南開大學
Nankai University

网络空间安全学院

密码学实验报告

实验一：古典密码算法及攻击方法

姓名：魏伯繁

学号：2011395

专业：信息安全

2022 年 11 月 17 日

目录

1 实验要求及实验目的

1.1 实验目的

通过 C ++编程实现移位密码和单表置换密码算法，加深对经典密码体制的了解。并通过对这两种密码实施攻击，了解对古典密码体制的攻击方法。

1.2 实验内容

（1）根据实验原理部分对移位密码算法的介绍，自己创建明文信息，并选择一个密钥，编写移位密码算法实现程序，实现加密和解密操作。

（2）两个同学为一组，互相攻击对方用移位密码加密获得的密文，恢复出其明文和密钥。

（3）自己创建明文信息，并选择一个密钥，构建置换表。编写置换密码的加解密实现程序，实现加密和解密操作。

（4）用频率统计方法，试译下面用单表置换加密的一段密文：

SIC GCBSPNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJHAY JBRCGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QCRRNEC GNB MBAY HC PCGMTCPCD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJXNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QCRRNEC HMH SIC PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRICR SM ENJB ZBNZSIMPJOCD GMBSPMA MF SIC QCRRNEC

写出获得的明文消息和置换表。

1.3 实验要求

要求上述密码算法提供最后的算法流程图，并写出明文、加解密的结果。字母频率统计攻击方法要求写明置换表中确定每个字母的原因和攻击的步骤。

运行 Windows 操作系统的 PC 机，具有 VC 等语言编译环境

2 密码算法简介

2.1 移位密码

1、移位密码

移位密码：将英文字母向前或向后移动一个固定位置。例如向后移动 3 个位置，即对字母表作置换（不分大小写）。

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

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

设明文为：public keys, 则经过以上置换就变成了：sxeolf nhbv。

如果将 26 个英文字母进行编码：A→0，B→1，…，Z→25，则以上加密过程可简单地写成：

明文：m = m1m2…mi…, 则有

密文：c=c1c2…ci…, 其中 ci=(mi+key mod26)，i = 1，2，…。

2.2 对移位密码的攻击

移位密码是一种最简单的密码，其有效密钥空间大小为 25。因此，很容易用穷举的方法攻破。穷举密钥攻击是指攻击者对可能的密钥的穷举，也就是用所有可能的密钥解密密文，直到得到有意义的明文，由此确定出正确的密钥和明文的攻击方法。对移位密码进行穷举密钥攻击，最多只要试译 25 次就可以得到正确的密钥和明文。

2.3 单表代换密码

单表置换密码就是根据字母表的置换对明文进行变换的方法，例如，给定置换

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

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

明文：public keys, 则有

密文：mckebw qxuo。

单表置换实现的一个关键问题是关于置换表的构造。置换表的构造可以有各种不同的途径，主要考虑的是记忆的方便。如使用一个短语或句子，删去其中的重复部分，作为置换表的前面的部分，然后把没有用到的字母按字母表的顺序依次放入置换表中。

为保证实现的统一性：输入密码表的方式采用一句话初始化，出现的字母按照对应顺序放在表中的前面，没出现的也按照顺序按顺序放在出现的字母之后，并要实现去重

2.4 对单表代换密码的攻击

在单表置换密码中，由于置换表字母组合方式有 26！种，约为 4.03×1026。所以采用穷举密钥的方法不是一种最有效的方法。对单表置换密码最有效的攻击方法是利用自然语言的使用频率：单字母、双字母组/三字母组、短语、词头/词尾等，这里仅考虑英文的情况。英文的一些显著特征如下:

短单词 (small words)：在英文中只有很少几个非常短的单词。因此，如果在一个加密的文本中可以确定单词的范围，那么就能得出明显的结果。一个字母的单词只有 a 和 I。如果不计单词的缩写，在从电子邮件中选取 500k 字节的样本中，只有两个字母的单词仅出现 35 次，而两个字母的所有组合为 26×26 = 676 种。而且，还是在那个样本中，只有三个字母的单词出现 196 次，而三个字母的所有组合为 26×26×26 = 17576 种。

