对称密码学安全性概述

> By 王超然

常见算法与基本参数

对称

AES(FIPS-197)

密钥128/192/256bits

块长度128bits

加密轮数10/12/14轮

SM4(GMT0002-2012)

密钥128bits

块长度128bits

加密轮数32

工作模式

基础：

ECB(SP 800-38A)

传统(SP 800-38A)：

CBC(with 128bits key,RFC3602)，OFB，CFB，CTR(RFC 3686)

新场景：

CTS(SP 800-38A)，XTS(XEX+CTS,IEEE P1619)

认证加密：

GCM(RFC4106)，CCM(RFC4309)，

【AES专有，XCBC_MAC96(RFC 3566)，XCBC_PRF_128(RFC 4434)】

padding

nopadding: 无padding

bitspadding:在流的结尾填补一个1，剩余填0

zeropadding: 用字节0补全

pkcs#7: 以缺少字节长度填充

HASH

SHA1(FIPS 180-2):160

SHA2(FIPS 180-4):224,256,384,512,512/224,512/256

SHA3(FIPS 202):224,256,384,512

SM3(GMT0004):256

MAC

HMAC(RFC 2104,RFC 2202)

GMAC(AES,RFC 4543)

CMAC

工具使用

os-openssl

命令行程序

enc

对称加密统一接口

指令：openssl enc -ciphername -in infilename -out outfilename -K hexkey [-nopad]

dgst

哈希统一接口，也支持hmac，但hmac不支持16进制输入

指令：openssl dgst -hashname -in infilename -out outfilename [-hmac key]

cms

打包证书进行使用

api

hmac：

```
HMAC_CTX* ctx = HMAC_CTX_new();
HMAC_Init_ex(ctx,key,strlen(key),EVP_sha1(),NULL);
HMAC_Update(ctx,data,strlen((char*)data));
HMAC_Final(ctx,md,&len);
```

```

   HMAC(EVP_sha1(),key,strlen(key),data,strlen((char*)data),md,&len);
```

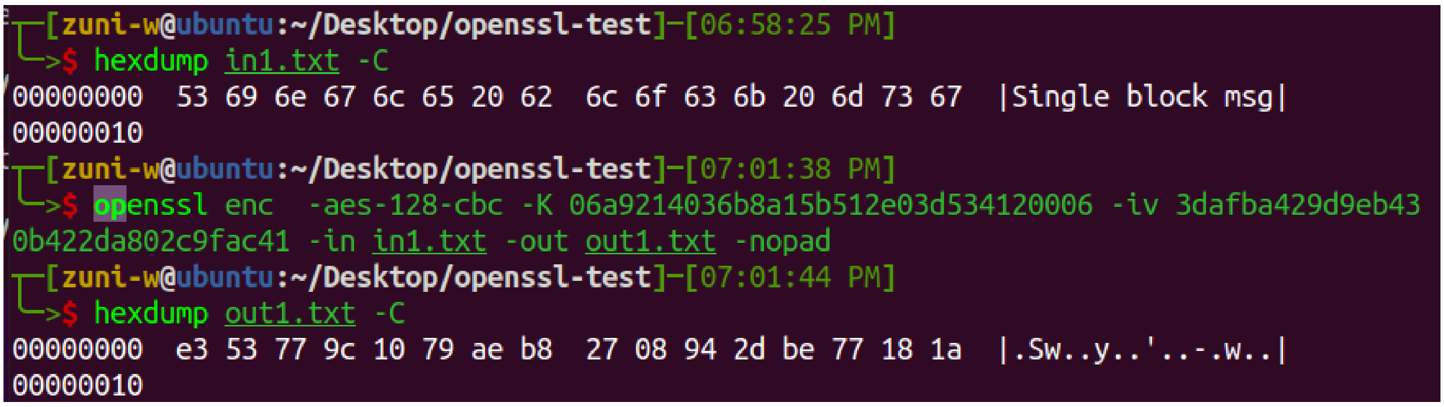
aes-gcm:

```
    void aes_gcm_encrypt(void)
{
        EVP_CIPHER_CTX *ctx;
        int outlen, tmplen;
        unsigned char outbuf[1024];
        ctx = EVP_CIPHER_CTX_new();
        EVP_EncryptInit_ex(ctx, EVP_aes_256_gcm(), NULL, NULL, NULL);
        EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_GCM_SET_IVLEN, sizeof(gcm_iv), NULL);
        EVP_EncryptInit_ex(ctx, NULL, NULL, gcm_key, gcm_iv);
        EVP_EncryptUpdate(ctx, NULL, &outlen, gcm_aad, sizeof(gcm_aad));
        EVP_EncryptUpdate(ctx, outbuf, &outlen, gcm_pt, sizeof(gcm_pt));
        EVP_EncryptFinal_ex(ctx, outbuf, &outlen);
        EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_GCM_GET_TAG, 16, outbuf);
        EVP_CIPHER_CTX_free(ctx);
}

void aes_gcm_decrypt(void)
{
        EVP_CIPHER_CTX *ctx;
        int outlen, tmplen, rv;
        unsigned char outbuf[1024];
        ctx = EVP_CIPHER_CTX_new();
        EVP_DecryptInit_ex(ctx, EVP_aes_256_gcm(), NULL, NULL, NULL);
        EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_GCM_SET_IVLEN, sizeof(gcm_iv), NULL);
        EVP_DecryptInit_ex(ctx, NULL, NULL, gcm_key, gcm_iv);
        EVP_DecryptUpdate(ctx, NULL, &outlen, gcm_aad, sizeof(gcm_aad));
        EVP_DecryptUpdate(ctx, outbuf, &outlen, gcm_ct, sizeof(gcm_ct));

        EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_GCM_SET_TAG, sizeof(gcm_tag), gcm_tag);
        rv = EVP_DecryptFinal_ex(ctx, outbuf, &outlen);
        printf("Tag Verify %s\n", rv > 0 ? "Successful!" : "Failed!");
        EVP_CIPHER_CTX_free(ctx);
}

