

密码学

平时实验报告

（ 2020 / 2021学年 第 二 学期）

题 目：古典密码实验

|  |  |
| --- | --- |
| **专 业** | 信息安全 |
| **组 长 姓 名** | 任远哲 |
| **组 员 姓 名** | 戴俊皓，郭稼逸 |
| **班 级 学 号** | B190307,B19031614，B19030733, B19030729 |
| **指 导 教 师** | 李琦 |
| **指 导 单 位** | 信息安全系 |
| **日 期** | 2021年3月12日 |

|  |  |
| --- | --- |
| **成员分工** | |
| 组长（1） | 任远哲，B19031614，编写Hill加解密，撰写报告 |
| 组员（1） | 戴俊皓, B19030733,编写仿射加解密 |
| 组员（2） | 郭稼逸,B19030729，编写置换加解密 |
| **简短评语：**  **指导教师：年月日** | |

1. **课题内容和要求**

**1.实验环境**

实验主机操作系统为Windows 7

**2.实验内容**

实验1置换密码：选择分组n=7的周期置换（自定义置换），进行加解密

实验2:Hill密码，给定加密矩阵，求出其解密矩阵，进行加密解密

实现3：仿射密码，设置参数，进行加密解密

**二、课题需求分析**

**实验1：置换密码：**

1. 将密文用#补齐，使其长度为周期的整数倍
2. 输入周期长度（这里输入7）
3. 输入对换形式（ 如(143)(56) ）
4. 根据（3）中对换形式，求出置换密钥
5. 求出密文
6. 根据（4）中置换密钥求出解密密钥
7. 将（5）中密文解密
8. 统计字符频率

**完整代码和注释：**

#include<stdio.h>

#include<map>

#include<string.h>

using namespace std;

int key[100]={0};//输入(143)(56),经过getkey，得到421365

int decodekey[100]={0};//解密密钥

void getkey(char\* pointerK){

int\* ktemp=key;

int i;

int t=0;

for(i=0;i<100;i++)

key[i]=i;

do{

if(\*pointerK=='(')

continue;

else{

t=\*pointerK-'0';

while(\*(pointerK+1)!=')'){

ktemp[\*pointerK-'0']=\*(pointerK+1)-'0';

pointerK++;

}

ktemp[\*pointerK-'0']=t;

pointerK++;

}

} while(\*(++pointerK)!='\0');

}

int main()

{

char plaintext[9999]="inseveraldistributedsystemsausershouldonlybeabletoaccessdataifauserpossesacertainsetofcredentialsorattributescurrentlytheonlymethodforenforcingsuchpoliciesistoemployatrustedservertostorethedataandmediateaccesscontrolhoweverifanyserverstoringthedataiscompromisedthentheconfidentialityofthedatawillbecompromisedinthispaperwepresentasystemforrealizingcomplexaccesscontrolonencrypteddatathatwecallciphertextpolicyattributebasedencryptionbyusingourtechniquesencrypteddatacanbekeptconfidentialevenifthestorageserverisuntrustedmoreoverourmethodsaresecureagainstcollusionattacks",cipher[9999],decodetext[9999];//明文，密文，由密文解密得到的明文

char K[100];//K=(143)(56)

int cyclelen,len;

//printf("请输入你要输入的明文：");

//scanf("%s",plaintext);

len=strlen(plaintext);

printf("明文为：\n");

for(int i=0; i<len; i++)

printf("%c",plaintext[i]);

printf("\n");

printf("输入周期长度：");

scanf("%d",&cyclelen);

getchar();

printf("\n");

//补齐末尾空缺

for(int i=len; i<len+cyclelen-(len%cyclelen); i++)

{plaintext[i]='#';

}

plaintext[len+cyclelen-len%cyclelen]='\0';

printf("请输入置换形式，形如(143)(56) \n");

gets(K);//K=(143)(56)

printf("加密密钥如下：\n");