常用单词 (common words): 再次分析 500k 字节的样本，总共有 5000 多个不同的单词出现。在这里，9 个最常用的单词出现的总次数占总单词数的 21%，20 个最常用的单词出现的总次数占总单词数的 30%，104 个最常用的单词占 50%，247 个最常用的单词占 60%。样本中最常用的 9 个单词占总词数的百分比为：

the 4.65 to 3.02 of 2.61 I 2.2 a 1.95

and 1.82 is 1.68 that 1.62 in 1.57

字母频率 (character frequency): 在 1M 字节旧的电子文本中，对字母” A” 到 “Z”（忽略大小写）分别进行统计。发现近似频率（以百分比表示）：

e 11.67 t 9.53 o 8.22 i 7.81 a 7.73 n 6.71 s 6.55

r 5.97 h 4.52 l 4.3 d 3.24 u 3.21 c 3.06 m 2.8

p 2.34 y 2.22 f 2.14 g 2.00 w 1.69 b 1.58 v 1.03

k 0.79 x 0.30 j 0.23 q 0.12 z 0.09

从该表中可以看出，最常用的单字母英文是 e 和 t，其他字母使用频率相对来说就小得多。这样，攻击一个单表置换密码，首先统计密文中最常出现的字母，并据此猜出两个最常用的字母，并根据英

文统计的其他特征（如字母组合等）进行试译。

3 算法实现

3.1 移位密码实现

算法核心：通过对输入密文进行 asc 码右移（加法）进行加密，并且要注意不要溢出，及时判断溢出并处理，对移位密码的流程图参考如下：

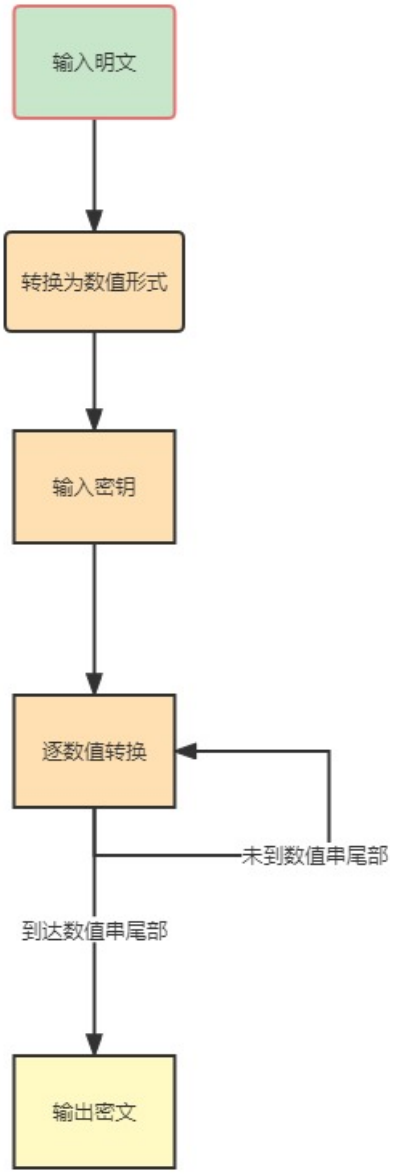

图 3.1: 移位密码流程图

```
1
2       int shiftword(){
```

```
    int n;//密钥
    cout << " 请输入密钥" << endl;
    cin >> n;
    n=n%26;//大于等于 26 没有意义

    char *message=new char[1000];
    cout << " 请输入明文" << endl;
    getchar();//防止 cin.getline 函数被跳过的情况发生
    cin.getline(message, 1000);
    int length=get_length(message);

    //存储密文
    char *secretmessage=new char[1000];
    int record;
    for(int i=0;message[i]!='\0'; i++) {
        record = i;
        if((message[i]>=65&&message[i]<=90)){
            secretmessage[i]=message[i]+n;
            if(secretmessage[i]>90){secretmessage[i]-=26;}
            continue;
        }
        if(message[i]>=97&&message[i]<=122){
            int j = message[i] + n;
            if (j > 122) { j -= 26; }
            secretmessage[i] = j;
            continue;
        }
        if(message[i]==32){
            secretmessage[i]=32;
            continue;
        }
        secretmessage[i] = message[i];
    }
    secretmessage[record+2] = '\0';
    cout<<" 您的密钥是:"<<n<<endl;
    cout<<" 您的明文是:"<<message<<endl;
    cout<<" 您的密文是:"<<secretmessage<<endl;
    return 0;
}

