

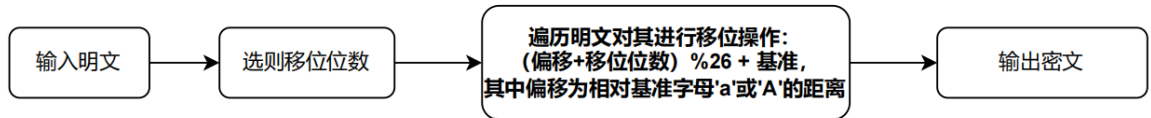
# 实验一 古典密码算法及攻击方法

学号： 姓名： 专业： 信息安全

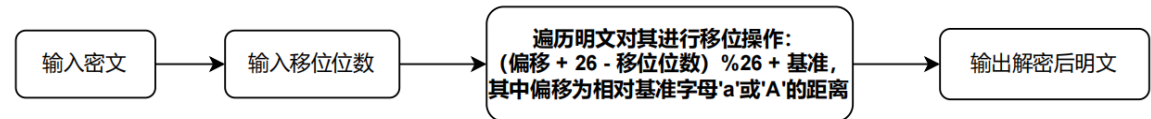
## 一. 移位密码

### 1. 算法流程图

#### ◦ 加密



#### ◦ 解密



### 2. 代码

```
1 // 移位密码加解密（自定义移位位数）
2 #include<iostream>
3 #include<string>
4 using namespace std;
5
6 // 加密函数
7 string Encrypt(string str, int offset)
8 {
9     int length = str.length();
10    string result;
11
12    for (int i = 0; i <= length - 1; i++)
13    {
14        if (str[i] >= 'A' && str[i] <= 'Z')
15        {
16            int shift = (str[i] - 'A' + offset) % 26 + 'A';
17            result = result + (char)shift;
18        }
19        if (str[i] == ' ')
20        {
21            result += ' ';
22            continue;
23        }
24        if (str[i] >= 'a' && str[i] <= 'z')
25        {
26            int shift = (str[i] - 'a' + offset) % 26 + 'a';
27            result = result + (char)shift;
28        }
29    }
30    cout << "加密结果为: " << result << endl;
31    return result;
32 }
33
34 //解密函数
```


```

35 void Decryption(string str, int offset)
36 {
37     int length = str.length();
38
39     string result;
40
41     for (int i = 0; i <= length - 1; i++)
42     {
43         if (str[i] >= 'A' && str[i] <= 'Z')
44         {
45             int shift = (str[i] - 'A' + 26 - offset) % 26 + 'A';
46             result = result + (char)shift;
47         }
48         if (str[i] == ' ')
49         {
50             result += ' ';
51             continue;
52         }
53
54         if (str[i] >= 'a' && str[i] <= 'z')
55         {
56             int shift = (str[i] - 'a' + 26 - offset) % 26 + 'a';
57             result = result + (char)shift;
58         }
59     }
60     cout << "解密结果为: " << result << endl;
61 }
62
63 int main()
64 {
65     string text, encode;
66     int length = 0;
67
68     //明文输入
69     cout << "请输入明文: " << endl;
70     getline(cin, text);
71     cout << "请输入移位密码移位位数 " << endl;
72     cin >> length;
73
74     // 加密
75     encode = Encrypt(text, length);
76
77     // 解密
78     Decryption(encode, length);
79
80 }

```

### 3. 验证

明文: public keys, 移位密码位数: 3



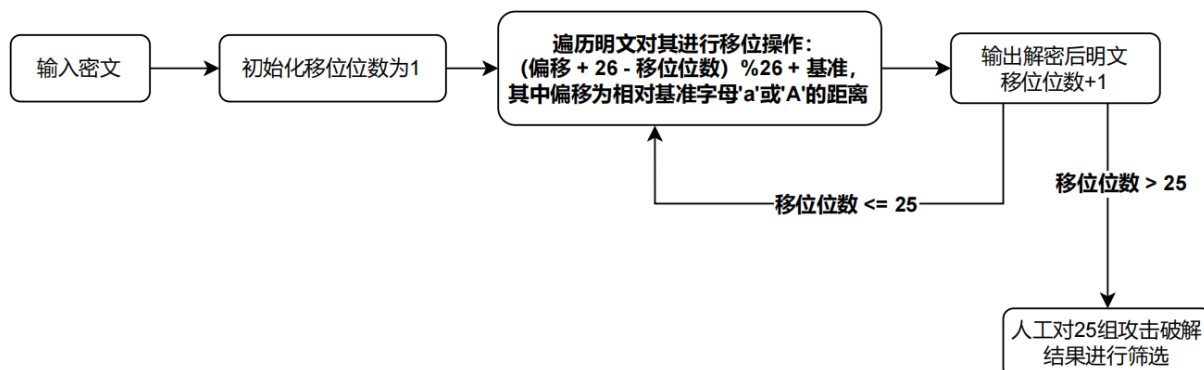
```

Microsoft Visual Studio 调试控制台
请输入明文:
public keys
请输入移位密码移位位数
3
加密结果为: sxeolf nhbv
解密结果为: public keys
D:\算法\密码学\1_古典密码算法及攻击方法\
按任意键关闭此窗口...