```

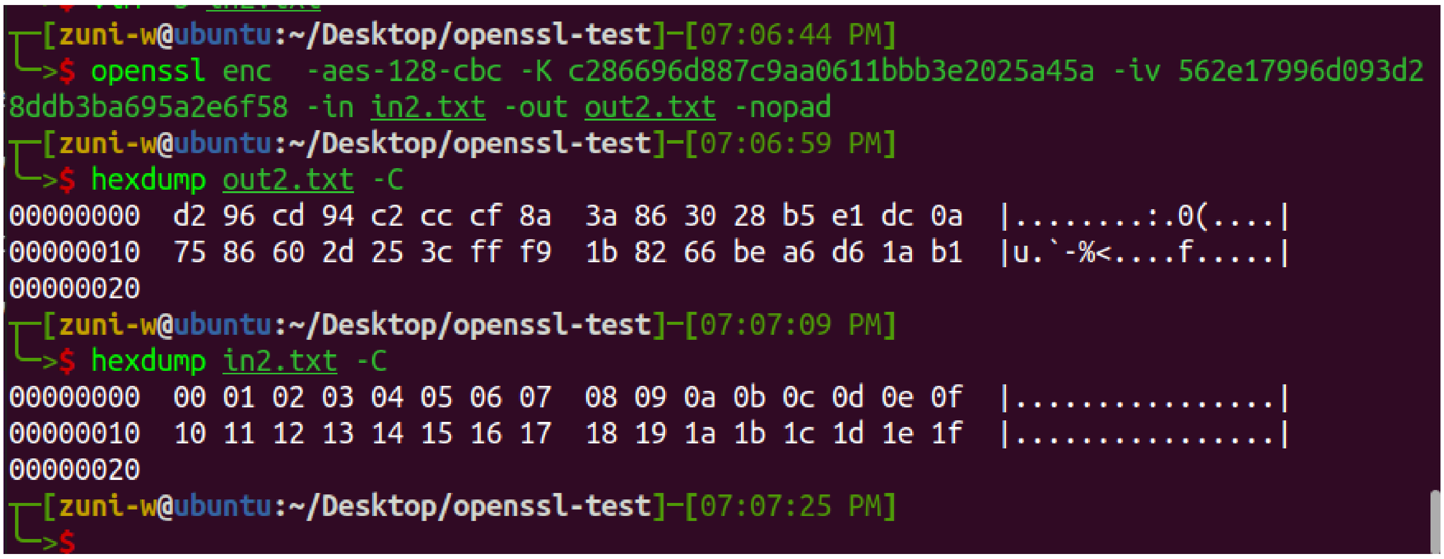
例子

aes-128-cbc

```
Case #1: Encrypting 16 bytes (1 block) using AES-CBC with 128-bit key
Key       : 0x06a9214036b8a15b512e03d534120006
IV        : 0x3dafba429d9eb430b422da802c9fac41
Plaintext : "Single block msg"
Ciphertext: 0xe353779c1079aeb82708942dbe77181a

Case #2: Encrypting 32 bytes (2 blocks) using AES-CBC with 128-bit key
Key       : 0xc286696d887c9aa0611bbb3e2025a45a
IV        : 0x562e17996d093d28ddb3ba695a2e6f58
Plaintext : 0x000102030405060708090a0b0c0d0e0f
              101112131415161718191a1b1c1d1e1f
Ciphertext: 0xd296cd94c2cccf8a3a863028b5e1dc0a
              7586602d253cfff91b8266bea6d61ab1
```

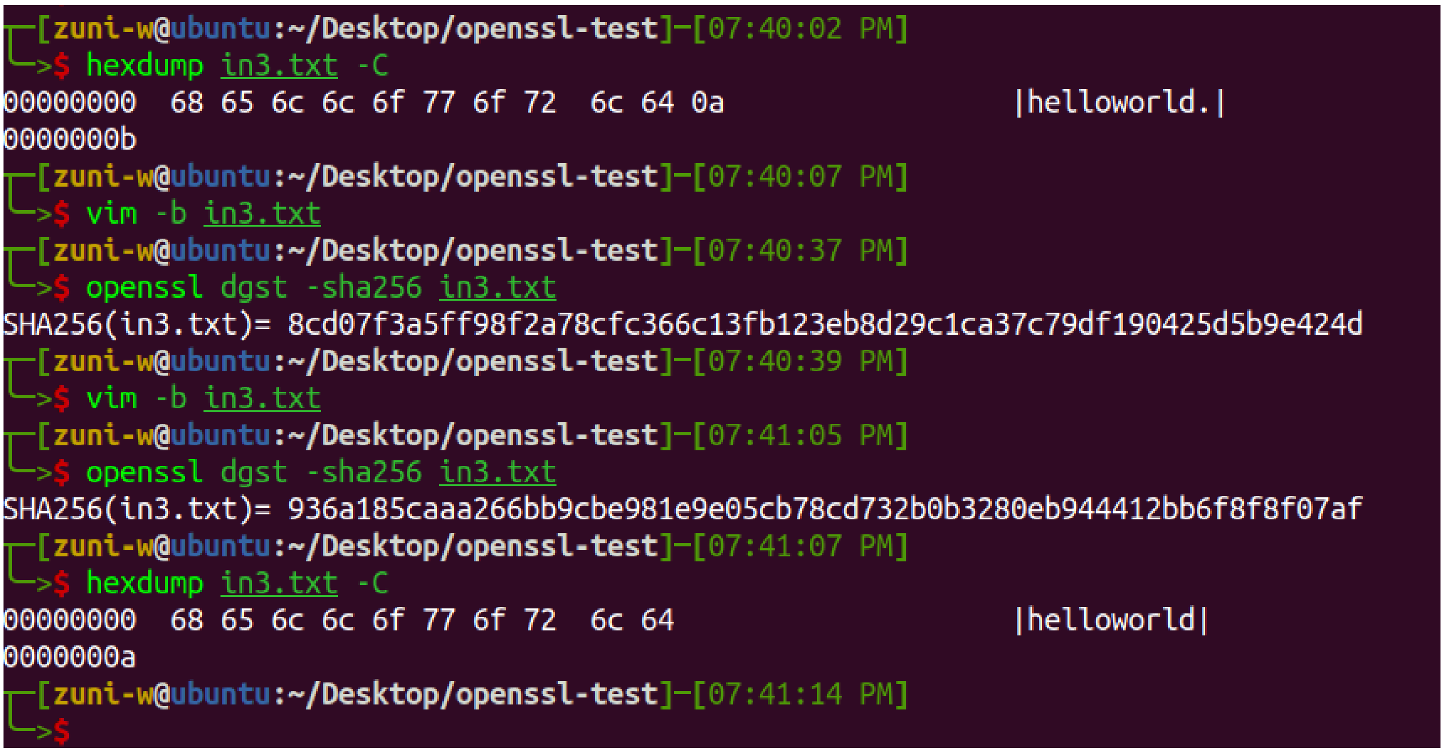


```
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[06:58:25 PM]
└─>$ hexdump in1.txt -C
00000000  53 69 6e 67 6c 65 20 62  6c 6f 63 6b 20 6d 73 67  |Single block msg|
00000010
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:01:38 PM]
└─>$ openssl enc  -aes-128-cbc -K 06a9214036b8a15b512e03d534120006 -iv 3dafba429d9eb43
0b422da802c9fac41 -in in1.txt -out out1.txt -nopad
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:01:44 PM]
└─>$ hexdump out1.txt -C
00000000  e3 53 77 9c 10 79 ae b8  27 08 94 2d be 77 18 1a  |.Sw..y..'..-.w..|
00000010
```