```

3.2 攻击移位密码

因为移位密码的密钥空间很短，只有 26，所以我们采取穷举攻击的方式对其进行攻击，攻击移位密码的流程图如下图所示

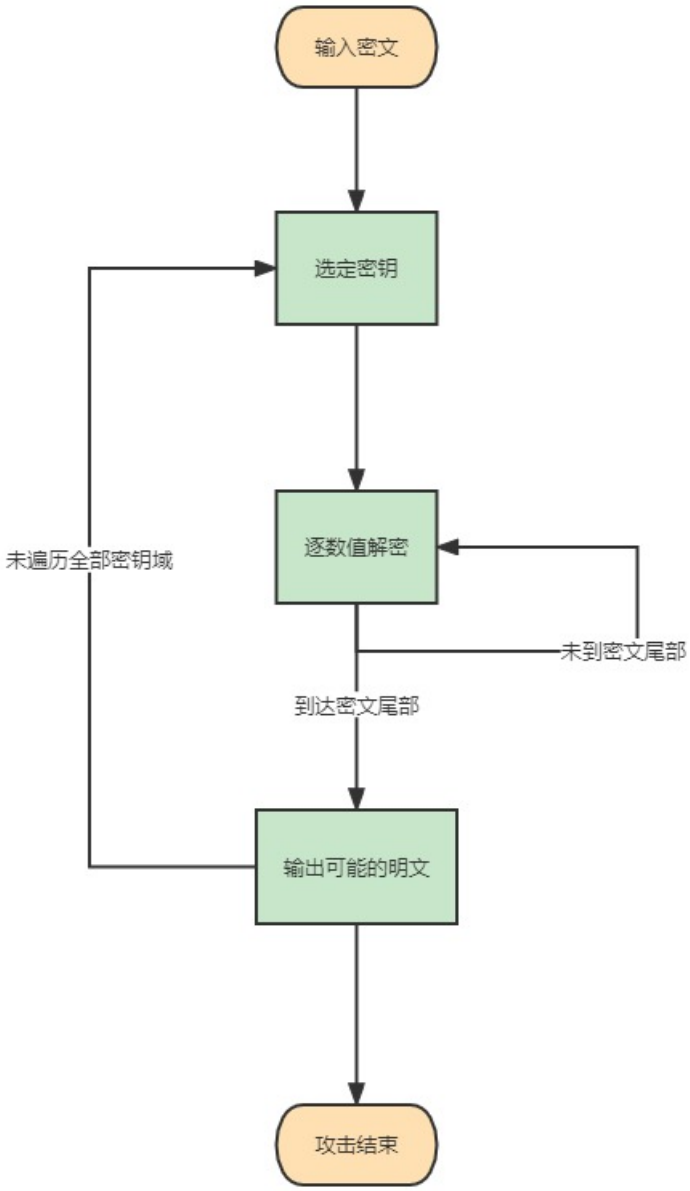

图 3.2: 攻击移位密码流程图

```
1  //攻击移位密码
2  int attackshift(){
3      char *secretmessage=new char [1000];
4      char *message=new char[1000];
5      cout<<" 请输入想要攻击的密文"<<endl;
6      cin.getline(secretmessage,1000);
```

```
7       int length=get_length(secretmessage);
8       for(int i=0;i<26;i++){
9           memcpy(message,secretmessage,length+1);
10          for(int j=0;j<length;j++){
11              if(message[j]>=65&&message[j]<=90){
12                  message[j]-=i;
13                  if(message[j]<65){message[j]+=26;}
14                  continue;
15              }
16              if(message[j]>=97&&message[j]<=122){
17                  message[j]-=i;
18                  if(message[j]<97){message[j]+=26;}
19                  continue;
20              }
21              if(message[j]==32){continue;}
22          }
23          cout<<" 当密钥是"<<i<<" 时解密的文件为: "<<message<<endl;
24      }
25      return 0;
26  }
```

