Snooper

APDU script program language

Programmer's Reference Manual

About this document

为实现 apdu 脚本简单化以及规范化，taoism 6 版对常规版的脚本工具进行了较大修改，本手册中涉及的软件为 snoopers taoism 6 版，适用于 0.0.6.7 以上版本，本手册为第 3 部分，脚本语法、使用以及函数列表，请见第 1、2、4 部分。

目　录

Set 函数列表

0.0.5.4 新增加部分

加入关键字

jcop22_ext_auth_2
jcop22_ext_auth_kmc_2
mac_auto_mac_2
upload_2
delete_2
select_2
des_des_mac_2
des_3des_mac_2

功能与原来的一样，对第二设备进行操作。

0.0.5.5 新增加部分

关于ZY_RF_Monitor的使用，请见Snooper.Spec-008-ZY_RF_Monitor_使用手册

map_m1_keya

map_m1_keya	变量名 = map_m1_keya(M1密钥keya)

map_m1_keyb

map_m1_keyb	变量名 = map_m1_keyb(M1密钥keyb)
	keya = ffffffffffff keyb = ffffffffffff m1key = call calc_m1_key($keya, $keyb) if $m1key != 0B 54 57 07 45 FE 3A E7 ? pause endif keya = a0a1a2a3a4a5 keyb = b0b1b2b3b4b5

	mlkey = call calc_ml_key($keya, $keyb) if $mlkey != 8C 7F 46 D7 6C E0 12 66 ? pause endif keya = 4D 3A 99 C3 51 DD keyb = 1A 98 2C 7E 45 9A mlkey = call calc_ml_key($keya, $keyb) if $mlkey != D8 09 C4 6A 74 84 A1 34 ? pause endif end calc_ml_key: keya1 = getpara keyb1 = getpara dkeya = map_ml_keya($keya1) dkeyb = map_ml_keyb($keyb1) key1 = reverse_block_byte($dkeya) key2 = reverse_block_byte($dkeyb) x = 3des_encode_ecb(0000000000000000, $key1 $key2) x = reverse_block_byte($x) return $x

reader_dialog

reader2_dialog(0.0.6.9 添加)

reader_dialog	变量名 = reader_dialog() 变量名 = reader_dialog(读卡器窗口从01开始的16进制索引) 变量名 = reader_dialog(“读卡器名称”) 变量名 = reader_dialog(, “标题”) 变量名 = reader_dialog(读卡器窗口从01开始的16进制索引, “标题”)

	变量名 = reader_dialog(“读卡器名称”, “标题”) 返回00成功 返回其他值失败
	a = reader_dialog() a = reader_dialog(03) a = reader_dialog("scm") a = reader2_dialog("scm", "xxxxxxxxxxxx")

close_first_device

close_first_device	变量名 = close_first_device() 返回空 注：返回值以后可能被修改
	a = messagebox("请使用接触界面") a = close_first_device() a = reader_dialog()

0.0.5.6 新增加部分

prime_gen

prime_gen	变量名 = prime_gen(16进制素数字节长度)

	prime = prime_gen(40)
	
	res = prime_test($prime)
	if $res != 00
	  ?
	  pause
	endif

	//prime = FE8477E417A6579A5B7266B30E0C5F4F68FE1E701904360F39EA09046A350392008732B16B0201655D1B0ED13A9230BF36911DD1742037A12967506F698363DF //res = 00

big_mod_inv_2_32

big_mod_inv_2_32	变量名 = big_mod_inv_2_32(4字节整数)

	a = 11223397 b = big_mod_inv_2_32($a) c = mul($a, $b) c1 = big_mul($a, $b) if $c != 00000001 ? pause endif

	//a = 11223397
	//b = 693B7C27
	//c = 00000001
	//c1 = 070B025A00000001

connectless_crc16_ab

connectless_crc16_ab	变量名 = connectless_crc16_ab(icv, data)
	a = connectless_crc16_ab(6363, e040) if $a != 3db5 ? pause endif a = connectless_crc16_ab(ffff, e040) if $a != daa4 ? pause endif

obe_crc16

obe_crc16	变量名 = obe_crc16(data)
	a = obe_crc16(D4C1423731374E5500000000FFFFFFFF) if $a != 2c0e ? pause endif

unpack80

unpack80	变量名 = unpack80(data) 此函数检查数据格式，去掉以80 00 00结尾的数据，如果不存在类似的尾部数据，则返回原值
	a = unpack80(aabb) a = unpack80(aabb80) a = unpack80(( )80)

	a = unpack80(00000000) a = unpack80(0000008000)

	//a = AABB
	//a = AABB
	//a = 00
	//a = 00000000
	//a = 000000

pack80_len

pack80_len	变量名 = pack80_len(data, 16进制对齐长度)
	a = pack80_len(1122, 20) b = unpack80($a) c = pack80_len(112233, 04) d = pack80_len(11223344, 04)
	此函数按16进制长度进行数据对齐，如果数据本身符合长度要求，则不再添加80 00 00 对齐

pack00_len

pack00_len	变量名 = pack00_len(data, 16进制对齐长度)
	a = pack00_len(1122, 20) c = pack80_len(112233, 04) d = pack80_len(11223344, 04)
	此函数按16进制长度进行数据对齐，如果数据本身符合长度要求，则不再添加00 00 00 对齐

oem_sm4_encrdata

oem_sm4_encrdata	变量名 = oem_sm4_encrdata(key, data, mode)
	将数据前面添加长度，后边补上80，然后用sm4做ecb加密 oem_sm4_encrdata(key, data, mode) 金融，加油数据加密，密钥直接加密/解密，不需要取随机数，不需要计算过程密钥 mode=0，加密 mode=1，解密

oem_sm4mac

oem_sm4mac	变量名 = oem_sm4mac(key, data, rand, mode)
	自动补80，然后按sm4算mac oem_sm4mac(key,data,rand,mode) mode bit 0位为0时，金融，加油计算MAC，当数据长度为8的整数倍时仍然补80，最低位为1时，长度为8的整数倍时不补80. mode bit 1位为0时，产生mac为4字节，bit1为1时，产生mac为8字节。 0 强制补80，4字节mac

	1 长度为8时，不自动补80，4字节mac 2 强制补80，8字节mac 3 长度为8时，不自动补80，8字节mac

oem_sm4js

oem_sm4js	变量名 = oem_desjs(key, data, mode)
	sm4做ecb模式计算 mode=00 加密 mode=01 解密

memset

memset	变量名 = memset(dest, offset, value, len)
	value长度可以超过1字节

memcpy

memcpy	变量名 = memset(dest, offset, value)

memxor

memxor	变量名 = memxor(dest, offset, value)

memor

memor	变量名 = memor(dest, offset, value)

memand

memand	变量名 = memand(dest, offset, value)

des_subkey

des_subkey	变量名 = des_subkey(数据，密钥)
	支持多级分散

des_diversify

des_diversify	变量名 = des_diversify(数据，密钥)
	支持多级分散

sm4_subkey

sm4_subkey	变量名 = sm4_subkey(数据，密钥)
	支持多级分散

sm4_diversify

sm4_diversify	变量名 = sm4_diversify(数据，密钥)
	支持多级分散

0.0.5.7 新增加部分

new_sm2_keyexchange

new_sm2_keyexchange	变量名 = new_sm2_exchange(角色（00表示发起方，01表示响应方），己方私钥，己方临时私钥，对方公钥，对方临时公钥，己方ID，对方ID)

```
a                = new_sm2_generate_keypair()
self_pri         = left( $a, 32 )
self_pub         = right( $a, 64 )

a                = new_sm2_generate_keypair()
self_tmp_pri     = left( $a, 32 )
self_tmp_pub     = right( $a, 64 )

a                = new_sm2_generate_keypair()
other_pri        = left( $a, 32 )
other_pub        = right( $a, 64 )

a                = new_sm2_generate_keypair()
other_tmp_pri    = left( $a, 32 )
other_tmp_pub    = right( $a, 64 )

self_id          = dup( 16, 11 )
other_id         = dup( 16, 22 )

keyab            = new_sm2_keyexchange( 00, $self_pri, $self_tmp_pri,
$other_pub, $other_tmp_pub, $self_id, $other_id )

keyba            = new_sm2_keyexchange( 01, $other_pri, $other_tmp_pri,
$self_pub, $self_tmp_pub, $other_id, $self_id )

if $keyab != $keyba
    ?
    pause
endif

self_pri         =
7D511F0A173101AD4B1E204956946F6C6E1B0A087C9F5AFC405751CD74C87429
self_pub         =
13F17A6A022AFFE04523A1BABE9A65C3BB3B90EDB57404F1D4CC80C048E57BB507EF820388F07
71D2C5AA92DAE53B75732C8A774F838E1095FC2E4E836F7FB29
```

```
self_tmp_pri     =
16332BBB07B929915707348063EF6B922A7578C7203D405514CB2B5A6A1F6D93
self_tmp_pub     =
D15AA0A9EB024EE12AF054D6BF25A8887CF6C5CC9910DD535C6E8D2DAD0DD2707A378F38876B3
7675B985591E403028FA8E10BD0020F7F27608CA4FDF2A95745

other_pri        =
581C3B64332F12386F874C02278249A0531906B8715F44B16A1F3155230C76CD
other_pub        =
1D9E4517A0483378B59CCD8CD65D5A6F180FA85850102E0CFCE0FD6E27B38C4D7434456860CF9
292B0E29505F7CE8A23031F6997EFBEF90C18FAA1EB596A2F94

other_tmp_pri    =
2B3B097C799A6F9333BA3D785CC26B37031456B663EB605F0C5233C718E337D8
other_tmp_pub    =
5FFE69C7756B2B14B3CC131A10AB0E0DAD9C5778F89E52CD5C94E354D8F80592ACD3B97849C18
FBFAA38E3456C60C5E7CC5581691681DE57FEDFE1E720165CB2

keyab            = new_sm2_keyexchange( 00, $self_pri, $self_tmp_pri,
$other_pub, $other_tmp_pub, $self_id, $other_id )

keyba            = new_sm2_keyexchange( 01, $other_pri, $other_tmp_pri,
$self_pub, $self_tmp_pub, $other_id, $self_id )

if $keyab != CF41314ECF7C0FEDB67CC92D62C5434C0E33C07264120F85990E5B98EC88B4F5
    ?
    pause
endif

if $keyba != CF41314ECF7C0FEDB67CC92D62C5434C0E33C07264120F85990E5B98EC88B4F5
    ?
    pause
endif
```

otp_sm3

otp_sm3	变量名 = otp_sm3(key, t, tc, c, q)
	// {0X12,0X34,0X56,0X78,0X90,0XAB,0XCD,0XEF,0X12,0X34,0X56,0X78,0X90,0XAB,0XCD,0XEF} , // 1313655030, 1234, "5678", 854787 key = 1234567890abcdef 1234567890abcdef

```
t = expr16( 1313655030 )
tc = 01
c = expr16( 1234 )
q = "5678"

a = otp_sm3( $key, $t, $tc, $c, $q )

the_correct_result = expr16( 854787, 4 )
if $a != $the_correct_result
    ?
    pause
endif

//
{0X12,0X34,0X56,0X78,0X90,0XAB,0XCD,0XEF,0X12,0X34,0X56,0X78,0X90,0XAB,0XCD,0XEF}
,
// 1313998979, 1234, "5678", 814095
t = expr16( 1313998979 )
tc = 01
c = expr16( 1234 )
q = "5678"

a = otp_sm3( $key, $t, $tc, $c, $q )

the_correct_result = expr16( 814095, 4 )
if $a != $the_correct_result
    ?
    pause
endif
```

otp_sm4

otp_sm4	变量名 = otp_sm4(key, t, tc, c, q)

```
//
{0X12,0X34,0X56,0X78,0X90,0XAB,0XCD,0XEF,0X12,0X34,0X56,0X78,0X90,0XAB,0XCD,0XEF}
,
// 1340783053, 1234, "5678", 446720

key = 1234567890abcdef 1234567890abcdef
t = expr16( 1340783053 )
tc = 01
c = expr16( 1234 )
```

```
q = "5678"

a = otp_sm4( $key, $t, $tc, $c, $q )

the_correct_result = expr16( 446720, 4 )
if $a != $the_correct_result
    ?
    pause
endif

//
{0X12,0X34,0X56,0X78,0X90,0XAB,0XCD,0XEF,0XAB,0XCD,0XEF,0X12,0X34,0X56,0X78,0X90}
,
// 1340783416, 5621, "3698", 49845
key = 1234567890abcdef abcdef1234567890
t = expr16( 1340783416 )
tc = 01
c = expr16( 5621 )
q = "3698"

a = otp_sm4( $key, $t, $tc, $c, $q )

the_correct_result = expr16( 49845, 4 )
if $a != $the_correct_result
    ?
    pause
endif
```

pkcs_encrypt_pad

pkcs_encrypt_pad	变量名 = pkcs_encrypt_pad(待加密数据, 16进制模长)
	返回填充后的明文

pkcs_encrypt_unpad

pkcs_encrypt_unpad	变量名 = pkcs_encrypt_unpad(填充后的明文)
	返回去掉填充的原始数据

pkcs_encrypt_pad_ff

pkcs_encrypt_pad_ff	变量名 = pkcs_encrypt_pad_ff(待加密数据, 16进制模长)
	返回填充后的明文，使用ff填充

pkcs_encrypt_unpad_ff

pkcs_encrypt_unpad_ff	变量名 = pkcs_encrypt_unpad_ff(填充后的明文)
	返回去掉填充的原始数据，与pkcs_encrypt_unpad相比，对填充的要求不

