实验二 分组密码实验报告

（一） 实验目的

1）熟悉一种密码库的使用；

2）掌握密码库的使用方法；

3）掌握 ECB、CBC、CTR 加密模式；

4）掌握 HMAC、GMAC 的构造方法

（二） 实验背景

为了更深入地理解加密和解密的基本原理，包括对称加密、非对称加密、哈希函数等，在实际项目中应用加密技术，熟悉使用加密算法库进行加密算法的编写，能让我们更好地运用密码学知识，强化运用密码学解决实际问题的能力。

1. AES 加密，使用对称密钥，对二进制原信息先分组，综合了轮密钥、流密钥、代换-置换网络等思想进行加密（加密细则见后），被认为是难以破译的现代高级加密标准。

2. AES 加密方式：对分组后的二进制信息分别进行至少 10 轮轮密钥加密，对于不同的分组，加密轮数不同（128bit 加密 10 轮，192bit 加密 12 轮，256bit 加密 14 轮）。对明文 x，第一轮与事先生成的密钥 key 异或，之后用 s 盒进行代换操作，先后进行行位移、列混淆（最后一轮没有列混淆），至此完成了 AES 加密的 ECB 模式。

3. 然而 ECB 模式虽然能够很好地混淆每一个分组，却不能很好地掩盖分组与分组之间的关系，对此，改用 CBC 模式，每一小段与初始块或者上一段的密文段进行异或运算后，再与密钥进行加密。CBC 模式在图像加密和安全性上都优于 ECB 模式。

4. CRT 模式中，通过一个自增的算子，这个算子用密钥加密之后的输出和明文异或的结果得到密文，相当于一次一密。这种加密方式简单快速，安全可靠，而且可以并行加密，但是在计算器不能维持很长的情况下，密钥只能使用一次。

5. MAC 值的意义：消息认证码，用于认证发送人身份。对于一段消息，通常使用一种带密钥的算法得到位数一定的 MAC 值，密钥是消息认证码安全性的保障。接收方去除末尾的 n 位 MAC 值，对信息再使用一次算法，比较得到的 MAC1 和原 MAC 值是否相同即可知道信息是否来自发送方。

6. HMAC，使用哈希函数构造消息认证码，构造方便，直接使用已有哈希函数即可，可靠性高。

7. GMAC，使用伽罗华域乘法运算得到的消息认证码。

实验内容

1. 熟悉一种密码库

（1）熟悉 crypto++的相关接口：从 https://www.cryptopp.com/下载 crypto++，并在自己的设备上进行安装。

或者（2）pycryptodome: pip3 install pycryptodome
https://www.pycryptodome.org/src/introduction

或者（3）cryptography: pip3 install cryptography
https://cryptography.io/en/latest/

选择（2）或者（3）的需要熟悉多精度整数库的使用
gmpy2: pip3 install gmpy2
https://gmpy2.readthedocs.io/en/latest/index.html

2. 读取文件 plaintext.txt，完成以下加密任务：

1)自动生成一个密钥，用 AES 对其进行加密，采用 ECB 模式，并把密文写到文件 cipher-ecb.txt

2)自动生成一个密钥，用 AES 对其进行加密，采用 CBC 模式，并把密文写到文件 cipher-cbc.txt

3)自动生成一个密钥，用 AES 对其进行加密，采用 CTR 模式，并把密文写到文件 cipher-ctr.txt

对以上得到的三个密文进行解密，对比解密是否正确。

4）随机生成一个认证码密钥，采用 HMAC，生成 plaintext.txt 的认证码，

并把认证码拼在 plaintext 的后面，一起写到文件 message-mac-hmac.txt

5）随机生成一个认证吗密钥，采用 GMAC，生成 plaintext.txt 的认证码，并把认证码拼在 plaintext 的后面，一起写到文件 message-mac-gmac.txt

（三） 实验过程

1) 自动生成一个密钥，用 AES 对其进行加密，采用 ECB 模式，并把密文写到文件 cipher-ecb.txt

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Random import get_random_bytes

def main():
    plain=''
    with open("cryptography exam\exam2\实验内容-20240401\plaintext.txt",'r') as file:
        plain=file.read()
        print(plain)
    b_plain=bytes(plain,encoding='utf-8')
    key=get_random_bytes(16)
    cipher=AES.new(key,AES.MODE_ECB)
    crypted_data=cipher.encrypt(pad(b_plain,AES.block_size))

    #with open("cryptography exam\exam2\实验内容-20240401\cipher-ecb.txt",'w') as file:
    #    file.write(str(crypted_data))

    file_out = open("cryptography exam\exam2\实验内容-20240401\cipher-ecb.txt", "wb")
    # 在文件中依次写入 key、iv 和密文 encrypted_data
    [file_out.write(x) for x in (key, crypted_data)]



if __name__=="__main__":
    main()
```

解密算法：

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

# 从前边文件中读取出加密的内容
file_in = open("cryptography exam\exam2\实验内容-20240401\cipher-ecb.bin", "rb")
# 依次读取 key、iv 和密文 encrypted_data，16 等是各变量长度，最后的-1 则表示读取到文件末尾
key, encrypted_data = [file_in.read(x) for x in (16,-1)]

# 实例化加密套件
cipher = AES.new(key, AES.MODE_ECB)
k=cipher.decrypt(encrypted_data)
data = unpad(k, AES.block_size)
print(data)
```