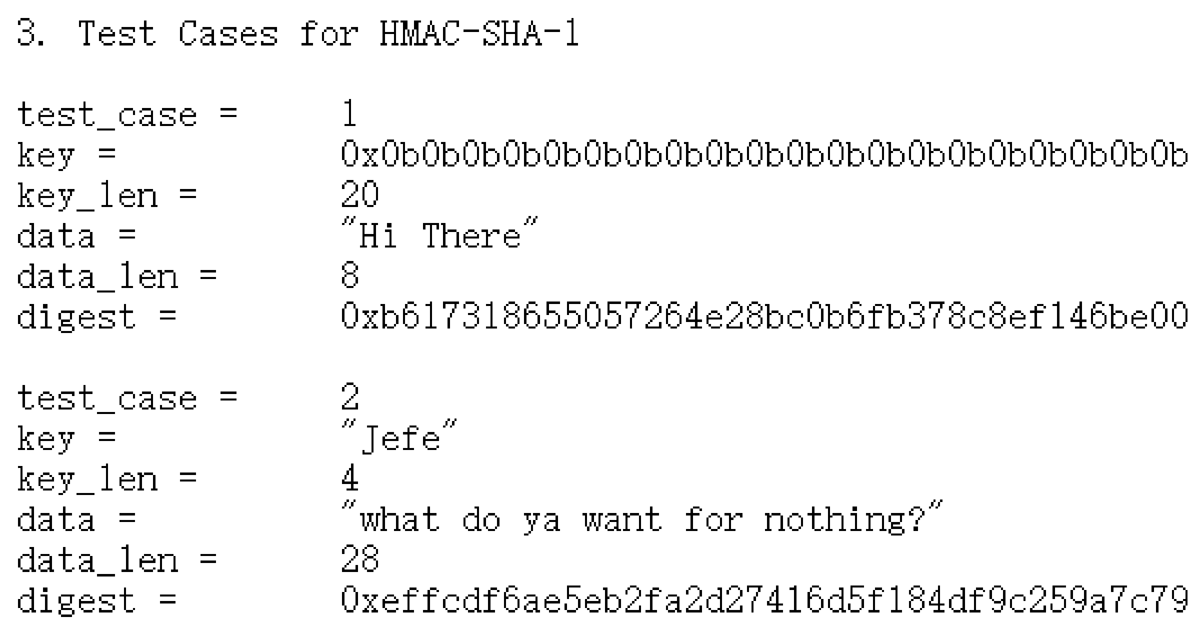
```
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:06:44 PM]
└─>$ openssl enc  -aes-128-cbc -K c286696d887c9aa0611bbb3e2025a45a -iv 562e17996d093d2
8ddb3ba695a2e6f58 -in in2.txt -out out2.txt -nopad
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:06:59 PM]
└─>$ hexdump out2.txt -C
00000000  d2 96 cd 94 c2 cc cf 8a  3a 86 30 28 b5 e1 dc 0a  |........:.0(....|
00000010  75 86 60 2d 25 3c ff f9  1b 82 66 be a6 d6 1a b1  |u.`-%<....f.....|
00000020
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:07:09 PM]
└─>$ hexdump in2.txt -C
00000000  00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f  |................|
00000010  10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f  |................|
00000020
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:07:25 PM]
└─>$
```

sha256

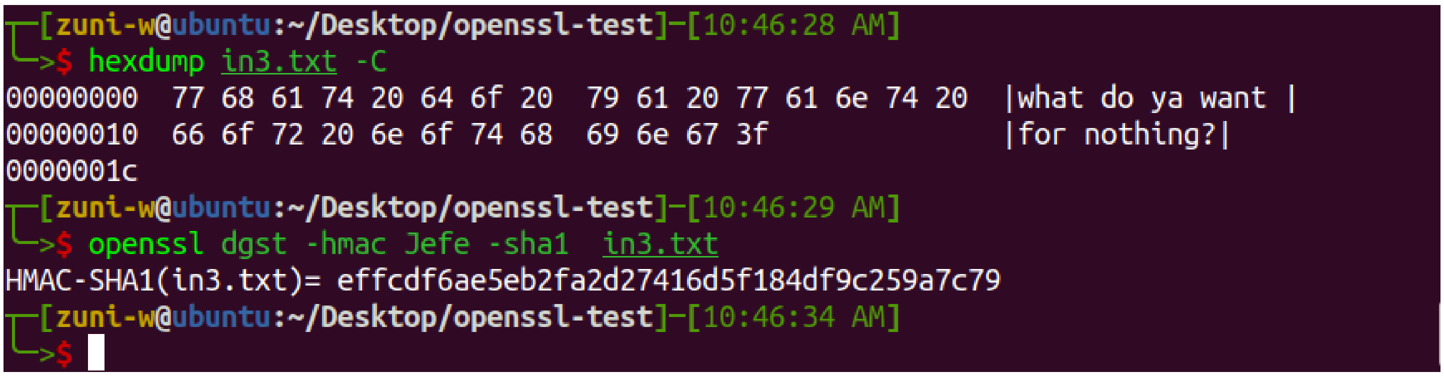


```
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:40:02 PM]
└─>$ hexdump in3.txt -C
00000000  68 65 6c 6c 6f 77 6f 72  6c 64 0a                 |helloworld.|
0000000b
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:40:07 PM]
└─>$ vim -b in3.txt
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:40:37 PM]
└─>$ openssl dgst -sha256 in3.txt
SHA256(in3.txt)= 8cd07f3a5ff98f2a78cfc366c13fb123eb8d29c1ca37c79df190425d5b9e424d
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:40:39 PM]
└─>$ vim -b in3.txt
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:41:05 PM]
└─>$ openssl dgst -sha256 in3.txt
SHA256(in3.txt)= 936a185caaa266bb9cbe981e9e05cb78cd732b0b3280eb944412bb6f8f8f07af
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:41:07 PM]
└─>$ hexdump in3.txt -C
00000000  68 65 6c 6c 6f 77 6f 72  6c 64                    |helloworld|
0000000a
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[07:41:14 PM]
└─>$
```

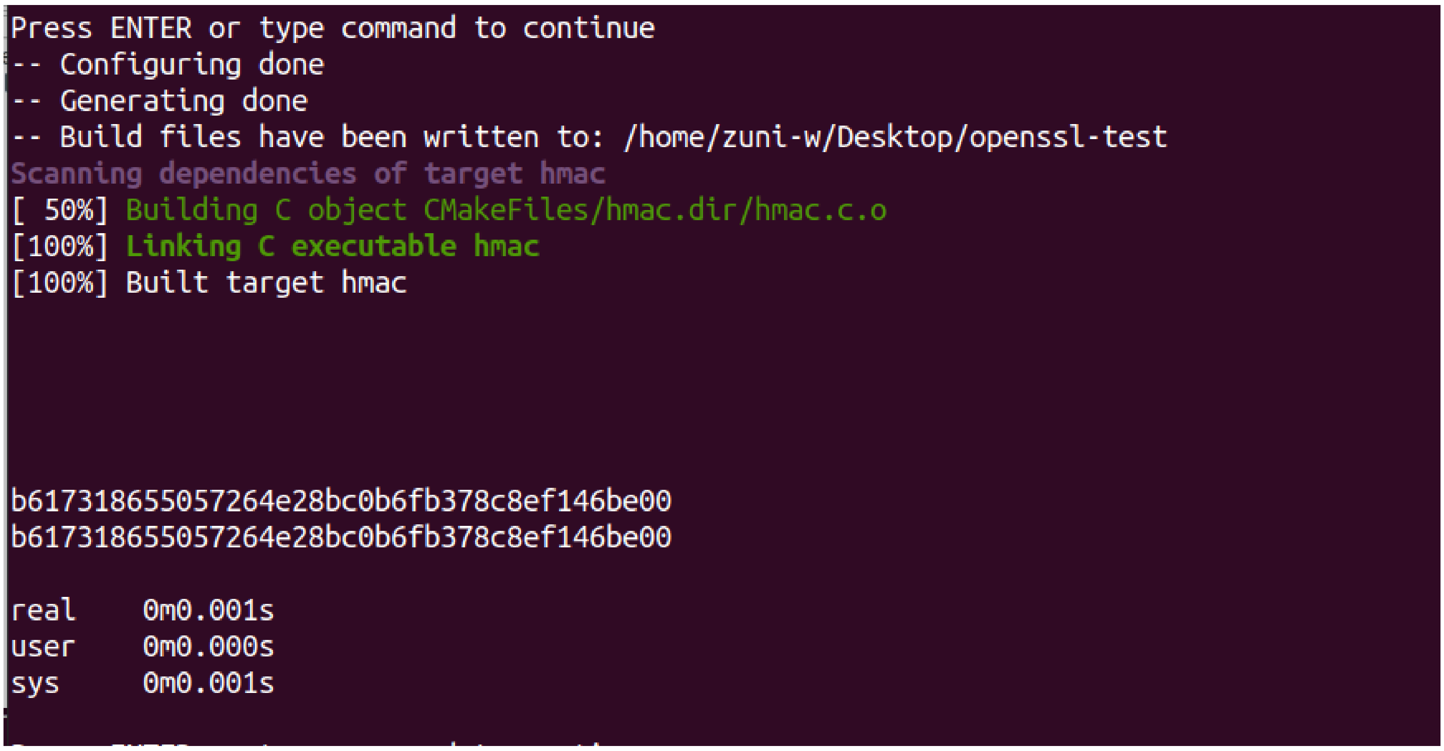
HMAC-SHA1



```
3. Test Cases for HMAC-SHA-1

test_case =     1
key =           0x0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
key_len =       20
data =          "Hi There"
data_len =      8
digest =        0xb617318655057264e28bc0b6fb378c8ef146be00

test_case =     2
key =           "Jefe"
key_len =       4
data =          "what do ya want for nothing?"
data_len =      28
digest =        0xeffcdf6ae5eb2fa2d27416d5f184df9c259a7c79
```

