

# 密码学第一次实验报告

## 古典密码算法及攻击方法

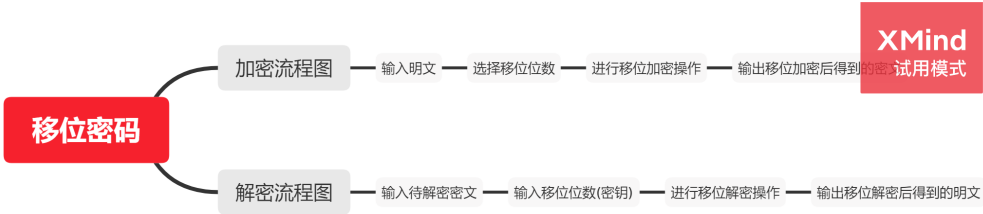
PS: 涉及到的项目代码位置导航, 移位密码的加密和解密位于**移位密码**文件夹内; 单表置换密码的加密解密(自己设计语句作为置换序列首, 后顺序排列)位于**单表置换**文件夹内; 单表置换密码的攻击我分成了两个代码, 首先是对文本中出现字母的统计与文档中提供的相似频率进行匹配, 得到一段置换序列, 该部分代码位于**单表移位替换**文件夹内, 然后是手工对置换序列修改得出明文, 此部分实则为上面单表置换加密解密中去除加密部分的代码, 为了方便操作单独提出来放在**单表置换改**文件夹中

### 1. 实验目的

通过借助C++编程实现移位密码和单表置换密码算法, 借助编程的手段进一步加深对经典密码体制的加密和解密体制, 并通过这两种密码进行攻击, 进一步了解古典密码体制的攻击方法。

### 2. 移位密码

#### ◦ 加密解密流程图



#### ◦ 程序实现结果

```
F:\Data\3-1\3-1\密码学\homework\1\移位密码\Debug\移位密码.exe
请选择加密过程或者解密过程: 0为加密, 1为解密, 2为爆破, 其余数字为退出: 0
请输入明文和偏移量: mimaxuetainanle 456789
加密得到的密文为: hdhvspzovdivigz
1
请输入密文和偏移量: hdhvspzovdivigz 456789
解密得到的明文为: mimaxuetainanle
```

```

请选择加密过程或者解密过程：0为加密，1为解密，2为爆破，其余数字为退出： 2
请输入密文： hdhvspzovdivigz
密文为： hdhvspzovdivigz
移位为： 0 解密为： hdhvspzovdivigz
移位为： 1 解密为： gcguroynuchuhfy
移位为： 2 解密为： fbftqnxmtbgtgex
移位为： 3 解密为： eaespmwlsafsfdw
移位为： 4 解密为： dzdrolvkrzerecv
移位为： 5 解密为： cycqnkujqyqdbu
移位为： 6 解密为： bxbpmjtipxpcat
移位为： 7 解密为： awaolishowbobzs
移位为： 8 解密为： zvznkhrgnvanayr
移位为： 9 解密为： yuymjgqfmuzmzxq
移位为： 10 解密为： txlilfpeltylywp
移位为： 11 解密为： wswkheodksxkxvo
移位为： 12 解密为： vrvjgdncjrwjwun
移位为： 13 解密为： uquifcmbiqivtm
移位为： 14 解密为： tptheblahpuhusl
移位为： 15 解密为： sosgdakzgotgtrk
移位为： 16 解密为： rnrfczjyfnfsqj
移位为： 17 解密为： qmqebyixemreri
移位为： 18 解密为： plpdaxhwdlqdqoh
移位为： 19 解密为： okoczgwvckpcpng
移位为： 20 解密为： njnbyvfubjobomf
移位为： 21 解密为： mimaxuetainanle
移位为： 22 解密为： lhlzwt dszhmzmkd
移位为： 23 解密为： kgkyvscryglyljc
移位为： 24 解密为： jfjxurbqxfkxkib
移位为： 25 解密为： ieiwtqapwejwjha
    
```

