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

学号：　2013484

姓名：　张世伟

年级：　2020 级

专业：　信息安全-法学

2022 年 11 月 14 日

摘　要

本文以 C++ 语言，对古典密码算法里面的移位密码和单表置换密码进行了加解密和破解程序的讲解

关键词: C++，古典密码，移位密码，单表置换加密，字母频率统计攻击

目　录

第 1 节　移位密码的加解密

1.1　移位密码的加解密流程

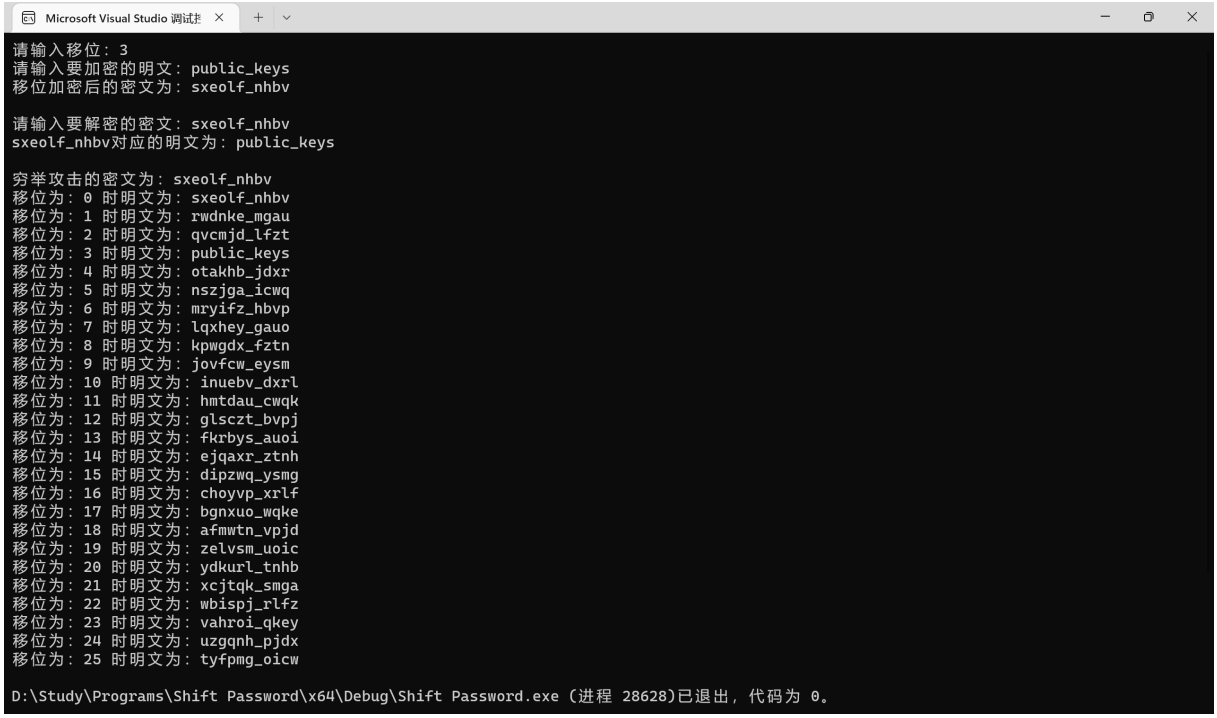
图 1.1: 移位密码的加解密流程

1.2　移位密码程序代码

```
#include <iostream>
using namespace std;

class shift_crypt
{
        private:
        int offset;//移位
        char* ciphertext;//密文
        public:
        shift_crypt()
        {
                offset = 0;
        }
        shift_crypt(int offset)
        {
                this->offset = offset;
        }
        char* get_ciphertext()
        {
                return this->ciphertext;
        }
        void shift_encrypt(char* plaintext) //加密
        {
                offset = offset % 26;
                //cout << "offset: " << offset << endl;
                int len = strlen(plaintext);
                //cout << "len: " << len << endl;
                this->ciphertext = new char[len];
                for (int i = 0; i < len; i++)
                {
                        //cout << "plaintext[i]: " << plaintext[i] << endl;
                        if (plaintext[i] >= 'a' && plaintext[i] <= 'z' || plaintext[i] >= 'A' && plaintext[i] <= 'Z')
                        {
                                char temp = plaintext[i] + offset;
                                //cout << "temp: " << temp << endl;
                                if (temp > 'Z' && plaintext[i]<='Z' || temp > 'z')
                                this->ciphertext[i] = temp - 26;
                                else
                                this->ciphertext[i] = temp;
                        }
                        else
                        {
                                this->ciphertext[i] = plaintext[i];
                        }
                        //cout << "ciphertext[i]: " << ciphertext[i] << endl;
```

```
                }
                this->ciphertext[len] = '\0';
        }
        char* shift_decrypt(char* ciphertext, int offset)//解密
        {
                int len = strlen(ciphertext);
                char* plaintext = new char[len];
                for (int i = 0; i < len; i++)
                {
                        if (ciphertext[i] >= 'a' && ciphertext[i] <= 'z' || ciphertext[i] >= 'A' && ciphertext[i] <= 'Z')
                        {
                                char temp = ciphertext[i] - offset;