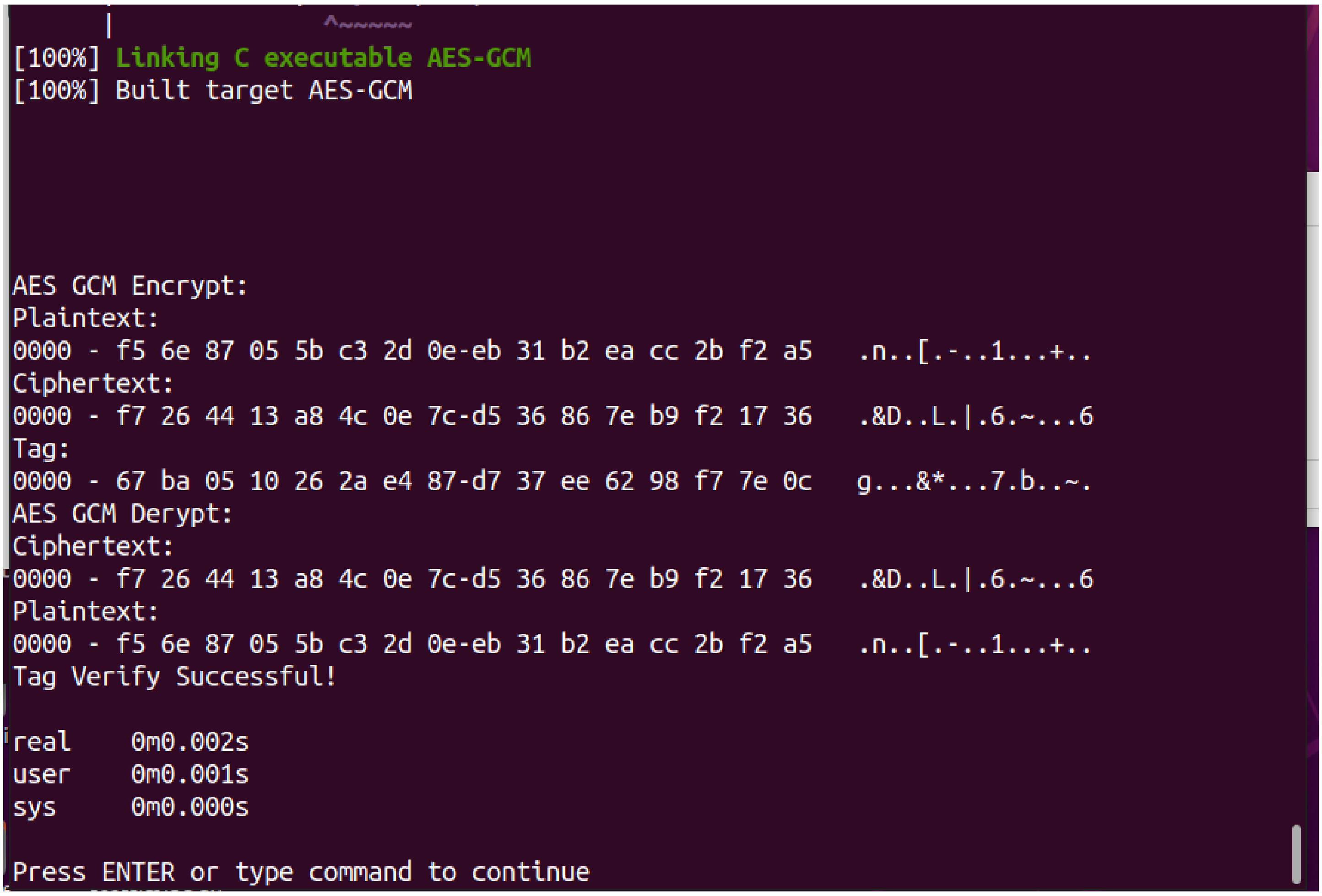


```
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[10:46:28 AM]
└─>$ hexdump in3.txt -C
00000000  77 68 61 74 20 64 6f 20  79 61 20 77 61 6e 74 20  |what do ya want |
00000010  66 6f 72 20 6e 6f 74 68  69 6e 67 3f              |for nothing?|
0000001c
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[10:46:29 AM]
└─>$ openssl dgst -hmac Jefe -sha1  in3.txt
HMAC-SHA1(in3.txt)= effcdf6ae5eb2fa2d27416d5f184df9c259a7c79
┌─[zuni-w@ubuntu:~/Desktop/openssl-test]─[10:46:34 AM]
└─>$ 
```



