南京信息工程大学 实验（实习）报告

|  |  |  |
| --- | --- | --- |
| 202283890036 | 黄家睿（Jerry Huang） | Internet of Things |

Playfair加密算法的实现与分析

1．实验目的：

1. 理解Playfair加解密算法；
2. 实现Playfair加解密算法
3. 对其性能进行测试（时间、雪崩效应等）。

2．实验内容：

1. 实现Playfair加解密算法；
2. 对其雪崩效应进行分析。

3．实验步骤

1. Playfair算法加密功能的实现
   1. 生成密码本：

def in\_matrix(matrix,item):

for row in matrix:

if item in row:

return True

return False

def generate\_cipher\_book(key):

key =''.join(filter(str.isalpha, key.lower()))

cipher\_book = np.full((5,5),None,dtype=object)

list\_key = list(key)

*# 密码本基于密钥生成，先将密钥按此输入到密码本中，如果存在相同字母就跳过*

i = 0

for value in list\_key:

if value == "j" or value == "i" :

value = "i"

if not in\_matrix(cipher\_book, value):

cipher\_book[int(i/5),int(i%5)] = value

i +=1

i = 0

*#用剩余的字母把密码本填满*

lower = string.ascii\_lowercase

for value in lower:

if value == "j" or value == "i" :

value = "i"

if not in\_matrix(cipher\_book, value):

for i in range(25):

if cipher\_book[int(i/5),i%5] == None:

cipher\_book[int(i/5),i%5] = value

else:

continue

break

return cipher\_book

* 1. 基于生成的密码本，对明文进行加密：

def find\_locations(cipher\_book,value):

value\_in\_function = value

if value\_in\_function == "j" or value\_in\_function == "i":

value\_in\_function = "i"

for i in range(25):

if value\_in\_function == cipher\_book[int(i/5)][int(i%5)]:

return int(i/5), int(i%5)

return 26

def playfair(plainText, key):

clean\_text = ''.join(filter(str.isalpha, plainText.lower()))

*#生成密码本*

cipher\_book = generate\_cipher\_book(key)

text\_list = list(clean\_text)

*#对明文进行编码，使其变为两个字母一组，例如JerryHuang变为：je rr yh ua ng，如果明文的字母个数为奇数，就在最后一个字母后面加一个紧随的字母，例如最后一个剩余的是a，就跟一个b*

if len(text\_list)%2 != 0:

last\_word = text\_list[-1]

if last\_word == "z":

next\_word = "a"

else:

next\_word = chr(ord(last\_word)+1)

text\_list.append(next\_word)

print(text\_list)

print(len(text\_list))

*#生成存储字母组的矩阵*

word\_list = [[None for \_ in range(2)] for \_ in range(int(len(text\_list) / 2))]

for i in range(int(len(text\_list)/2)):

for j in range(2):

word\_list[i][j] = text\_list[2\*i+j]

*#生成存储密文用的string类型变量*

cipherText = ""

*#找到每个字母在密码本中的位置，并按照规则改变位置*

for i in range(len(word\_list)):

index = 0

value\_row0, value\_col0 = find\_locations(cipher\_book,word\_list[i][index])

index = 1

value\_row1, value\_col1 = find\_locations(cipher\_book,word\_list[i][index])

if value\_row0 == value\_row1:

if value\_col0 ==4 and value\_col1 != 4:

value\_col0 = 0

value\_col1 += 1

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

elif value\_col1 ==4 and value\_col0 !=4:

value\_col1 = 0

value\_col0 += 1

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

elif value\_col0 ==4 and value\_col1 ==4:

value\_col0 = 0

value\_col1 = 0

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

else:

value\_col0 += 1

value\_col1 += 1

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

elif value\_col0 == value\_col1:

if value\_row0 ==4 and value\_row1 !=4:

value\_row0 = 0

value\_row1 += 1

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

elif value\_row1 ==4 and value\_row0 !=4:

value\_row1 = 0

value\_row0 += 1

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

else:

value\_row0 += 1

value\_row1 += 1

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

else:

temp = value\_col0

value\_col0 = value\_col1

value\_col1 = temp

cipherText += cipher\_book[value\_row0][value\_col0]

cipherText += cipher\_book[value\_row1][value\_col1]

return cipherText

1. Playfair算法解密功能的实现
   1. 通过密钥实现对密文的解密

def playfair\_discode(cipherText, key):

plain\_text = ""

code\_book = generate\_cipher\_book(key)

text\_list = list(cipherText)

for i in range(int(len(text\_list)/2)):

value\_row0, value\_col0 = find\_locations(code\_book,text\_list[i\*2])

value\_row1, value\_col1 = find\_locations(code\_book,text\_list[i\*2+1])

if value\_row0 == value\_row1:

if value\_col0 ==0 and value\_col1 !=0:

value\_col0 = 4

value\_col1 -= 1

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

elif value\_col0 != 0 and value\_col1 ==0:

value\_col1 = 4

value\_col0 -= 1

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

elif value\_col0 == 0 and value\_col1 ==0:

value\_col0 = 4

value\_col1 = 4

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

else:

value\_col0 -= 1

value\_col1 -= 1

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

elif value\_col0 == value\_col1:

if value\_row0 ==0 and value\_row1 !=0:

value\_row0 = 4

value\_row1 -= 1

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

elif value\_row1 ==0 and value\_row0 !=0:

value\_row1 = 4

value\_row0 -= 1

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

else:

value\_row0 -= 1

value\_row1 -= 1

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

else:

temp = value\_col0

value\_col0 = value\_col1

value\_col1 = temp

plain\_text += code\_book[value\_row0][value\_col0]

plain\_text += code\_book[value\_row1][value\_col1]

print(plain\_text)

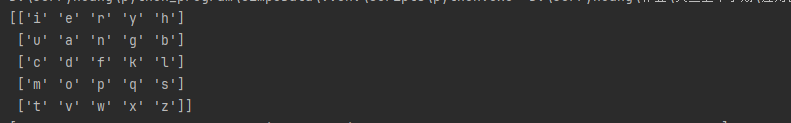
1. 加密解密过程和结果的输出：

我们设置密钥为“Jerryhaung”，明文为：“my name is jerryhuang”



使用上述的方向进行加密可得到：

1. 基于密钥生成的密码本：



1. 基于密码本生成的密文：

qignoihmeryyhiangb

1. 使用解密功能对密文进行解密：



我们可以得到原文：

mynameisierryhuang

其中因为密码本设置i和j这两个字母合并为一个字母，导致j和i出现翻译错误，这在预料之中，因此我们得到了正确的明文。

4．实验分析和总结

1. Playfair生成密码本的方式

Playfair算法基于一个5\*5的字母矩阵，该矩阵使用一个关键词构造，方法是按从左到右、从上到下顺序，填入关键词的字母(去除重复字母)后，将字母表其作余字母填入。

它依据一个5x 5的正方形组成的密码表来编写，密码表里排列有25个字母。如果一种语言字母超过 25 个，可以去掉使用频率最少的一个。在本次实验中，我们把i或者j去点一个。

在上述的实验中，我们的密钥为“jerryhuang”，我们已经密码本生成法则可以得到这样的密码本：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J/I | E | R | Y | H |
| U | A | N | G | B |
| C | D | F | K | L |
| M | O | P | Q | S |
| T | V | W | X | Z |

具体构造方法如下：

1. 构造一个基于5×5的字母矩阵
2. 该矩阵使用一个关键词（密钥）来构建
3. 然后从左到右、从上到下；依次填入密钥的字母（注意：密钥中重复的字母不填），然后再以字母表顺序依次填入其他字母
4. 字母 I 和 J 算作一个字母
5. 基于密码本生成密文的方法
6. 首先对明文进行分组，两个字母为一组，例如My name is jerryhaung 被分为：my na me is je rr yh au ng
7. 如果明文的字母数量为奇数个，最后一组会剩下一个字母，就在最后一个字母后面添加这个字母的下一个字母，例如剩下的一个是u，就在后面添加v。
8. 如果一组中的两个字母在密码本的同一行，那么密文就是这两字母在密码本中的下一列字母，如果字母是最后一列，那么最后一列的密文就是第一列。
9. 如果一组中的两个字母在密码本的同一列，那么密文就是这两字母在密码本中的下一行字母，如果字母是最后一行，那么最后一行的密文就是第一行。
10. 如果一组中的两个字母在密码本的不同行也不同列，则取其同行且与同组另一字母同列的字母为密文。
11. 雪崩效应的分析（基于实例进行图文分析）

雪崩效应（Avalanche effect）指加密算法（尤其是块密码和加密散列函数）的一种理想属性。雪崩效应就是一种不稳定的平衡状态也是加密算法的一种特征，它指明文或密钥的少量变化会引起密文的很大变化。雪崩效应是指当输入发生最微小的改变（例如，反转一个二进制位）时，也会导致输出的剧变（如，输出中一半的二进制位发生反转）。在高品质的块密码中，无论密钥或明文的任何细微变化都应当引起密文的剧烈改变。

在上述的实验中，我们使用了“jerryhuang”作为密钥来对密文进行加密，得到的密文为：

Qignoihmeryyhiangb

我们变换密钥，将“jerryhuang”变为“notjerryhuang”，明文不变，我们来看得到的结果：

Lherqtnxenyyhuarrk

可以发现密文发生了非常大的变化，但是雪崩效应还需要明文的变化也能给密文带来巨大的变化，我们把密钥设置为“jerryhuang”把明文变为“my name is not jerryhuang”：

qignoihmeryyhiangb

Qignoihmapiuryyhibngby

密文并没有出现非常大的变化，因此认为playfair加密法不具有雪崩效应。