



南開大學  
Nankai University

南 开 大 学  
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

---

罗功成 1910487

年级：2019 级

专业：信息安全

2022 年 4 月 20 日

## 一、 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

## 二、 实验原理

### 1. 移位密码和攻击策略：

(1) 移位密码通过各个字母按照字母表位置向后同步循环式移动若干位来实现加密。

(2) 由于密钥空间有限且很小（除明文外最多 25 种可能），可直接通过穷举法破解。

### 2. 单表置换密码及其攻击策略：

(1) 单表置换密码通过字母表对应关系置换实现加密。

(2) 由于置换关系不确定，至多有  $26!$  种可能，这使得穷举的效率很低。因此可通过对短单词，常用词汇，固定搭配等单词出现频率的统计关系来分析破译。

## 三、 程序流程图

### 1. 移位密码：

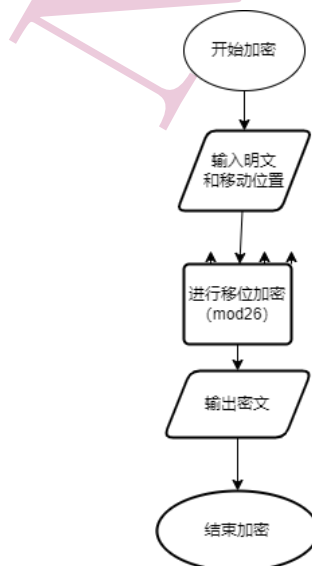


图 1: 移位密码加密

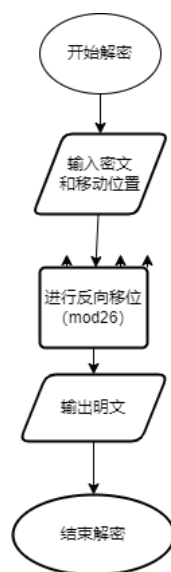


图 2: 移位密码解密

## 2. 置换密码:

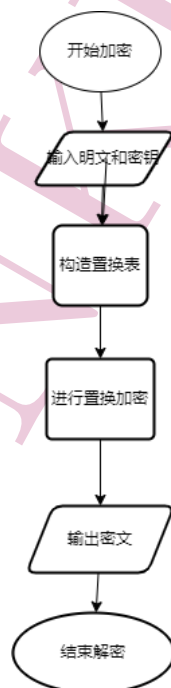


图 3: 置换密码加密

对应的解密即为加密过程的逆过程。

## 四、 代码说明

1. 具体使用过程中在命令行有相应提示。

2. 移位密码:

E 为加密标志, D 为解密标志, 涉及的数字主要为字母个数 26, 字母对应的 ascii 码等

3. 单表置换密码:

(1) 这里以大写字母作为例子实现的, 密钥构建和明文密文输出时对英文字母加密, 其他字符不加密。本程序中添加了大小写转换, 可以同时支持大小写两种输入方式, 输出加密内容或解密内容时都转换成为大写字母。

(2) 注意要输满 26 个字母。

(3) s1 是需要输入的置换对应关系; s2 是输入明文, 之后可加密后输出为密文; s3 是输入密文解明文。

## 五、 设计思路

1. 移位密码:

(1) 加密: 先减 65, 将字符转化为在字母表中的下标位置; 对移位个数取模 ( $\text{mod}26$ ), 实现循环移位。

(2) 解密: 加密的逆过程, 但注意  $\text{mod}$  运算转换为循环加的过程。

(3) 攻击: 穷举法, 遍历移位后进行解密, 和解密过程类似。

2. 单表置换密码:

(1) 置换表 (输入密钥): 按对应关系逐个输入即可。

(2) 加密: 本质是找到明文对应密文的哪个字符。将明文减 65 变成下标关系, 用密文串中的对应下标处的字母替换即可。

(3) 解密: 加密逆过程, 输入密文的各个字母先找到对应密文的下标, 利用密文下标对应明文。注意下标之间对应关系看清楚即可。

(4) 攻击: 统计字符出现频率。

## 六、 核心代码

### 移位加密

```
1  if (c == 'E')
2      {
3
4      for (int i = 0; i < s.length(); i++)
5      {
6          if (s[i] >= 65 && s[i] <= 90) //大写字母
7          {
8              int t = s[i] + a - 65;
9              t = t % 26;
10             s[i] = (char)(t + 65);
11             continue;
12         }
13
14         if (s[i] >= 97 && s[i] <= 122) //小写字母
15         {
16             int t = s[i] + a - 97;
17             t = t % 26;
18             s[i] = (char)(t + 97);
19             continue;
20         }
21     }
22     cout << s;
23
24
25
26 }
```