//(143)(56),经过getkey，得到421365

getkey(K);

//打印加密密钥

for(int i=1;i<=cyclelen;i++)

{printf("%d ", key[i]) ;

}

//加密过程

int flag=1;

for(int i=0; i<len+cyclelen-len%cyclelen; i++)//plaintext[len+cyclelen-len%cyclelen]='\0'

{ if(flag==cyclelen+1){flag=1;}

cipher[i]=plaintext[i-(flag-key[flag])];

flag++;

}

//打印密文

printf("\n\n");

printf("密文如下：\n");

for(int i=0; i<len+cyclelen-len%cyclelen; i++)

printf("%c",cipher[i]);

printf("\n\n");

printf("通过加密密钥，构造解密密钥：\n");

//通过加密密钥，构造解密密钥

for(int i=1;i<=cyclelen;i++)

{

decodekey[key[i]]=i;

}

//打印解密密钥

for(int i=1;i<=cyclelen;i++)

{printf("%d ", decodekey[i]) ;

}

//解密过程

flag=1;

for(int i=0; i<len; i++)

{ if(flag==cyclelen+1){flag=1;}

decodetext[i]=cipher [i-(flag-decodekey[flag])];

flag++;

}

//打印解密后密文

printf("\n\n");

printf("密文经过解密结果如下：\n");

for(int i=0; i<len; i++)

printf("%c",decodetext[i]);

printf("\n");

printf("密文统计：\n");

map<char,int>statistics;

for(int i=0; i<26; i++)

statistics.insert(pair<int, int>(i+97, 0));

for(int i=0; i<len; i++)

{

statistics[cipher[i]]++;

}

for(int i=0; i<26; i++)

printf("%c：%d个",i+97,statistics[i+97]);

return 0;

}

**实验2：Hill密码：**

1. 将密文转成5\*5的明文矩阵（a-z映射到1-26）
2. 使用给定的加密矩阵，与明文矩阵做矩阵乘法，得到密文矩阵
3. 输出密文（1-26映射回a-z）
4. 矩阵A在Z26上的逆是[(det A)^-1(mod26)\*伴随(A)](mod26),根据此公式求出解密矩阵
5. 密文矩阵与解密矩阵运算，得到明文矩阵
6. 得到明文（1-26映射回a-z）
7. 统计字符频率

**完整代码和注释：**

import numpy as np

def encode(string, size):

#把 string 转换成 size x size 的 int32 numpy.ndarray

blocks = (string[i:i + size] for i in range(0, len(string), size))

# 将 a-z 编码为 0-25

return np.array([list(map(ord, block)) for block in blocks], dtype='int32') - ord('a')

def matrixMul(A, B):

res = [[0] \* len(B[0]) for i in range(len(A))]

for i in range(len(A)):

for j in range(len(B[0])):

for k in range(len(B)):

res[i][j] += A[i][k] \* B[k][j]

return res

#把一个 0-25 数字映射到字母

def modify(x):

return chr(x%26+ord('a'))

def matrixT(A):

B=[[0]\*len(A)]\*len(A[0])

for i in range (len(A)):

for j in range (len(A[0])):

B[j][i]=A[i][j]

return list(map(list, zip(\*A)))

def Statistics(plaintext ,cipher):

Plain = {chr(i):0 for i in range(97,123)}

Cipher={chr(i):0 for i in range(97,123)}

for i in plaintext:

Plain[i]=Plain[i]+1

for i in cipher:

Cipher[i]=Cipher[i]+1

print("明文字频统计",Plain)

print("密文字频统计",Cipher)