#### ○ 程序实现代码

```

#include<iostream>
using namespace std;
int get_length(char* text) {
    int i = 0;
    while (text[i] != '\0') {
        i++;
    }
    return i;
}
class shift_crypt {
private:
    int offset;
public:
    shift_crypt() {
        offset = 0;
    }
    shift_crypt(int offset) {
        this->offset = offset;
    }
    char* shift_encrypt(char* plain, int offset) {
        int real_offset = offset % 26;
        int length = get_length(plain);
        char* cipher = new char[length];
        for (int i = 0; i < length; i++) {
            if (plain[i] >= 65 && plain[i] <= 90) {
                int temp = plain[i] + real_offset;
                if (temp > 90)
                    temp -= 26;
                cipher[i] = (char)temp;
                continue;
            }
            if (plain[i] >= 97 && plain[i] <= 122) {
                int temp = plain[i] + real_offset;
                if (temp > 122)
                    temp -= 26;
                cipher[i] = (char)temp;
            }
        }
    }
}
    
```

```

        temp -= 26;
        cipher[i] = (char)temp;
        continue;
    }
    cipher[i] = plain[i]; //非字母的位默认不加密
}
cipher[length] = '\0';
return cipher;
}

char* shift_decrypt(char* cipher, int offset) {
    int real_offset = offset % 26;
    int length = get_length(cipher);
    char* plain = new char[length];
    for (int i = 0; i < length; i++) {
        if (cipher[i] >= 65 && cipher[i] <= 90) {
            int temp = cipher[i] - real_offset;
            if (temp < 65)
                temp += 26;
            plain[i] = (char)temp;
            continue;
        }
        if (cipher[i] >= 97 && cipher[i] <= 122) {
            int temp = cipher[i] - real_offset;
            if (temp < 97)
                temp += 26;
            plain[i] = (char)temp;
            continue;
        }
        plain[i] = cipher[i]; //非字母的位默认不加密
    }
    plain[length] = '\0';
    return plain;
}

void brute_force(char* cipher) {
    cout << "密文为: " << cipher << endl;
    for (int i = 0; i < 26; i++) {
        cout << "移位为: " << i << "    解密为: " <<
shift_decrypt(cipher, i) << endl;
    }
    cout << endl;
}

};

int main() {
    int flag;
    cout << "请选择加密过程或者解密过程: 0为加密, 1为解密, 2为爆破, 其余数字为退出: ";
    while (true) {
        cin >> flag;
        if (flag == 0) {
            char* plaintext = new char[100];
            int offset;
            cout << "请输入明文和偏移量: ";
            cin >> plaintext >> offset;
            shift_crypt encrypt(offset);
            cout << "加密得到的密文为: " <<
encrypt.shift_encrypt(plaintext, offset) << endl;
        }
        else if (flag == 1) {

```

```

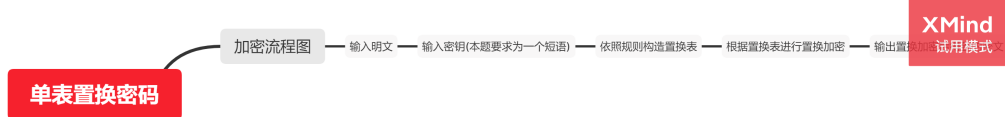
char* ciphertext = new char[100];
int offset;
cout << "请输入密文和偏移量: ";
cin >> ciphertext >> offset;
shift_crypt decrypt(offset);
cout << "解密得到的明文为: " <<
decrypt.shift_decrypt(ciphertext, offset) << endl;
}
else if (flag == 2) {
char* ciphertext = new char[100];
cout << "请输入密文: ";
cin >> ciphertext;
shift_crypt decrypt;
decrypt.brute_force(ciphertext);
}
else {
break;
}
}
return 0;
}

```

移位密码加密解密以及暴力破解的代码相对而言较为清晰易懂，主要是将每一位char的ascii进行增加或减少，如果因为移位造成超出或者不足字母的ascii范围，对其进行减或加26操作，爆破即offset从0-25依次进行解密操作

### 3. 单表置换密码

- 加密流程图



- 程序实现结果

```

Microsoft Visual Studio 调试控制台
输入密钥，如果密钥长度大于26位，则只取前26位
zhihuanmimabiao

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
z h i u a n m b o c d e f g j k l p q r s t v w x y
输入明文： zhe shi yi ge ming wen
您的密文为： yba qbo xo ma fogm vag

您的密文为： yba qbo xo ma fogm vag!
您的明文为： zhe shi yi ge ming wen!

F:\Data\3-1\3-1\密码学\homework\1\单表置换\Debug\单表置换.exe (进程 23768) 已退出，代码为 0。
按任意键关闭此窗口。 . . .

