**封面及考核表已删除**

1. **实验目的和要求**

实验目的：建立系统设计的整体思想，锻炼编写程序、调试程序的能力，学习文档编写规范，培养独立学习、吸取他人经验、探索前言知识的习惯，树立团队协作精神。

实验要求：使用Python、C、Java等语言根据题目要求编写程序。

1. **实验原理**
2. **移位加密**

输入一串字母组成的字符s，再输入一个密钥k，将明文字符串中的每个字符向后移动k位进行加密，将密文字符串中每个字符向前移动k位进行解密。

举例：加密（密钥为3）：abc -> def

解密（密钥为3）：abc -> xyz

1. **替换加密**

输入一串由字母、数字组成的字符s，通过随机生成的一张替换表（由26个英文字母和数字组成）进行加密解密。

举例：若替换表为：abcdefghijklmnopqrstuvwxyz0123456789

h2ugim7vfaqw0ctpr9j3bdlyox81se456nzk

则明文abcd对应的密文为h2ug

1. **数据置乱**

输入由可见字符组成的字符串，根据字符个数生成随机置乱表，通过置换表进行加密解密。

举例：若明文为：abc123[]\啊喔呃

根据字符数生成的置乱表为：9, 4, 8, 1, 3, 2, 12, 10, 6, 5, 7, 11

则对应的密文为：\1]acb呃啊32[喔

其中第9位的\移动到了第1位，第4位的1移动到了第二位......以此类推。

1. **分组加密**

将明文根据密钥中字符数进行分组，然后通过依次移位进行加密。

若密钥为k位，则将明文每k位分为一组，然后每一组的明文依次向后移动1位、2位、3位......k位

举例：明文：abcdefghi 密钥：abc

将明文每三个字符（密钥abc由3个字符组成）分为一组，然后每一组依次移一位、两位、三位进行加密：a -> b（移1位），b -> d（移2位），c -> f（移3位），d -> e（移1位），e -> g（移2位），f -> i（移3位），g -> h（移1位），h -> j（移2位），i ->l（移3位）。

最终得到的密文为：edfbgihjl

1. **字符串转二进制和十六进制**

先将字符串中的字符分别转为十进制的ASCII码，然后再将十进制的ASCII码转化为二进制、十六进制。

举例：字母a：转为十进制ASCII码为97，97再转为二进制为01100001，97转为十六进制为61。

1. **流式加密**

先将明文字符转为对应的十进制ASCII码，然后将其转为对应的二进制明文。随机生成等长的二进制密钥，二进制明文和二进制密钥通过按位异或进行加密，最终得到密文。

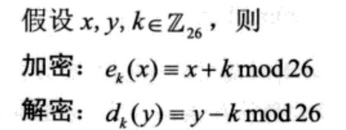
举例：明文：ab -> 二进制明文：01100001 01100010

随机生成等长二进制密钥：01100110 00100100

通过按位异或得到密文：00000111 01000110

1. **实验内容及步骤**
2. **移位加密**

输入字母组成的明文字符串，设定一个密钥，采用移位密码的方式进行加密。并可以通过再次输入改密码并输入相应的密码进行解密。移位密码的加解密过程：



示例：

输入：加密字符串 abcdef， 密钥2

输出：cdefgh

代码：

# 加密函数  
def f1(x, k):  
 word = list(map(ord,x)) #使用list()和map()和ord()把x转换为ASCII码形式的列表  
 lens = len(word)  
 k = int(k)  
 for i in range(lens):  
 temp = word[i]  
 if 65 <= temp <= 90: #大写字母  
 temp += k  
 while temp > 90:  
 temp -= 26  
 elif 97 <= temp <= 122: #小写字母  
 temp += k  
 while temp > 122:  
 temp -= 26  
 word[i] = temp  
 word = list(map(chr, word)) #转为字母形式的列表  
 word = ''.join(word)  
 return word  
  