3.3 单表代换密码

单表代换密码的处理主要在于能否处理好输入的密钥所构成的代换表，注意在去重时需要格外小心，对单表代换密码计算的流程图如下：

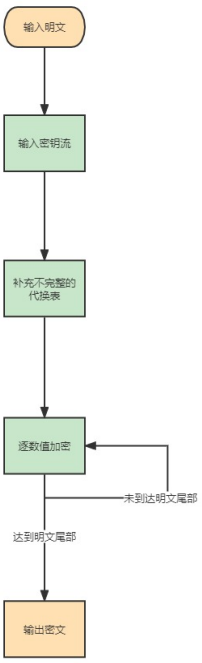

图 3.3: 单表代换流程图

```
1
2   \\单表代换密码
3   int singleTableChange(){
4       map<char, char>table;
5       vector<char>already;
6       char* message = new char[26];
7       memset(message, 0, 26);
8       int nowbase = 0;
9       cout << " 请输入置换序列，如果输入大于 26 位则只取前 26 位" << endl;
10      cin.getline(message, 26);
11      for (int i = 0; i < 26; i++) {
12          if (message[i] != 0) {
13              if (find(already.begin(), already.end(), message[i]) == already.end()) {
14                  char big;
15                  char small;
16                  if (message[i] >= 65 && message[i] <= 90) {
17                      big = message[i];
18                      small = message[i] + 32;
19                  }
20                  else {
21                      big = message[i] - 32;
22                      small = message[i];
23                  }
```

```
24                  table.insert(pair<char, char>(char(65+nowbase), big));
25                  table.insert(pair<char, char>(char(97 + nowbase), small));
26                  already.push_back(big);
27                  already.push_back(small);
28                  nowbase++;
29              }
30              else {
31                  continue;
32              }
33          }
34          else {
35              break;
36          }
37      }
38      for (int i = 0; i < 26; i++) {
39          if (find(already.begin(), already.end(), char(65 + i)) == already.end()) {
40              table.insert(pair<char, char>(char(65 + nowbase), char(65 + i)));
41              table.insert(pair<char, char>(char(97 + nowbase), char(97 + i)));
42              nowbase++;
43          }
44      }
45      cout << " 您的代换表为" << endl;
46      for (int i = 0; i < 26; i++) {
47          cout << char(97 + i) << " " << table.find(char(97 + i))->second <<" " << char(65 + i)
48      }
49      cout << " 请输入您的加密信息" << endl;
50      message = new char[1000];
51      cin.getline(message, 1000);
52      char* secretmessage = new char[1000];
53      int length = get_length(message);
54      for (int i = 0; i < length; i++) {
55          secretmessage[i] = table.find(message[i])->second;
56      }
57      secretmessage[length + 1] = '\0';
58      cout << " 您的明文是:" << message << endl;
59      cout << " 您的密文是:" << secretmessage << endl;
60      return 0;
61  }
62
```

3.4 对单表代换密码的攻击

对单表代换密码的攻击仅仅依靠机器进行分析的效率是比较低的，因为有很多情况下一个单词的的大部分单词已经被正确解密，剩下部分单词的拆解就可以由人来手工完成。

但是固定的程序也有很强的功能，他可以帮助我们记录统计文本中字符、单词出现的概率并绘制表格并记录已经有的翻译结果。我们则可以根据单词之间的统计概率以及已经被翻译的文本进行人为的调整翻译。对单表代换的攻击流程图如下图所示：

