南京信息工程大学 实验（实习）报告

实验（实习）名称 日期 得分 指导教师

系 专业 年级 班次 姓名 学号

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．Experiment Content

[Code]

def locate(char: chr, cipher\_table: list) -> tuple:

"""查找字符在矩阵中的位置"""

for i in range(1, 6):

for j in range(1, 6):

if cipher\_table[i][j] == char:

return i, j

return -1, -1

def encrypt(plaintext\_ls: list, cipher\_table: list) -> list:

"""加密函数"""

ciphertext\_ls = []

cur = 0

while cur < len(plaintext\_ls) - 1:

x1, y1 = locate(plaintext\_ls[cur], cipher\_table)

x2, y2 = locate(plaintext\_ls[cur + 1], cipher\_table)

if x1 == x2: *# 同一行，右移*

ciphertext\_ls.append(cipher\_table[x1][(y1 % 5) + 1])

ciphertext\_ls.append(cipher\_table[x2][(y2 % 5) + 1])

elif y1 == y2: *# 同一列，下移*

ciphertext\_ls.append(cipher\_table[(x1 % 5) + 1][y1])

ciphertext\_ls.append(cipher\_table[(x2 % 5) + 1][y2])

else: *# 不同行不同列，交换列坐标*

ciphertext\_ls.append(cipher\_table[x1][y2])

ciphertext\_ls.append(cipher\_table[x2][y1])

cur += 2

return ciphertext\_ls

def decrypt(ciphertext\_ls: list, cipher\_table: list) -> list:

"""解密函数"""

plaintext\_ls = []

cur = 0

while cur < len(ciphertext\_ls) - 1:

x1, y1 = locate(ciphertext\_ls[cur], cipher\_table)

x2, y2 = locate(ciphertext\_ls[cur + 1], cipher\_table)

if x1 == x2: *# 同一行，左移*

plaintext\_ls.append(cipher\_table[x1][y1 - 1 if y1 > 1 else 5])

plaintext\_ls.append(cipher\_table[x2][y2 - 1 if y2 > 1 else 5])

elif y1 == y2: *# 同一列，上移*

plaintext\_ls.append(cipher\_table[x1 - 1 if x1 > 1 else 5][y1])

plaintext\_ls.append(cipher\_table[x2 - 1 if x2 > 1 else 5][y2])

else: *# 不同行不同列，交换列坐标*

plaintext\_ls.append(cipher\_table[x1][y2])

plaintext\_ls.append(cipher\_table[x2][y1])

cur += 2

return plaintext\_ls

def generate\_cipher\_table(key: str) -> list:

"""生成密码表"""

letters = "ABCDEFGHIKLMNOPQRSTUVWXYZ" *# I 和 J 合并*

used\_letters = set(key) *# 密钥中的字符*

cipher\_table = [[0] \* 6 for \_ in range(6)] *# 初始化矩阵*

*# 填入密钥*

idx = 0

for i in range(1, 6):

for j in range(1, 6):

if idx < len(key):

cipher\_table[i][j] = key[idx]

idx += 1

else:

break

if idx >= len(key):

break

*# 填入剩余字母*

letters = [ch for ch in letters if ch not in used\_letters]

for i in range(1, 6):

for j in range(1, 6):

if cipher\_table[i][j] == 0:

cipher\_table[i][j] = letters.pop(0)

return cipher\_table

def preprocess\_plaintext(plaintext: str) -> list:

"""处理明文字符串"""

plaintext = plaintext.upper().replace(" ", "").replace("J", "I")

plaintext\_ls = list(plaintext)

cur = 0

while cur < len(plaintext\_ls) - 1:

if plaintext\_ls[cur] == plaintext\_ls[cur + 1]: *# 相同字符插入 X*

plaintext\_ls.insert(cur + 1, 'X')

cur += 2

if len(plaintext\_ls) % 2: *# 补充 X*

plaintext\_ls.append('X')

return plaintext\_ls

*# 主函数流程*

print('请输入明文：', end='')

plaintext = input().strip()

print('请输入密钥：', end='')

key = ''.join(dict.fromkeys(input().strip().upper())) *# 去重并保持顺序*

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

*# 展示密码本*

print('本次加密所使用的密码本如下：')

for i in range(1, 6):

print(' '.join(cipher\_table[i][1:]))

*# 处理明文*

plaintext\_ls = preprocess\_plaintext(plaintext)

print(f'处理后的明文为：{"".join(plaintext\_ls)}')