# 解密函数  
def f2(x, k):  
 word = list(map(ord, x))  
 lens = len(word)  
 k = int(k)  
 for i in range(lens):  
 temp = word[i]  
 if 65 <= temp <= 90:  
 temp -= k  
 while temp < 65:  
 temp += 26  
 elif 97 <= temp <= 122:  
 temp -= k  
 while temp < 97:  
 temp += 26  
 word[i] = temp  
 word = list(map(chr, word))  
 word = ''.join(word)  
 return word  
  
words = input("加密(1)还是解密(0)？")  
msg = input("输入信息：")  
if not msg.isalpha():  
 print("错误！请重新输入信息：", end='')  
 msg = input()  
  
k = input("输入密钥：")  
if not k.isdigit():  
 print("错误！请重新输入密钥：", end='')  
 k = input()  
  
if words == "1":  
 ans = f1(msg, k)  
else:  
 ans = f2(msg, k)  
print("加密/解密结果为:", ans)

1. **替换加密**

输入一个字母和数字组成明文字符串，并能随机产生一个替换表作为密钥保存到文件并显示出来，然后根据替换表进行数据变换加密。在解密时导入文件中的替换表，根据该替换表进行解密。

示例：

输入：加密字符串 abcdef， 随机生成的替换表为

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j |
| b | e | f | g | h | a | j | i | c | d |

输出：befgha

以上为简易实例，要生成完整的替换表。

代码：

import random  
import os  
  
Plaintext = list(input("输入明文："))  
for i in range(0, len(Plaintext)):  
 if not Plaintext[i].isalnum():  
 print("错误！请重新输入信息：")  
 Plaintext = input("输入明文：")  
 break  
  
l = len(Plaintext)  
  
key\_chart\_1 = [] # 新建空字母表  
for i in range(97, 123):  
 key\_chart\_1.append(chr(i))  
for i in range(48, 58):  
 key\_chart\_1.append(chr(i))  
  
if not os.path.exists("替换加密.txt"): # 判断替换表是否存在，若不存在则生成替换表  
 key\_chart\_2 = [] # 新建空替换表  
 for i in range(97, 123):  
 key\_chart\_2.append(chr(i))  
 for i in range(48, 58):  
 key\_chart\_2.append(chr(i))  
 random.shuffle(key\_chart\_2) # 生成随机替换表  
 chart\_1 = dict(zip(key\_chart\_1, key\_chart\_2)) # 字母表和替换表组成字典  
 # 加密  
 Ciphertext = [] # 新建空密文列表  
 for i in range(0, l): # 使用随机替换表进行加密  
 for x in chart\_1.keys():  
 if Plaintext[i] == x:  
 Ciphertext.append(chart\_1[x])  
 Ciphertext = ''.join(Ciphertext) # 密文列表转为字符串  
 print("加密结果为：", Ciphertext)  
 key\_chart\_2 = ''.join(key\_chart\_2)  
 f = open("替换加密.txt", mode='w') # 新建txt文件  
 f.write(key\_chart\_2) # 写入替换表  
 f.close() # 关闭文件  
else:  
 file = open("替换加密.txt", mode='r') # 打开文件  
 key = list(file.readline()) # 读取替换表  
 file.close() # 关闭文件  
 Ciphertext = [] # 新建空密文列表  
 chart\_1 = dict(zip(key\_chart\_1, key)) # 字母表和替换表组成字典  
 for i in range(0, l): # 使用随机替换表进行加密  
 for x in chart\_1.keys():  
 if Plaintext[i] == x:  
 Ciphertext.append(chart\_1[x])  
 Ciphertext = ''.join(Ciphertext) # 密文列表转为字符串  
 print("加密结果为：", Ciphertext)  
  
# 解密  
file = open("替换加密.txt", mode='r') # 打开文件  
key = list(file.readline()) # 读取替换表  
file.close() # 关闭文件  
  
Ciphertext = input("输入要解密的密文：")  
  
chart\_2 = dict(zip(key, key\_chart\_1)) # 读取到的替换表和字母表组成字典chart\_2  
ans = [] # 新建解密列表  
print("解密结果为： ", end='')  
for i in range(0, l): # 解密  
 for x in chart\_2.keys():  
 if Ciphertext[i] == x:  
 ans.append(chart\_2[x])  
ans = ''.join(ans) # 解密结果转为字符串形式  
print(ans)

1. **数据置乱**

输入可包含所有可见字符的字符串，并随机生成一个并能随机产生一个置乱表作为密钥保存到文件并显示出来，然后根据置乱表进行数据变换加密。在解密时导入文件中的替换表，根据该置乱表进行解密。

示例：

输入字符串：abcdef

随机产生的置乱表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 4 | 5 | 1 | 3 | 6 |