                                if (temp < 'a' && ciphertext[i] >= 'a' || temp < 'A')
                                plaintext[i] = temp + 26;
                                else
                                plaintext[i] = temp;
                        }
                        else
                        plaintext[i] = ciphertext[i];
                }
                plaintext[len] = '\0';
                return plaintext;
        }
        void exhaust_decrypt(char* ciphertext)//穷举
        {
                int offset;
                char* plaintext;
                for (offset = 0; offset <= 25; offset++)
                {
                        plaintext = shift_decrypt(ciphertext, offset);
                        cout << "移位为：" << offset << " 时明文为：" << plaintext << endl;
                }
        }
};
int main()
{
        char* plaintext = new char[1024];
        char* ciphertext = new char[1024];
        int offset;
        cout << "请输入移位：";
        cin >> offset;
        cout << "请输入要加密的明文：";
        cin >> plaintext;

        shift_crypt pro = shift_crypt(offset);
        pro.shift_encrypt(plaintext);
        char* temp1 = pro.get_ciphertext();
        cout << "移位加密后的密文为：" << temp1 << endl << endl;
        cout << "请输入要解密的密文：";
        cin >> ciphertext;
        char* temp2 = pro.shift_decrypt(ciphertext, offset);
        cout << ciphertext << "对应的明文为：" << temp2 << endl << endl;

        cout << "穷举攻击的密文为："<<temp1 << endl;
        pro.exhaust_decrypt(temp1);
        return 0;
}
```

1.3 程序运行结果



```
请输入移位: 3
请输入要加密的明文: public_keys
移位加密后的密文为: sxeolf_nhbv

请输入要解密的密文: sxeolf_nhbv
sxeolf_nhbv对应的明文为: public_keys

穷举攻击的密文为: sxeolf_nhbv
移位为: 0 时明文为: sxeolf_nhbv
移位为: 1 时明文为: rwdnke_mgau
移位为: 2 时明文为: qvcmjd_lfzt
移位为: 3 时明文为: public_keys
移位为: 4 时明文为: otakhb_jdxr
移位为: 5 时明文为: nszjga_icwq
移位为: 6 时明文为: mryifz_hbvp
移位为: 7 时明文为: lqxhey_gauo
移位为: 8 时明文为: kpwgdx_fztn
移位为: 9 时明文为: jovfcw_eysm
移位为: 10 时明文为: inuebv_dxrl
移位为: 11 时明文为: hmtdau_cwqk
移位为: 12 时明文为: glsczt_bvpj
移位为: 13 时明文为: fkrbys_auoi
移位为: 14 时明文为: ejqaxr_ztnh
移位为: 15 时明文为: dipzwq_ysmg
移位为: 16 时明文为: choyvp_xrlf
移位为: 17 时明文为: bgnxuo_wqke
移位为: 18 时明文为: afmwtn_vpjd
移位为: 19 时明文为: zelvsm_uoic
移位为: 20 时明文为: ydkurl_tnhb
移位为: 21 时明文为: xcjtqk_smga
移位为: 22 时明文为: wbispj_rlfz
移位为: 23 时明文为: vahroi_qkey
移位为: 24 时明文为: uzgqnh_pjdx
移位为: 25 时明文为: tyfpmg_oicw

D:\Study\Programs\Shift Password\x64\Debug\Shift Password.exe (进程 28628)已退出, 代码为 0。
```