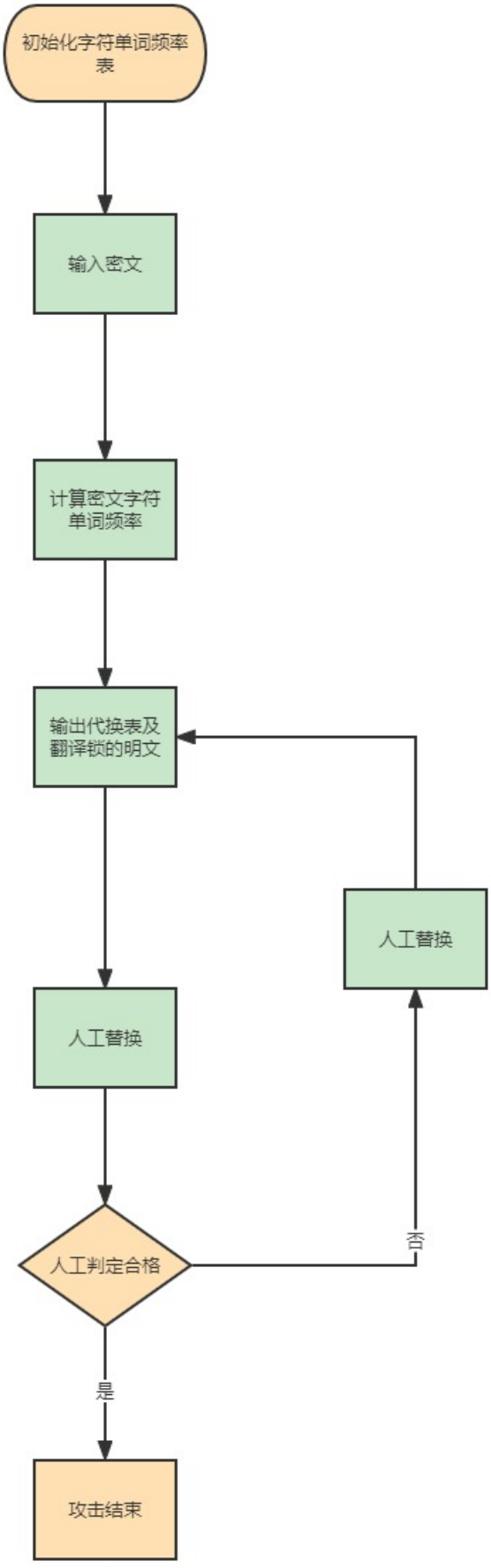
初始化字符单词频率表
输入密文
计算密文字符单词频率
输出代换表及翻译锁的明文
人工替换
人工替换
人工判定合格
否
是
攻击结束