```

- 程序实现代码

```

#include<iostream>
#include<map>
#include<string>
#include<stdio.h>
using namespace std;
int get_length(char* text) {
int i = 0;
while (text[i] != '\0') {

```

```

        i++;
    }
    return i;
}
map<char, char> list;
int* a = new int[26];
int main() {
    char* text = new char[100];
    for (int i = 0; i < 26; i++) {
        a[i] = 0;
    }
    cout << "输入密钥, 如果密钥长度大于26位, 则只取前26位" << endl;
    cin >> text;
    if (get_length(text) > 26) {
        text[26] = '\0';
    }
    //全部改为小写 去重
    int t = 0;
    for (int i = 0; i < get_length(text); i++) {
        text[i] = tolower(text[i]);
        if (text[i] < 97 || text[i] > 122) {
            continue;
        }
        if (a[(int)(text[i] - 'a')]) {
            continue;
        }
        else {
            a[(int)(text[i] - 'a')] = 1;
            list[(char)('a' + t)] = text[i];
            t++;
        }
    }
    for (int j = t; j < 26; j++) {
        char temp = '\0';
        for (int k = 0; k < 26; k++) {
            if (a[k] == 0) {
                a[k] = 1;
                temp = (char)('a' + k);
                break;
            }
        }
        list[(char)('a' + j)] = temp;
    }
    cout << endl << "替换表为:  " << endl;
    for (int j = 0; j < 26; j++) {
        cout << (char)('a' + j) << " ";
    }
    cout << endl;
    for (int j = 0; j < 26; j++) {
        cout << list[(char)('a' + j)] << " ";
    }
    cout << endl;
    char* plaintext = new char[100];
    cout << "输入明文:  ";
    cin.get();
    cin.getline(plaintext, 100);
    int len = get_length(plaintext);
    char* ciphertext = new char[len];

```

```

for (int j = 0; j < len; j++) {
    plaintext[j] = tolower(plaintext[j]);
    if (plaintext[j] < 97 || plaintext[j] > 122) {
        ciphertext[j] = plaintext[j];
    }
    else {
        ciphertext[j] = list.at((char)(plaintext[j]));
    }
}
ciphertext[len] = '\0';
cout << "您的密文为:  " << ciphertext << endl;
cout << endl << endl;
cout << "您的密文为:  ";
cin.getline(ciphertext, 100);
len = get_length(ciphertext);
for (int j = 0; j < len; j++) {
    ciphertext[j] = tolower(ciphertext[j]);
    if (ciphertext[j] < 97 || ciphertext[j] > 122) {
        plaintext[j] = ciphertext[j];
    }
    else {
        for (std::map<char, char>::iterator it = list.begin(); it !=
list.end(); it++) {
            if (it->second == ciphertext[j]) {
                plaintext[j] = it->first;
            }
        }
    }
}
plaintext[len] = '\0';
cout << "您的明文为:  " << plaintext << endl;
return 0;
}

```

主要借助了C++STL编程中的MAP结构进行操作，默认原字母表为key，置换后为value，因此可以借助at通过key取value，也可以借助iterator迭代器通过value取key（因为构造是无重叠，所以不考虑单value多key的情况）

#### 4. 移位密码的攻击

移位密码肯定不适合穷举攻击，在较大规模的数据下，攻击的思路为：将密文中的每个字母出现的频率进行统计，并且和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，但是因为给出的文本较为较少，所以我们需要在置换过程后进行校正

## 各个字母

### o 统计字母频率

主要统计密文中出现的字母（默认为小写）的频率，进行排序，与提供的近似频率进行匹配比较得到置换序列，二者均为顺序排序。

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				

```
Microsoft Visual Studio 调试控制台
SIC GCBSPNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJH
JBRGGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QRRNEC GNB MBAY HC PCGMTCPD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJ
JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QRRNEC HMH SIC PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRICR S
B ZBNZSIMPJOCDB MBSPMA MF SIC QRRNEC
您的文本中出现的字母频率为：
a 0.0296736
b 0.0830861
c 0.106825
d 0.00890208
e 0.0267062
f 0.0207715
g 0.041543
h 0.0267062
i 0.0534125
j 0.0830861
k 0
l 0
m 0.0860534
n 0.0919881
o 0.00296736
p 0.0682493
q 0.0237389
r 0.0623145
s 0.0979228
t 0.00593472
u 0
v 0.00890208
w 0
x 0.0356083
y 0.0207715
z 0.0148368
置换序列：bvexcyzimloghjnrpsatdkfw
```