图 1.2: 移位密码加解密程序运行结果

第 2 节　单表置换密码的加解密

2.1　单表置换密码的加解密流程

图 2.1: 单表置换密码的加解密流程

2.2　单表置换密码程序代码

```
#include <iostream>
#include<string.h>
using namespace std;

//处理输入的密钥
int* build_table(char* inputtext)
{
        int len = strlen(inputtext);
        //输入的密钥长度大于26，则只取前26个
        if (len > 26)
        {
                inputtext[26] = '\0';
                len = 26;
        }
        int* temp = new int[26];
        int j = 0;
        temp[j] = -1;
        for (int i = 0; i < len; i++)
        {
                if (inputtext[i] >= 'A' && inputtext[i] <= 'Z')
                {
                        inputtext[i] = inputtext[i] - 'A' + 'a';
                }
                if (inputtext[i] >= 'a' && inputtext[i] <= 'z')
                {
                        bool t = true;
                        int n = inputtext[i] - 'a';
                        for (int k = 0; k <= j; k++)
                        {
                                if (temp[k] == n)
                                {
                                        t = false;
                                        break;
                                }
                        }
                        if (t)
                        {
                                temp[j] = n;
                                j++;
                        }
                }
        }
        for (int i = 0; i < 26; i++)
        {
                bool t = true;
                for (int k = 0; k < j; k++)
```

```
                {
                        if (temp[k] == i)
                        {
                                t = false;
                                break;
                        }
                }
                if (!t)
                        continue;
                else
                {
                        temp[j] = i;
                        j++;
                }
        }
        //for (int i = 0; i < 26; i++)
        //{
        //    cout << temp[i] << " ";
        //}
        return temp;
}

//解密时使用
int retnum(int* replacetable, int num)
{
        for (int i = 0; i < 26; i++)
        {
                if (replacetable[i] == num)
                {
                        return i;
                }
        }
}

class table_replace_crypt
{
private:
        int* replacetable = new int[26];//置换表
public:
        table_replace_crypt()
        {
                for (int i = 0; i < 26; i++)
                {
                        replacetable[i] = i;
                }
        }
        table_replace_crypt(char* input)
        {
                replacetable = build_table(input);
        }
        int* getreplacetable()
        {
                return replacetable;
        }
        char* table_replace_encrypt(char*plaintext)//加密
        {
                int len = strlen(plaintext);
                char* ciphertext = new char[len];
                for (int i = 0; i < len; i++)
                {
                        if (plaintext[i] >= 'a' && plaintext[i] <= 'z')
                        {
                                int n = plaintext[i] - 'a';
                                ciphertext[i] = plaintext[i] + replacetable[n] - n;
                        }
                        else if (plaintext[i] >= 'A' && plaintext[i] <= 'Z')
```

```
                        {
                                int n = plaintext[i] - 'A';
                                ciphertext[i] = plaintext[i] + replacetable[n] - n;
                        }
                        else
                        ciphertext[i] = plaintext[i];
                }
                ciphertext[len] = '\0';
                return ciphertext;
        }
        char* table_replace_decrypt(char* ciphertext)//解密
        {
                int len = strlen(ciphertext);
                char* plaintext = new char[len];
                for (int i = 0; i < len; i++)
                {
                        if (ciphertext[i] >= 'a' && ciphertext[i] <= 'z')
                        {
                                int n = retnum(replacetable, ciphertext[i] - 'a');
                                plaintext[i] = 'a' + n;
                        }
                        else if (ciphertext[i] >= 'A' && ciphertext[i] <= 'Z')
                        {
                                int n = retnum(replacetable, ciphertext[i] - 'A');
                                plaintext[i] = 'A' + n;
                        }
                        else
                        plaintext[i] = ciphertext[i];
                }
                plaintext[len] = '\0';
                return plaintext;
        }
};
int main()
{

        cout << "请输入密钥: ";
        char* input = new char[24];
        cin.getline(input, 20);

        table_replace_crypt pro = table_replace_crypt(input);
        cout << "置换表为: " << endl;
        for (int i = 0; i < 26; i++)
        {
                cout << char('a' + i) << " ";
        }
        cout << endl;
        int* temp = pro.getreplacetable();
        for (int i = 0; i < 26; i++)
        {
                cout << char('a' + temp[i]) << " ";
        }
        cout << endl << endl;

        cout << "请输入要加密的明文: ";
        char* plaintext = new char[24];
        char* temp1;
        cin.getline(plaintext, 20);
        temp1 = pro.table_replace_encrypt(plaintext);
        cout << "明文" << plaintext << "加密后的密文为: " << temp1 << endl << endl;

        cout << "请输入要解密的密文: ";
        char* ciphertext = new char[24];
        char* temp2;
        cin.getline(ciphertext, 20);
        temp2 = pro.table_replace_decrypt(ciphertext);
```

```
        cout << "密文" << ciphertext << "解密后的明文为：" << temp2 << endl << endl;
        return 0;
}
```