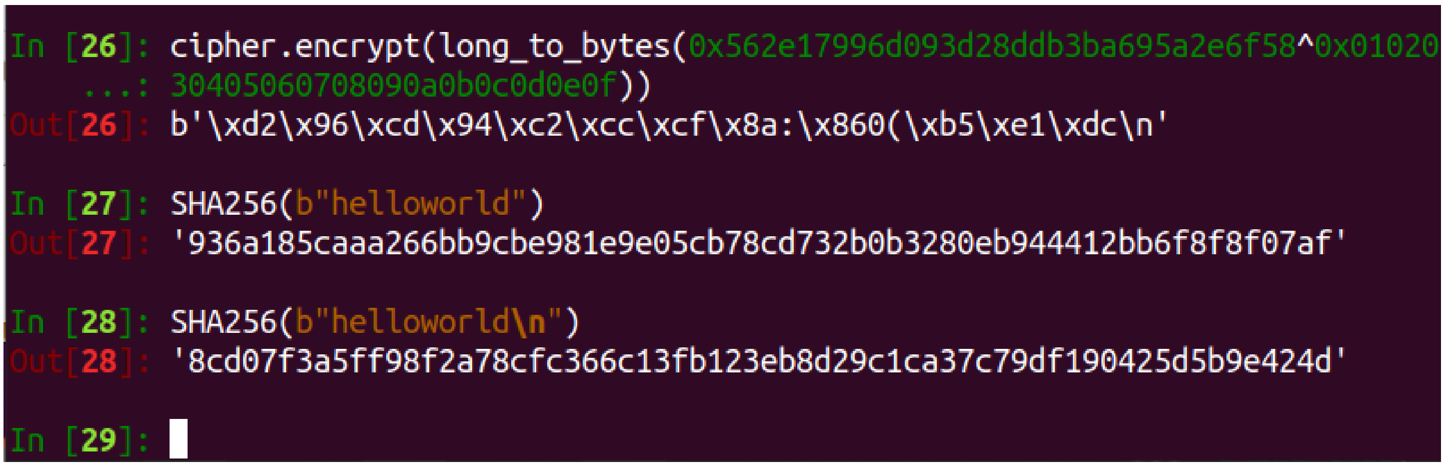
```
Press ENTER or type command to continue
-- Configuring done
-- Generating done
-- Build files have been written to: /home/zuni-w/Desktop/openssl-test
Scanning dependencies of target hmac
[ 50%] Building C object CMakeFiles/hmac.dir/hmac.c.o
[100%] Linking C executable hmac
[100%] Built target hmac






b617318655057264e28bc0b6fb378c8ef146be00
b617318655057264e28bc0b6fb378c8ef146be00

real	0m0.001s
user	0m0.000s
sys	0m0.001s
```

aes-128-gcm



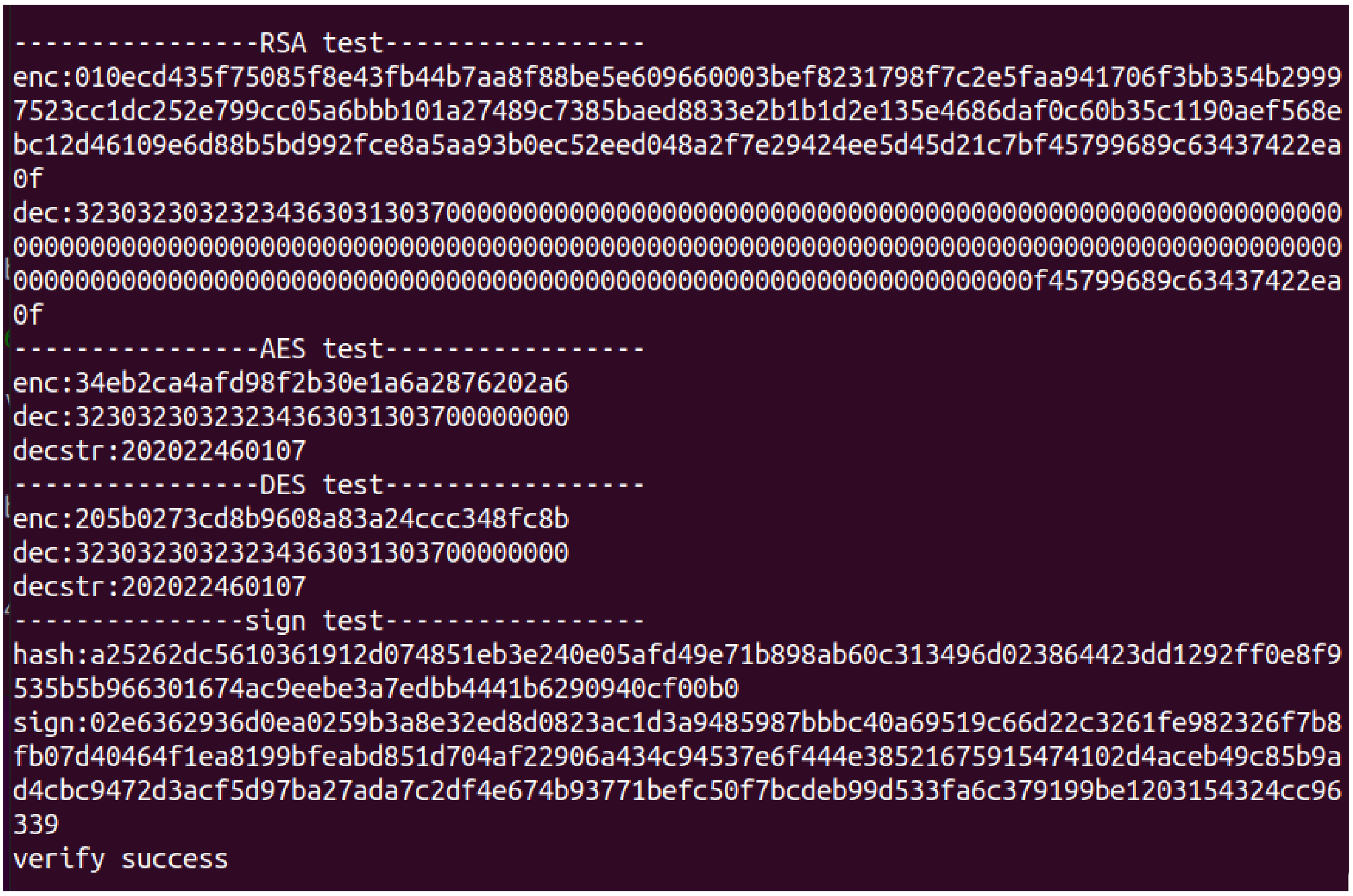
```
[100%] Linking C executable AES-GCM
[100%] Built target AES-GCM





AES GCM Encrypt:
Plaintext:
0000 - f5 6e 87 05 5b c3 2d 0e-eb 31 b2 ea cc 2b f2 a5   .n..[.-..1...+..
Ciphertext:
0000 - f7 26 44 13 a8 4c 0e 7c-d5 36 86 7e b9 f2 17 36   .&D..L.|.6.~...6
Tag:
0000 - 67 ba 05 10 26 2a e4 87-d7 37 ee 62 98 f7 7e 0c   g...&*...7.b..~.
AES GCM Derypt:
Ciphertext:
0000 - f7 26 44 13 a8 4c 0e 7c-d5 36 86 7e b9 f2 17 36   .&D..L.|.6.~...6
Plaintext:
0000 - f5 6e 87 05 5b c3 2d 0e-eb 31 b2 ea cc 2b f2 a5   .n..[.-..1...+..
Tag Verify Successful!

real	0m0.002s
user	0m0.001s
sys	0m0.000s

Press ENTER or type command to continue
```