2) 自动生成一个密钥，用 AES 对其进行加密，采用 CBC 模式，并把密文写到文件 cipher-cbc.txt

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Random import get_random_bytes

def main():
    plain=''
    with open("cryptography exam\exam2\实验内容-20240401\plaintext.txt",'r') as file:
        plain=file.read()
    b_plain=bytes(plain,encoding='utf-8')
    key=get_random_bytes(16)

    cipher=AES.new(key,AES.MODE_CBC)
    crypted_data=cipher.encrypt(pad(b_plain,AES.block_size))

    #with open("cryptography exam\exam2\实验内容-20240401\cipher-ecb.txt",'w') as file:
    #    file.write(str(crypted_data))
```

```
    file_out = open("cryptography exam\exam2\实验内容-20240401\cipher-cbc.txt", "wb")
    # 在文件中依次写入 key、iv 和密文 encrypted_data
    [file_out.write(x) for x in (key,cipher.iv,crypted_data)]
```

解密算法：

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

# 从前边文件中读取出加密的内容
file_in = open("cryptography exam\exam2\实验内容-20240401\cipher-cbc.txt", "rb")
# 依次读取 key、iv 和密文 encrypted_data，16 等是各变量长度，最后的-1 则表示读取到文件末尾
key,iv, encrypted_data = [file_in.read(x) for x in (16,AES.block_size,-1)]

# 实例化加密套件
cipher = AES.new(key, AES.MODE_CBC,iv)
k=cipher.decrypt(encrypted_data)
data = unpad(k, AES.block_size)
print(data)
```

3) 自动生成一个密钥，用 AES 对其进行加密，采用 CTR 模式，并把密文写到文件 cipher-ctr.txt

```
from Crypto.Cipher import AES
from Crypto.Util import Counter
from Crypto.Random import get_random_bytes
from Crypto.Util.number import bytes_to_long
import os

def main():
    plain=''
    with open("cryptography exam\exam2\实验内容-20240401\plaintext.txt",'r') as file:
        plain=file.read()
```

```
    b_plain=bytes(plain,encoding='utf-8')
    key=get_random_bytes(16)

    ctr = Counter.new(AES.block_size * 8, initial_value=bytes_to_long(bytes(os.urandom(16))))
    cipher=AES.new(key,counter=ctr,mode=AES.MODE_CTR)
    crypted_data=cipher.encrypt(b_plain)

    #with open("cryptography exam\exam2\实验内容-20240401\cipher-ecb.txt",'w') as file:
    #    file.write(str(crypteddata))

    file_out = open("cryptography exam\exam2\实验内容-20240401\cipher-ctr.txt", "wb")
    # 在文件中依次写入 key、iv 和密文 encrypted_data
    [file_out.write(x) for x in (key,ctr,crypted_data)]



if __name__=="__main__":
    main()
```

解密算法：

```
from Crypto.Cipher import AES

# 从前边文件中读取出加密的内容
file_in = open("cryptography exam\exam2\实验内容-20240401\cipher-ctr.txt", "rb")
# 依次读取 key、iv 和密文 encrypted_data，16 等是各变量长度，最后的-1 则表示读取到文件末尾
key,ctr, encrypted_data = [file_in.read(x) for x in (16,AES.block_size,-1)]

# 实例化加密套件
cipher = AES.new(key, AES.MODE_CTR)
k=cipher.decrypt(encrypted_data)
print(str(k,encoding='utf-8'))
```

对以上得到的三个密文进行解密，对比解密完全正确。

4） 随机生成一个认证码密钥，采用 HMAC，生成 plaintext.txt 的认证码，并把认证码拼在 plaintext 的后面，一起写到文件 message-mac-hmac.txt

```
from Crypto.Hash import HMAC
from Crypto.Hash import SHA256
from Crypto.Random import get_random_bytes

def main():
    plain=''
    with open("cryptography exam\exam2\实验内容-20240401\plaintext.txt",'r') as file:
        plain=file.read()
    b_plain=bytes(plain,encoding='utf-8')

    key=get_random_bytes(32)
    hmac=HMAC.new(key,msg=b_plain,digestmod=SHA256)
    hmac_code=hmac.hexdigest()
    final=plain+hmac_code

    with open("cryptography exam\exam2\实验内容-20240401\plaintext_HMAC.txt",'w') as file:
        file.write(final)

if __name__=="__main__":
    main()
```

5） 随机生成一个认证吗密钥，采用 GMAC，生成 plaintext.txt 的认证码，并把认证码拼在 plaintext 的后面，一起写到文件 message-mac-gmac.txt

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

def main():
    plain=''
    with open("cryptography exam\exam2\实验内容-20240401\plaintext.txt",'r') as file:
        plain=file.read()
    b_plain=bytes(plain,encoding='utf-8')
```

```
    key=get_random_bytes(32)
    gmac = AES.new(key, AES.MODE_GCM)
    ciphertext, tag = gmac.encrypt_and_digest(b_plain)

    with open("cryptography exam\exam2\实验内容-20240401\plaintext_GMAC.txt",'w') as file:
        file.write(plain+tag.hex())

if __name__=="__main__":
    main()
```

（四） 实验心得

本次实验包含了几种典型的 AES 加密模式和消息认证码。对于我个人的学习有着相当大的帮助，能够为接下来的密码学学习打好基础，帮助我理解各种信息传递方式优劣和作用。