Nanjing University of Information Science & Technology

Experiment (Internship) Report

|  |  |  |  |
| --- | --- | --- | --- |
| 202283890036 | 黄家睿(Jerry Huang) | Internet of Things | Zhiguo Qu(瞿治国） |

Implementation and Analysis of the Playfair Encryption Algorithm

1．Experimental Purpose：

1. Understanding the Playfair Encryption and Decryption Algorithm；
2. Understanding the Playfair Encryption and Decryption Algorithm
3. Test its performance (time, avalanche effect, etc.)

2．Experimental content：

1. Implement the Playfair Encryption and Decryption Algorithm；
2. Analyze its avalanche effect.

3．Experimental steps

1. Implementation of Playfair Algorithm Encryption Function
   1. Generate a password book：

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. Encrypt the plaintext using the generated cipher.：

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. Implementation of the Playfair cipher decryption feature
   1. Decrypting ciphertext using a key.

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]

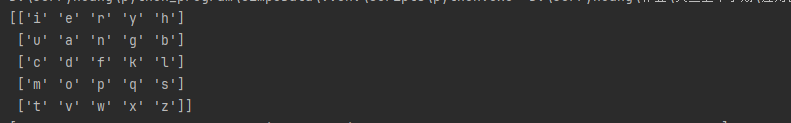
print(plain\_text)

1. The output of the encryption and decryption process and results.：

We set the key as "Jerryhaung", and the plaintext as: "my name is jerryhuang".

Using the above directions for encryption can yield.：

1. Password book generated based on the key：



1. Ciphertext generated based on a cipher book：

qignoihmeryyhiangb

1. Use the decryption function to decrypt the ciphertext.：



Use the decryption function to decrypt the ciphertext.：

mynameisierryhuang

Because the cipher text merges the letters 'i' and 'j' into one letter, resulting in translation errors for 'j' and 'i', as expected, we obtain the correct plaintext.

4．Experimental Analysis and Summary

1. The way Playfair generates a cipher text

The Playfair algorithm is based on a 5x5 letter matrix, which is constructed using a keyword. The method involves filling in the letters of the keyword (removing duplicate letters) in order from left to right, top to bottom, and then filling in the remaining letters of the alphabet.

It uses a code table made up of a 5x5 square in which 25 letters are arranged. If a language has more than 25 letters, the least frequently used one can be omitted. In this experiment, we removed either 'i' or 'j'.

In the experiment mentioned above, our key was "jerryhuang", and we were able to generate a codebook using the method described：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

The specific construction method is as follows.：

* 1. Construct a 5x5 letter matrix.
  2. The matrix is constructed using a keyword (key).
  3. Then fill in the letters of the key from left to right, from top to bottom; then fill in the remaining letters in alphabetical order. (Note: Do not fill in repeated letters from the key.)
  4. The letters I and J count as one letter.

1. Method for generating ciphertext based on a cipher text.
   1. First, group the plaintext into pairs of two letters, for example, "My name is jerryhaung" is divided into：my na me is je rr yh au ng
   2. If the number of letters in the plaintext is odd, the last group will have one letter left . Add the next letter of this letter after the last one. For example, if the last letter is 'u', add 'v' after it.
   3. If two letters in a set are in the same row of the cipher book, then the ciphertext is the next column letter of these two letters in the cipher book. If the letter is in the last column, then the ciphertext for the last column is the first column.
   4. If two letters in a set are in the same column of the cipher book, then the ciphertext is the letter below these two letters in the cipher book. If the letter is in the last row, then the ciphertext for the last row is the first row.
   5. If two letters in a set are in different rows and columns in the cipher book, then take the letter in the same row and column as the other letter in the same set as the ciphertext.。
2. Analysis of the Avalanche Effect (Illustrated with examples)

The avalanche effect refers to an ideal property of cryptographic algorithms, especially block ciphers and cryptographic hash functions. It is a characteristic of an unstable equilibrium state in cryptographic algorithms, indicating that a small change in the plaintext or key will cause a significant change in the ciphertext. The avalanche effect means that even the slightest change in the input (for example, inverting a single binary bit) will result in a dramatic change in the output (such as, half of the binary bits in the output being inverted). In high-quality block ciphers, any minor change in the key or plaintext should cause a drastic change in the ciphertext.

In the aforementioned experiment, we used "jerryhuang" as the key to encrypt the ciphertext, resulting in the following ciphertext.：

Qignoihmeryyhiangb

We change the key, turning "jerryhuang" into "notjerryhuang", the plaintext remains the same, let's look at the result.：

Lherqtnxenyyhuarrk

It can be observed that the ciphertext has undergone significant changes, but the avalanche effect requires that changes in the plaintext should also cause substantial changes in the ciphertext. We set the key to "jerryhuang" and change the plaintext to "my name is not jerryhuang".：

qignoihmeryyhiangb

Qignoihmapiuryyhibngby

The ciphertext has not undergone very large changes, thus it is believed that the Playfair encryption method does not exhibit the avalanche effect.