输出：daebcf

代码：

import random  
import os  
  
Plaintext = list(input("输入明文："))  
lens = len(Plaintext)  
# 加密函数  
def encode():  
 key = [] # 新建空置换列表  
 for i in range(1, lens + 1):  
 key.append(i)  
 random.shuffle(key) # 生成随机置换表  
 Ciphertext = [] # 新建空密文列表  
 for i in key: # 根据置换表生成密文  
 a = Plaintext[i - 1]  
 Ciphertext.append(a)  
 Ciphertext = ''.join(Ciphertext)  
 print("密文为：", Ciphertext)  
 for i in range(0, lens): # 数字列表变成字符列表  
 key[i] = str(key[i])  
 key = str(list(key))  
 # 将置换列表写入文件  
 f = open("数据置乱.txt", mode='w')  
 f.writelines(key)  
 f.close()  
  
if not os.path.exists("数据置乱.txt"): # 判断替换表是否存在，若不存在则生成置乱表  
 encode()  
else: # 若置乱表存在  
 file = open("数据置乱.txt", mode='r') # 读取文件  
 key = list(file.readline()) # 密钥（字符串列表）  
 file.close()  
 if len(key) != lens: # 判断密钥长度  
 encode()  
 else:  
 Ciphertext = [] # 新建空密文列表  
 for i in range(0, lens): # 变为数字列表  
 key[i] = int(key[i])  
 for i in key: # 根据置换表生成密文  
 a = Plaintext[i-1]  
 Ciphertext.append(a)  
 Ciphertext = ''.join(Ciphertext)  
 print("密文为：", Ciphertext)  
# 解密  
Ciphertext = input("输入要解密的密文：")  
  
file = open("数据置乱.txt", mode='r') # 从文件中读取置换表  
key = file.readline() # 密钥（字符串列表）  
file.close()  
  
for i in range(0, len(key)):  
 key = key.replace("'", '')  
 key = key.replace('[', '')  
 key = key.replace(']', ',')  
 key = key.replace(' ', '')  
  
keys = []  
temp = []  
cnt = 0  
for i in range(0, len(key)):  
 if key[i] != ',':  
 cnt += 1  
 temp.append(key[i])  
 else:  
 cnt = 0  
 if cnt == 0:  
 keys.append(''.join(temp))  
 temp = []  
  
for i in range(0, lens): # 变为数字列表  
 keys[i] = int(keys[i])  
ans = dict(zip(keys, Ciphertext)) # 创建字典，密钥和密文一一对应  
ans = sorted(ans.items(), key=lambda item:item[0]) # 将字典按照键的大小排序  
print("解密结果为： ", end='')  
for x in range(0, lens):  
 print(ans[x][1], end='')

1. **分组加密**

输入一个字母组成明文字符串，并输入一个字母组成的密钥，对明文进行分组后与密钥进行移位加密，输入密文以及密钥可以进行解密

实例：

输入字符串：abcdef

输入密钥：abc

输出：bdfegi

注意：输入的明文在分组时不足的部分要采用特殊字符进行标记进行处理。

代码：

# 分组函数  
def Grouping(msg, k):  
 group\_t = []  
 group = []  
 cnt = 0  
 for i in range(0, len(msg)):  
 group\_t.append(msg[i])  
 if (i + 1) % len(k) == 0:  
 group\_t = ''.join(group\_t)  
 group.append(group\_t)  
 group\_t = []  
 return group  
# 加密函数  
def encode(group):  
 ans = []  
 l\_group = len(group)  
 for i in range(0, l\_group):  
 temp = group[i]  
 for j in range(0, len(temp)):  
 t = temp[j]  
 t = ord(t) + j + 1  
 if (t > 122) or ((t < 97) and (t > 90)):  
 t -= 26  
 t = chr(t)  
 ans.append(t)  
 ans = ''.join(ans)  
 print("加密结果为：", ans)  
# 解密函数  
def decode(group):  
 ans = []  
 l\_group = len(group)  
 for i in range(0, l\_group):  
 temp = group[i]  
 for j in range(0, len(temp)):  
 t = temp[j]  
 t = ord(t) - j - 1  
 if (t < 65) or ((t < 97) and (t > 90)):  
 t += 26  
 t = chr(t)  
 ans.append(t)  
 ans = ''.join(ans)  
 print("解密结果为：", ans)  
  
