实验一 古典密码实验报告

（一） 实验目的

1） 掌握单表代替密码；

2） 掌握多表代替密码，重点是多表代替密码的实现；

3） 掌握置换密码；

4） 掌握对单表、多表密码体制的攻击；

5） 掌握 16 进制转换与读写。

（二） 实验背景

代替密码：

将字符用另一特定字符代替而成，容易表现出统计特征，容易被攻击。核心思路在于代替的规则。解密只需反向替代即可。

维吉尼亚密码：

一种经典的密码加密技术，基于多表替换密码的原理，安全性高于代替密码，适用于手工加密的简单而有效的加密方法，核心思路在于使用一个密钥，一个字符串，用来确定字母的偏移量。解密需要对照密钥和维吉尼亚表将密文还原。

希尔密码：

基于线性代数的密码加密技术,使用矩阵乘法来加密和解密文本,相对于简单的代替密码，希尔密码提供了更高的安全性。核心思路在于将原文本分块，与特定的矩阵相乘加密。对于解密则乘以逆矩阵后模 26 还原原文本。

置换密码：

将明文中的字符按照固定的规则重新排列来生成密文，本质上是希尔密码的一种特定的表现形式，将明文中的字符按照一定的规则重新排列，对于现代密码分析方法来说并不安全。解密需要掌握原本的置换规则后反向置换（就是与逆矩阵相乘）

密码统计分析：

对于出现的字符频率进行统计，英文字符中不同的字母出现的频率不同，当密文量较大时会出现明显的统计特征，据此可以分析出原文字符的部分对应关系。

base64 编码：

本质上不是一种密码，而是一种传输的编码方式。将信息转化为二进制字符串，以 3 个字节转化为 4 位 base64 编码的方式用于传输，可以在不同的操作系统、不同的识别方式下确保信息本身没有因兼容性问题而受损。译码过程则时将 4 位 base64 编码转为 3 个字节得到原文本。

（三） 实验内容

plaintext = cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms, to transform messages in ways that are hard to decipher. These deterministic algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet and confidential communications such as credit card transactions and email.

注意：标点符号和空格不加密

1）实现单表代替密码：凯撒密码,实现对英文消息 plainxtext 的代替加密；

2）实现多表代替密码：维吉尼亚密码、希尔密码,实现对英文消息的代替加密；

3）实现分组大小为 8 的置换密码，实现对英文消息 plainxtext 的置换加密；

4）实现对代替密码的攻击,找到 1）中密文对应的解密密钥；

5）实现二进制文件读写、字节替换密码，读取二进制文件 cryptointro.txt，并对其进行加密，把密文写到文件 cipher.txt。

6）了解 base64 编码，读取上述 cryptointro.txt 和 cipher.txt，并以 base64 编码输出并显示。

（四） 实验过程

1）实现单表代替密码：凯撒密码,实现对英文消息 plainxtext 的代替加密；

```
def GenKey():
    #生成加密密钥和解密密钥
    K=3
    return K

def Encrypt(str):
    #加密函数
    temp=''
    Key=GenKey()
```

```
    for j in str:
        if ord(j)<=ord('z') and ord(j)>=ord('a'):
            t=chr(ord(j)+Key)
            if ord(t)>ord('z'):
                temp+=chr(ord(t)-26)
            else:
                temp+=t
        else:
            temp+=j
    return temp
```

2）实现多表代替密码：维吉尼亚密码、希尔密码,实现对英文消息的代替加密；

```
def Vigenere_GenKey():
    #生成维吉尼亚加密密钥和解密密钥
    Key=[random.randint(1,10) for i in range(random.randint(2,6))]
    return Key
def Vigenere_Encrypt(str,key):
    #维吉尼亚加密函数
    En_Key=key
    m=len(key)
    cipher=''
    i=0
    for j in str:
        if (ord(j)>=ord('a') and ord(j)<=ord('z')) or (ord(j)>=ord('A') and ord(j)<=ord('Z')):
            j=chr(ord(j)+En_Key[i%m])
            i+=1
            if ord(j)>ord('z'):
                j=chr(ord(j)-26)
            cipher+=j
        else:
            cipher+=j
    return cipher
def Vigenere_Decrypt(str,key):
    #维吉尼亚解密函数
```

```
    De_Key=key
    m=len(De_Key)
    plain=''
    i=0
    for j in str:
        if (ord(j)>=ord('a') and ord(j)<=ord('z')) or (ord(j)>=ord('A') and ord(j)<=ord('Z')):
            j=chr(ord(j)-De_Key[i%m])
            i+=1
            if ord(j)<ord('a'):
                j=chr(ord(j)+26)
            plain+=j
        else:
            plain+=j
    return plain
```

