748CE176D83477DE71C80

#temp public key RB of B：

set rPB 1799B2A2C778295300D9A2325C686129B8F2B5337B3DCF4514E8BBC19D900EE554C9288C82733EFDF7808AE7F27D0E732F7C73A7D9AC98B7D8740A91D0DB3CF4

#sender

sm2\_kex 0 16 $ZA $ZB ${dA}${PA} ${rdA}${rPA} $PB $rPB

#receiver

sm2\_kex 1 16 $ZB $ZA ${dB}${PB} ${rdB}${rPB} $PA $rPA

#the result is

#55b0ac62a6b927ba23703832c853ded4 #284c8f198f141b502e81250f1581c7e9eeb4ca6990f9e02df388b45471f5bc5c #23444daf8ed7534366cb901c84b3bdbb63504f4065c1116c91a4c00697e6cf7a

### 6.3.7 encryption

command:

sm2\_enc Random Pubkey M

Random : random number of 16 bytes

Pubkey: public key of (Px, Py)

M :messages

return code :

(c1||C3||C2)

example:

set rand C9FFCE9C0FDFCD9962C5BF9CF5891F881531760D7D45E2D2E5E54A4006499243

set pk BC4EADB005F9AADF6BB8573DE5C430A12B023A2471402813CB4D066FC3D681648F98951D3EE032E6F4A4AB2B79510D5721767492E94F31B82C1603731E6CB92A

set m ce8f1ce36e5e62b16772

sm2\_enc $rand $pk $m

#the result is E594A5745BBBD5539D68711C64CA55898A284C9081B65CA36E388062045A357C97AEFE68641FAB6E6A3E4E10855C7C3DE9B8F9417381E4FBB020E9303926BC77126d6f6f74993dd43233c284a0840040ef2e77b0383b9ef5f73b1803db7f4503C66D9F3544BCB59239D5

### 6.3.8 decryption

command:

sm2\_dec Prikey C

Prikey: private key of 16 bytes

C : ciphertext

example:

set sk 700BE499A4EFE27A8369F58BFFE0F5563CDFF772E11832254DDE10E324A81755

set c E594A5745BBBD5539D68711C64CA55898A284C9081B65CA36E388062045A357C97AEFE68641FAB6E6A3E4E10855C7C3DE9B8F9417381E4FBB020E9303926BC77126d6f6f74993dd43233c284a0840040ef2e77b0383b9ef5f73b1803db7f4503C66D9F3544BCB59239D5

sm2\_dec $sk $c

#the result is CE8F1CE36E5E62B16772

# 7. random number

command:

rand NumByte smin smax

NumByte: the size of random number

smin : the minimum value of one byte random number

smax: the maximum value of on byte random number

example:

rand 16 0 255

#the result is 7BEB05FC7F348FA2901088B1B2AD0A62

# 8. CRC

## 8.1 crc16

command:

crc16Ibm data

for CRC-16, CRC-IBM, CRC-16/ARC, CRC-16/LHA

crc16AugCcitt data

for CRC-16/AUG-CCITT, CRC-16/SPI-FUJITSU

crc16Buypass data

for CRC-16/BUYPASS, CRC-16/VERIFONE

crc16CcittFalse data

for CRC-16/CCITT-FALSE

crc16Cdma2000 data

for CRC-16/CDMA2000

crc16Dds110 data

for CRC-16/DDS-100

crc16DectR data

for CRC-16/DECT-R, R-CRC-16

crc16DectX data

for CRC-16/DECT-X, X-CRC-16

crc16Dnp data

for CRC-16/DNP

crc16En13757 data

for CRC-16/EN-13757

crc16Genibus data

for CRC-16/GENIBUS, CRC-16/EPC, CRC-16/I-CODE, CRC-16/DARC

crc16Maxim data

for CRC-16/MAXIM

crc16Mcrf4xx data

for CRC-16/MCRF4XX

crc16Riello data

for CRC-16/RIELLO

crc16T10Dif data

for CRC-16/T10-DIF

crc16Teledisk data

for CRC-16/TELEDISK

crc16Tms37157 data

for CRC-16/TMS37157

crc16Usb data

for CRC-16/USB

crcA data

for CRC-A

crcB data

for CRC-B

crc16Ccitt data

for CRC-16/CCITT, CRC-16/CCITT-TRUE, CRC-16/KERMIT, CRC-CCITT

crc16Modbus data

for MODBUS

crc16X25 data

for X-25, CRC-16/IBM-SDLC, CRC-16/ISO-HDLC, CRC-B

crc16Xmodem data

for XMODEM, ZMODEM, CRC-16/ACORN

# 9. util

## 9.1 dec2hex

command:

dec2hex dec

example：

dec2hex 16

#the result is 10

## 9.2 endian

command:

endian str

example：

endian 00112233445566778899

#the result is 99887766554433221100

## 9.3 hex2bin

command:

Hex2bin str

example：

hex2bin 303132333435

#the result is 012345

## 9.4 bin2hex

command:

bin2hex str

example：

bin2hex 012345

#the result is 303132333435