	同

pkcs_sign_pad

pkcs_sign_pad	变量名 = pkcs_sign_pad(待签名数据, 16进制模长)
	返回填充后的明文

pkcs_sign_unpad

pkcs_sign_unpad	变量名 = pkcs_sign_unpad(填充后的明文)
	返回去掉填充的原始数据

show_ansi_string

show_ansi_string	变量名 = show_ansi_string(数据, 转义标识)

show_utf8_string

show_utf8_string	变量名 = show_utf8_string(数据, 转义标识)

show_utf7_string

show_utf7_string	变量名 = show_utf7_string(数据, 转义标识)

show_utf16_little_string

show_utf16_little_string	变量名 = show_utf16_little_string(数据, 转义标识)

show_utf16_big_string

show_utf16_big_string	变量名 = show_utf16_big_string(数据, 转义标识)

show_string

show_string	变量名 = show_string(数据,代码页, 转义标识)

0.0.5.8 新增加部分

new_sm2_ecc_kdf

new_sm2_ecc_kdf	变量名 = new_sm2_ecc_kdf(Z, 16进制k长度)

u2f_escape_command

u2f_escape_command	变量名 = u2f_escape_command(1字节控制字符, apdu指令)

open_process_lock

open_process_lock	变量名 = open_process_lock(字符串格式互斥量名字)
	a = open_process_lock("abc") aa = 88 b = release_process_lock("abc") 进程锁可以跨进程使用，同一个工具的不同脚本窗口无效，open与release之间的操作被视为一个整体执行，其他进程则等待此进程。

release_process_lock

release_process_lock	变量名 = release_process_lock(字符串格式互斥量名字)

u2f_escape_send

u2f_escape_send	变量名 = u2f_escape_send(tpdu数据)

u2f_escape_recv

u2f_escape_recv	变量名 = u2f_escape_recv()

getbit

getbit	变量名 = getbit(数据, 16进制从0开始字节索引，高位，低位)
	xx = getbit(aabbccdd, 02, 07, 00)

getbit_right

getbit_right	变量名 = getbit_right(数据, 16进制从0开始字节索引，高位，低位)
	xx = getbit_right(aabbccdd, 02, 07, 00)

aes128_encode_ofb

aes128_encode_ofb	变量名 = aes128_encode_ofb(icv, data, key)

aes128_decode_ofb

aes128_decode_ofb	变量名 = aes128_decode_ofb(icv, data, key)

aes192_encode_ofb

aes192_encode_ofb	变量名 = aes192_encode_ofb(icv, data, key)

aes192_decode_ofb

aes192_decode_ofb	变量名 = aes192_decode_ofb(icv, data, key)

aes256_encode_ofb

aes256_encode_ofb	变量名 = aes256_encode_ofb(icv, data, key)

aes256_decode_ofb

aes256_decode_ofb	变量名 = aes256_decode_ofb(icv, data, key)

sm4_encode_ofb

sm4_encode_ofb	变量名 = sm4_encode_ofb(icv, data, key)

sm4_decode_ofb

sm4_decode_ofb	变量名 = sm4_decode_ofb(icv, data, key)

aes128_encode_ctr

aes128_encode_ctr	变量名 = aes128_encode_ctr(icv, data, key)

aes128_decode_ctr

aes128_decode_ctr	变量名 = aes128_decode_ctr(icv, data, key)

aes192_encode_ctr

aes192_encode_ctr	变量名 = aes192_encode_ctr(icv, data, key)

aes192_decode_ctr

aes192_decode_ctr	变量名 = aes192_decode_ctr(icv, data, key)

aes256_encode_ctr

aes256_encode_ctr	变量名 = aes256_encode_ctr(icv, data, key)

aes256_decode_ctr

aes256_decode_ctr	变量名 = aes256_decode_ctr(icv, data, key)

sm4_encode_ctr

sm4_encode_ctr	变量名 = sm4_encode_ctr(icv, data, key)

sm4_decode_ctr

sm4_decode_ctr	变量名 = sm4_decode_ctr(icv, data, key)

gbk_gb18030

gbk_gb18030	变量名 = gbk_gb18030(gbk)

utf8_gb18030

utf8_gb18030	变量名 = utf8_gb18030(utf8)

utf16_little_gb18030

utf16_little_gb18030	变量名 = utf16_little_gb18030(utf16_little)

gb18030_utf16_little

gb18030_utf16_little	变量名 = gb18030_utf16_little_gb18030(gb18030)

utf16_big_gb18030

utf16_big_gb18030	变量名 = utf16_big_gb18030(utf16_big)

aes128_cmac

aes128_cmac	变量名 = aes128_cmac(icv, data, key)

aes192_cmac

aes192_cmac	变量名 = aes192_cmac(icv, data, key)

aes192_cmac

aes192_cmac	变量名 = aes192_cmac(icv, data, key)

sm4_cmac

sm4_cmac	变量名 = sm4_cmac(icv, data, key)

ansi_utf16_little

ansi_utf16_little	变量名 = ansi_utf16_little(ansi_format_string)

ansi_utf16_big

ansi_utf16_big	变量名 = ansi_utf16_big(ansi_format_string)

utf8_utf16_little

utf8_utf16_little	变量名 = utf8_utf16_little(utf8_format_string)

utf8_utf16_big

utf8_utf16_big	变量名 = utf8_utf16_big(utf8_format_string)

aes128_encode_cfb

aes128_encode_cfb	变量名 = aes128_encode_cfb(icv, data, key, slen)

aes128_decode_cfb

aes128_decode_cfb	变量名 = aes128_decode_cfb(icv, data, key, slen)

aes192_encode_cfb

aes192_encode_cfb	变量名 = aes192_encode_cfb(icv, data, key, slen)

aes192_decode_cfb

aes192_decode_cfb	变量名 = aes192_decode_cfb(icv, data, key, slen)

aes256_encode_cfb

aes256_encode_cfb	变量名 = aes256_encode_cfb(icv, data, key, slen)

aes256_decode_cfb

aes256_decode_cfb	变量名 = aes256_decode_cfb(icv, data, key, slen)

sm4_encode_cfb

sm4_encode_cfb	变量名 = sm4_encode_cfb(icv, data, key, slen)

sm4_decode_cfb

sm4_decode_cfb	变量名 = sm4_decode_cfb(icv, data, key, slen)

des_encode_cfb

des_encode_cfb	变量名 = des_encode_cfb(icv, data, key, slen)

des_decode_cfb

des_decode_cfb	变量名 = des_decode_cfb(icv, data, key, slen)

3des_encode_cfb

3des_encode_cfb	变量名 = 3des_encode_cfb(icv, data, key, slen)

3des_decode_cfb

3des_decode_cfb	变量名 = 3des_decode_cfb(icv, data, key, slen)

3des24_encode_cfb

3des24_encode_cfb	变量名 = 3des24_encode_cfb(icv, data, key, slen)

3des24_decode_cfb

3des24_decode_cfb	变量名 = 3des24_decode_cfb(icv, data, key, slen)

des_encode_ofb

des_encode_ofb	变量名 = des_encode_ofb(icv, data, key)

des_decode_ofb

des_decode_ofb	变量名 = des_decode_ofb(icv, data, key)

3des_encode_ofb

3des_encode_ofb	变量名 = 3des_encode_ofb(icv, data, key)

3des_decode_ofb

3des_decode_ofb	变量名 = 3des_decode_ofb(icv, data, key)

3des24_encode_ofb

3des24_encode_ofb	变量名 = 3des24_encode_ofb(icv, data, key)

3des24_decode_ofb

3des24_decode_ofb	变量名 = 3des24_decode_ofb(icv, data, key)

des_encode_ctr

des_encode_ctr	变量名 = des_encode_ctr(icv, data, key)

des_decode_ctr

des_decode_ctr	变量名 = des_decode_ctr(icv, data, key)

3des_encode_ctr

3des_encode_ctr	变量名 = 3des_encode_ctr(icv, data, key)

3des_decode_ctr

3des_decode_ctr	变量名 = 3des_decode_ctr(icv, data, key)

3des24_encode_ctr

3des24_encode_ctr	变量名 = 3des24_encode_ctr(icv, data, key)

3des24_decode_ctr

3des24_decode_ctr	变量名 = 3des24_decode_ctr(icv, data, key)

des_cmac

des_cmac	变量名 = des_cmac(icv, data, key)

3des_cmac

3des_cmac	变量名 = 3des_cmac(icv, data, key)

3des24_cmac

3des24_cmac	变量名 = 3des24_cmac(icv, data, key)

0.0.5.9 新增加部分

rockey4_encode

rockey4_encode	变量名 = rockey4_encode(data, key)

rockey4_decode

rockey4_decode	变量名 = rockey4_decode(data, key)

cbor 函数组

cbor_init(index, cbor_data)
cbor_stream_get(index)
	用于cbor编码、解码，index是cbor编码对象索引，有效范围是00-03。
	简单用法
	a = cbor_init(00, "")
	// 或
	a = cbor_init(00, 有效数据)
	
	使用cbor_stream_get(index)取得相应的值

cbor_stream_analy(cbor_data) cbor_stream_to_encode(index, cbor_data) cbor_stream_to_decode(index, cbor_data)	调用init之后，可以调用几个函数进行分析、或生成编码
cbor_get_current(index)	生成解码脚本，可逐项取得对应的值
cbor_write_positive(index, value) cbor_write_negative(index, value) cbor_write_any(index, value) cbor_write_map_start(index, value) cbor_write_byte_string(index, value) cbor_write_text_string(index, value) cbor_write_array_start(index, value) cbor_write_bool(index, true_of_false)	cbor编码函数

blakehash 函数组

blake224_hash (data) blake256_hash (data) blake384_hash (data) blake512_hash (data)	

blake224_hash_init (data) blake224_hash_update (data) blake224_hash_dofinal (data) blake256_hash_init (data) blake256_hash_update (data) blake256_hash_dofinal (data) blake384_hash_init (data) blake384_hash_update (data) blake384_hash_dofinal (data) blake512_hash_init (data) blake512_hash_update (data) blake512_hash_dofinal (data)	

oid_encrypt

oid_encrypt	变量名 = oid_encrypt(字符串格式的oid)
	x = oid_encrypt("1.2.840.113549.2.5") //x = //————————————MD5 //—2A864886F70D0205

oid_decrypt

oid_decrypt	变量名 = oid_decrypt(oid) 返回utf16-little格式的字符串

	x = oid_encrypt("1.2.840.113549.2.5")
	y = show_utf16_little_string(oid_decrypt($x))
	
	//x =
	//————————————MD5
	//—2A864886F70D0205
	//y =
	//————————————1.2.840.113549.2.5
	
	//————————————MD5
	
	//[]=======================[]
	//[] 1.2.840.113549.2.5 []
	//[]=======================[]

crc32

crc32	变量名 = crc32(4字节crc初始化向量, 待crc的数据)
	x = crc32(00000000, 74)

	x = crc32($x, 68656D653A206A656B796C6C2D7468656D652D736C6174650A)

rc4_crypt

rcr_crypt	变量名 = rcr_crypt(数据，密钥) x = rc4_crypt(1122, 1122) y = rc4_crypt($x, 1122)

getopenfilename

getopenfilename	变量名 = getopenfilename(“图片文件\0*.jpg”)

getsavefilename

getsavefilename	变量名 = getsavefilename(“图片文件\0*.jpg”)

CreateProcess

CreateProcess	变量名 = CreateProcess(EXE, Para, b_wait_to_quit)

0.0.5.9 版----删除记录

1) 删除了superlong关键字，对应功能可以写在一行
2) 去掉了 pause 关键字的部分功能，pause 只支持暂停
3) 删除了runtime_replace关键字，不再支持此功能