# 调用函数  
Pt = [] # 新建明文列表  
Ct = [] # 新建密文列表  
  
text = input("加密(1)还是解密(0)？")  
text = int(text)  
if text != 1 and text != 0:  
 print("请重新选择：加密(1)还是解密(0)？", end='')  
 text = input()  
 text = int(text)  
if text == 1:  
 Pt = list(input("输入明文："))  
 key = input("输入密钥：")  
 if len(Pt) % len(key) != 0: # 判断密钥是否合理  
 print("错误！请重新输入密钥：", end='') # 输入明文在分组不足时进行处理  
 key = input()  
 encode(Grouping(Pt, key))  
else:  
 Ct = list(input("输入密文："))  
 key = input("输入密钥：")  
 decode(Grouping(Ct, key))

1. **字符串转二进制和十六进制**

输入任意长的可见字符，将该字符串的数据转为字符型的二进制01和十六进制0-f的字符串在终端显示。

示例：

输入：abcdef

在屏幕输出二进制：01100001 01100010 01100011 01100100 01100101 01100110

十六进制：61 62 63 64 65 66

代码：

msg = list(input("输入字符串："))  
l = len(msg)  
for i in range(0, l): # 字符串转为十进制数字  
 msg[i] = ord(msg[i])  
  
def Change2(lx): # 转为二进制并输出  
 print("二进制：", end='')  
 for i in range(0, l):  
 t = lx[i]  
 t = bin(t)  
 t = t.replace("0b", "0")  
 print(t, end=' ')  
  
def Change\_16(lx): # 转为十六进制并输出  
 print("十六进制：", end='')  
 for i in range(0, l):  
 t = lx[i]  
 t = hex(t)  
 t = t.replace("0x", "")  
 print(t, end=' ')  
  
Change2(msg) # 调用函数  
print('\n')  
Change\_16(msg)

1. **流式加密**

输入一个任意长度可见字符串组成的字符串，并随机生成一个等长度的随机密钥并保存在文件中，采用按位异或的方式进行二进制加密，并将该加密结果保存在一个文件中，解密时课导入该密文和密钥进行解密，将解密结果进行输出。

示例：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 输入 | a | b | c | d | e | f |
| 明文的二进制 | 01100001 | 01100010 | 01100011 | 01100100 | 01100101 | 01100110 |
| 二进制密钥 | 01111001 | 01110101 | 01100100 | 01100101 | 01101111 | 01110111 |
| 二进制明文 | 00011000 | 00010111 | 00000111 | 00000001 | 00001010 | 00010001 |

代码：

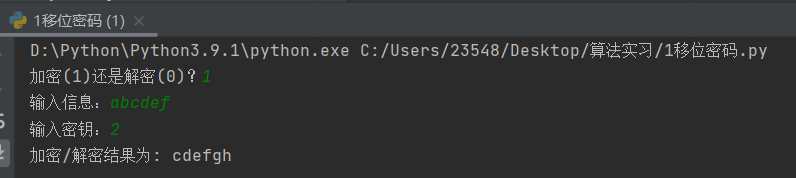
import random  
import os  
  
msg = list(input("输入字符串："))  
l = len(msg)  
for i in range(0, l): # 字符串转为十进制数字  
 msg[i] = ord(msg[i])  
 msg[i] = bin(msg[i]) # 转为二进制  
 if len(msg[i]) > 8:  
 msg[i] = msg[i].replace("0b", "0")  
 else:  
 msg[i] = msg[i].replace("0b", "00")  
  
# 创建密钥  
def create\_keys():  
 random\_key = [] # 新建空二进制密钥列表  
 for i in range(0, len(msg[0]) \* len(msg)): # 生成二进制密钥  
 random\_number = ''.join(random.sample(["0", "1"], 1))  
 random\_key.append(random\_number)  
 random\_key = ''.join(random\_key)  
 # 保存密钥到文件里  
 file = open("流式加密.txt", mode='w')  
 file.write(random\_key)  
 file.close()  
  
