实验二 分组密码实验报告

1. 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

1. 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

1. 实验过程
2. 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)

1. 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)

1. 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。

1. 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()

1. 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()

1. 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。