```

## 二. 对移位密码的攻击

### 1. 算法流程图



### 2. 代码

```
1 // 对移位密码攻击
2 // 遍历移位数从并输出解密结果，人工识别哪个单词可读
3
4 #include<iostream>
5 #include<string>
6 using namespace std;
7
8 void Decryption(string str) //因为不区分大小写，所以统一按照小写字母来处
   理的;
9 {
10     int length = str.length();
11     string result;
12
13     for (int j = 1; j <= 25; j++)
14     {
15         result = { 0 };
16         for (int i = 0; i <= length - 1; i++)
17         {
18             if (str[i] >= 'A' && str[i] <= 'Z')
19                 str[i] = str[i] + 32;
20             if (str[i] >= 'a' && str[i] <= 'z')
21             {
22                 int shift = (str[i] - 97 + 26 - j) % 26 + 97;
23                 result += (char)shift;
24             }
25             else
26                 result += " ";
27         }
28         cout << "移位位数为      : " << j << endl;
29         cout << "攻击解密结果为: " << result << endl;
30     }
31 }
32
33 int main()
34 {
35     string ciphertext;
```

```

36     int length = 0;
37
38     //密文输入
39     cout << "请输入密文: " << endl;
40     getline(cin, ciphertext);
41     cout << "===== " << endl <<
endl;
42
43     // 对移位密码进行攻击
44     Decryption(ciphertext);
45 }

```

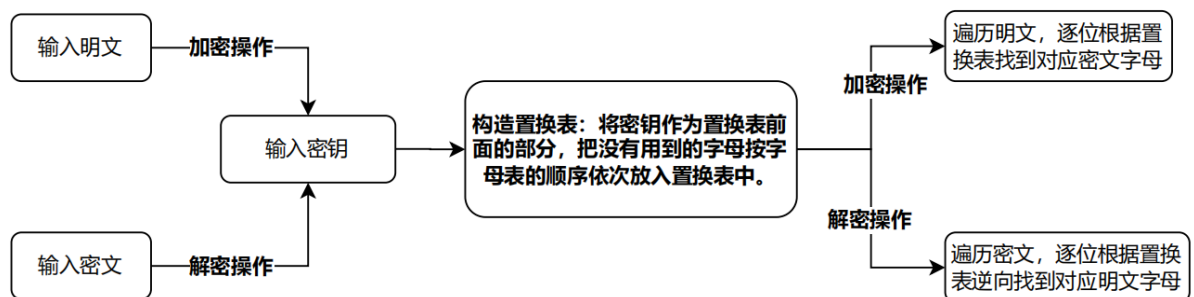
### 3. 结果

密文为: sxeolf nhbv

可以通过观察结果, 得到密钥(偏移量)为3

## 三. 单表置换密码

### 1. 算法流程图



### 2. 代码

```

1  #include<iostream>
2  #include<cstdio>
3  #include<string>
4  #include<map>
5  using namespace std;
6
7  // 用来标记该字母之前是否已被添加到置换表: 未添加 - bool值为1, 添加 - bool值为0
8  map<char, bool> Book_Use;
9  char Rep_Table[26][2];
10
11 // 构造置换表
12 void setTable(string str)
13 {