### 移位解密

```
1  if (c == 'D')
2      {
3
4      for (int i = 0; i < s.length(); i++)
5      {
6          if (s[i] >= 65 && s[i] <= 90) //大写字母
7          {
8              //int t = s[i] - a - 65; 注意，这里不取模时就不用先减再加了！
9              a = a % 26; //对移位数字取模
10             int t = s[i] - a ;
11             if (t < 65)
12                 t = t + 26; //越减越小，要按26加
13             //s[i] = (char)(t + 65);
14             s[i] = (char)(t);
15
16             continue;
```

```
17         }
18
19         if (s[i] >= 97 && s[i] <= 122)//小写字母
20         {
21             a = a % 26;
22             int t = s[i] - a ;
23             if (t < 97)
24                 t = t + 26;
25             s[i] = (char)(t);
26             continue;
27         }
28
29     }
30     cout << s;
```

## 移位攻击

```
1     cout << "请输入密文" << endl;
2
3     getline(cin, s);
4     for (int a = 0; a < 26; a++)
5     {
6         cout << "移位位数为: " << a << "时" << endl;
7         for (int i = 0; i < s.length(); i++)
8         {
9             if (s[i] >= 65 && s[i] <= 90)//大写字母
10            {
11
12                int t = s[i] - a;
13                if (t < 65)
14                    t = t + 26;//越减越小, 要按26加
15
16                s[i] = (char)(t);
17
18                continue;
19            }
20
21            if (s[i] >= 97 && s[i] <= 122)//小写字母
22            {
23
24                int t = s[i] - a;
25                if (t < 97)
26                    t = t + 26;
27                s[i] = (char)(t);
28                continue;
29            }
30
31        }
32        cout << s << endl;
```

## 构建置换表

```
1 string s1, s2, s3;// 输入字符串
2 cout << "请输入密钥s1以构建置换表" << endl;
3 cout << "要求: 1.填满26位.2.大写字母.3.按字母表顺序给出对应关系" << endl;
4 getline(cin,s1);
5 cout << "置换表为: " << endl;
6 for (int i = 0; i < 26; i++)
7 {
8     cout << (char)('A' + i) << " ";
9 }
10 cout << endl;
11 for (int i = 0; i < s1.length(); i++)
12 {
13     if (s1[i] >= 65 && s2[i] <= 90)
14         cout << s1[i] << " ";
15     else
16         continue;
17 }
```

## 单表代换加密

```
1
2 cout << "输入明文，进行加密后得密文" << endl;
3 getline(cin,s2);
4 int count1 = s2.length();
5 int* a;
6 a = new int[count1];
7 int t;
8 for (int i = 0; i < count1; i++)//字母转换成坐标关系。
9 {
10     if (s2[i] >= 97 && s2[i] <= 122)
11         s2[i] = s2[i] - 97 + 65;//如果输入小写时转换为大写字母进行加密。
12     if (s2[i] >= 65 && s2[i] <= 90)
13     {
14         t = s2[i] - 65;
15         a[i] = s1[t];
16     }
17     else
18         a[i] = s2[i];
19 }
20 for (int i = 0; i < count1; i++)
21     cout << char(a[i]);
```

## 单表代换解密

```
1 cout << "输入密文，进行解密后得明文" << endl;
2 getline(cin,s3);
```

```
3   int count2 = s3.length();
4   int* b;
5   b = new int[count2];
6   //对应的密文不是连续关系，考虑逐个做减法（密-明）来记录明密文之间的关系
7   //密-（密-明）=明
8   // 不可行！注意s3,s1对应关系不一致！
9   int *p=new int [count2]; // 偏移数表
10
11
12  //输入密文-》对应到s1[i]->i->明文i+65
13  for (int i = 0; i < count2; i++)//字母转换成坐标关系。
14  {
15      for (int j = 0; j < 26; j++)//s3中各位对应的是密文中的第几个
16      {
17          if (s3[i] >= 97 && s3[i] <= 122)
18              s3[i] = s3[i] - 97 + 65;//如果输入小写时转换为大写字母进行加
              密。
19          if (s3[i] >= 65 && s3[i] <= 90)
20          {
21              if (s3[i] == s1[j]) //!!双等号!
22                  p[i] = j;
23          }
24          else
25              p[i] = s3[i];
26      }
27  }
28
29
30
31  for (int i = 0; i < count2; i++)//第几个密文，下标加上65便是明文
32  {
33      if (s3[i] >= 65 && s3[i] <= 90)
34          b[i] = p[i] + 65;
35      else
36          b[i] = p[i];
37  }
38  for (int i = 0; i < count2; i++)
39      cout << char(b[i]);
```