维吉尼亚密码实现较为简单，使用以上代码能够有效对维吉尼亚密码实现加密和解密。

希尔密码实现过程较为复杂，因此单独放在下方

```
def Hill_GenKey():
    #生成希尔加密密钥和解密密钥
    m=4
    Key=np.random.randint(2,10,(m,m))
    Key_det=np.around(np.linalg.det(Key))
    egcd_det=egcd(Key_det,26)
    if egcd_det[0]!=1:
        Key,Key_inverse=Hill_GenKey()
        return Key,Key_inverse
    anti_det=egcd_det[1]
    if anti_det<0:
        anti_det+=26
    Key_=[0 for i in range(m)]
    Key_=[Key_.copy() for i in range(m)]
    for i in range(m):
        for j in range(4):
            temp=Key.T.copy()
```

```
            temp=np.delete(temp,i,0)
            temp=list(temp)
            for t in range(m-1):
                temp[t]=np.delete(temp[t],j,0)
            temp=np.array(temp)
            if (i+j)%2==1:
                Key_[i][j]=-1*np.linalg.det(temp)
            elif (i+j)%2==0:
                Key_[i][j]=np.linalg.det(temp)
    Key_inverse=((np.around(np.array(Key_))).astype(np.int32))*anti_det
    return Key,Key_inverse
def Hill_Encrypt(str,En_Key):
    #希尔加密函数
    cipher=['#' for i in range(len(str))]
    m=len(En_Key[0])
    i,j,n=0,0,0
    ls_temp=[]
    for temp in str:
        if (ord(temp)>=ord('a') and ord(temp)<=ord('z')) or (ord(temp)>=ord('A') and
ord(temp)<=ord('Z')):
            ls_temp.append(alphabet[temp])
            i+=1
            j+=1
        else:
            cipher[i]=temp
            i+=1
        if j==m:
            ls_temp=np.dot(ls_temp,En_Key)
            for k in range(m):
                for n in range(len(cipher)):
                    if ord(cipher[n])==ord('#'):
                        break
                cipher[n]=alpha_str[ls_temp[k]%26]
```

```
            ls_temp=[]
            j=0
    return ''.join(cipher)
def Hill_Decrypt(str,De_Key):
    #希尔解密函数
    plain=['#' for i in range(len(str))]
    m=len(De_Key[0])
    i,j,n=0,0,0
    ls_temp=[]
    for temp in str:
        if (ord(temp)>=ord('a') and ord(temp)<=ord('z')) or (ord(temp)>=ord('A') and
ord(temp)<=ord('Z')):
            ls_temp.append(alphabet[temp])
            i+=1
            j+=1
        else:
            plain[i]=temp
            i+=1
        if j==m:
            ls_temp=(np.dot(np.array(ls_temp),De_Key)).astype(np.int32)
            for k in range(m):
                for n in range(len(plain)):
                    if ord(plain[n])==ord('#'):
                        break
                plain[n]=alpha_str[ls_temp[k]%26]
            ls_temp=[]
            j=0
    return ''.join(plain)
```

3）实现分组大小为 8 的置换密码，实现对英文消息 plainxtext 的置换加密；

```
def GenKey():
    #生成加密密钥和解密密钥
    m=8
    Key=[[0 for j in range(m)]for i in range(m)]
```

```
    t=sample(range(m),m)
    for i in range(m):
        Key[i][t[i]]=1
    Key=np.array(Key)
    Key_inverse=np.linalg.inv(Key)
    return Key,Key_inverse.astype(np.int32)


def Encrypt(plain,En_Key):
    #加密函数
    mark=',!.- '
    t=''
    m=len(En_Key)
    cipher=['#' for i in range(len(plain))]
    for j,val in enumerate(plain):
        if val not in mark:
            t+=val
        else:
            cipher[j]=val
        if len(t)==m:
            temp=[ord(i) for i in t]
            temp=list(np.dot(temp,En_Key))
            t=''.join([chr(i) for i in temp])
            for n in range(m):
                for k,val in enumerate(cipher):
                    if val=='#':
                        cipher[k]=t[0]
                        t=t[1:]
                        break
    if t:
        for i in range(m-len(t)):
            t+='A'
            cipher+='#'
        temp=[ord(i) for i in t]
```

```
            temp=list(np.dot(temp,En_Key))
            t=''.join([chr(i) for i in temp])
            for n in range(m):
                for k,val in enumerate(cipher):
                    if val=='#':
                        cipher[k]=t[0]
                        t=t[1:]
                        break
        return ''.join(cipher)


def Decrypt(cipher,De_Key):
    #解密函数
    mark=',!.- '
    t=''
    m=len(De_Key)
    plain=['#' for i in range(len(cipher))]
    for j,val in enumerate(cipher):
        if val not in mark:
            t+=val
        else:
            plain[j]=val
        if len(t)==m:
            temp=[ord(i) for i in t]
            temp=list(np.dot(temp,De_Key))
            t=''.join([chr(i) for i in temp])
            for n in range(m):
                for k,val in enumerate(plain):
                    if val=='#':
                        plain[k]=t[0]
                        t=t[1:]
                        break

    return ''.join(plain).lower()
```