0.0.6.0 新增加部分

libusb 函数族，返回 00 成功，非 00 失败

libusb_init	
libusb_exit	
libusb_finddevice_by_interface	vid, pid, interface_class
libusb_opendevice	
libusb_closedevice	
libusb_claim_interface	interface_u1
libusb_control_transfer	request_type_u1, request_u1, value_u2, index_u2, read_len_u2
libusb_control_transfer_raw	raw_data
libusb_bulk_transfer	endpoint, data
iap2_checksum	data
libusb_set_timeout	time_ms
libusb_reset_device	

libusb_set_config	config_ul

PLC 机械控制函数族，返回 00 成功，非 00 失败

fn_openplc	port_num, baudrate
fn_closeplc	
fn_runlamp_control	onflag_01_or_00, fixtype_t409c_01_t409e_02
fn_arm_zero	armid_based_0, speed_hz, low_speed_hz, direct_00_indirect_01, isplus
fn_arm_motion	armid_based_0, speed_hz, low_speed_hz, direct_00_indirect_01, ispuleses, rate, iscam, isplus
fn_calculate_pulse_num	micro_step, unit_range, goal_distance

记号控制函数族，可用于实现全局变量功能，被注册的记号在脚本窗口打开期间有效

mark_test	记号名
mark_set	记号名, 记号值 用于注册或更新记号内容
mark_get	记号名 取得记号内容
mark_del	记号名 删除某个记号
mark_remove_all	删除所有记号

```
clear

x = mark_test( "already_reset" )

if $x == 00
    // already_reset is not exist
    x = mark_set( "already_reset", 88 )
    x = mark_set( "eee", 88 )
    zzz = 00
else
    zzz = add( $zzz, 01 )
endif

x = mark_get( "already_reset" )
if $x == ""
    ?
    pause
endif
```

```
x = mark_del( "eee" )

x = mark_test( "eee" )
if $x != 00
    ?
    pause
endif
```

json 函数族（目前只支持读取功能）

json_load	“json文件名” 返回00表示成功 其他值表示失败

	x json_load($settings_name)
	if $x != 00
	?
	? "找不到配置文件"
	x = show_ansi_string($settings_name)
	pause
	endif
	
	// 加载成功会列出json的所有节点的名字与内容概要
	//---int KeyVersion = 24 (0x18)
	//---int Option = 2 (0x02)
	//-----string SCP11c.CardGroupID = 01020304
	//-------string SCP11c.Curve.k = 01
	//-----int SCP11c.Digest = 33 (0x21)
	//-----string SCP11c.HostID = 80808080808080
	//-----bool SCP11c.IncludeHostID = true
	//-----int SCP11c.KID_CA_KLOC = 16 (0x10)
	//-----string SCP11c.KPR =
	//-----int SCP11c.KeyLength = 16 (0x10)
	//-----string SCP11c.Key_DEK =
	//---string SCPType = _
	//---int sLevel = 0 (0x00)

json_dump	显示json文件的内容与概要
json_get	“json节点名”
	scp11c_oce_1_0 = json_get("scp11c.oce[1][0]")
	注意： 这里取得的内容与原始内容有差异 ● 对于int型，如果文件中的内容是0x11，取出来的是

	00000011，要注意长度 ● 对于string型，如果文件中的内容是”1234”，取出来的是31323334，要使用parse函数转换成二进制

h2s

h2s	h2s函数是之前的num2txt的别名，这样的做法是因为代码中使用h2s进行转换，有了这样的别名用起来比较熟悉
num2txt	变量名 = num2txt(输入数据)
	输入是00，输出是”00”，也就是3030，无空格。

parse

parse	parse函数是之前的txt2num的别名，这样的做法是因为代码中使用parse进行转换，有了这样的别名用起来比较熟悉
txt2num	变量名 = txt2num (输入数据)
	输入是”00”， 也就是3030，输出是00

fthub_init

fthub_open

fthub_close

fthub_init	x = fthub_init()
	为简单起见，只支持一个hub
fthub_open	x = fthub_open(16进制端口号，00-07)
	为简单起见，只支持一个hub
fthub_close	x = fthub_close(16进制端口号，全部关闭)
	为简单起见，只支持一个hub

```
x    = fthub_init()
if $x != 00
    ?
    pause
endif

num = 00
for i = 0 to 400
    x    = fthub_open( $num )
    if $x != 00
        ?
        pause
    endif

    sleep 100
    num = add( $num, 01 )
```

```
    if $num > 07
        num = 00
    endif

    x   = fthub_close( $num )        // 全部关闭

    if $x != 00
        ?
        pause
    endif

next i
```

unix_utc

utc_unix

unix_utc	x = unix_utc(11228844)
	4字节unix时间戳
utc_unix	y = utc_unix(790210062300) 或 y = utc_unix(19790210062300)
	6或7个字节的utc时间

fthub_init_2

fthub_open_2

fthub_close_2

fthub_init_2	与fthub_init相同，此函数支持带电源的hub
fthub_open_2	与fthub_open相同，此函数支持带电源的hub
fthub_close_2	与fthub_close相同，此函数支持带电源的hub

check_tlv

check_dcep_tlv

check_tlv	y = check_tlv($input) clear prompt off t = dup(130, 55) x = 40($t) y = check_tlv($x) if $y != 00

```
    ?
    pause
endif

t = dup( 130, 55 )
x = 40 81( $t )
y = check_tlv( $x )
if $y != 00
    ?
    pause
endif
```

```
//--(40)-82==55==55- 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55
//------------------------ Compare ------------------------
//------ 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ----  00 01 02
// 0000  FF                                                 ----  00
//
//
//------
//------ Pause at Line 10
// The script has paused, press 'F5' to continue
//--(40)-81--82- 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
//-- 55 55 55 55 55
```

此函数检查tlv，如果正确返回00，如果不正确，返回非00

check_dcep_tlv

```
y = check_dcep_tlv( $input )
```

此函数与check_tlv类似

aes128_encode_ccm

aes128_decode_ccm

aes192_encode_ccm

aes192_decode_ccm

aes256_encode_ccm

aes256_decode_ccm

aes_128_encode_ccm	key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
	nonce = 00 00 00 03 02 01 00 A0 A1 A2 A3 A4 A5
	Adata = 00 01 02 03 04 05 06 07
	// Adata =
	len_adata = datalen($adata)
	m = 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E
	
	x = aes128_encode_ccm($nonce, $adata, $m, $key, 08)
	
	cipher = mid($x, int(16 + 0x$len_adata), 23)
	返回数据分为4个部分
	● 前0x10个字节为B0
	● 外部传入的adata，与输入数据长度相同
	● 密文，与明文长度相同
	● mac，长度由外面指定

aes_128_decode_ccm	y = aes128_decode_ccm($nonce, $adata, $cipher, $key, 08)

0.0.6.1 新增加部分

aes128_encode_gcm

aes128_decode_gcm

aes192_encode_gcm

aes192_decode_gcm

aes256_encode_gcm

aes256_decode_gcm

aes_128_encode_gcm	Key = f5a2b27c74355872eb3ef6c5feafaa740e6ae990d9d48c3bd9bb8235e589f010 IV = 58d2240f580a31c1d24948e9 Tag = 15e051a5e4a5f5da6cea92e2ebee5bac tem_xx = aes256_encode_gcm($iv, "", "", $key, 10) if $tem_xx != $tag ? pause endif xx = aes256_decode_gcm($iv, , , $key, $tag) if $xx != "" ? pause endif Key = 11754cd72aec309bf52f7687212e8957 IV = 3c819d9a9bed087615030b65 Tag = 250327c674aaf477aef2675748cf6971 tem_xx = aes128_encode_gcm($IV, , , $Key, 10) if $tem_xx != $tag ? pause endif xx = aes128_decode_gcm($iv, , , $key, $tag) if $xx != "" ?

	pause endif mac_length = 04 Key = 0e00c76561d2bd9b40c3c15427e2b08f IV = 492cadaccd3ca3fbc9cf9f06eb3325c4e159850b0dbe98199b89b7af528806610b6f63998e1eae80c348e74cbb921d8326631631fc6a5d304f39166daf7ea15fa1977f101819adb510b50fe9932e12c5a85aa3fd1e73d8d760af218be829903a77c63359d75edd91b4f6ed5465a72662f5055999e059e7654a8edc921aa0d496 PT = fef03c2d7fb15bf0d2df18007d99f967c878ad59359034f7bb2c19af120685d78e32f6b8b83b032019956ca9c0195721476b85 AAD = d8f1163d8c840292a2b2dacf4ac7c36aff8733f18fabb4fa5594544125e03d1e6e5d6d0fd61656c8d8f327c92839ae5539bb469c9257f109ebff85aad7bd220fdaa95c022dbd0c7bb2d878ad504122c943045d3c5eba8f1f56c0 CT = 4f6cf471be7cbd2575cd5a1747aea8fe9dea83e51936beac3e68f66206922060c697ffa7af80ad6bb68f2cf4fc97416ee52abe Tag = e20b6655 tem_xx = aes128_encode_gcm($IV, $PT, $AAD, $Key, 04) if $tem_xx != $CT $Tag ? pause endif xx = aes128_decode_gcm($iv, $tem_xx, $aad, $key, $tag) x = datalen($xx) x = sub($x, $mac_length) yy = hmid($xx, 00, $x) if $yy != $pt ? pause endif 返回数据分为2个部分 ● 第一部分密文，与明文长度一致 ● mac，长度由外面指定
aes_128_decode_gcm	y = aes128_decode_gcm(iv, data, aad, key, mac)

asn1_reset

asn1_length_begin

asn1_data

asn1_pushtlv

asn1_length_end

asn1_final

用法	x = asn1_reset() x = asn1_data(31) inc_indent x = asn1_length_begin() x = asn1_data(30) inc_indent x = asn1_length_begin() x = asn1_data(06) inc_indent x = asn1_length_begin() x = asn1_data(04 00 7F 00 07 02 02 02) x = asn1_length_end() dec_indent x = asn1_data(02) inc_indent x = asn1_length_begin() x = asn1_data(02) x = asn1_length_end() dec_indent x = asn1_length_end() dec_indent x = asn1_pushtlv(02, 9999) x = asn1_pushtlv(02, 88888888) x = asn1_pushtlv(02, 777777777777777777777) x = asn1_length_end() dec_indent x = asn1_final()
	//x = 3126300D060804007F000702020202010202029999020488888888020B777777777777

	7777777777 (T:31, 0:0000, L:0026) SET (T:30, 0:0002, L:000D) SEQUENCE (T:06, 0:0004, L:0008) OBJECT IDENTIFIER : bsiTA ((T:02, 0:000E, L:0001) INTEGER : 02 (T:02, 0:0011, L:0002) INTEGER : 9999 (T:02, 0:0015, L:0004) INTEGER : 88888888 (T:02, 0:001B, L:000B) INTEGER : 77777777777777777777

settlv

用法	z = settlv($x, 31 30 02, aaaaaaaaaaaaaaaaaaaaaaa)
	//z = 31313018060804007F0007020202020CAAAAAAAAAAAAAAAAAAAAAAAA02029999020488888888020B77777777777777777777 (T:31, 0:0000, L:0031) SET (T:30, 0:0002, L:0018) SEQUENCE (T:06, 0:0004, L:0008) OBJECT IDENTIFIER : bsiTA (0.4.0.127.0.7.2.2.2) (T:02, 0:000E, L:000C) INTEGER : AAAAAAAAAAAAAAAAAAAAAAAA (T:02, 0:001C, L:0002) INTEGER : 9999 (T:02, 0:0020, L:0004) INTEGER : 88888888 (T:02, 0:0026, L:000B) INTEGER : 77777777777777777777
用法	z = settlv($x, 31 02, aaaaaaaaaaaaaaaaaaaaaaa, 02)
	//z = 3127300D060804007F0007020202020102020299990204888888888020CAAAAAAAAAAAAAAAAAAAAAAA (T:31, 0:0000, L:0027) SET (T:30, 0:0002, L:000D) SEQUENCE (T:06, 0:0004, L:0008) OBJECT IDENTIFIER : bsiTA (0.4.0.127.0.7.2.2.2) (T:02, 0:000E, L:0001) INTEGER : 02 (T:02, 0:0011, L:0002) INTEGER : 9999 (T:02, 0:0015, L:0004) INTEGER : 88888888 (T:02, 0:001B, L:000C) INTEGER : AAAAAAAAAAAAAAAAAAAAAAAA

gettlv_bypath

settlv_bypath

用法	der = 3037A003020102020A34A20E85000000002CD6300D06092A864886F70D010105050030153113301106 0A0992268993F22C6401191603636F6D path = "/0/3/0/0/0" path = "/0/0/0" value = gettlv_bypath($der, $path)

	new_der = settlv_bypath($der, $path, 888888) compare $der, $new_der
	//---------- Compare ---------- //------ 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ---- 00 01 02 03 04 05 06 07 08 09 0A 0B //0000 30 37 A0 03 02 01 02 02 0A 34 A2 0E 85 00 00 00 ---- 30 39 A0 05 02 03 88 88 88 02 0A 34 //0010 00 2C D6 30 0D 06 09 2A 86 48 86 F7 0D 01 01 05 ---- 00 00 00 2C D6 30 0D 06 09 2A 86 48 //0020 05 00 30 15 31 13 30 11 06 0A 09 92 26 89 93 F2 ---- 01 05 05 00 30 15 31 13 30 11 06 0A //0030 2C 64 01 19 16 03 63 6F 6D -- -- ---- 93 F2 2C 64 01 19 16 03 63 6F 6D //

passport_calcdigit

用法	x = passport_calcdigit("11223344") //x = 38

ext_euclid

用法	根据扩展欧几里得算法，根据输入a，b的值，得到x，y与sign标志
	满足a * x - b * y = 1，或b * y - a * x = 1
	
	// 扩展欧几里得算法，普通小数计算
	a = 80
	b = 5b
	xy = ext_euclid($a, $b)
	x = mid($xy, 0, 4)
	y = mid($xy, 4, 4)
	gcd = mid($xy, 8, 4)
	sign = mid($xy, 12, 1)
	
	x_a = big_mul($a, $x)
	y_b = big_mul($b, $y)
	
	res = big_sub($x_a, $y_b)
	