## 七、 实验结果



```

Microsoft Visual Studio 调试控制台
请输入明/密文
I love NANKAI 1919!@NKUCS
输入E加密，输入D解密
E
请输入移位个数
3
L oryh QDQNDL 1919!@QNXFV

```

图 4: 移位加密

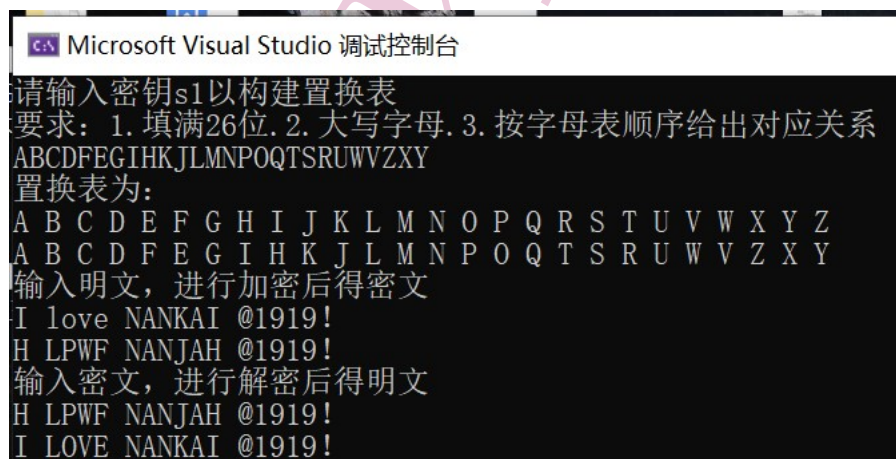


```

Microsoft Visual Studio 调试控制台
请输入明/密文
L oryh QDQNDL 1919!@QNXFV
输入E加密，输入D解密
D
请输入移位个数
3
I love NANKAI 1919!@NKUCS

```

图 5: 移位解密



```

Microsoft Visual Studio 调试控制台
请输入密钥s1以构建置换表
要求: 1. 填满26位. 2. 大写字母. 3. 按字母表顺序给出对应关系
ABCDEFGHIKJLMNPOQTSRUWVZXY
置换表为:
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
A B C D F E G I H K J L M N P O Q T S R U W V Z X Y
输入明文，进行加密后得密文
I love NANKAI @1919!
H LPWF NANJAH @1919!
输入密文，进行解密后得明文
H LPWF NANJAH @1919!
I LOVE NANKAI @1919!

```

图 6: 单表置换加密解密

移位密码攻击：输入一串被加密过的密文：k nqyg pcpmck

```

请输入密文
k nqxs pcpmck
移位位数为: 0时
k nqxs pcpmck
移位位数为: 1时
j mpwf obolbj
移位位数为: 2时
h knud mzmjzh
移位位数为: 3时
e hkra jwigwe
移位位数为: 4时
a dgnw fsfcsa
移位位数为: 5时
v ybir anaxnv
移位位数为: 6时
p svcl uhurhp
移位位数为: 7时
i love nankai
移位位数为: 8时
a dgnw fsfcsa
移位位数为: 9时
r uxen wjwtjr
移位位数为: 10时
h knud mzmjzh
移位位数为: 11时
w zcjs bobyow
移位位数为: 12时
k nqxs pcpmck
移位位数为: 13时
x adkt cpczpx
移位位数为: 14时
j mpwf obolbj
移位位数为: 15时
u xahq zmzwmu
移位位数为: 16时
e hkra jwigwe
移位位数为: 17时
n qtaj sfspfn
移位位数为: 18时
v ybir anaxnv
移位位数为: 19时

```

图 7: 移位攻击

穷举攻击后，可见在移位位数为 20 时恢复明文：i love nankai

```

n qtaj sfspfn
移位位数为: 18时
v ybir anaxnv
移位位数为: 19时
c fipy huheuc
移位位数为: 20时
i love nankai
移位位数为: 21时
n qtaj sfspfn
移位位数为: 22时
r uxen wjwtjr
移位位数为: 23时
u xahq zmzwmu
移位位数为: 24时
w zcjs bobyow
移位位数为: 25时
x adkt cpczpx