python-pycryptodome+hashlib

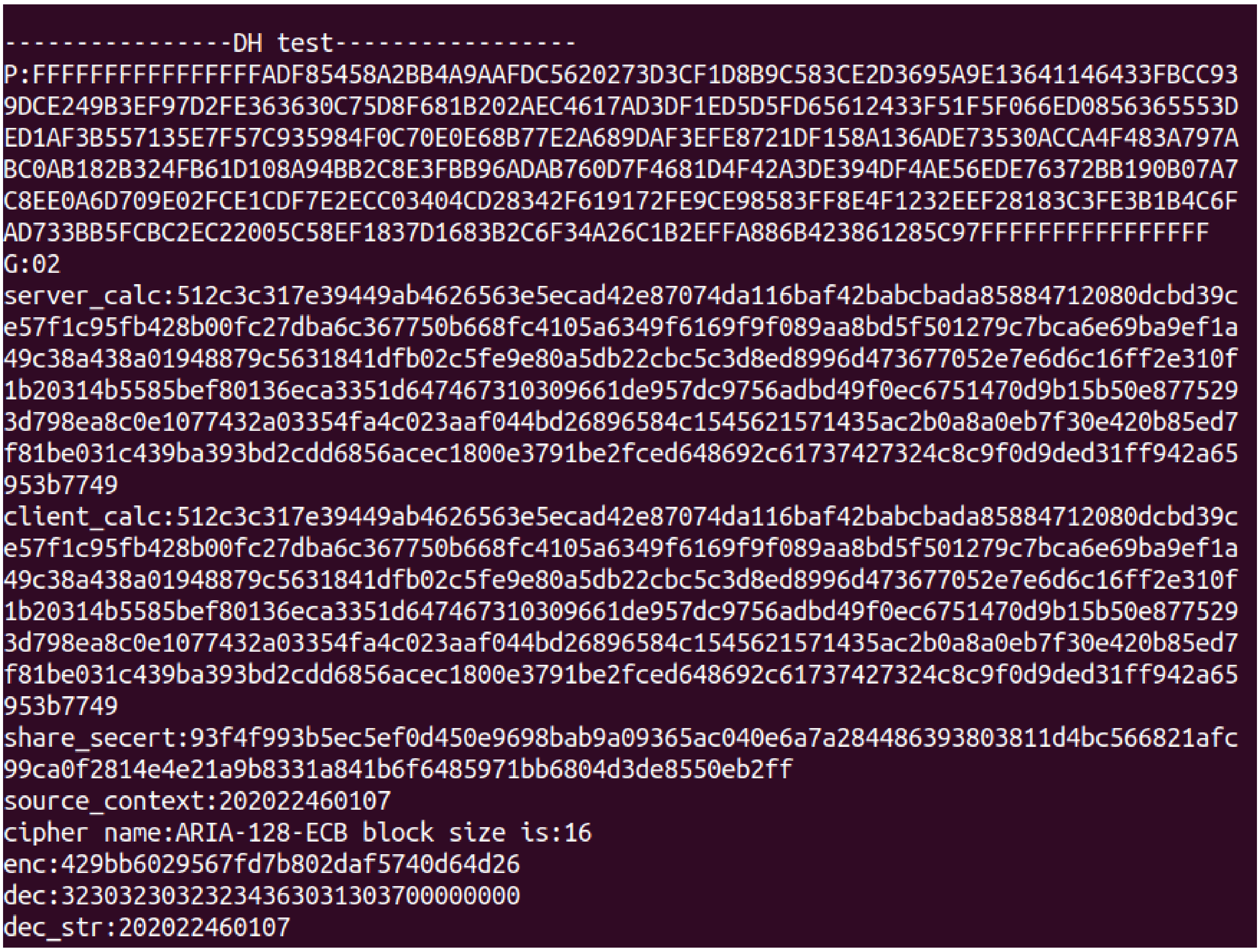


```
In [26]: cipher.encrypt(long_to_bytes(0x562e17996d093d28ddb3ba695a2e6f58^0x01020
    ...: 30405060708090a0b0c0d0e0f))
Out[26]: b'\xd2\x96\xcd\x94\xc2\xcc\xcf\x8a:\x860(\xb5\xe1\xdc\n'

In [27]: SHA256(b"helloworld")
Out[27]: '936a185caaa266bb9cbe981e9e05cb78cd732b0b3280eb944412bb6f8f8f07af'

In [28]: SHA256(b"helloworld\n")
Out[28]: '8cd07f3a5ff98f2a78cfc366c13fb123eb8d29c1ca37c79df190425d5b9e424d'

In [29]: 
```