```

```

14     int len = str.length();
15     int mappingnum = 0;
16
17     // 使用一个短语或句子作为置换表的前面的部分
18     for (int i = 0; i <= len - 1; i++)
19     {
20         bool exist = false;
21         // 不区分大小写
22         if (str[i] >= 'A' && str[i] <= 'Z')
23             str[i] += 32;
24
25         if (str[i] < 'a' || str[i] > 'z')
26             continue;
27
28         for (int j = 0; j <= mappingnum - 1; j++)
29         {
30             if (Rep_Table[j][1] == str[i])
31                 exist = true;
32         }
33         if (!exist)
34         {
35             Rep_Table[mappingnum][1] = str[i];
36             mappingnum++;
37             Book_Use[str[i]] = 0;
38         }
39     }
40
41     // 把没有用到的字母按字母表的顺序依次放入置换表中
42     map<char, bool>::iterator iter;
43     for (iter = Book_Use.begin(); iter != Book_Use.end(); iter++)
44     {
45         if (iter->second == 1)
46         {
47             Rep_Table[mappingnum][1] = iter->first;
48             iter->second = 0;
49             mappingnum++;
50         }
51     }
52 }
53
54 // 加密函数
55 string Encrypt(string str)
56 {
57     int length = str.length();
58     string result;
59     int flag = 0;
60     for (int i = 0; i <= length - 1; i++)
61     {
62         if (str[i] >= 'A' && str[i] <= 'Z')
63             str[i] += 32;
64         if (str[i] == ' ')
65         {
66             result += ' ';
67             continue;
68         }
69         flag = 0;
70         if (str[i] < 'a' || str[i] > 'z')
71             continue;

```

```

72
73     for (int j = 0; j <= 25; j++)
74         if (Rep_Table[j][0] == str[i])
75         {
76             flag = j;
77             break;
78         }
79     result += Rep_Table[flag][1];
80 }
81 cout << "解密结果为: " << result << endl << endl;
82 return result;
83 }
84
85 // 解密函数
86 void Decryption(string str)
87 {
88     int length = str.length();
89     string result;
90     int flag = 0;
91     for (int i = 0; i <= length - 1; i++)
92     {
93         if (str[i] >= 'A' && str[i] <= 'Z')
94             str[i] += 32;
95         if (str[i] == ' ')
96         {
97             result += ' ';
98             continue;
99         }
100         flag = 0;
101         if (str[i] < 'a' || str[i] > 'z')
102             continue;
103
104         for (int j = 0; j <= 25; j++)
105             if (Rep_Table[j][1] == str[i])
106             {
107                 flag = j;
108                 break;
109             }
110         result += Rep_Table[flag][0];
111     }
112     cout << "解密结果为: " << result << endl;
113 }
114
115
116 int main()
117 {
118     for (int i = 0; i <= 25; i++)
119     {
120         Rep_Table[i][0] = 97 + i;
121         Rep_Table[i][1] = '\0';
122         Book_Use[Rep_Table[i][0]] = 1;
123     }
124     // 构造置换表
125     string key;
126     cout << "请输入密钥: ";
127     getline(cin, key);
128     setTable(key);
129

```

```


130 // 输出置换表
131 for (int i = 0; i <= 25; i++)
132     cout << Rep_Table[i][0] << " ";
133 cout << endl;
134 for (int i = 0; i <= 25; i++)
135     cout << Rep_Table[i][1] << " ";
136 cout << endl;
137
138 // 输入明文字符串
139 string text, Encode;
140 cout << "请输入明文字符串: " << endl;
141 getline(cin, text);
142 Encode = Encrypt(text);
143 Decryption(Encode);
144 }

```

### 3. 验证

创建明文信息为: Cryptography is very important for cyberspace security major

选则密钥为: helloworld

 Microsoft Visual Studio 调试控制台

```

请输入密钥: helloworld
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 e l o w r d a b c f g i j k m n p q s t u v x y z
请输入明文字符串:
Cryptography is very important for cyberspace security major
=====

```

```

解密结果为: lpymskdphmay bq uwpy bimkpshjs rkp lyewpqmhlw qwltpbsy ihckp
解密结果为: cryptography is very important for cyberspace security major

```

D:\算法\密码学\1\_古典密码算法及攻击方法\单表置换密码\Debug\单表置换密码.exe (进程 ID: 12345) 按任意键关闭此窗口...