	//a = 80
	//b = 5B
	//xy = 000000200000002DFF
	//x = 00000020
	//y = 0000002D
	//gcd = 00000001
	//sign = FF
	//x_a = 1000
	//y_b = 0FFF

	//res = 0001

```
a               = new_sm2_generate_keypair()
self_pri        = left( $a, 32 )
self_pub        = right( $a, 64 )

a               = new_sm2_generate_keypair()
self_tmp_pri    = left( $a, 32 )
self_tmp_pub    = right( $a, 64 )

a               = new_sm2_generate_keypair()
other_pri       = left( $a, 32 )
other_pub       = right( $a, 64 )

a               = new_sm2_generate_keypair()
other_tmp_pri   = left( $a, 32 )
other_tmp_pub   = right( $a, 64 )

self_id         = dup( 16, 11 )
other_id        = dup( 16, 22 )

keyab           = new_sm2_keyexchange( 00, $self_pri, $self_tmp_pri, $other_pub,
$other_tmp_pub, $self_id, $other_id )

keyba           = new_sm2_keyexchange( 01, $other_pri, $other_tmp_pri, $self_pub,
$self_tmp_pub, $other_id, $self_id )

if $keyab != $keyba
    ?
    pause
endif

self_pri        = 7D511F0A173101AD4B1E204956946F6C6E1B0A087C9F5AFC405751CD74C87429
self_pub        =
13F17A6A022AFFE04523A1BABE9A65C3BB3B90EDB57404F1D4CC80C048E57BB507EF820388F0771D2C5AA92DAE53
B75732C8A774F838E1095FC2E4E836F7FB29

self_tmp_pri    = 16332BBB07B929915707348063EF6B922A7578C7203D405514CB2B5A6A1F6D93
self_tmp_pub    =
D15AA0A9EB024EE12AF054D6BF25A8887CF6C5CC9910DD535C6E8D2DAD0DD2707A378F38876B37675B985591E403
028FA8E10BD0020F7F27608CA4FDF2A95745

other_pri       = 581C3B64332F12386F874C02278249A0531906B8715F44B16A1F3155230C76CD
```

```
other_pub       =
1D9E4517A0483378B59CCD8CD65D5A6F180FA85850102E0CFCE0FD6E27B38C4D7434456860CF9292B0E29505F7CE
8A23031F6997EFBEF90C18FAA1EB596A2F94

other_tmp_pri   = 2B3B097C799A6F9333BA3D785CC26B37031456B663EB605F0C5233C718E337D8
other_tmp_pub   =
5FFE69C7756B2B14B3CC131A10AB0E0DAD9C5778F89E52CD5C94E354D8F80592ACD3B97849C18FBFAA38E3456C60
C5E7CC5581691681DE57FEDFE1E720165CB2

keyab           = new_sm2_keyexchange( 00, $self_pri, $self_tmp_pri, $other_pub,
$other_tmp_pub, $self_id, $other_id )

keyba           = new_sm2_keyexchange( 01, $other_pri, $other_tmp_pri, $self_pub,
$self_tmp_pub, $other_id, $self_id )

if $keyab != CF41314ECF7C0FEDB67CC92D62C5434C0E33C07264120F85990E5B98EC88B4F5
    ?
    pause
endif

if $keyba != CF41314ECF7C0FEDB67CC92D62C5434C0E33C07264120F85990E5B98EC88B4F5
    ?
    pause
endif
```

big_mod_u32

用法	

	big_a = random(13)
	small_b = 5b
	m = big_div_u32($big_a, $small_b)
	r = big_mod_u32($big_a, $small_b)
	xx = big_mul($m, $small_b)
	xx = big_add($xx, $r)
	if $xx != $big_a
	    ?
	    pause
	endif
	
	//big_a = 8A696809FB9895936B5B547E6D
	//small_b = 5B
	//m = 000000018560CAA8BE29073F11E1FBC3
	//r = 0000001C
	//xx = 8A696809FB9895936B5B547E51

	//xx = 8A696809FB9895936B5B547E6D

0.0.6.2 新增加部分

new_sm9_gen_sign_master_key

new_sm9_get_sign_master_key

new_sm9_gen_sign_pri_key

new_sm9_sign

new_sm9_verify

用法	// 生成主公私钥
	x = new_sm9_gen_sign_master_key()
	pri = mid($x, 0, 32)
	pub = mid($x, 32)
	
	// 生成签名私钥
	dek = new_sm9_gen_sign_pri_key(ansi_string("Alice"), 01, $pri)
	
	// 采用规范中的标准数据
	// 签名主密钥和用户签名密钥产生过程中的相关值：
	// 签名主私钥ks：0130E7 8459D785 45CB54C5 87E02CF4 80CE0B66 340F319F 348A1D5B 1F2DC5F4
	// 签名主公钥Ppub-s = [ks]P2 = (xPpub-s , yPpub-s)：
	// 坐标xPpub-s：(9F64080B 3084F733 E48AFF4B 41B56501 1CE0711C 5E392CFB 0AB1B679 1B94C408,
	// 29DBA116 152D1F78 6CE843ED 24A3B573 414D2177 386A92DD 8F14D656 96EA5E32)
	// 坐标yPpub-s：(69850938 ABEA0112 B57329F4 47E3A0CB AD3E2FDB 1A77F335 E89E1408 D0EF1C25,
	// 41E00A53 DDA532DA 1A7CE027 B7A46F74 1006E85F 5CDFF073 0E75C05F B4E3216D)
	
	ks = 000130E7 8459D785 45CB54C5 87E02CF4 80CE0B66 340F319F 348A1D5B 1F2DC5F4
	pub = new_sm9_get_sign_master_key($ks)
	if $pub != 9F64080B3084F733E48AFF4B41B565011CE0711C5E392CFB0AB1B6791B94C40829DBA116152D1F786CE843ED24A3B573414D2177386A92DD8F14D65696EA5E3269850938ABEA

```
0112B57329F447E3A0CBAD3E2FDB1A77F335E89E1408D0EF1C2541E00A53DDA532DA1A
7CE027B7A46F741006E85F5CDFF0730E75C05FB4E3216D
    ?
    pause
endif

ida = ansi_string( "Alice" )
hid = 01

// 签名私钥dsA = [t2]P1 = (xdsA, ydsA):
// 坐标xdsA: A5702F05 CF131530 5E2D6EB6 4B0DEB92 3DB1A0BC F0CAFF90
523AC875 4AA69820
// 坐标ydsA: 78559A84 4411F982 5C109F5E E3F52D72 0DD01785 392A727B
B1556952 B2B013D3

dsa = new_sm9_gen_sign_pri_key( $ida, $hid, $ks )
if $dsa !=
A5702F05CF1315305E2D6EB64B0DEB923DB1A0BCF0CAFF90523AC8754AA6982078559A
844411F9825C109F5EE3F52D720DD01785392A727BB1556952B2B013D3
    ?
    pause
endif

// 签名步骤中的相关值:
// 待签名消息M: Chinese IBS standard
// M 的16 进制表示: 4368696E 65736520 49425320 7374616E 64617264

m   = ansi_string( "Chinese IBS standard" )

r_s = new_sm9_sign( $m, $pub, $dsa )
r   = mid( $r_s, 0, 32 )
s   = mid( $r_s, 32 )

res = new_sm9_verify( $m, $ida, $hid, $pub, $r_s )
if $res != 00
    ?
    pause
endif

? "success"
```

new_sm9_get_saved_random

用法	取得sm9运算中用到的随机数

new_sm9_gen_encrypt_master_key

new_sm9_get_encrypt_master_key

new_sm9_gen_encrypt_pri_key

new_sm9_encrypt_block

new_sm9_encrypt_stream

new_sm9_decrypt_block

new_sm9_decrypt_stream

用法

```
x       = new_sm9_gen_encrypt_master_key()
pri     = mid( $x, 0, 32 )
pub     = mid( $x, 32 )

enc_pri = new_sm9_gen_encrypt_pri_key( ansi_string( "Bob" ), 03,
$pri )

// 加密主私钥ke：01EDEE 3778F441 F8DEA3D9 FA0ACC4E 07EE36C9 3F9A0861
8AF4AD85 CEDE1C22
// 加密主公钥Ppub-e = [ke]P1= (xPpub-e , yPpub-e)：
// 坐标xPpub-e：787ED7B8 A51F3AB8 4E0A6600 3F32DA5C 720B17EC A7137D39
ABC66E3C 80A892FF
// 坐标yPpub-e：769DE617 91E5ADC4 B9FF85A3 1354900B 20287127 9A8C49DC
3F220F64 4C57A7B1
// 加密私钥生成函数识别符hid：0x03
// 实体B 的标识IDB: Bob
// IDB 的16 进制表示：426F62

pri     = 0001EDEE 3778F441 F8DEA3D9 FA0ACC4E 07EE36C9 3F9A0861
8AF4AD85 CEDE1C22
pub     = new_sm9_get_encrypt_master_key( $pri )
if $pub !=
787ED7B8A51F3AB84E0A66003F32DA5C720B17ECA7137D39ABC66E3C80A892FF769DE6
1791E5ADC4B9FF85A31354900B202871279A8C49DC3F220F644C57A7B1
    ?
    pause
endif
```

```
idb     = ansi_string( "Bob" )
hid     = 03

// 计算deB=[t2]P2=(xdeB, ydeB):
// 21
// 坐标xdeB: (94736ACD 2C8C8796 CC4785E9 38301A13 9A059D35 37B64141
40B2D31E ECF41683,
//          115BAE85 F5D8BC6C 3DBD9E53 42979ACC CF3C2F4F 28420B1C
B4F8C0B5 9A19B158)
// 坐标ydeB: (7AA5E475 70DA7600 CD760A0C F7BEAF71 C447F384 4753FE74
FA7BA92C A7D3B55F,
//          27538A62 E7F7BFB5 1DCE0870 4796D94C 9D56734F 119EA447
32B50E31 CDEB75C1)

deb     = new_sm9_gen_encrypt_pri_key( $idb, $hid, $pri )
if $deb !=
94736ACD2C8C8796CC4785E938301A139A059D3537B6414140B2D31EECF41683115BAE
85F5D8BC6C3DBD9E5342979ACCCF3C2F4F28420B1CB4F8C0B59A19B1587AA5E47570DA
7600CD760A0CF7BEAF71C447F3844753FE74FA7BA92CA7D3B55F27538A62E7F7BFB51D
CE08704796D94C9D56734F119EA44732B50E31CDEB75C1
    ?
    pause
endif

m       = ansi_string( "Chinese IBE standard" )

cipher  = new_sm9_encrypt_block( $m, $idb, $hid, $pub )
plain   = new_sm9_decrypt_block( $cipher, $idb, $hid, $deb )

if $plain != $m
    ?
    pause
endif

cipher  = new_sm9_encrypt_stream( $m, $idb, $hid, $pub )
plain   = new_sm9_decrypt_stream( $cipher, $idb, $hid, $deb )

if $plain != $m
    ?
    pause
endif
```