## ○ 置换字母

随后借助上面单表置换的代码简单精简后进行破译得到相关信息

```
Microsoft Visual Studio 调试控制台
输入密钥，如果密钥长度大于26位，则只取前26位
bvexcyzimloghjnrpsatdkfw

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
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
您的明文为：
SIC GCBSPNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJH
AY JBRGGZPC 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 PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRI
CR SM ENJB ZBNZSIMPJOCDB MBSPMA MF SIC QRRNEC
您的明文为： the leatsou dsimuep na lsfdticsodhf nr thot iy tsoarpttnac nayisptonia ysip o dinat o ti o dinat m mf peo
ar iy o dirrrnmuf narelge lhoaau na rglh o bof thot the isncnaou perroce loa iauf me selivesew mf the snchtygu selndnea
tr the dostlndoatr na the tsoaroltnia ose ounle the isncnaotis iy the perroce mim the selenves oaw irlos o dirrrnmue idd
iaeat bhi bnrher ti cona gaogthisnkew liatsiu iy the perroce

F:\Data\3-1\3-1\密码学\homework\1\单表置换改\Debug\单表置换改.exe (进程 18240) 已退出，代码为 0。
按任意键关闭此窗口。 . . .
```

```
#include<iostream>
#include<string>
using namespace std;
int get_length(char* text) {
    int i = 0;
    while (text[i] != '\0') {
        i++;
    }
    return i;
}
class letter {
private:
    int num;
    char letter_old;
    char letter_new;
public:
    letter() {
        num = 0;
        letter_old = '\0';
        letter_new = '\0';
    }
    letter(char a) {
        num = 0;
        letter_old = a;
        letter_new = '\0';
    }
};
```

```

    }
    void set_old(char a) {
        letter_old = a;
    }
    void set_num(int a) {
        num = a;
    }
    void num_plus() {
        num++;
    }
    void set_new(char a) {
        letter_new = a;
    }
    char get_old() {
        return letter_old;
    }
    char get_new() {
        return letter_new;
    }
    int get_num() {
        return num;
    }
};

void letter_sort(letter a[]) {
    for (int i = 1; i < 26; i++) {
        for (int j = 0; j < 26 - i; j++) {
            if (a[j].get_num() < a[j + 1].get_num()) {
                char t1 = a[j].get_old();
                a[j].set_old(a[j + 1].get_old());
                a[j + 1].set_old(t1);
                int t2 = a[j].get_num();
                a[j].set_num(a[j + 1].get_num());
                a[j + 1].set_num(t2);
            }
        }
    }
}

int main() {
    char re[27];
    memset(re, '\0', 27);
    letter text_letter[26];
    for (int i = 0; i < 26; i++) {
        text_letter[i].set_old('a' + i);
    }
    cout << "请输入您的文本段:" << endl;
    char* text = new char[1000];
    cin.getline(text, 1000);
    int len = get_length(text);
    double num = 0;
    for (int i = 0; i < len; i++) {
        if (text[i] >= 65 && text[i] <= 90) {
            text_letter[(int)(text[i] - 'A')].num_plus();
            num++;
        }
        if (text[i] >= 97 && text[i] <= 122) {
            text_letter[(int)(text[i] - 'a')].num_plus();
            num++;
        }
    }
}

```



```

    }
    cout << "您的文本中出现的字母频率为: " << endl;
    for (int i = 0; i < 26; i++) {
        text_letter[i].set_old('a' + i);
        cout << (char)('a' + i) << "      " << (double)
(text_letter[i].get_num() / num) << endl;
    }

    char real_freq[26] = {
'e','t','o','i','a','n','s','r','h','l','d','u','c','m','p','y','f','g',
'w','b','v','k','x','j','q','z' };
    letter_sort(text_letter);
    for (int i = 0; i < 26; i++) {
        text_letter[i].set_new(real_freq[i]);
    }
    for (int i = 0; i < 26; i++) {
        re[text_letter[i].get_new() - 'a'] = text_letter[i].get_old();
    }
    re[26] = '\0';
    cout << "置换序列: " << re;
    cout << endl;
    return 0;
}