## 四. 对单表置换密码的攻击

### 1. 算法核心思想

首先将密文中每个字母出现的频率进行统计, 并与实验文档中给出的自然语言中的字母统计频率进行比较匹配。但是, 由于所给定的密文段较短, 因此仅依靠自然语言的统计规律, 难以准确地构造正确的置换表。因此, 根据字母频率构造置换表完成置换之后, 还要手工进行相应的调整。

### 2. 实验过程

#### 。 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
PCGMTPCPD 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

## 。统计密文中每个字母出现的频率

```
Microsoft Visual Studio 调试控制台

Please input encrypted message:
SIC GCBSRNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJH
AY JBRCGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QCRRNEC GNB MBAY HC PCGMTCPD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJ
XNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QCRRNEC HMH SIC PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRI
CR SM ENJB ZBNZSIMPJOCD GMBSPMA MF SIC QCRRNEC

=====
各个字母频率为:
a: 10  b: 28  c: 36  d: 3  e: 9  f: 7  g: 14  h: 9  i: 18  j: 28  k: 0  l: 0  m: 29
n: 31  o: 1  p: 23  q: 8  r: 21  s: 33  t: 2  u: 0  v: 3  w: 0  x: 12  y: 7  z: 5
```

## 。比照统计结果，初步构造置换表

已知信息：

- 字母频率：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

将根据此明文得到的字母频率高低排序结果，与自然语言中字母出现的频率（上述已知信息）进行匹配，可得到如下初步的匹配结果：

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  
b v e x c y z i m l o g h j n q u r p s a t d k f w

根据初步构造的置换表进行解密，得到如下结果：

```
Microsoft Visual Studio 调试控制台

Please input encrypted message:
SIC GCBSRNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJH
AY JBRCGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QCRRNEC GNB MBAY HC PCGMTCPD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJ
XNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QCRRNEC HMH SIC PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRI
CR SM ENJB ZBNZSIMPJOCD GMBSPMA MF SIC QCRRNEC

=====
各个字母频率为:
a: 10  b: 28  c: 36  d: 3  e: 9  f: 7  g: 14  h: 9  i: 18  j: 28  k: 0  l: 0  m: 29
n: 31  o: 1  p: 23  q: 8  r: 21  s: 33  t: 2  u: 0  v: 3  w: 0  x: 12  y: 7  z: 5

解密结果为:
the leatsou dsimuep na lsfdticsodhf nr thot iy tsoarpnttnac navispotnia ysip o dinat o ti o dinat m mf peoar iy o dirrmn
if narelge lhoaeeu na rgih o bof thot the isncnaou perroce loa iauf me selivesew mf the snchtygu selndneatr the dostnl
loatr na the tsoaroltnia ose oumle the isncnaotis iy the perroce mim the selenves oaw irlos o dirrmnue iddiaeat bhi bnrh
r ti cona gaogthisnkew liatsiu iy the perroce

D:\算法\密码学\1_古典密码算法及攻击方法\对单表置换密码的攻击\Debug\对单表置换密码的攻击.exe (进程 18896) 已退出，返回代码
为: 0。
按任意键关闭此窗口...
```



## 。手工调整置换表

观察解密得到的明文，第一行中有“thot”一词，根据英文中常用单词 “that” 可以初步判断应该是在解密过程中将原本被翻译为o的密文翻译为a，将原本被翻译为a的密文翻译为o，因此需要对换置表中明文字母o、a对应的密文字母，调整后置换表如下：

```
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
n v e x c y z i m l o g h j b q u r p s a t d k f w
```

根据调整后的置换表进行解密，得到如下结果：

```
Microsoft Visual Studio 调试控制台
Please input encrypted message:
SIC GCBS PNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNR MF N XMRRJH
AY JBRCGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QRRNEC GNB MBAY HC PCGMTCPD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJ
XNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QRRNEC HMH SIC PCGJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRI
CR SM ENJB ZBNZSIMPJOC D GMBSPMA MF SIC QRRNEC

=====
各个字母频率为:
a: 10 b: 28 c: 36 d: 3 e: 9 f: 7 g: 14 h: 9 i: 18 j: 28 k: 0 l: 0 m: 29
n: 31 o: 1 p: 23 q: 8 r: 21 s: 33 t: 2 u: 0 v: 3 w: 0 x: 12 y: 7 z: 5

解密结果为:
the leotsau dsimuep no lsfdticsadhf nr that iy tsaorpnntnoc noyispatnio ysip a dinot a ti a dinot m mf peaoir iy a dirrmn
uf norelgse lhaoou no rglh a baf that the isncnoau perrace lao iouf me selivesew mf the snchtygu selndneotr the dastnl
daotr no the tsaoraltnio ase aunle the isncnoatis iy the perrace mim the selenves aow irlas a dirrmue iddioeot bhi bnrh
er ti cano goagthisnkew liotsiu iy the perrace

D:\算法\密码学\1_古典密码算法及攻击方法\对单表置换密码的攻击\Debug\对单表置换密码的攻击.exe (进程 10828) 已退出, 返回代码
为: 0。
按任意键关闭此窗口...
```