	? "success"

new_sm9_key_pack

new_sm9_key_unpack

用法	// 加密主私钥ke：01EDEE 3778F441 F8DEA3D9 FA0ACC4E 07EE36C9 3F9A0861 8AF4AD85 CEDE1C22
	// 加密主公钥Ppub-e = [ke]P1= (xPpub-e , yPpub-e)：
	// 坐标xPpub-e：787ED7B8 A51F3AB8 4E0A6600 3F32DA5C 720B17EC A7137D39 ABC66E3C 80A892FF
	// 坐标yPpub-e：769DE617 91E5ADC4 B9FF85A3 1354900B 20287127 9A8C49DC 3F220F64 4C57A7B1
	// 加密私钥生成函数识别符hid：0x03
	// 实体B 的标识IDB: Bob
	// IDB 的16 进制表示：426F62
	
	pri = 0001EDEE 3778F441 F8DEA3D9 FA0ACC4E 07EE36C9 3F9A0861 8AF4AD85 CEDE1C22
	pub = new_sm9_get_encrypt_master_key($pri)
	if $pub != 787ED7B8A51F3AB84E0A66003F32DA5C720B17ECA7137D39ABC66E3C80A892FF769DE61791E5ADC4B9FF85A31354900B202871279A8C49DC3F220F644C57A7B1
	    ?
	    pause
	endif
	
	idb = ansi_string("Bob")
	hid = 03
	
	// 计算deB=[t2]P2=(xdeB, ydeB)：
	// 21
	// 坐标xdeB：(94736ACD 2C8C8796 CC4785E9 38301A13 9A059D35 37B64141 40B2D31E ECF41683,
	// 115BAE85 F5D8BC6C 3DBD9E53 42979ACC CF3C2F4F 28420B1C B4F8C0B5 9A19B158)
	// 坐标ydeB：(7AA5E475 70DA7600 CD760A0C F7BEAF71 C447F384 4753FE74 FA7BA92C A7D3B55F,
	// 27538A62 E7F7BFB5 1DCE0870 4796D94C 9D56734F 119EA447 32B50E31 CDEB75C1)
	
	deb = new_sm9_gen_encrypt_pri_key($idb, $hid, $pri)

```
if $deb !=
94736ACD2C8C8796CC4785E938301A139A059D3537B6414140B2D31EECF41683115BAE
85F5D8BC6C3DBD9E5342979ACCCF3C2F4F28420B1CB4F8C0B59A19B1587AA5E47570DA
7600CD760A0CF7BEAF71C447F3844753FE74FA7BA92CA7D3B55F27538A62E7F7BFB51D
CE08704796D94C9D56734F119EA44732B50E31CDEB75C1
    ?
    pause
endif

len = 10
x   = new_sm9_key_pack( $idb, $hid, $pub, $len )
k   = hmid( $x, 00, $len )
c   = hmid( $x, $len )

y   = new_sm9_key_unpack( $c, $idb, $deb, $len )
if $y != $k
    ?
    pause
endif

// 换一个长度
len = 30
x   = new_sm9_key_pack( $idb, $hid, $pub, $len )
k   = hmid( $x, 00, $len )
c   = hmid( $x, $len )

y   = new_sm9_key_unpack( $c, $idb, $deb, $len )
if $y != $k
    ?
    pause
endif


? "success"
```

new_sm9_h1

用法	x = new_sm9_h1(112233, B640000002A3A6F1D603AB4FF58EC74449F2934B18EA8BEEE56EE19CD69ECF25)

new_sm9_key_exchange

用法	// 加密主密钥和用户加密密钥产生过程中的相关值：
	// 加密主私钥ke：02E65B 0762D042 F51F0D23 542B13ED 8CFA2E9A 0E720636 1E013A28 3905E31F
	// 加密主公钥Ppub-e = [ke]P1 = (xPpub-e , yPpub-e)：
	// 坐标xPpub-e：91745426 68E8F14A B273C094 5C3690C6 6E5DD096 78B86F73 4C435056 7ED06283
	// 坐标yPpub-e：54E598C6 BF749A3D ACC9FFFE DD9DB686 6C50457C FC7AA2A4 AD65C316 8FF74210
	// 加密私钥生成函数识别符hid：0x02
	// 实体A 的标识IDA：Alice
	
	pri = 0002E65B 0762D042 F51F0D23 542B13ED 8CFA2E9A 0E720636 1E013A28 3905E31F
	pub = new_sm9_get_encrypt_master_key($pri)
	if $pub != 9174542668E8F14AB273C0945C3690C66E5DD09678B86F734C4350567ED0628354E598C6BF749A3DACC9FFFEDD9DB6866C50457CFC7AA2A4AD65C3168FF74210
	?
	pause
	endif
	
	ida = ansi_string("Alice")
	idb = ansi_string("Bob")
	hid = 02
	
	dea = new_sm9_gen_encrypt_pri_key($ida, $hid, $pri)
	if $dea != 0FE8EAB395199B56BF1D75BD2CD610B6424F08D1092922C5882B52DCD6CA832A7DA57BC50241F9E5BFDDC075DD9D32C7777100D736916CFC165D8D36E0634CD783A457DAF52CAD464C903B26062CAF937BB40E37DADED9EDA401050E49C8AD0C6970876B9AAD1B7A50BB4863A11E574AF1FE3C5975161D73DE4C3AF621FB1EFB
	?
	pause
	endif
	
	deb = new_sm9_gen_encrypt_pri_key($idb, $hid, $pri)
	if $deb != 74CCC3AC9C383C60AF083972B96D05C75F12C8907D128A17ADAFBAB8C5A4ACF701092FF4DE89362670C21711B6DBE52DCD5F8E40C6654B3DECE573C2AB3D29B244B0294AA04290E1524FF3E3DA8CFD432BB64DE3A8040B5B88D1B5FC86A4EBC18CFC48FB4FF37F1E27727464F3C34E2153861AD08E972D1625FC1A7BD18D5539

```
    ?
    pause
endif

// 取rA 为：5879 DD1D51E1 75946F23 B1B41E93 BA31C584 AE59A426 EC1046A4
D03B06C8
// 计算RA=[rA]QB=( xRA , yRA)：
// 坐标xRA：7CBA5B19 069EE66A A79D4904 13D11846 B9BA76DD 22567F80
9CF23B6D 964BB265
// 坐标yRA：A9760C99 CB6F7063 43FED056 37085864 958D6C90 902ABA7D
405FBEDF 7B781599

ra      = 00005879 DD1D51E1 75946F23 B1B41E93 BA31C584 AE59A426
EC1046A4 D03B06C8
rra     = 7CBA5B19 069EE66A A79D4904 13D11846 B9BA76DD 22567F80
9CF23B6D 964BB265 A9760C99 CB6F7063 43FED056 37085864 958D6C90
902ABA7D 405FBEDF 7B781599

// 取rB 为: 018B98 C44BEF9F 8537FB7D 071B2C92 8B3BC65B D3D69E1E
EE213564 905634FE
// 计算RB=[rB]QA=( xRB , yRB)：
// 坐标xRB：861E9148 5FB7623D 2794F495 031A3559 8B493BD4 5BE37813
ABC710FC C1F34482
// 坐标yRB：32D906A4 69EBC121 6A802A70 52D5617C D430FB56 FBA729D4
1D9BD668 E9EB9600
rb      = 00018B98 C44BEF9F 8537FB7D 071B2C92 8B3BC65B D3D69E1E
EE213564 905634FE
rrb     = 861E9148 5FB7623D 2794F495 031A3559 8B493BD4 5BE37813
ABC710FC C1F34482 32D906A4 69EBC121 6A802A70 52D5617C D430FB56
FBA729D4 1D9BD668 E9EB9600

len     = 10
keyb    = new_sm9_key_exchange( 00, $ida, $hid, $idb, $pub, $dea, $ra,
$rra, $rrb, $len )
if $keyb != C5C13A8F59A97CDEAE64F16A2272A9E7
    ?
    pause
endif

keya    = new_sm9_key_exchange( 01, $idb, $hid, $ida, $pub, $deb, $rb,
$rrb, $rra, $len )
if $keya != C5C13A8F59A97CDEAE64F16A2272A9E7
    ?
```

	pause endif ? "success"

无版本

show_wsq_image

用法	wsq = FFA0FFA8007B4E4953545F434F4D20390A5049585F574944544820353135 0A5049585F 48454947485420343837.....FECC7FC4FC0F46E146D4A1FA9DA45F5BC5FC4162CF8D DA79FC85100D4E4667103C24323F61D47B0ED122C7B4276CED0B287B549DDEB63763AB 4D34FA1ED3FA4FFFA1 x = show_wsq_image($wsq)

save_wsq_to_tmp

用法	wsq = FFA0FFA8007B4E4953545F434F4D20390A5049585F574944544820353135 0A5049585F 48454947485420343837.....FECC7FC4FC0F46E146D4A1FA9DA45F5BC5FC4162CF8D DA79FC85100D4E4667103C24323F61D47B0ED122C7B4276CED0B287B549DDEB63763AB 4D34FA1ED3FA4FFFA1 x = save_wsq_to_bmp($wsq, "finger.bmp")

big_div

big_mod

big_mod_div

```
x   = random( 80 )
y   = random( 60 )

q   = big_div( $x, $y )
r   = big_mod( $x, $y )

z   = big_mul( $q, $y )
z   = big_add( $z, $r )
if $z != $x
    ?
```

```
    pause
endif
```