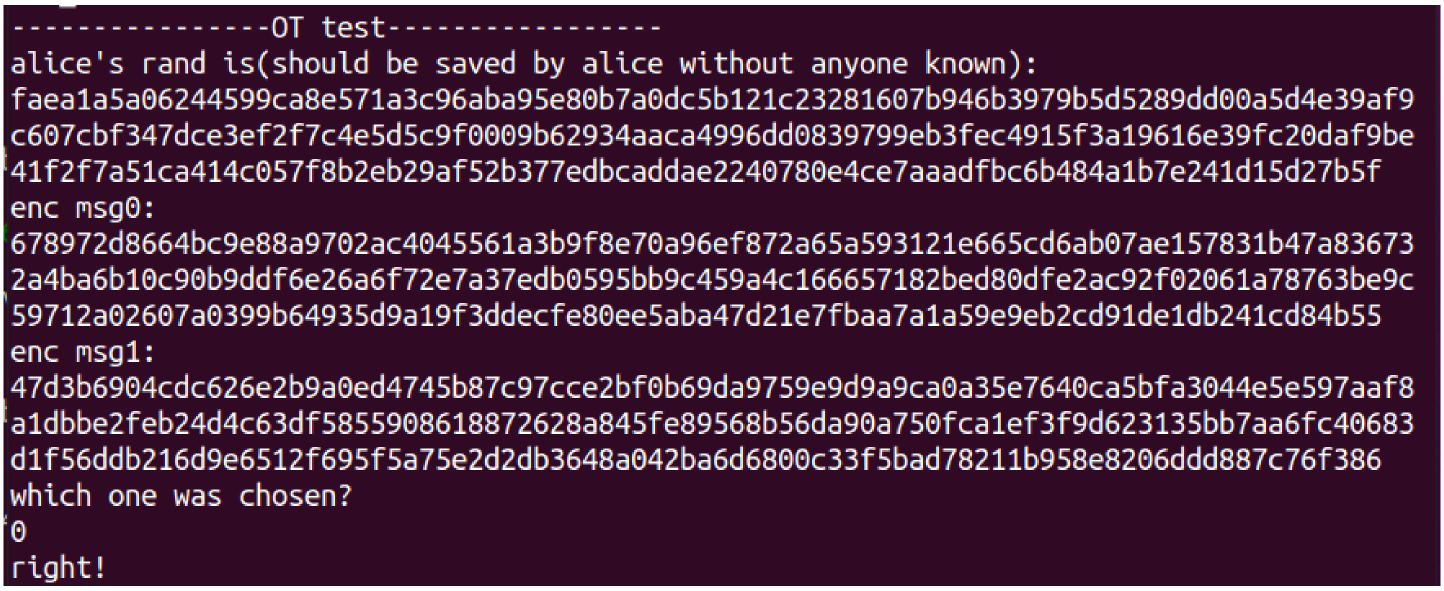
C-mbedtls



```
----------------RSA test----------------
enc:010ecd435f75085f8e43fb44b7aa8f88be5e609660003bef8231798f7c2e5faa941706f3bb354b2999
7523cc1dc252e799cc05a6bbb101a27489c7385baed8833e2b1b1d2e135e4686daf0c60b35c1190aef568e
bc12d46109e6d88b5bd992fce8a5aa93b0ec52eed048a2f7e29424ee5d45d21c7bf45799689c63437422ea
0f
dec:3230323032323436303130370000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000f45799689c63437422ea
0f
----------------AES test----------------
enc:34eb2ca4afd98f2b30e1a6a2876202a6
dec:32303230323234363031303700000000
decstr:202022460107
----------------DES test----------------
enc:205b0273cd8b9608a83a24ccc348fc8b
dec:32303230323234363031303700000000
decstr:202022460107
---------------sign test----------------
hash:a25262dc5610361912d074851eb3e240e05afd49e71b898ab60c313496d023864423dd1292ff0e8f9
535b5b966301674ac9eebe3a7edbb4441b6290940cf00b0
sign:02e6362936d0ea0259b3a8e32ed8d0823ac1d3a9485987bbbc40a69519c66d22c3261fe982326f7b8
fb07d40464f1ea8199bfeabd851d704af22906a434c94537e6f444e385216759154741 02d4aceb49c85b9a
d4cbc9472d3acf5d97ba27ada7c2df4e674b93771befc50f7bcdeb99d533fa6c379199be1203154324cc96
339
verify success
```



```
----------------DH test-----------------
P:FFFFFFFFFFFFFFFFADF85458A2BB4A9AAFDC5620273D3CF1D8B9C583CE2D3695A9E13641146433FBCC93
9DCE249B3EF97D2FE363630C75D8F681B202AEC4617AD3DF1ED5D5FD65612433F51F5F066ED0856365553D
ED1AF3B557135E7F57C935984F0C70E0E68B77E2A689DAF3EFE8721DF158A136ADE73530ACCA4F483A797A
BC0AB182B324FB61D108A94BB2C8E3FBB96ADAB760D7F4681D4F42A3DE394DF4AE56EDE76372BB190B07A7
C8EE0A6D709E02FCE1CDF7E2ECC03404CD28342F619172FE9CE98583FF8E4F1232EEF28183C3FE3B1B4C6F
AD733BB5FCBC2EC22005C58EF1837D1683B2C6F34A26C1B2EFFA886B423861285C97FFFFFFFFFFFFFFFF
G:02
server_calc:512c3c317e39449ab4626563e5ecad42e87074da116baf42babcbada85884712080dcbd39c
e57f1c95fb428b00fc27dba6c367750b668fc4105a6349f6169f9f089aa8bd5f501279c7bca6e69ba9ef1a
49c38a438a01948879c5631841dfb02c5fe9e80a5db22cbc5c3d8ed8996d473677052e7e6d6c16ff2e310f
1b20314b5585bef80136eca3351d647467310309661de957dc9756adbd49f0ec6751470d9b15b50e877529
3d798ea8c0e1077432a03354fa4c023aaf044bd26896584c1545621571435ac2b0a8a0eb7f30e420b85ed7
f81be031c439ba393bd2cdd6856acec1800e3791be2fced648692c61737427324c8c9f0d9ded31ff942a65
953b7749
client_calc:512c3c317e39449ab4626563e5ecad42e87074da116baf42babcbada85884712080dcbd39c
e57f1c95fb428b00fc27dba6c367750b668fc4105a6349f6169f9f089aa8bd5f501279c7bca6e69ba9ef1a
49c38a438a01948879c5631841dfb02c5fe9e80a5db22cbc5c3d8ed8996d473677052e7e6d6c16ff2e310f
1b20314b5585bef80136eca3351d647467310309661de957dc9756adbd49f0ec6751470d9b15b50e877529
3d798ea8c0e1077432a03354fa4c023aaf044bd26896584c1545621571435ac2b0a8a0eb7f30e420b85ed7
f81be031c439ba393bd2cdd6856acec1800e3791be2fced648692c61737427324c8c9f0d9ded31ff942a65
953b7749
share_secert:93f4f993b5ec5ef0d450e9698bab9a09365ac040e6a7a284486393803811d4bc566821afc
99ca0f2814e4e21a9b8331a841b6f6485971bb6804d3de8550eb2ff
source_context:202022460107
cipher name:ARIA-128-ECB block size is:16
enc:429bb6029567fd7b802daf5740d64d26
dec:32303230323234363031303700000000
dec_str:202022460107
```



```
----------------OT test-----------------
alice's rand is(should be saved by alice without anyone known):
faea1a5a06244599ca8e571a3c96aba95e80b7a0dc5b121c23281607b946b3979b5d5289dd00a5d4e39af9
c607cbf347dce3ef2f7c4e5d5c9f0009b62934aaca4996dd0839799eb3fec4915f3a19616e39fc20daf9be
41f2f7a51ca414c057f8b2eb29af52b377edbcaddae2240780e4ce7aaadfbc6b484a1b7e241d15d27b5f
enc msg0:
678972d8664bc9e88a9702ac4045561a3b9f8e70a96ef872a65a593121e665cd6ab07ae157831b47a83673
2a4ba6b10c90b9ddf6e26a6f72e7a37edb0595bb9c459a4c166657182bed80dfe2ac92f02061a78763be9c
59712a02607a0399b64935d9a19f3ddecfe80ee5aba47d21e7fbaa7a1a59e9eb2cd91de1db241cd84b55
enc msg1:
47d3b6904cdc626e2b9a0ed4745b87c97cce2bf0b69da9759e9d9a9ca0a35e7640ca5bfa3044e5e597aaf8
a1dbbe2feb24d4c63df5855908618872628a845fe89568b56da90a750fca1ef3f9d623135bb7aa6fc40683
d1f56ddb216d9e6512f695f5a75e2d2db3648a042ba6d6800c33f5bad78211b958e8206ddd887c76f386
which one was chosen?
0
right!
```