图 3.4: 移位密码流程图

```
int attackSingleTableChange(){
    initialftable();
    char *message=new char[1000];
    ifstream infile;
    infile.open("message.txt",ios::in);
    if(!infile.is_open()){
        cout<<" 读取文件失败"<<endl;
        return -1;
    }
    char buff[1000];
    while(infile.getline(buff,1000)){
        for (int i = 0; i < 1000; i++) {
            if (buff[i] != '\0') {
                message[i] = buff[i];
                continue;
            }
            if (buff[i] == '\0') {
                message[i] = '\0';
                break;
            }
        }
    }
    int n = get_length(message);
    int cn = get_length_without_blank(message);
    cout<<" 您要攻击的密文长度为: " << get_length(message) << endl;
    cout << " 您要攻击的密文信息为: " << message << endl;
    map<char, double>temprecord;
    map<char, double>record;//记录中每个字符串出现频率
    //记录每个字符出现的次数并转换为小写存储便于识别
    for (int i = 0; i < n; i++) {
        char temp;
        if (message[i] >= 65 && message[i] <= 90) {
            temp = message[i] + 32;
            message[i] = temp;
        }else{
            if (message[i] >= 97 && message[i] <= 122) {
                temp = message[i];
            }
            else {
                continue;
            }
```

```
42          }
43          if (temprecord.find(temp) != temprecord.end()) {
44              temprecord[temp] += 1;
45              record[temp] += 1;
46              continue;
47          }
48          else {
49              temprecord.insert(pair<char, double>(temp, 1.0));
50              record.insert(pair<char, double>(temp, 1.0));
51          }
52      }
53      cout << message << endl;
54      //找那些没出现的
55      for (int i = 97; i <= 122; i++) {
56          if (record.find(char(i)) == record.end()) {
57              record.insert(pair<char, char>(char(i), 0.0));
58              temprecord.insert(pair<char, char>(char(i), 0.0));
59          }
60      }
61      char myrank[26];//密文中字符串排名
62      for (int i = 0; i < 26; i++) {
63          myrank[i] = findmax(temprecord);
64      }
65      //展示单表代换对比结果
66      cout << " 下面展示初步单表代换结果" << endl;
67      cout << " 统计字符" << "    " << " 统计字符频率" << "    " << " 密文字符" << "    " << " 密
68      for (int i = 0; i < 26; i++) {
69          cout <<"  " << frank[i] << "           " << ftable[frank[i] - 97] << "           " <<
70      }
71      //构造代换表
72      //以后需要手动更改
73      map<char, char>stctable;
74      for (int i = 0; i < 26; i++) {
75          stctable.insert(pair<char, char>(frank[i], myrank[i]));
76      }
77      cout << " 根据频率的单表代换结果解密如下: " << endl;
78      for (int i = 0; i < n; i++) {
79          if (message[i] >= 97 && message[i] <= 122) { cout << stctable[message[i]]; }
80          else { cout << message[i]; }
81      }
82      cout << " 自动处理部分结束, 其余部分需要手动替换解密" << endl;
83      cout << " 请输入想要设置的代换对, 格式为: a b, 他表示 a 的密文将被翻译为 b 的明文" << endl;
```

```
84      while (true) {
85          cout << " 请输入想要设置的代换对数目" << endl;
86          int number;
87          cin >> number;
88          for (int i = 1; i <= number; i++) {
89              cout << " 请输入第" << i << " 组代换" << endl;
90              char a, b;
91              cin >> a >> b;
92              char temp = stctable[a];
93              for (auto it : stctable) {
94                  if (it.second == b) {
95                      char c=it.first;
96                      stctable[c] = temp;
97                      break;
98                  }
99              }
100             stctable[a] = b;
101         }
102         cout << " 代换表: " << endl;
103         cout << " 统计字符" << "    " << " 统计字符频率" << "    " << " 密文字符" << "    " << "
104         for (auto it:stctable) {
105             cout << "  " << it.first << "           " << ftable[it.first - 97] << "
106         }
107         cout << " 解密结果" << endl;
108         cout << " 根据频率的单表代换结果解密如下: " << endl;
109         cout << message << endl;
110         for (int i = 0; i < n; i++) {
111             if (message[i] >= 97 && message[i] <= 122) { cout << stctable[message[i]]; }
112             else { cout << message[i]; }
113         }
114         cout << endl;
115     }
116 }
```