getnextprime

getprevprime

```
x    = random( 80 )
y    = getnextprime( $x )
z    = getprevprime( $x )
compare $z, $x
compare $x, $y
compare $z, $y


//---------------------- Compare ----------------------
//------ 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ----  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
//0000  1B 66 90 A0 B9 BD 36 EA 62 70 52 3A C9 77 C9 6D  ----  1B 66 90 A0 B9 BD 36 EA 62 70 52 3A C9 77 C9 6D
//0010  A9 B0 D6 64 AB 2F 09 FD 8A D5 8F 42 75 24 B3 E1  ----  A9 B0 D6 64 AB 2F 09 FD 8A D5 8F 42 75 24 B3 E1
//0020  B5 51 A5 DD 8F 9B 7A 82 6C DC 75 9E 37 5E 85 EE  ----  B5 51 A5 DD 8F 9B 7A 82 6C DC 75 9E 37 5E 85 EE
//0030  07 EF 0E D7 FE 38 EC 56 71 4A B6 3A 48 7A 3F 91  ----  07 EF 0E D7 FE 38 EC 56 71 4A B6 3A 48 7A 3F 91
//0040  A8 6F 63 60 D0 79 00 97 41 24 42 43 22 0E 23 DF  ----  A8 6F 63 60 D0 79 00 97 41 24 42 43 22 0E 24 06
//
//
//---------------------- Compare ----------------------
//------ 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ----  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
//0000  1B 66 90 A0 B9 BD 36 EA 62 70 52 3A C9 77 C9 6D  ----  1B 66 90 A0 B9 BD 36 EA 62 70 52 3A C9 77 C9 6D
//0010  A9 B0 D6 64 AB 2F 09 FD 8A D5 8F 42 75 24 B3 E1  ----  A9 B0 D6 64 AB 2F 09 FD 8A D5 8F 42 75 24 B3 E1
//0020  B5 51 A5 DD 8F 9B 7A 82 6C DC 75 9E 37 5E 85 EE  ----  B5 51 A5 DD 8F 9B 7A 82 6C DC 75 9E 37 5E 85 EE
//0030  07 EF 0E D7 FE 38 EC 56 71 4A B6 3A 48 7A 3F 91  ----  07 EF 0E D7 FE 38 EC 56 71 4A B6 3A 48 7A 3F 91
//0040  A8 6F 63 60 D0 79 00 97 41 24 42 43 22 0E 24 06  ----  A8 6F 63 60 D0 79 00 97 41 24 42 43 22 0E 24 65
//
//
//---------------------- Compare ----------------------
//------ 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ----  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
//0000  1B 66 90 A0 B9 BD 36 EA 62 70 52 3A C9 77 C9 6D  ----  1B 66 90 A0 B9 BD 36 EA 62 70 52 3A C9 77 C9 6D
//0010  A9 B0 D6 64 AB 2F 09 FD 8A D5 8F 42 75 24 B3 E1  ----  A9 B0 D6 64 AB 2F 09 FD 8A D5 8F 42 75 24 B3 E1
//0020  B5 51 A5 DD 8F 9B 7A 82 6C DC 75 9E 37 5E 85 EE  ----  B5 51 A5 DD 8F 9B 7A 82 6C DC 75 9E 37 5E 85 EE
//0030  07 EF 0E D7 FE 38 EC 56 71 4A B6 3A 48 7A 3F 91  ----  07 EF 0E D7 FE 38 EC 56 71 4A B6 3A 48 7A 3F 91
//0040  A8 6F 63 60 D0 79 00 97 41 24 42 43 22 0E 23 DF  ----  A8 6F 63 60 D0 79 00 97 41 24 42 43 22 0E 24 65
//
```

```
new_sm2_get_yflag( point )

new_sm2_computer_y( yflag, point_x )

new_sm2_check_point( point )

new_sm2_point_add( p, q )

new_sm2_point_double( p )

new_sm2_check_pubkey( pubkey )

new_sm2_ecdh_gm_map( k, point )

new_sm2_kp( k, p )

new_sm2_kp_add_lq( k, p, l, q )

new_ecc_get_yflag( point )
```

```
// sm2
P               = FF FF FF FE FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00
FF FF FF FF FF FF FF FF
A               = FF FF FF FE FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00
FF FF FF FF FF FF FF FC
B               = 28 E9 FA 9E 9D 9F 5E 34 4D 5A 9E 4B CF 65 09 A7 F3 97 89 F5 15 AB 8F 92
DD BC BD 41 4D 94 0E 93
GX              = 32 C4 AE 2C 1F 19 81 19 5F 99 04 46 6A 39 C9 94 8F E3 0B BF F2 66 0B E1
71 5A 45 89 33 4C 74 C7
GY              = BC 37 36 A2 F4 F6 77 9C 59 BD CE E3 6B 69 21 53 D0 A9 87 7C C6 2A 47 40
02 DF 32 E5 21 39 F0 A0
N               = FF FF FF FE FF FF FF FF FF FF FF FF FF FF FF FF 72 03 DF 6B 21 C6 05 2B
53 BB F4 09 39 D5 41 23

h               = 01
hex_len_in_byte = 20

r               = new_ecc_initialize( $p, $a, $b, $gx, $gy, $n, $h, $hex_len_in_byte )
if $r != 00
    ?
    pause
endif

keypair         = new_ecc_generate_keypair()
len             = datalen( $keypair )
prilen          = big_div( $len, 03 )
```

```
if $len > 00
    pri = hmid( $keypair, 00, $prilen )
    pub = hmid( $keypair, $prilen )
else
    // error
    pause
endif

r               = new_ecc_check_pubkey( $pub )
if $r != 00
    ?
    pause
endif

pri             = 127C3DEF3718230E28B0134A41B22F1312F5676A47C9073005A77DA51FCE4101
pub             =
D72A6958CC1540E7BE420F07D3C1E51F77B5E12EAF050F9F28E6E19ACED4E99BA272A6AB31EC61C7FD0A28A972
E1B02569ED6ADF98DD2F32D9987EF59CE974F2
x               = new_ecc_get_pubkey( $pri )
if $x != $pub
    ?
    pause
endif

r               = new_ecc_check_pubkey( $pub )
if $r != 00
    ?
    pause
endif

pri1            = hrandom( $prilen )
pri2            = hrandom( $prilen )
pub1            = new_ecc_get_pubkey( $pri1 )
pub2            = new_ecc_get_pubkey( $pri2 )

x               = new_ecc_ecdh_agreement( $pri1, $pub2 )
y               = new_ecc_ecdh_agreement( $pri2, $pub1 )
if $x != $y
    ?
    pause
endif

pri1            = 86F31E7A3A3291CD81DB5A6A028CEB0A66DAD60C5C45EB0CE16AA0B6BE7BB7DC
```

```
pri2            = 52ED1AC791700ACB4D24781EB50AB3EB1BB944A1FA52169137B5DBB8A7F5AAD9
pub1            =
4F6B8DF5ED018EFEE3E3EA65E71CE7F2C8A49FB1DD7E0BD97A673B4BCC4A00A45D1884822638037477A7BFE359
ED4173F344383984521D7F2802B43AEEC46E8C
pub2            =
9CB4DA1FE75CA340187D63ACFDFCAA226E217376FAA0DB7AB496E29A697E7A3ACA04262C5537304C16C03BBCC2
2764A602AE41DEF043F4936B6EB96FBCD7DAB9
stx             =
EFF2E277542ABBF7CB371376B13B6FF9643ACE7E654464E72C6543312C901BC0A002201195DFAB4801BD5F69E6
6B5FAFF2231EA49ECCB1B41701A6EB5887225D
x               = new_ecc_ecdh_agreement( $pri1, $pub2 )
y               = new_ecc_ecdh_agreement( $pri2, $pub1 )
if $x != $stx
    ?
    pause
endif
if $y != $stx
    ?
    pause
endif

hash            = hrandom( $prilen )
r_s             = new_ecc_sign( $pri1, $hash ) // or new_ecc_sign( $pri1, $hash, $random )

kk              = new_ecc_verify( $pub1, $hash, $r_s )
if $kk != 00
    ?
    pause
endif

pub1            =
4F6B8DF5ED018EFEE3E3EA65E71CE7F2C8A49FB1DD7E0BD97A673B4BCC4A00A45D1884822638037477A7BFE359
ED4173F344383984521D7F2802B43AEEC46E8C
hash            = 22E4FCC99451AF042CF1DBA03964ED662FE28F6AA82DCC98AAC47590285AC4B4
r_s             =
0D9A85758DCEB848C02AC9692989A31D9CEE4FBB92BC46FD2D8872A661045A1972A05C0A789DF5174614BA58DD
0355E178346F83C73388E08B604048AB50BFA2

kk              = new_ecc_verify( $pub1, $hash, $r_s )
if $kk != 00
    ?
    pause
endif
```

```
// 计算y值功能
keypair          = new_ecc_generate_keypair()
len              = datalen( $keypair )
prilen           = big_div( $len, 03 )
if $len > 00
    pri = hmid( $keypair, 00, $prilen )
    pub = hmid( $keypair, $prilen )
else
    // error
    pause
endif
yflag            = new_sm2_get_yflag( $pub )

pointy           = new_sm2_computer_y( $yflag, $pub )
pub_y            = hmid( $pub, $prilen )
if $pub_y != $pointy
    ?
    pause
endif

pri              = 0B0A7D0D759505E56F5C38DF5D1359901CB6238D31B77B5673C94B7420A64CB6
pub              =
DA2EDEA262AFC868C70AB76A057D24A3B0784C2FCC830E375544E30643007369E224F16425487C2CFA813DE499
5A9FC0F3C592B2E3ABA4E775D1927B69ABF6C1
yflag            = new_sm2_get_yflag( $pub )

pointy           = new_sm2_computer_y( $yflag, $pub )
pub_y            = hmid( $pub, $prilen )
if $pub_y != $pointy
    ?
    pause
endif

yflag            = new_ecc_get_yflag( $pub )

pointy           = new_ecc_computer_y( $yflag, $pub )
pub_y            = hmid( $pub, $prilen )
if $pub_y != $pointy
    ?
    pause
endif
```

```
SM2_P1          =
0555AF337D1994F08E44A738BDE8A37A0066C03726B84EAD7B83F9B2F5CE95F2A705FCCC0A6703A3C29DA01FC9
1FEF9D7EF52AC4D98B256B01724C91E65B8E84
r               = new_sm2_check_point( $sm2_p1 )
if $r != 00
    ?
    pause
endif

SM2_P2          =
5E7DF5818844E122238A067284749144A3DE8D180D9AE58D014C77E302C6A2FA94DF18470F8563D87FEC74C16D
0A60E13CEEEADB1F3157AA58622AD5B6D272FC
r               = new_sm2_check_point( $sm2_p2 )
if $r != 00
    ?
    pause
endif


SM2_P1_ADD_P2   =
8C52A0DDF1294926E6AC640CD591C7E9E0728BFCE7F54027BB18B012AB6EADBB20B96FCDE41160B73C6E461F8E
00B1D6403C022CDE0F7BA16344250B46C88C09
r               = new_sm2_check_point( $SM2_P1_ADD_P2 )
if $r != 00
    ?
    pause
endif

a               = new_ecc_point_add( $sm2_p1, $sm2_p2 )
if $a != $sm2_p1_add_p2
    ?
    pause
endif

b               = new_sm2_point_add( $sm2_p1, $sm2_p2 )
if $b != $sm2_p1_add_p2
    ?
    pause
endif

sm2_p1_double   =
67B0230B2872D8130F3DD64D5EDC13366B55DA9FEB6F46666DA52DBF9659125B5E41306D3F11C232861FEF844F
36E8E25829F2F1A9D59B81D43BBA93E43F2339
```

```
a               = new_sm2_point_double( $sm2_p1 )
if $a != $sm2_p1_double
    ?
    pause
endif

x               = new_ecc_check_pubkey( $a )
if $x != 00
    ?
    pause
endif

x               = new_sm2_check_pubkey( $a )
if $x != 00
    ?
    pause
endif

a               = new_ecc_point_double( $pub1 )
if $a !=
42CDA3170F0413A7FF57D47C77AA31E8463987BB92FE2A4613210BC6564FEF32C8D58CE31F66F099F531BB1E34
814316D221FE3D04773067B305DFE0A818EE95
    ?
    pause
endif

x               = new_ecc_check_pubkey( $a )
if $x != 00
    ?
    pause
endif


x               = hrandom( $prilen )
a               = new_ecc_kp( $x, $pub1 )
x               = new_ecc_check_pubkey( $a )
if $x != 00
    ?
    pause
endif

l               = hrandom( $prilen )
k               = hrandom( $prilen )
```

```
a               = new_ecc_kp_add_lq( $k, $pub1, $l, $pub2 )
a1              = new_ecc_kp( $k, $pub1 )
a2              = new_ecc_kp( $l, $pub2 )
a3              = new_ecc_point_add( $a1, $a2 )
if $a3 != $a
    ?
    pause
endif
x               = new_ecc_check_point( $a )
if $x != 00
    ?
    pause
endif


x               = new_sm2_check_pubkey( $a )
if $x != 00
    ?
    pause
endif


a               = new_sm2_point_double( $pub1 )
if $a !=
42CDA3170F0413A7FF57D47C77AA31E8463987BB92FE2A4613210BC6564FEF32C8D58CE31F66F099F531BB1E34
814316D221FE3D04773067B305DFE0A818EE95
    ?
    pause
endif

x               = new_sm2_check_pubkey( $a )
if $x != 00
    ?
    pause
endif


x               = hrandom( $prilen )
a               = new_sm2_kp( $x, $pub1 )
x               = new_sm2_check_pubkey( $a )
if $x != 00
    ?
    pause
```

```
endif

l               = hrandom( $prilen )
k               = hrandom( $prilen )
a               = new_sm2_kp_add_lq( $k, $pub1, $l, $pub2 )
a1              = new_sm2_kp( $k, $pub1 )
a2              = new_sm2_kp( $l, $pub2 )
a3              = new_sm2_point_add( $a1, $a2 )
if $a3 != $a
    ?
    pause
endif
x               = new_ecc_check_point( $a )
if $x != 00
    ?
    pause
endif


k               = FFF3A31F73D09FAC889D4DA2EBE9521D15D29F5F15BF50219051D42706964F17
h               =
3334C5922D7F14F3C29C52BC0F88014F2E65A52B1BC8C73A38254D99CE1B7B4FE38B126B602E41B11357DC96758C4B465CFF95CA25B2C25BDF26B7300A43AF7A
z               =
9F84D746A33BF6ACAEAA9157D95161332AFBD00C5847001A49E00B72534CECF9B0EB79B512474B0B638AFE5C96F54C801679999814DB0071B0AB9DF5FA988E14
t               = new_ecc_ecdh_gm_map( $k, $h )
if $z != $t
    ?
    pause
endif


k               = FFF3A31F73D09FAC889D4DA2EBE9521D15D29F5F15BF50219051D42706964F17
h               =
3334C5922D7F14F3C29C52BC0F88014F2E65A52B1BC8C73A38254D99CE1B7B4FE38B126B602E41B11357DC96758C4B465CFF95CA25B2C25BDF26B7300A43AF7A
z               =
9F84D746A33BF6ACAEAA9157D95161332AFBD00C5847001A49E00B72534CECF9B0EB79B512474B0B638AFE5C96F54C801679999814DB0071B0AB9DF5FA988E14
t               = new_sm2_ecdh_gm_map( $k, $h )
if $z != $t
    ?
```

```
    pause
endif
```

file_read_linehex

get_line_count

get_linehex_from_mem

```
x               = file_read_linehex( "所有密钥.txt" )
count           = get_line_count()

line            = get_linehex_from_mem( 00 )
line            = get_linehex_from_mem( 01 )
```

big_sqrt

```
x   = random( 20 )
y   = big_mul( $x, $x )
z   = big_sqrt( $y )
compare $x, $z
```

rockey4_smart_getcrc

```
a = rockey4_smart_getcrc( 88 d7  85 48 c1 91  77 b1 aa 89  d7 eb 80 73  d6 f6 a0 4b  3a 7b
c1 55  bd dc 7b a2  53 44 e2 cb 19 d9 cf c9  32 60 b6 05  61 b1 47 74  6b 59 7f 6a  1b f7
14 4a  89 ae bf 0d  5f 03 6c aa  9a cd 42 0b b5 bf a5 91  6a f6 )
```