# 要加密的信息

msg= 'inseveraldistributedsystemsausershouldonlybeabletoaccessdataifauserpossesacertainsetofcredentialsorattributescurrentlytheonlymethodforenforcingsuchpoliciesistoemployatrustedservertostorethedataandmediateaccesscontrolhoweverifanyserverstoringthedataiscompromisedthentheconfidentialityofthedatawillbecompromisedinthispaperwepresentasystemforrealizingcomplexaccesscontrolonencrypteddatathatwecallciphertextpolicyattributebasedencryptionbyusingourtechniquesencrypteddatacanbekeptconfidentialevenifthestorageserverisuntrustedmoreoverourmethodsaresecureagainstcollusionattacks'

print("要加密的明文为：",msg,"\r\n")

# 加密矩阵

encryptermatrix = [[10, 5, 12, 0, 0], [3, 14, 21, 0, 0], [8, 9, 11, 0, 0], [0, 0, 0, 11, 8], [0, 0, 0, 3, 7]]

#print(encryptermatrix)

# 加密

encrypted = matrixMul(encode(msg, 5),encryptermatrix )#密文矩阵

encryptedlist =sum(encrypted,[])#转一维

cipher = ''.join(map(modify, encryptedlist))#映射到字母

print("密文为：",cipher,"\r\n")

# 解密矩阵decrypter。矩阵A的逆是(det A)^-1\*伴随(A)

inverseelement=[0,1,0,9,0,21,0,15,0,3,0,19,0,0,0,7,0,23,0,11,0,5,0,17,0,25]#0到25在mod(26)下逆元

det=(int)(np.linalg.det(encryptermatrix))#求行列式

#np.linalg.det(encryptermatrix)\*np.linalg.inv(encryptermatrix为伴随矩阵

decryptermatrix =[[(round(z)\*(inverseelement[det%26]))%26 for z in y ]for y in (np.linalg.det(encryptermatrix)\*np.linalg.inv(encryptermatrix)).tolist()]

#decryptermatrix=[[21,15,17,0,0],[23,2,16,0,0],[25,4,3,0,0],[0,0,0,7,18],[0,0,0,23,11]]

#print(decryptermatrix)

# 解密代码

decrypted = matrixMul(encode(cipher, 5),decryptermatrix)#解密后明文矩阵

decryptedlist=sum(decrypted,[])#list转一维

# 明文

plaintext = ''.join(map(modify,decryptedlist))#map对list批量操作

print("解密后明文为",plaintext,"\r\n")

Statistics(plaintext,cipher)#统计

**实验3：仿射密码**

1. 将明文字符串映射为1-26的list
2. 选择算法的参数a=11.b=6
3. 根据e(x)=a\*x+b(mod26),加密明文list,得到密文列表
4. 求a在模26下的逆元a^-1
5. 根据d(e(x))=a^-1((e(x)-b)(mod26),解密密文列表，再将1-26映射回a-z，即为解密后的明文。

**完整代码和注释：**

def myExtGCD(a, b):

# a: 模的取值,b: 想求逆的值

if b == 0:

return 1, 0, a

else:

x, y, q = myExtGCD(b, a % b)

# q = gcd(a, b) = gcd(b, a%b)

x, y = y, (x - (a // b) \* y)

return x, y, q

def lettermodify(x):

return ord(x)-ord('a')

def numbermodify(x):

return chr(x%26+ord('a'))

def Statistics(plaintext ,cipher):

Plain = {chr(i):0 for i in range(97,123)}

Cipher={chr(i):0 for i in range(97,123)}

for i in plaintext:

Plain[i]=Plain[i]+1

for i in cipher:

Cipher[i]=Cipher[i]+1

print("明文字频统计",Plain)

print("密文字频统计",Cipher)

msg = 'inseveraldistributedsystemsausershouldonlybeabletoaccessdataifauserpossesacertainsetofcredentialsorattributescurrentlytheonlymethodforenforcingsuchpoliciesistoemployatrustedservertostorethedataandmediateaccesscontrolhoweverifanyserverstoringthedataiscompromisedthentheconfidentialityofthedatawillbecompromisedinthispaperwepresentasystemforrealizingcomplexaccesscontrolonencrypteddatathatwecallciphertextpolicyattributebasedencryptionbyusingourtechniquesencrypteddatacanbekeptconfidentialevenifthestorageserverisuntrustedmoreoverourmethodsaresecureagainstcollusionattacks'