```

#### o 校正

得到的明文为the leatsou dsimuep na lsfdticsodhf nr thot iy tsoarpnttnac nayisopotnia  
ysip o dinat o ti o dinat m mf peoar iy o dirnmuf narelgse lhooaeu na rglh o bof thot  
the isncnaou perroce loa iauf me selivesew mf the snchtygu selndneatr the  
dostnlndoatr na the tsoaroltnia ose ounle the isncnaotis iy the perroce mim the  
selenves oaw irlos o dirnmue iddiaeat bhi bnrher ti cona gaogthisnkew liatsiu iy the  
perroce, 很明显并不通顺, 很明显, 单个单词o其实应该是a, 所以我们校正置换序列, 将  
o, a所对应的替换密码进行替换, 修正置换序列: nvexcyzimloghjbqurpsatdkfw

得到新的明文为the leotsau dsimuep no lsfdticsadhf nr that iy tsaorpnttnoc noyispatnio  
ysip a dinot a ti a dinot m mf peoar iy a dirnmuf norelgse lhaooeu no rglh a baf that the  
isncnoau perrace lao iouf me selivesew mf the snchtygu selndneotr the dastnlndoatr no  
the tsaoraltnio ase aunle the isncnoatis iy the perrace mim the selenves aow irlas a  
dirnmue iddioeot bhi bnrher ti cano gaogthisnkew liotsiu iy the perrace

我们又看到其中存在iy的存在, 所以应该是改成if, 即和y进行交换, 校正置换序列:  
nvexcfzimloghjbqurpsatdkyw

得到新的明文为: the leotsau dsimuep no lsydticsadhy nr that if tsaorpnttnoc  
nofispatnio fsip a dinot a ti a dinot m my peoar if a dirnmuy norelgse lhaooeu no rglh a  
bay that the isncnoau perrace lao iouy me selivesew my the snchtfgu selndneotr the  
dastnlndoatr no the tsaoraltnio ase aunle the isncnoatis if the perrace mim the selenves  
aow irlas a dirnmue iddioeot bhi bnrher ti cano goagthisnkew liotsiu if the perrace

我们有发现存在nr that的句子, 根据主谓语的关系, 应该是is that, 所以n与i互换, s与r互  
换, 得到新的置换序列: nvexcfzijloghmbqurpsatdkyw

得到新的明文为: the leotrau drnmuep io lrydtncradhy is that nf traospittioc iofnrpatino  
frnp a dniot a tn a dniot m my peaos nf a dnssimuy ioselgre lhaooeu io sglh a bay that  
the nrccioau pessace lao nouy me relnverew my the richtfgu relidieots the dartilidaots io  
the traosaltino are auile the nrccioatnr nf the pessace mnm the releiver aow nslar a  
dnssimue nddnoeot bhn bishes tn caio goagthnrkew lnotrnu nf the pessace

nf明显为of, 所以n和o互换, 得到新的置换序列: nvexcfzijloghbmqupsatdkyw

得到新的明文为：the lentrau dromuep in lrydtocradhy is that of transpittinc inforpation  
frop a doint a to a doint m my peans of a dossimuy inselgre lhanneu in sglh a bay that  
the oricinau pessace lan onuy me reloverew my the richtfgu relidients the dartilidants in  
the transaltion are auile the oricinator of the pessace mom the releiver anw oslar a  
dossimue oddonent bho bishes to cain gnagthorikew lontrou of the pessace

doint改成point, 置换序列：nveqcfzijloghbm xuprsatdkyw

得到新的明文为：the lentrau promued in lryptocraphy is that of transdittinc infordation  
frod a point a to a point m my deans of a possimuy inselgre lhanneu in sglh a bay that  
the oricinau dessace lan onuy me reloverew my the richtfgu relipients the partilipants in  
the transaltion are auile the oricinator of the dessace mom the releiver anw oslar a  
possimue opponnet bho bishes to cain gnagthorikew lontrou of the dessace

inforpation->information, probuem->problem, uentral->central, cryptouraphy->  
cryptography, day->way, unauthoriked->unauthorized。得到新的置换序列为：  
nhgdcfeijlwaqbm xuprsztvkyo

得到新的明文为：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 opponet who wishes to gain unauthorized control of the message

经过简单审查，该英文能过读顺，破译成功，最终的置换表为

Microsoft Visual Studio 调试控制台

输入密钥，如果密钥长度大于26位，则只取前26位  
nhgdcfeijlwaqbm xuprsztvkyo

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