ed25519_generate_keypair

ed25519_get_pubkey

ed25519_sign

ed25519_verify

x25519_generate_keypair

x25519_get_pubkey

x25519

```
clear

x           = ed25519_generate_keypair()
```

```
pri         = mid( $x, 0, 32 )
pub         = mid( $x, 32 )

test        = ed25519_get_pubkey( $pri )
if $test != $pub
    ?
    pause
endif

msg         = 11223344
rs          = ed25519_sign( $pri, $msg )

ver         = ed25519_verify( $pub, $msg, $rs )
if $ver != 00
    ?
    pause
endif

// Alice's private key, a:
aprikey = 77076d0a7318a57d3c16c17251b26645df4c2f87ebc0992ab177fba51db92c2a
// Alice's public key, X25519(a, 9):
apubkey = 8520f0098930a754748b7ddcb43ef75a0dbf3a0d26381af4eba4a98eaa9b4e6a
tmp     = x25519_get_pubkey( $aprikey )
if $tmp != $apubkey
    ?
    pause
endif

// Bob's private key, b:
bprikey = 5dab087e624a8a4b79e17f8b83800ee66f3bb1292618b6fd1c2f8b27ff88e0eb
// Bob's public key, X25519(b, 9):
bpubkey = de9edb7d7b7dc1b4d35b61c2ece435373f8343c85b78674dadfc7e146f882b4f
tmp     = x25519_get_pubkey( $bprikey )
if $tmp != $bpubkey
    ?
    pause
endif

// Their shared secret, K:
share   = 4a5d9d5ba4ce2de1728e3bf480350f25e07e21c947d19e3376f09b3c1e161742
tmp     = x25519( $aprikey, $bpubkey )
if $tmp != $share
    ?
```

```
    pause
endif

tmp     = x25519( $bprikey, $apubkey )
if $tmp != $share
    ?
    pause
endif
```

shake128_hash

shake128_hash_init

shake128_hash_update

shake128_hash_dofinal

shake256_hash

shake256_hash_init

shake256_hash_update

shake256_hash_dofinal

```
x = shake128_hash( "" )
if $x != 7F9C2BA4E88F827D616045507605853ED73B8093F6EFBC88EB1A6EACFA66EF26
    ?
    pause
endif

x = shake256_hash( "" )
if $x !=
46B9DD2B0BA88D13233B3FEB743EEB243FCD52EA62B81B82B50C27646ED5762FD75DC4DDD8C0F200CB05019D67
B592F6FC821C49479AB48640292EACB3B7C4BE
    ?
    pause
endif

x = shake128_hash_init()
x = shake128_hash_update( "" )
x = shake128_hash_dofinal( 32 )

x = shake256_hash_init()
```

```
x = shake256_hash_update( "" )
x = shake256_hash_dofinal( 32 )
```

vpn_direct_onlysend

vpn_direct_sendrecv

```
x = vpn_direct_sendrecv( 48535345 0000 0084000008 )

x = vpn_direct_onlysend( 48535345 0800 0084000008 )
x = vpn_direct_sendrecv( 99999999999999999999999999 )
```

systemtime_sub

systemtime_add

systemtime_diff

```
// 时间 - 秒数 = 新时间
x       = systemtime_sub( 20220425120000, hex( 86399 ) )
// 时间 + 秒数 = 新时间
x       = systemtime_add( 20220425120000, hex( 86399 ) )
// 得到两个时间的秒数差
x       = systemtime_diff( 20200425120000, 20060101000000 )
// 由秒数计算分钟值
minute  = big_div( $x, hex( 60 ) )
```

tlcrc

```
x       = tlcrc( 11223344 )
```

sm4_encode_gcm

sm4_decode_gcm

```
iv          = 000012345678000000000ABCD
key         = 0123456789ABCDEFFEDCBA9876543210
plaintext   =
AAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCCDDDDDDDDDDDDDDDDEEEEEEEEEEEEEEEEFFFFFFFFFF
FFFFFFFEEEEEEEEEEEEEEEEAAAAAAAAAAAAAAAA
aad         = FEEDFACEDEADBEEFFEEDFACEDEADBEEFABADDAD2
//    CipherText:              17F399F08C67D5EE19D0DC9969C4BB7D
//                             5FD46FD3756489069157B282BB200735
//                             D82710CA5C22F0CCFA7CBF93D496AC15
//                             A56834CBCF98C397B4024A2691233B8D
//    Authentication Tag:      83DE3541E4C2B58177E065A9BF7B62EC
```

```
x           = sm4_encode_gcm( $iv, $plaintext, $aad, $key, 10 )

mac         = right( $x, 16 )
len         = strlen( $x )
len         = sub( $len, 10 )
encryptd    = hmid( $x, 00, $len )

y           = sm4_decode_gcm( $iv, $encryptd, $aad, $key, $mac )
```

scrypt

```
x = scrypt( "", "", 10, 01, 01, 40 )
if $x !=
77D6576238657B203B19CA42C18A0497F16B4844E3074AE8DFDFFA3FEDE21442FCD0069DED0948F8326A753A0F
C81F17E8D3E0FB2E0D3628CF35E20C38D18906
    ?
    pause
endif

// 该函数为内存型函数
```

pbkdf1_md2

pbkdf1_md4

pbkdf1_md5

pbkdf1_ripemd128

pbkdf1_ripemd160

pbkdf1_sha1

pbkdf1_sha224

pbkdf1_sha256

pbkdf1_sha384

pbkdf1_sha512

pbkdf1_sha512_224

pbkdf1_sha512_256

pbkdf1_sha3_224

pbkdf1_sha3_256

pbkdf1_sha3_384

pbkdf1_sha3_512

pbkdf1_blake2b_160

pbkdf1_blake2b_256

pbkdf1_blake2b_384

pbkdf1_blake2b_512

pbkdf1_blake2s_128

pbkdf1_blake2s_160

pbkdf1_blake2s_224

pbkdf1_blake2s_256

pbkdf1_tiger

pbkdf1_whirlpool

pbkdf1_sm3_160

pbkdf1_sm3_192

pbkdf1_sm3_256

```
? "所有函数参数类似，示例如下"
p       = "password"
s       = "salt"
c       = hex( 4096 )
len     = hex( 64 )
x       = pbkdf1_sha512( $p, $s, $c, $len )
if $x !=
412A2DC289E35A975B8374F1644995F9FDBD025CA87AE363B7A019228B7411EAE09EC0DB9DEA35CD1F0F91F55F
07750A52121DA270C180ED6BC1CC5E98401324
    ?
    pause
endif
```

pbkdf2_md2

pbkdf2_md4

pbkdf2_md5

pbkdf2_ripemd128

pbkdf2_ripemd160

pbkdf2_sha1

pbkdf2_sha224

pbkdf2_sha256

pbkdf2_sha384

pbkdf2_sha512

pbkdf2_sha512_224

pbkdf2_sha512_256

pbkdf2_sha3_224

pbkdf2_sha3_256

pbkdf2_sha3_384

pbkdf2_sha3_512

pbkdf2_blake2b_160

pbkdf2_blake2b_256

pbkdf2_blake2b_384

pbkdf2_blake2b_512

pbkdf2_blake2s_128

pbkdf2_blake2s_160

pbkdf2_blake2s_224

pbkdf2_blake2s_256

pbkdf2_tiger

pbkdf2_whirlpool

pbkdf2_sm3_160

pbkdf2_sm3_192

pbkdf2_sm3_256

```
? "所有函数参数类似，示例如下"
p       = "passwordPASSWORDpassword"
s       = "saltSALTsaltSALTsaltSALTsaltSALTsalt"
c       = hex( 4096 )
len     = hex( 64 )
x       = pbkdf2_sha512( $p, $s, $c, $len )
if $x != 
8C0511F4C6E597C6AC6315D8F0362E225F3C501495BA23B868C005174DC4EE71115B59F9E60CD9532FA33E0F75
AEFE30225C583A186CD82BD4DAEA9724A3D3B8
    ?
    pause
endif
```

hkdf_md2

hkdf_md4

hkdf_md5

hkdf_ripemd128

hkdf_ripemd160

hkdf_sha1

hkdf_sha224

hkdf_sha256

hkdf_sha384

hkdf_sha512

hkdf_sha512_224

hkdf_sha512_256

hkdf_sha3_224

hkdf_sha3_256

hkdf_sha3_384

hkdf_sha3_512

hkdf_blake2b_160

hkdf_blake2b_256

hkdf_blake2b_384

hkdf_blake2b_512

hkdf_blake2s_128

hkdf_blake2s_160

hkdf_blake2s_224

hkdf_blake2s_256

hkdf_tiger

hkdf_whirlpool

hkdf_sm3_160

hkdf_sm3_192

hkdf_sm3_256

```
? "所有函数参数类似，示例如下"
IKM     = 0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b// (22 octets)
salt    = 000102030405060708090a0b0c// (13 octets)
info    = f0f1f2f3f4f5f6f7f8f9// (10 octets)
L       = hex( 42 )
okm     = hkdf_sha256( $ikm, $salt, $info, $l )
if $okm !=
3CB25F25FAACD57A90434F64D0362F2A2D2D0A90CF1A5A4C5DB02D56ECC4C5BF34007208D5B887185865
    ?
    pause
endif
```

hmac_md2

hmac_md4

hmac_md5

hmac_ripemd128

hmac_ripemd160

hmac_sha1

hmac_sha224

hmac_sha256

hmac_sha384

hmac_sha512

hmac_sha512_224

hmac_sha512_256

hmac_sha3_224

hmac_sha3_256

hmac_sha3_384

hmac_sha3_512

hmac_blake2b_160

hmac_blake2b_256

hmac_blake2b_384

hmac_blake2b_512

hmac_blake2s_128

hmac_blake2s_160

hmac_blake2s_224

hmac_blake2s_256

hmac_tiger

hmac_whirlpool

hmac_sm3_160

hmac_sm3_192

hmac_sm3_256

hmac_shake128

hmac_shake256

```
? "所有函数参数类似，示例如下"
Key     = 0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b  //                  (20 bytes)
Data    = 4869205468657265          //        ("Hi There")

x       = hmac_sha224( $data, $key )
if $x !=  896fb1128abbdf196832107cd49df33f47b4b1169912ba4f53684b22
    ?
    pause
endif
x       = hmac_sha256( $data, $key )
if $x != b0344c61d8db38535ca8afceaf0bf12b881dc200c9833da726e9376c2e32cff7
    ?
    pause
endif
```

a3a8_fun

comp128_1

comp128_2

comp128_3

```
? "所有函数参数类似，示例如下"
// 参数1-随机数 - 8byte
// 参数2-密钥 - 16byte
// 其中 a3a8_fun 与 comp128_1 功能一致
x = a3a8_fun( 0000000000000000, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA )
if $x != 0E9FF8FF 24119D2D4ED18C00
    ?
    pause
endif
```

```
y = comp128_1( 0000000000000003, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA )
if $y != $x
    ?
    pause
endif

z = comp128_2( 0000000000000003, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA )
if $z != 9299E7F994D3288FA2E38000
    ?
    pause
endif

e = comp128_3( 0000000000000003, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA )
if $e != 9299E7F994D3288FA2E38216
    ?
    pause
endif
```

milenage_f1

milenage_f2345

milenage_f1star

milenage_f5star

milenage_compute_opc

milenage_opc_f1

milenage_opc_f2345

milenage_opc_f1star

milenage_opc_f5star

milenage_gsm

milenage_opc_gsm

```
//milenage_f1( k_16B, rand_16B, sqn_6B, amf_2B, op_16B )
//milenage_f2345( k_16B, rand_16B, op_16B )
//milenage_f1star( k_16B, rand_16B, sqn_6B, amf_2B, op_16B )
//milenage_f5star( k_16B, rand_16B, op_16B )
```

```
//milenage_compute_opc( op_16B )
//milenage_opc_f1( k_16B, rand_16B, sqn_6B, amf_2B, opc_16B )
//milenage_opc_f2345( k_16B, rand_16B, opc_16B )
//milenage_opc_f1star( k_16B, rand_16B, sqn_6B, amf_2B, opc_16B )
//milenage_opc_f5star( k_16B, rand_16B, opc_16B )
//milenage_gsm( k_16B, rand_16B, op_16B )
//milenage_opc_gsm( k_16B, rand_16B, opc_16B )

k               = dup( 16, aa )
op              = dup( 16, aa )
randx           = dup( 16, 00 )
sqn             = dup( 6, 00 )
amf             = dup( 2, 00 )

f1_mac_a        = milenage_f1( $k, $randx, $sqn, $amf, $op )
if $f1_mac_a != AFDBE3276549ACBF
    ?
    pause
endif

tmp             = milenage_f2345( $k, $randx, $op )
f2_res          = mid( $tmp, 0, 8 )
f3_ck           = mid( $tmp, 8, 16 )
f4_ik           = mid( $tmp, 24, 16 )
f5_ak           = mid( $tmp, 40 )
if $f2_res != C42F454F5FDEC1A1
    ?
    pause
endif
if $f3_ck != 701FB9D446EE896D96B0E97685E35693
    ?
    pause
endif
if $f4_ik != 173438EF8FBD185DE382834F72209F7E
    ?
    pause
endif
if $f5_ak != 600D51C33584
    ?
    pause
endif

autn            = $f5_ak $amf $f1_mac_a
```

```
if $autn != 600D51C335840000AFDBE3276549ACBF
    ?
    pause
endif

f1star_mac_s    = milenage_f1star( $k, $randx, $sqn, $amf, $op )
if $f1star_mac_s != A53F5C8E60EEF2C4
    ?
    pause
endif

f5star          = milenage_f5star( $k, $randx, $op )
if $f5star != 1A1BC57B2D4E
    ?
    pause
endif

//---------- 使用opc模式
opc             = milenage_compute_opc( $op )
f1_mac_a        = milenage_opc_f1( $k, $randx, $sqn, $amf, $opc )
if $f1_mac_a != AFDBE3276549ACBF
    ?
    pause
endif

tmp             = milenage_opc_f2345( $k, $randx, $opc )
f2_res          = mid( $tmp, 0, 8 )
f3_ck           = mid( $tmp, 8, 16 )
f4_ik           = mid( $tmp, 24, 16 )
f5_ak           = mid( $tmp, 40 )
if $f2_res != C42F454F5FDEC1A1
    ?
    pause
endif
if $f3_ck != 701FB9D446EE896D96B0E97685E35693
    ?
    pause
endif
if $f4_ik != 173438EF8FBD185DE382834F72209F7E
    ?
    pause
endif
if $f5_ak != 600D51C33584
```

```
    ?
    pause
endif

autn            = $f5_ak $amf $f1_mac_a
if $autn != 600D51C335840000AFDBE3276549ACBF
    ?
    pause
endif

f1star_mac_s    = milenage_opc_f1star( $k, $randx, $sqn, $amf, $opc )
if $f1star_mac_s != A53F5C8E60EEF2C4
    ?
    pause
endif

f5star          = milenage_opc_f5star( $k, $randx, $opc )
if $f5star != 1A1BC57B2D4E
    ?
    pause
endif

tmp             = milenage_gsm( $k, $randx, $op )
sres            = mid( $tmp, 0, 4 )
kc              = mid( $tmp, 4 )
if $sres != 9BF184EE
    ?
    pause
endif
if $kc != 1219EB023E9058DD
    ?
    pause
endif

tmp             = milenage_opc_gsm( $k, $randx, $opc )
sres            = mid( $tmp, 0, 4 )
kc              = mid( $tmp, 4 )
if $sres != 9BF184EE
    ?
    pause
endif
if $kc != 1219EB023E9058DD
    ?
```

```
    pause
endif
```