a=11

b=6

plaincode=map(lettermodify,list(msg))

cipercode=[]

for i in plaincode:

cipercode.append((i\*a+b)%26)

ciper=''.join(map(numbermodify,cipercode))

print('密文为：',ciper)

cipercode=map(lettermodify,list(ciper))

plaincode=[]

for i in cipercode:

plaincode.append((myExtGCD(26,a)[1]\*(i-b))%26)

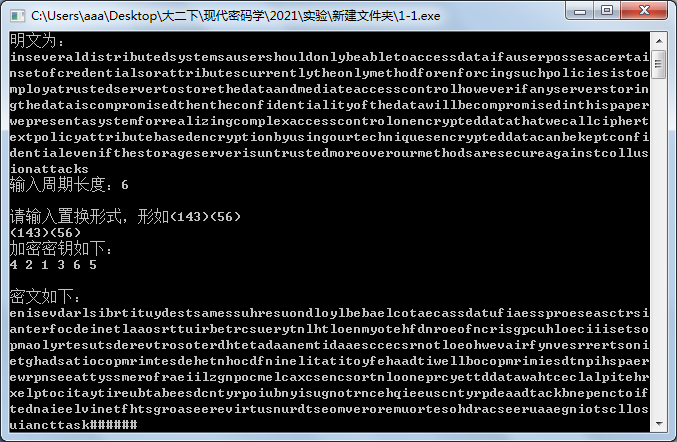
plaintext=''.join(map(numbermodify,plaincode))

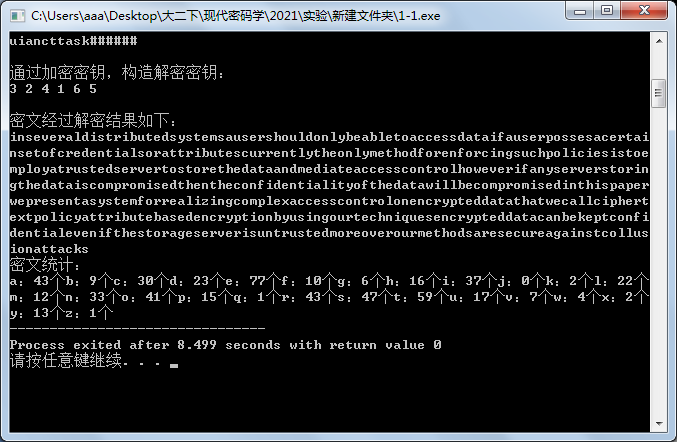
print("解密后明文为",plaintext,"\r\n")

Statistics(msg,ciper)

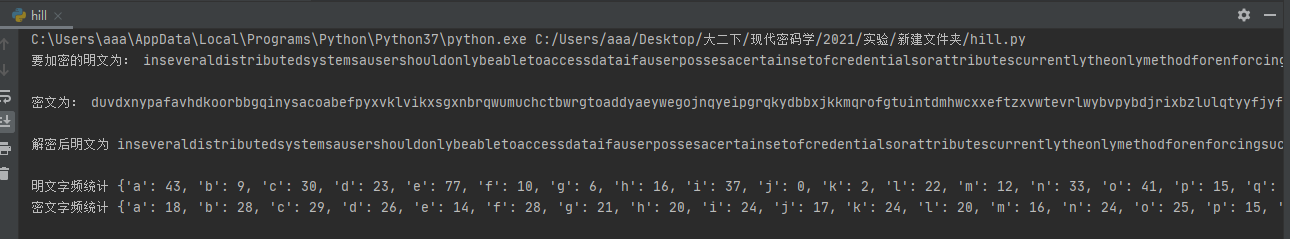
**三.测试数据及其结果分析**

明文字频统计 {'a': 43, 'b': 9, 'c': 30, 'd': 23, 'e': 77, 'f': 10, 'g': 6, 'h': 16, 'i': 37, 'j': 0, 'k': 2, 'l': 22, 'm': 12, 'n': 33, 'o': 41, 'p': 15, 'q': 1, 'r': 43, 's': 47, 't': 59, 'u': 17, 'v': 7, 'w': 4, 'x': 2, 'y': 13, 'z': 1}