# 按位异或加密  
def encode():  
 f = open("流式加密.txt", mode='r')  
 random\_key = f.readline() # 读取文件中保存的密钥  
 f.close()  
 t2 = []  
 for i in range(0, len(msg)):  
 t1 = msg[i]  
 for j in range(0, len(msg[i])):  
 t2.append(str(int(t1[j]) ^ int(random\_key[j + i \* len(msg[i])])))  
 t2 = ''.join(t2)  
 msg[i] = t2  
 t2 = []  
 print("加密结果为：", end='')  
 for i in range(0, l):  
 print(msg[i], end=' ')  
 print('\n')  
  
if not os.path.exists("流式加密.txt"):  
 create\_keys()  
 encode()  
else:  
 f = open("流式加密.txt", mode='r') # 读取文件中保存的密钥  
 key = f.readline()  
 f.close()  
 if len(key) / len(msg) < len(msg):  
 create\_keys()  
 encode()  
  
# 解密  
Ciphertext = input("输入要解密的密文：")  
Ciphertext = list(Ciphertext.replace(" ", ''))  
  
f = open("流式加密.txt", mode='r') # 读取文件中保存的密钥  
key = f.readline()  
f.close()  
  
de\_code = []  
to\_str = ''  
t2 = []  
for i in range(0, len(Ciphertext)):  
 to\_str += Ciphertext[i]  
 if (i + 1) % 8 == 0:  
 de\_code.append(to\_str)  
 to\_str = ''

# 按位异或解密  
for i in range(0, len(de\_code)):  
 t1 = de\_code[i]  
 for j in range(0, len(de\_code[i])):  
 t2.append(str(int(t1[j]) ^ int(key[j + i \* len(de\_code[i])])))  
 t2 = ''.join(t2)  
 de\_code[i] = t2  
 t2 = []

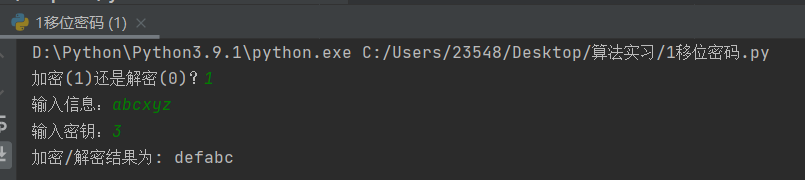
print("解密结果为：", end='')  
ans = []  
for i in range(0, len(de\_code)):  
 de\_code[i] = chr(int(de\_code[i], 2)) # 字符串转十进制数字后转为字符  
 ans = ''.join(de\_code)  
print(ans)