sm4_encode_ccm

sm4_decode_ccm

sm4_encode_ccm	m = a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9
	
	m1 = 03 80 14 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9
	
	key = ca 44 ef 8d f3 25 ab b3 8d ac 37 43 dd 32 43 df
	
	nonce = 72 43 52 3C 65 17 8B 96 68 37 A3 6F
	
	//SM4CCM对数据进行对称加密结果: fb 78 40 15 24 ca 9c 2d 68 3b c4 e9 5d dc 71 b8 28 07 77 81 aa 8e 3f e8 1c e4 de 21 38 76 49 19 59 ee 87 63 e2 21 55
	// x = sm4_encode_ccm($nonce, "", $m, $key, 10)
	x = sm4_encode_ccm($nonce, "", $m1, $key, 10)
	
	cipher = mid($x, 16)
	if $cipher != FB78401524CA9C2D683BC4E95DDC71B828077781AA8E3FE81CE4DE213876491959EE8763E22155
	?
	pause
	endif
	
	返回数据分为4个部分
	● 前0x10个字节为B0
	● 外部传入的adata，与输入数据长度相同
	● 密文，与明文长度相同
	● mac，长度由外面指定

sm4_decode_ccm	len = strlen($m1) y = sm4_decode_ccm($nonce, "", $cipher, $key, 10) y = hmid($y, 00, $len) if $y != $m1 ? pause

	endif

new_rsa_calculate_d2

new_rsa_calculate_d2	newd = new_rsa_calculate_d2($module_len, $e, $p, $q, $old_d)

new_ecc_prilen_random

new_ecc_prilen_random	random = new_ecc_prilen_random ()

new_ecc_prilen

new_ecc_prilen	len = new_ecc_prilen ()

new_ecc_publen

new_ecc_publen	len = new_ecc_publen ()

UrlDownloadToFile

UrlDownloadToFile	k = UrlDownloadToFile(“https:\\a.com\\b.txt”, “c:\\aa”) k = UrlDownloadToFile(“https:\\a.com\\b.txt”, “savepath”) 或 k = UrlDownloadToFile(“https:\\a.com\\b.txt”, “savepath”, 01, 08) 第三个参数表示是否要加入一个4字节序数，第四个参数是序数

Replace_in_file

Replace_in_file	k = Replace_in_file(“c:\\b.txt”, “old”, “new”, “txt\0html\0\0”) k = Replace_in_file(“c:”, “old”, “new”)

Delete_in_file

Delete_in_file	k = Delete_in_file(“C:\\b.txt”, “start”, “end” , “txt\0html\0\0”) k = Delete_in_file(“C:”, “start”, “end”)

crc16_x25

Crc_16_x25	crc16_x25(polynomial, icv_crc, xout, data) crc = crc16_x25(8408, ffff, ffff, $data)

pkcs7_pad

pkcs7_unpad

pkcs7_pad	pkcs7_pad(input, pack_len) pkcs7_unpad(input) a = dup(7, 88) x = pkcs7_pad($a) y = pkcs7_unpad($x)

```
x = pkcs7_pad( $a, 10 )
y = pkcs7_unpad( $x )
x = pkcs7_pad( $a, 14 )
y = pkcs7_unpad( $x )

x = 88888888888880901090909090909090
y = pkcs7_unpad( $x )
x = 88888888888880109090909090909090
y = pkcs7_unpad( $x )
x = 8888888888888010D0D0D0D0D0D0D0D0D0D0D0D
y = pkcs7_unpad( $x )
x = 88888888888888888888888888888888
y = pkcs7_unpad( $x )
```

pkcs12_sha1_diver_by_password

pkcs12_sha1_diver_by_password

```
clear

ansi_password   = "1234"
utf16_password  = ansi_utf16_big($ansi_password)
utf16_password  = $utf16_password 0000

salt            = D9A5B98E36B81236

id              = 01

iteration       = hex( 2000 )
key_len         = hex( 24 )

x               = pkcs12_sha1_diver_by_password( $utf16_password,
$salt, $id, $iteration, $key_len )
if $x != 13B0F4E8EAA3F9E840F709C140A10F6F62659B786555FD99
    ?
    pause
endif
```

hotp_sha1_int

hotp_sha256_int

hotp_sha1_hex

hotp_sha256_hex

hotp_sha1_int

```
seedkey = 01020304050607080102030405060708
```

hotp_sha256_int hotp_sha1_hex hotp_sha256_hex	counter = 00 b = hotp_sha256_int($seedkey, $counter) b = hotp_sha1_int($seedkey, $counter) b = hotp_sha256_int($seedkey, $counter, 06) b = hotp_sha1_int($seedkey, $counter, 08) b = hotp_sha256_hex($seedkey, $counter) b = hotp_sha1_hex($seedkey, $counter)

totp_sha1_int

totp_sha256_int

totp_sha1_hex

totp_sha256_hex

totp_sha1_int totp_sha256_int totp_sha1_hex totp_sha256_hex	seedkey = 0102030405060708010203040506070 8 counter = 00 b = totp_sha256_int($seedkey, $counter) b = totp_sha1_int($seedkey, $counter) b = totp_sha256_int($seedkey, $counter, 06) b = totp_sha1_int($seedkey, $counter, 08) b = totp_sha256_hex($seedkey, $counter) b = totp_sha1_hex($seedkey, $counter)

get_reader_name

get_reader_name	x = get_reader_name(index)

utf16_little_ansi

utf16_little_ansi	a = 4A00410056004100430 04F00530020005600690072007400750061006C00200043006F006E00740061006300740020005200650061006400650072002000300 0 b = show_utf16_little_string($a) c = utf16_little_ansi($a)

strstr

strstr	a = "123456" b = "34" c = strstr($a, $b)

0.0.6.6

LoadLibrary

GetProcAddress

FreeLibrary

```
dll_handle  = loadlibrary( "UserAlgoDll.dll" )
if $dll_handle != 00
    ? "load success"
else
    ? "load dll error"
endif

fun_add     = getprocaddress( $dll_handle, "Fun_Add" )
if $fun_add != 00
    t   = dllfun( $fun_add, 00000023, 00000022 )
    if $t != 00000045
        ?
        pause
    endif
endif
x           = freelibrary( $dll_handle )
```

dllFun

dllfun	t = dllfun($fun_add, 00000023, 00000022) 最多支持12个参数 dllfun(funaddr, p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12)

new_rsa_oaep_sha1_pub_encode

new_rsa_oaep_sha224_pub_encode

new_rsa_oaep_sha256_pub_encode

new_rsa_oaep_sha384_pub_encode

new_rsa_oaep_sha512_pub_encode

new_rsa_oaep_sha1_std_decode

new_rsa_oaep_sha224_std_decode

new_rsa_oaep_sha256_std_decode

new_rsa_oaep_sha384_std_decode

new_rsa_oaep_sha512_std_decode

new_rsa_oaep_sha1_crt_decode

new_rsa_oaep_sha224_crt_decode

new_rsa_oaep_sha256_crt_decode

new_rsa_oaep_sha384_crt_decode

new_rsa_oaep_sha512_crt_decode

new_rsa_oaep_sha1_pub_encode	x1 = new_rsa_oaep_sha512_pub_encode(0800, $e, $n, 12345678, 9999) y1 = new_rsa_oaep_sha512_std_decode(0800, $n, $d, $x1) if $y1 != 12345678 ? pause endif y2 = new_rsa_oaep_sha512_crt_decode(0800, $p, $q, $dp, $dq, $qinv, $x1) if $y2 != 12345678 ? pause endif

new_rsa_pss_sha1_pub_verify

new_rsa_pss_sha224_pub_verify

new_rsa_pss_sha256_pub_verify

new_rsa_pss_sha384_pub_verify

new_rsa_pss_sha512_pub_verify

new_rsa_pss_sha1_std_sign

new_rsa_pss_sha224_std_sign

new_rsa_pss_sha256_std_sign

new_rsa_pss_sha384_std_sign

new_rsa_pss_sha512_std_sign

new_rsa_pss_sha1_crt_sign

new_rsa_pss_sha224_crt_sign

new_rsa_pss_sha256_crt_sign

new_rsa_pss_sha384_crt_sign

new_rsa_pss_sha512_crt_sign

new_rsa_pss 示例	hash = sha256_hash($plain) a = new_rsa_pss_sha256_crt_sign(0800, $p, $q, $dp, $dq, $qinv, $hash, $saltlen) k = new_rsa_pss_sha256_pub_verify(0800, $e, $n, $a, $hash, $saltlen) if $k != 00 ? pause endif b = new_rsa_pss_sha256_std_sign(0800, $n, $d, $hash, $saltlen) k = new_rsa_pss_sha256_pub_verify(0800, $e, $n, $b, $hash, $saltlen) if $k != 00 ? pause endif

new_sm9_encrypt_block_2018

new_sm9_encrypt_stream_2018

new_sm9_decrypt_block_2018

new_sm9_decrypt_stream_2018

用法	与之前不带2018的版本一致

s2k_md5

s2k_sha1

s2k_sha224

s2k_sha256

s2k_sha384

s2k_sha512

用法

```
salt            = 3031323334353637

passphrase      = "123456"      // 用户口令
iteration_count = hex( 100000 )  // 迭代次数

len             = hex( 32 )

x               = s2k_sha256( $passphrase, $salt, $iteration_count,
$len )
if $x !=
773784A602B6C81E3F092F4D7D00E17CC822D88F7360FCF2D2EF2D9D901F44B6
    ?
    pause
endif

salt            = 41 42 43 44 45 46 47 48
passphrase      = "12345678"      // 用户口令
x               = s2k_sha256( $passphrase, $salt, $iteration_count,
$len )
if $x !=
2675D6164A0D4827D1D00C7EEA620D015C00030A1CAB38B4D0DD600B27DC9630
    ?
    pause
endif
```

crc_13239_lsb

crc_13239_msb

用法	xx = 6988698269856a81 yy = $credential_private_id $xx crc = crc_13239_lsb($yy)

var_exist

用法	xx = var_exist(“key1”) 存在返回01，不存在返回00

kdf_md5

kdf_sha1

kdf_sha224

kdf_sha256

kdf_sha384

kdf_sha512

kdf_sm3

用法	结果 = kdf_sha256(first, second, keylen) xx = kdf_sha256($k, $tmp1, 30) key = mid($xx, 0, 16) mackey = mid($xx, 16, 32)

crc16_buypass

crc16_umts

crc16_modbus

crc16_xmodem

用法	x = crc16_buypass(D26B20B067BE3974288FC708F5D1421136BFDDE4FE483CF32311D037AD4CBEB4) x = crc16_umts(D26B20B067BE3974288FC708F5D1421136BFDDE4FE483CF32311D037AD4CBEB4)

```
if $x != 17a5
    ?
    pause
endif

x = crc16_umts( 3BF824F3EAF35428B5598D56B4DA6FC9118D94818C8A3C49F8 )
if $x != dadb
    ?
    pause
endif
```