## 。按照上述手工调整置换表的方法进行多次调整

最终的置换表如下所示：

```
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
n h g d c f e i j l w a q b m x u p r s z t v k y o
```

根据置换表进行解密，得到如下结果，可以看出得到了明文，因此上面置换表不需再调整

```
Microsoft Visual Studio 调试控制台
Please input encrypted message:
SIC GCBS PNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNR MF N XMRRJH
AY JBRCGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QRRNEC GNB MBAY HC PCGMTCPD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJ
XNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QRRNEC HMH SIC PCGJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRI
CR SM ENJB ZBNZSIMPJOC D GMBSPMA MF SIC QRRNEC

=====
各个字母频率为:
a: 10 b: 28 c: 36 d: 3 e: 9 f: 7 g: 14 h: 9 i: 18 j: 28 k: 0 l: 0 m: 29
n: 31 o: 1 p: 23 q: 8 r: 21 s: 33 t: 2 u: 0 v: 3 w: 0 x: 12 y: 7 z: 5

解密结果为:
the central problem in cryptography is that of transmitting information from a point a to a point b by means of a possib
ly insecure channel in such a way that the original message can only be recovered by the rightful recipients the partici
pants in the transaction are alice the originator of the message bob the receiver and oscar a possible opponent who wish
s to gain unauthorized control of the message

D:\算法\密码学\1_古典密码算法及攻击方法\对单表置换密码的攻击\Debug\对单表置换密码的攻击.exe (进程 14020) 已退出, 返回代码
为: 0。
按任意键关闭此窗口...
```

## 3. 代码

```
1 // 对单表置换攻击
2 #include<iostream>
3 #include<string>
4 using namespace std;
5
6 char Rep_Table[26][26]; //置换表
7 int frequency[26]; //统计频次
```

```

8
9 // 解密函数
10 void Decryption(string str)
11 {
12     int length = str.length();
13     string result;
14     int flag = 0;
15     for (int i = 0; i <= length - 1; i++)
16     {
17         if (str[i] >= 'A' && str[i] <= 'Z')
18             str[i] += 32;
19         if (str[i] == ' ')
20         {
21             result += ' ';
22             continue;
23         }
24         flag = 0;
25         if (str[i] < 'a' || str[i] > 'z')
26             continue;
27
28         for (int j = 0; j <= 25; j++)
29             if (Rep_Table[j][1] == str[i])
30             {
31                 flag = j;
32                 break;
33             }
34         result += Rep_Table[flag][0];
35     }
36     cout << "解密结果为: " << endl << result << endl;
37 }
38
39
40 void Attack(string text) {
41     // 遍历密文，统计频次
42     int index;
43     for(index = 0; text[index] != '\0'; index += 1)
44     {
45         if (text[index] <= 'Z' && text[index] >= 'A') {
46             // 如果是大写字母
47             int i = text[index] - 'A';
48             // 获取到的代换后的字母是小写字母，需要利用ASCII的关系转换成大写字
49             frequency[i] += 1;
50         }
51         else
52         if (text[index] <= 'z' && text[index] >= 'a')
53         {
54             // 如果是小写字母
55             int i = text[index] - 'a';
56             frequency[i] += 1;
57         }
58     }
59
60     // 输出统计的频次
61     cout << "各个字母频率为: ";
62     for (int i = 0; i < 26; i++) {
63         if (i % 13 != 0)
64             cout << char(97 + i) << ": " << frequency[i] << '\t';

```

```

65         else
66             cout << endl << char(97 + i) << ": " << frequency[i] << '\t';
67     }
68     cout << endl << endl;
69
70     // 根据多轮更正，最终根据分析结果得到正确置换表
71     char replaceTable[26] = {
72         'n','h','g','d','c','f','e','i','j','0','0','a','q','b','m','x','0','p','r',
73         ',','s','z','t','v','0','y','o' };
74
75     // 构造置换表
76     for (int i = 0; i <= 25; i++)
77     {
78         Rep_Table[i][0] = 97 + i;
79         Rep_Table[i][1] = replaceTable[i];
80     }
81
82     //根据置换表进行解密
83     Decryption(text);
84
85 }
86
87 int main()
88 {
89     // 输入要统计的密文
90     cout << "Please input encrypted message:\n";
91     string ciphertext;
92     getline(cin, ciphertext);
93     cout << "===== " << endl
<< endl;
94     Attack(ciphertext);
95 }

```