1. **实验结果分析**
2. **移位加密**

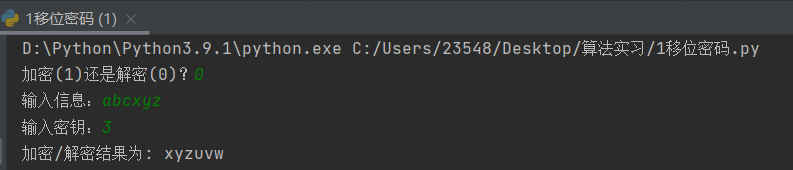
题目示例：

****

加密：

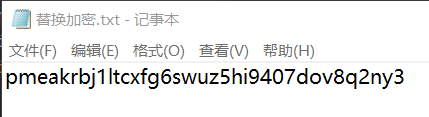
****

解密：

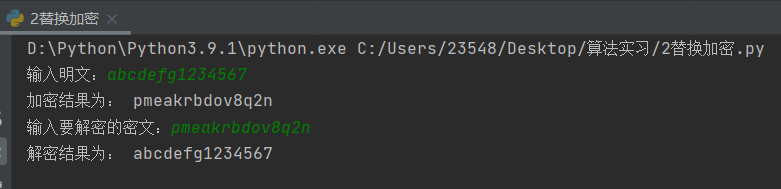
****

1. **替换加密**

随机生成替换表：

****

替换加密解密：

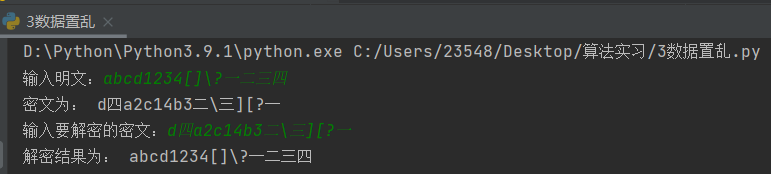
****

1. **数据置乱**

随机生成置乱表：

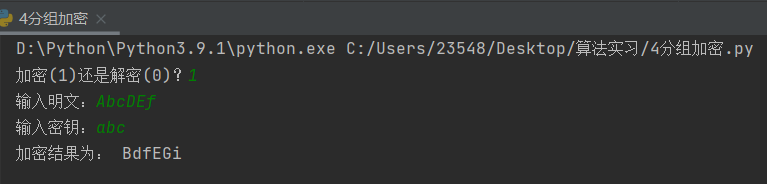


加密解密：

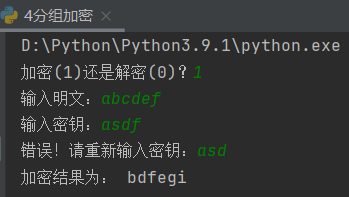


1. **分组加密**

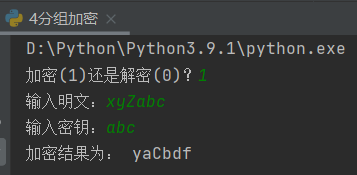
大小写混合：



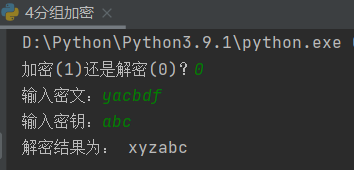
输入与要求不符时的报错：



边界情况（xyz)：

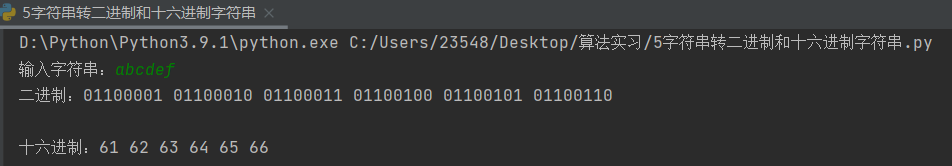


解密：



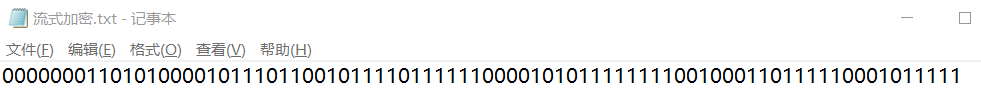
1. **字符串转二进制和十六进制**

题目示例：

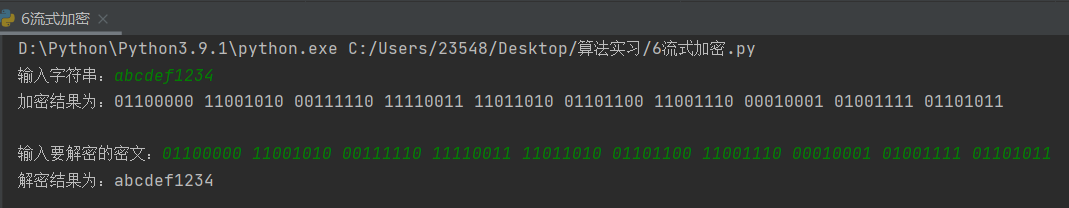


1. **流式加密**

生成随机二进制密钥：



加密解密：

****

1. **实验总结及心得体会**

通过本次实验学习，我使用Python实现了六个信息安全的基础算法。这次试验锻炼了我编写程序、调试程序的能力，提高了我对Python的熟练度，使我重新熟悉了在学习过程中没有完全掌握的知识，学习了一些新的Python函数，并在编写代码时使用它们。

在实验过程中，我认为最重要的是思路。当我在拿到这六道题的时候，我会先把题目读懂，然后根据题目的意思先建立一个整体的框架，对整体流程有一个大致的想法。之后，在真正开始编写程序前，根据大致的流程，将细节添加进去。最后，对代码进行测试，在发现bug后修改代码，经过不断完善后得到最终的代码。

在这次为期两周的算法实习中，指导老师也给了我很多帮助，帮我找到了一些代码中存在的bug。虽然在实验结束后代码里依旧有一些我无法修改完善的问题，但我的代码在整体上比一开始写出来的样子已经完善了很多，程序也合理了很多，变得更加贴近于实际应用。

这次实习不仅使我学到了知识，丰富了经验，也帮我缩小了实践与理论的差距。让我能够更好理解课本上所提到过的各种加密算法，也让我能够在未来专业课的学习中，更好的把书本内容和实际内容结合起来，更加深入的理解信息安全方面的知识。