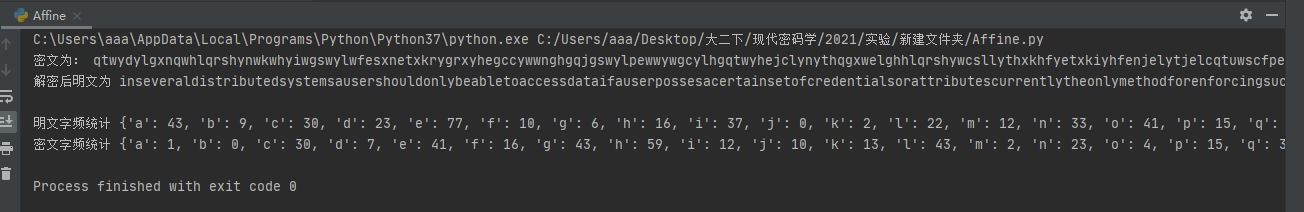
** 实验1运行结果**

****

**实验2运行结果**

****

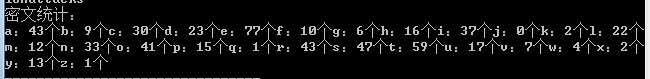
**实验3运行结果**

****

**密文字符统计工作**

明文字频统计 {'a': 43, 'b': 9, 'c': 30, 'd': 23, 'e': 77, 'f': 10, 'g': 6, 'h': 16, 'i': 37, 'j': 0, 'k': 2, 'l': 22, 'm': 12, 'n': 33, 'o': 41, 'p': 15, 'q': 1, 'r': 43, 's': 47, 't': 59, 'u': 17, 'v': 7, 'w': 4, 'x': 2, 'y': 13, 'z': 1}

**置换加密**



**Hill加密**

{'a': 18, 'b': 28, 'c': 29, 'd': 26, 'e': 14, 'f': 28, 'g': 21, 'h': 20, 'i': 24, 'j': 17, 'k': 24, 'l': 20, 'm': 16, 'n': 24, 'o': 25, 'p': 15, 'q': 24, 'r': 21, 's': 13, 't': 32, 'u': 26, 'v': 25, 'w': 23, 'x': 16, 'y': 24, 'z': 17}

**仿射加密**

{'a': 1, 'b': 0, 'c': 30, 'd': 7, 'e': 41, 'f': 16, 'g': 43, 'h': 59, 'i': 12, 'j': 10, 'k': 13, 'l': 43, 'm': 2, 'n': 23, 'o': 4, 'p': 15, 'q': 37, 'r': 9, 's': 17, 't': 33, 'u': 6, 'v': 1, 'w': 47, 'x': 22, 'y': 77, 'z': 2}

**四、课题完成过程中遇到的问题及解决方法**

问题：写置换密码程序时遇到没有提示的运行崩溃。

解决方法：仔细检查代码，发现一处a%0错误，改正后即可正常运行。

**五、总结**

经过本次实验，我明白了古代密码中置换密码，单表代换密码和多表代换密码具体的应用。在搞清楚加密算法每一步的原理后，程序的编写是很简单的。

根据密文字符统计可以发现**置换加密**中的明密文字符出现频率完全相等，名字也相同，很不安全；**仿射加密**中除了字符名改变外，字符的频率和组合方式完全相同，也不太安全；

**Hill加密**中明文统计特性可以很好的隐藏，相对安全。