对比

openssl：接口较老，命令行功能较尴尬，部分功能仅有接口但未成型；

pycryptodome：python，方便检验的自动化开发；

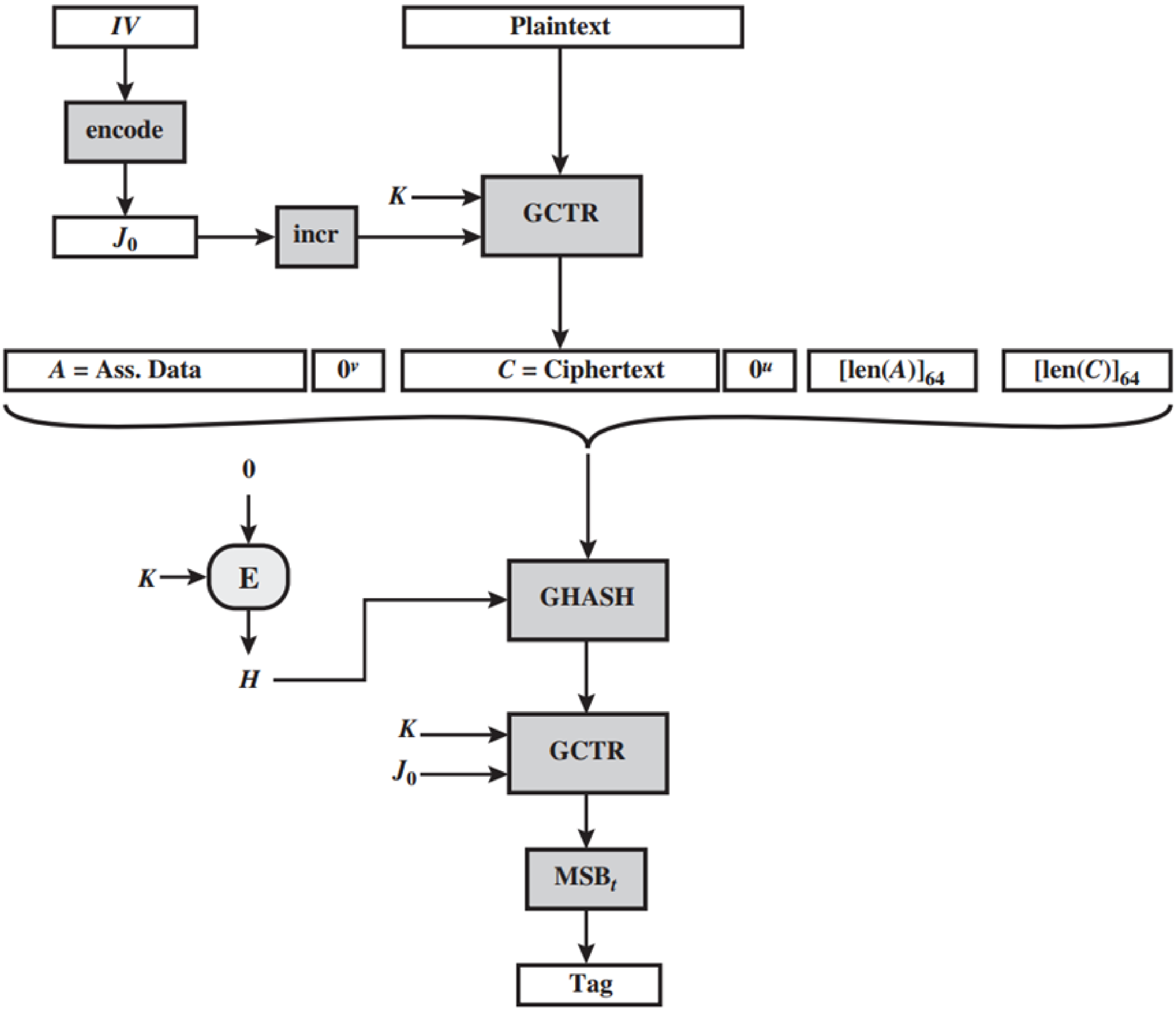
hashlib：原生接口，正确性最高；

mbedtls：嵌入式专用，编程函数较长（？），开发自由度较高；

重点算法解读

AES-GCM

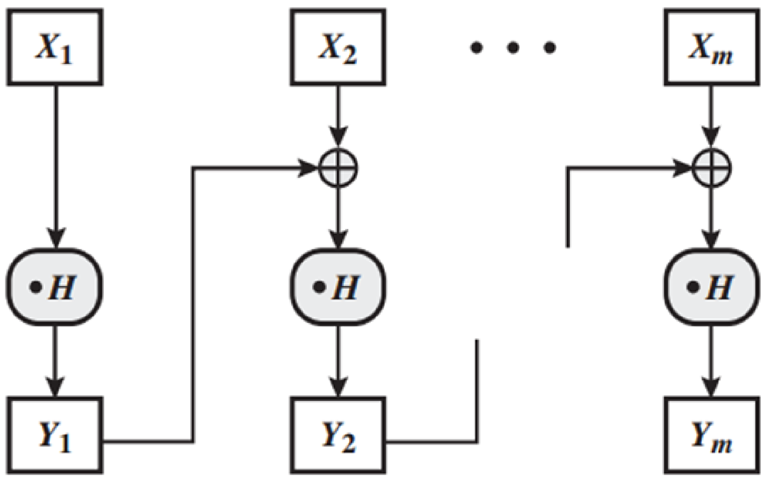
AES-GCM模式是一种认证加密模式，通过CTR与GMAC的组合实现了加密并认证。



$$H = E(k, 0^{128})$$

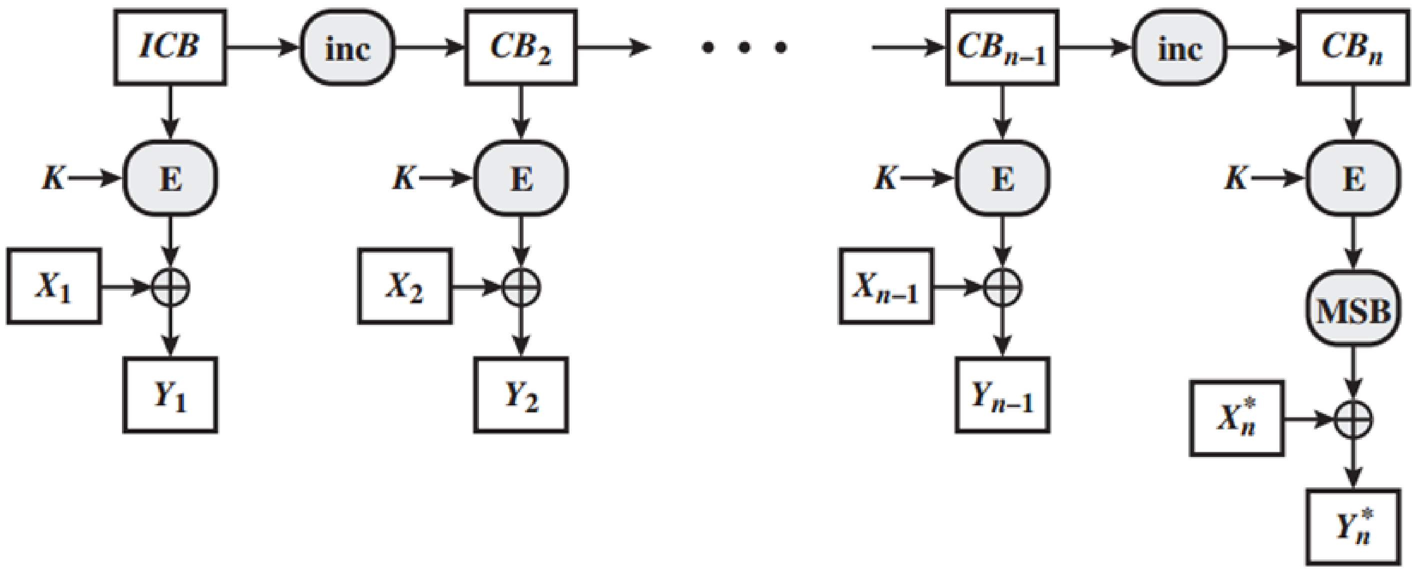
$$Y_i = (Y_{i-1} \oplus X_i) \cdot H$$

$$GF(2^{128})$$



(a) GHASH$_H(X_1 \| X_2 \| \ldots \| X_m) = Y_m$

$$GHASH_H(X) = (X_1 \cdot H^m) \oplus (X_2 \cdot H^{m-1}) \oplus \cdots \oplus (X_m \cdot H \quad)$$



(b) GCTR$_K(ICB, X_1 \| X_2 \| \ldots \| X_n^*) = Y_n^*$

非对称算法选讲

RSA

安全性1024bits

细节?

ECC（Elgamal）

安全性160bits

细节?

DH

密钥交换

细节?

参考文献

32.2 asn1parse_OpenSSL 中文手册

SP 800-38D

» RFC Editor

密码技术(图解密码技术的学习总结) - 走看看