```

图 8: 移位攻击

#### 4. 单表置换攻击:

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 SPN-  
 BRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QCRRNEC HMH SIC PCGCJTCP

NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRICR SM ENJB ZBNZSIMPJ OCD GMB-  
SPMA MF SIC QCRRNEC

首先，我们观察文本中大量出现的短单词和搭配：

(1) SIC, N 经常出现在一个较长的词汇前面，如 SIC GCBSPNA, SIC PJEISFZA, N XM-  
RRJHAY 等。根据经验，较长的单词前还需要加一个长度为 3 或 1 的短单词，那么这个长单词  
应该是一个名词，前面的长度为 3 的是定冠词 the, 长度为 1 的是 a. 因此 S->T, I->H, C->E, N->A

(2) JB 经常出现在两个长单词（大概率是名词）之间，如 XPMHACQ JB GPYXSMEP-  
NXIY, SIC XNPSJGJXNBSR JB SIC SPNBRNGSJMB 等，可能是 in, 也可能是 of; 但是注  
意到 GPYXSMEPNXIY JR SINS MF 中 JR 的出现，若 J 是 o 的话 JR 无法和常见短单词匹  
配，因而 J->I, B->N。以 i 开头 JR 处在一个长单词和一个短单词之间，很可能是在解释，常见  
的是 is, 因此 R->S

(3) s 为 T, 那么 SM 作为以 t 开头的常用短语只能是 to, M->O, MF 可翻译成 OF, 得  
F->F; 另一个常用的 SINS 现在可翻译为 THAT; FPMQ 变成 FPOQ, 和 FROM 很相似，因此  
P->R, Q->M

其次，统计各字母出现频率：

本文的字母频率 (归一化)

1	a	0.03	b	0.08	c	0.11	d	0.01	e	0.03	f	0.02	g	0.04
2	h	0.03	i	0.05	j	0.08	k	0.00	l	0.00	m	0.09	n	0.09
3	o	0.002	p	0.07	q	0.02	r	0.06	s	0.10	t	0.01	u	0.00
4	v	0.01	w	0.00	x	0.04	y	0.02	z	0.01				

参考字母使用频率

1	e	11.67	t	9.53	o	8.22	i	7.81	a	7.73	n	6.71	s	6.55
2	r	5.97	h	4.52	l	4.3	d	3.24	u	3.21	c	3.06	m	2.8
3	p	2.34	y	2.22	f	2.14	g	2.00	w	1.69	b	1.58	v	1.03
4	k	0.79	x	0.30	j	0.23	q	0.12	z	0.09				

将本文档中出现的频率顺序和参考文件中的使用频率对照，既可以验证前面的短单词解法  
关系是否合理，还可以解出尚未找到的对应关系。比如本文中最高频率出现的 c, 应该是对应明  
文中最常出现的 e. 注意，可以按顺序直接将其作为置换表，和前面推理不一致时可以分别将两  
者代入，看哪个更合理。比如 z 在使用频率中是最小的，可能对应 l, k, u 等根本没出现的，也可  
能是 o 这样出现了但次数极少的。可分别代入后，发现代入 o 后变为单词 unauthorized, 显然  
更合理，这时就选 o 作为对应明文 z.

最后，结合上面破译出的信息，阅读文字，看哪些单词和常用单词中极为相似，不同之处  
的字母就是剩下尚未破解的置换关系。比如 probuem 和 problem 很像，那么根据上下文意思  
发现这里填入 problem 语意也很通顺，这样就可以将 U 对应为 L. 类似的还有 inforpation 和  
information, day 和 way, uentral 和 central 等

最终解密的数据: 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 unauthorized control of the message

对应的置换表为:

替换表为:

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

图 9: 置换表

## 八、 总结和收获

1. 移位密码加解密和攻击的实现较为简单, 但缺点就是密钥空间过小, 安全性不佳, 仅需要 25 次穷举密钥即可攻破。
2. 单表代换密码实现难度得以提升, 穷举难度较大, 但是通过常用词汇分析和词频统计的方式还是可以被攻破, 因而若想提高安全性可以考虑在置换的基础上再进行移位、代换等多重加密
3. 正是由于古典密码体制面对当今飞速发展的算力时, 在安全性上的急剧下降, 因而采用现代密码体制是时代潮流所向, 也是保护个人隐私和国家安全的必要所需。