4）实现对代替密码的攻击,找到 1）中密文对应的解密密钥；

考虑到密码本身较为简单，攻击方使用唯密攻击，且知道使用的密码体制

```
def Attack(cipher):
    #默认已知密码体制，进行移位密码穷尽
    plain=[]
    alpha_str='abcdefghijklmnopqrstuvwxyz'
    for i in range(25):
        temp=''
        for j in cipher:
            if j in alpha_str:
                t=chr(ord(j)-i)
                if t in alpha_str:
                    temp+=t
                else:
                    t=chr(ord(t)+26)
                    temp+=t
            else:
                temp+=j
        plain.append(temp)
    return plain
```

5）实现二进制文件读写、字节替换密码，读取二进制文件 cryptointro.txt，并对其进行加密，把密文写到文件 pmt-cipher.txt。

二进制字符串加密后直接以字符形式存储，故密文为二进制字符串

```
def GenKey():
    #生成加密密钥和解密密钥
    m=128
    Key=[[0 for j in range(m)]for i in range(m)]
    t=random.sample(range(m),m)
    for i in range(m):
        Key[i][t[i]]=1
    Key=np.array(Key)
    return Key
```

```
def Encrypt(plain):
    #加密函数
    En_Key=GenKey()
    cipher=''
    t=''
    mark=',!.- 。，、\n'
    b_plain=[bin(ord(i)).replace('0b','') if i not in mark else i for i in plain]
    temp_text=''.join(b_plain)
    m=len(En_Key)
    cipher=['#' for i in range(len(temp_text))]
    for j,val in enumerate(temp_text):
        if val not in mark:
            t+=val
        else:
            cipher[j]=val
        if len(t)==m:
            temp=[ord(i) for i in t]
            temp=list(np.dot(temp,En_Key))
            t=''.join([chr(i) for i in temp])
            for n in range(m):
                for k,val in enumerate(cipher):
                    if val=='#':
                        cipher[k]=t[0]
                        t=t[1:]
                        break
    if t:
        for i in range(m-len(t)):
            t+='1'
            cipher+='#'
        temp=[ord(i) for i in t]
        temp=list(np.dot(temp,En_Key))
        t=''.join([chr(i) for i in temp])
```

```
        for n in range(m):
            for k,val in enumerate(cipher):
                if val=='#':
                    cipher[k]=t[0]
                    t=t[1:]
                    break
    return ''.join(cipher)
```

6）了解 base64 编码，读取上述 cryptointro.txt 和 pmt-cipher.txt，并以 base64 编码输出并显示。

```
import base64
def main():
    str1,str2='',''
    with open("cryptography exam\exam1\古典密码实验内容
-20240226\cryptointro.txt",'r',encoding='utf-8') as file:
        str1=file.read()
    encodestr=base64.b64encode(str1.encode('utf-8'))
    print(encodestr)
    with open("cryptography exam\exam1\古典密码实验内容
-20240226\pmt-cipher.txt",'r',encoding='utf-8') as file:
        str2=file.read()
    encodestr=base64.b64encode(str2.encode('utf-8'))
    print(encodestr)
```

对于 base64 的理解：

base64 并不是加密方式而是文本传输的编码方式。将信息转化为二进制字符串，以 3 个字节转化为 4 位 base64 编码的方式用于传输，可以在不同的操作系统、不同的识别方式下确保信息本身没有因兼容性问题而受损。译码过程则时将 4 位 base64 编码转为 3 个字节得到原文本。

（五） 实验心得

本次实验包含了几种典型的古典密码及其分析。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，帮助我理解各种加密方式的优缺点。