4 实验结果

4.1 移位密码效果图

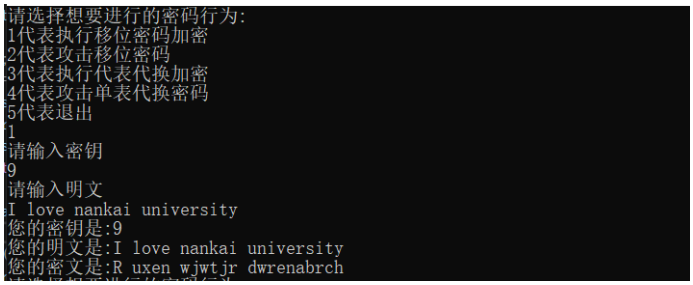

图 4.5: 移位密码效果图

4.2 攻击移位密码效果图

和同班级的周延霖同学组队，根据密文试图破译出明文

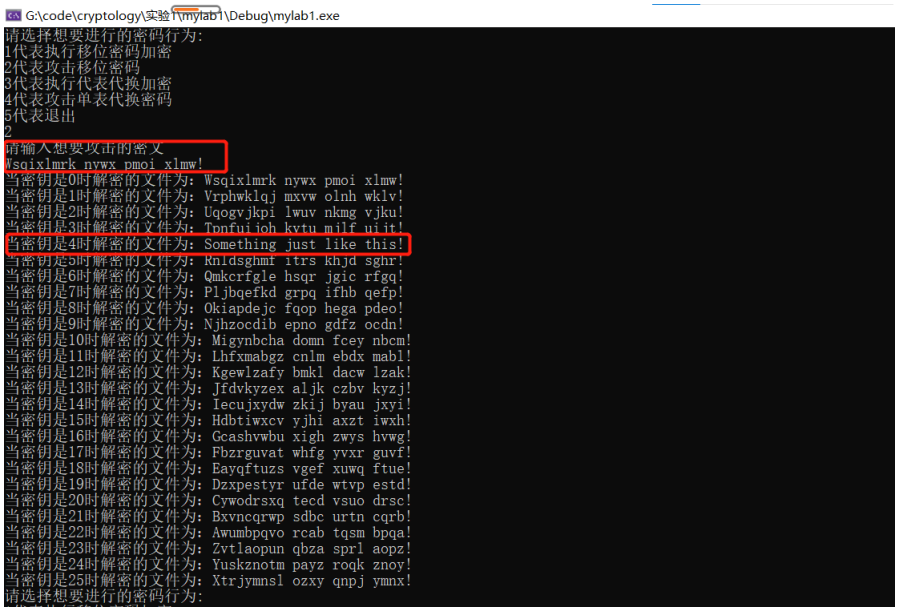

图 4.6: 移位密码效果图

4.3 单表代换密码效果图

```
请选择想要进行的密码行为:
1代表执行移位密码加密
2代表攻击移位密码
3代表执行单表代换加密
4代表攻击单表代换密码
5代表退出
3
请输入置换序列，如果输入大于26位则只取前26位
ILOVENankaiuniversity
您的代换表为
a i A I
b l B L
c o C O
d v D V
e e E E
f n F N
g a G A
h k H K
i u I U
j r J R
k s K S
l t L T
m y M Y
n b N B
o c O C
p d P D
q f Q F
r g R G
s h S H
t j T J
u m U M
v p V P
w q W Q
x w X W
y x Y X
z z Z Z
请输入您的加密信息
IlovenankaiUniversity
您的明文是:IlovenankaiUniversity
您的密文是:UtcpebibsiuMbupeghujx
```

图 4.7: 移位密码效果图

4.4 攻击移位密码效果图

最后的代换表以及解密结果如图所示

```
代换表:
统计字符    统计字符频率    密文字符    密文字符频率
 a          7.73          l          0
 b          1.58          n          7.63547
 c          3.06          e          2.21675
 d          3.24          d          0.738916
 e          11.67          g          3.44828
 f          2.14          f          1.72414
 g          2             c          8.867
 h          4.52          b          6.89655
 i          7.81          h          2.21675
 j          0.23          i          4.4335
 k          0.79          j          6.89655
 l          4.3           x          2.95567
 m          2.8           o          0.246305
 n          6.71          a          2.46305
 o          8.22          z          1.23153
 p          2.34          r          5.17241
 q          0.12          m          7.14286
 r          5.97          s          8.12808
 s          6.55          t          0.492611
 t          9.53          v          0.738916
 u          3.21          k          0
 v          1.03          w          0
 w          1.69          q          1.97044
 x          0.3           p          5.66502
 y          2.22          y          1.72414
 z          0.09          u          0
解密结果
根据频率的单表代换结果解密如下:
sic gcbspna xpmhacq jb gpyxsmepnxiy jr sins mf spnbrqjssjbe jbfmpqnsjmb fpmq n xmjbs n sm n xmjbs h hy qcnbr mf n xmrrjhay jbrcgzpc ginbbca jb rzgi n vny sins sic mpjejbna qcrrnec gnb mbay
hc pcgmtcpcd hy sic pjeisfza pcgjxjcbsr sic xnpsjgjxnbsr jb sic spnbrngsjmb npc najgc sic mpjejbnsmp mf sic qcrrnec hmh sic pcgcjtcp nbd mrgnp n xmrrjhac mxxmbcbs vim vjricr sm enjb zbnzsim
pjocd gmbspma mf sic qcrrnec
the central problem in cryptography is that of transmitting information from a point a to a point b by means of a possibly insecure channel in such a way that the original message can only
be recovered by the rightful recipients the participants in the transaction are alice the originator of the message bob the receiver and oscar a possible opponent who wishes to gain unautho
rized control of the message
请输入想要设置的代换对数目
```