*# 加密*

ciphertext\_ls = encrypt(plaintext\_ls, cipher\_table)

ciphertext = ''.join(ciphertext\_ls)

print(f'加密后的密文为：{ciphertext}')

*# 解密*

print('请输入想要解密的密文：', end='')

ciphertext = input().strip()

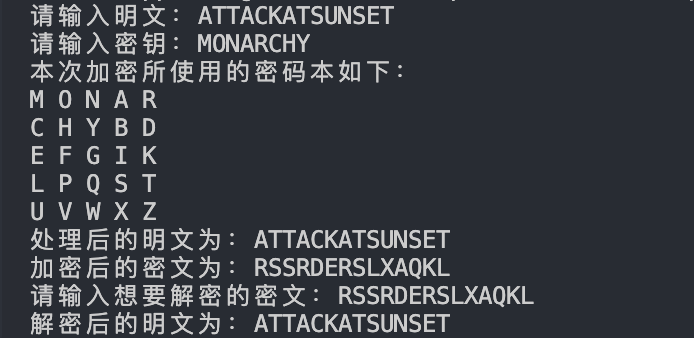
ciphertext\_ls = list(ciphertext)

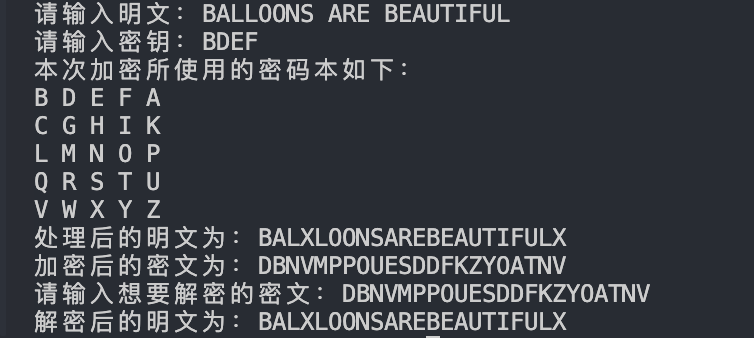
plaintext\_ls = decrypt(ciphertext\_ls, cipher\_table)

plaintext = ''.join(plaintext\_ls)

print(f'解密后的明文为：{plaintext}')

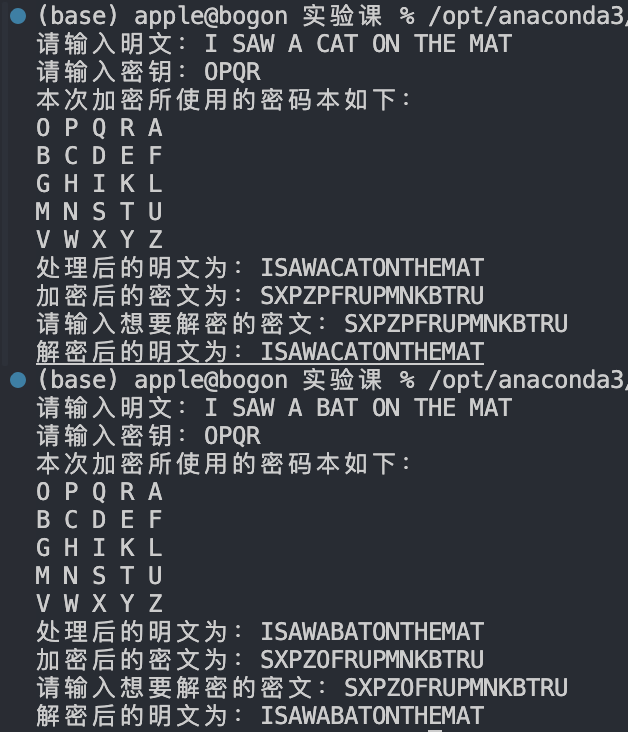
[Output]





Inputting “attack at sunset” does not require processing, as every two letters can naturally form a pair. However, when inputting “balloons are beautiful,” the two ‘l’ letters are placed in the same group, so an ‘X’ is inserted to separate them. Since the sentence has an odd length, an additional ‘X’ is appended at the end to complete the grouping. After decryption, both examples will yield the correct plaintext results.

4．Experiment Analyzation and Summarization



The plaintexts differ only in the first letter between “cat” and “bat.” As observed, the resulting ciphertexts are nearly identical, with only the third pair, PF and OF, differing.

Therefore, the Playfair cipher does not exhibit the avalanche effect.