图 4.8: 移位密码效果图

下面将根据人的主观能动性以及统计结果逐步进行单表代换密码解密步骤的说明:
首先，根据单词出现的频率以及开头的一个 sic 判断应该将其替换为 the

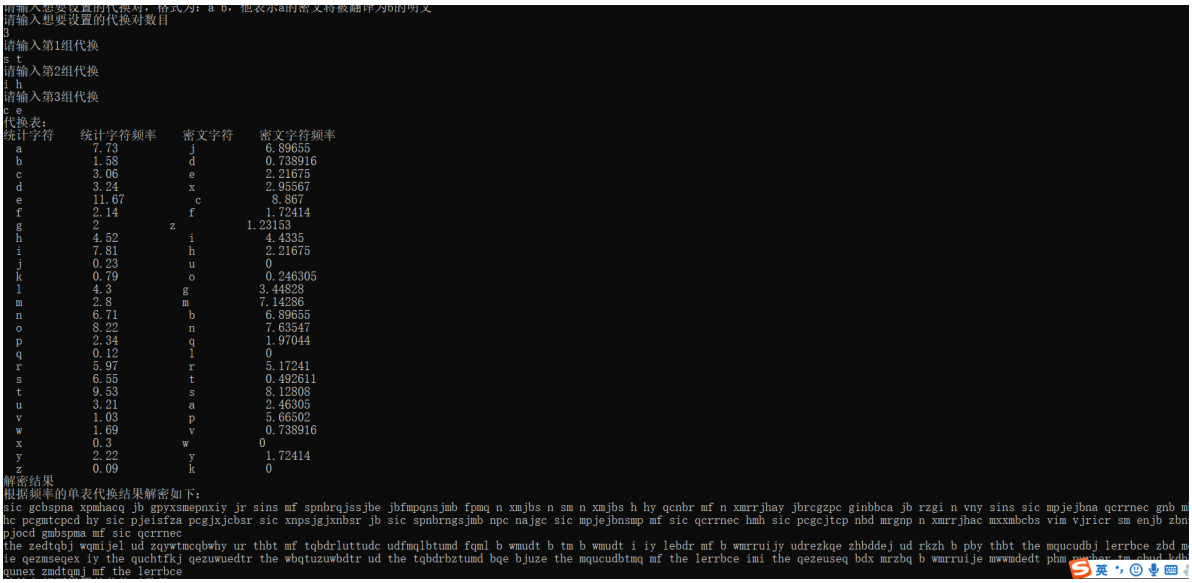

图 4.9: 置换 1

紧接着，我们看到密文中 n 总是单独出现出现，于是判定他应该翻译为 a

图 4.10: 置换 2

根据出现的连续两个单词的频率猜测书 mf 应该对应于 of,jr 应该对应为 is

图 4.11: 置换 3

图 4.12: 置换 4

然后根据最后一个单词我们已经解密出的字符猜测最后一个单词为 message 完成剩余字符的替换，同理猜测 wishes 用 n 代替 w

图 4.13: 置换 5

图 4.14: 置换 6

以此类推就可以利用前面翻译的结果不断推测出新的单词。