2.3 程序运行结果

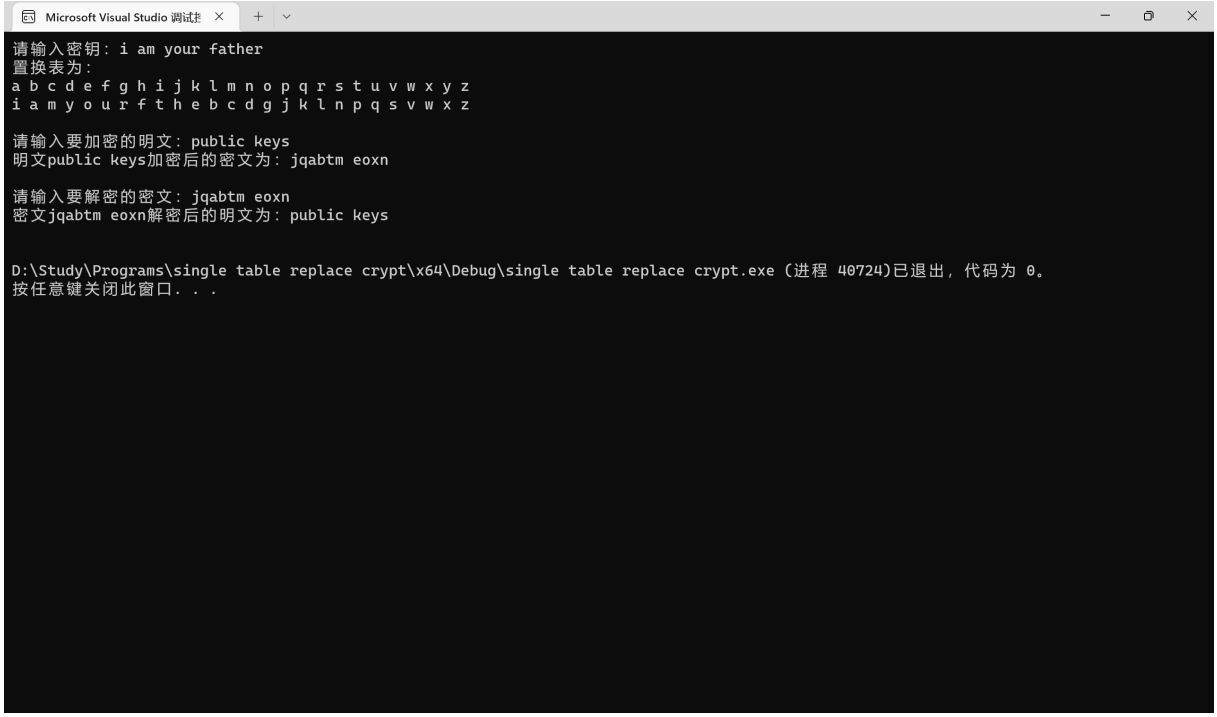


图 2.2: 单表置换密码加解密程序运行结果

第 3 节　字母频率统计攻击方法

3.1　字母频率统计攻击方法流程

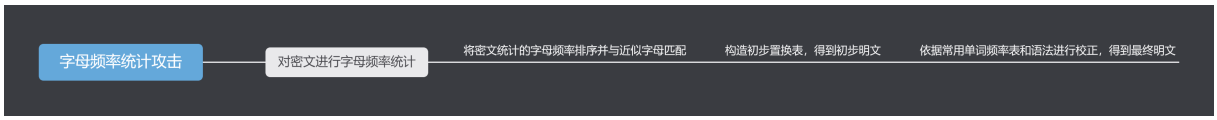


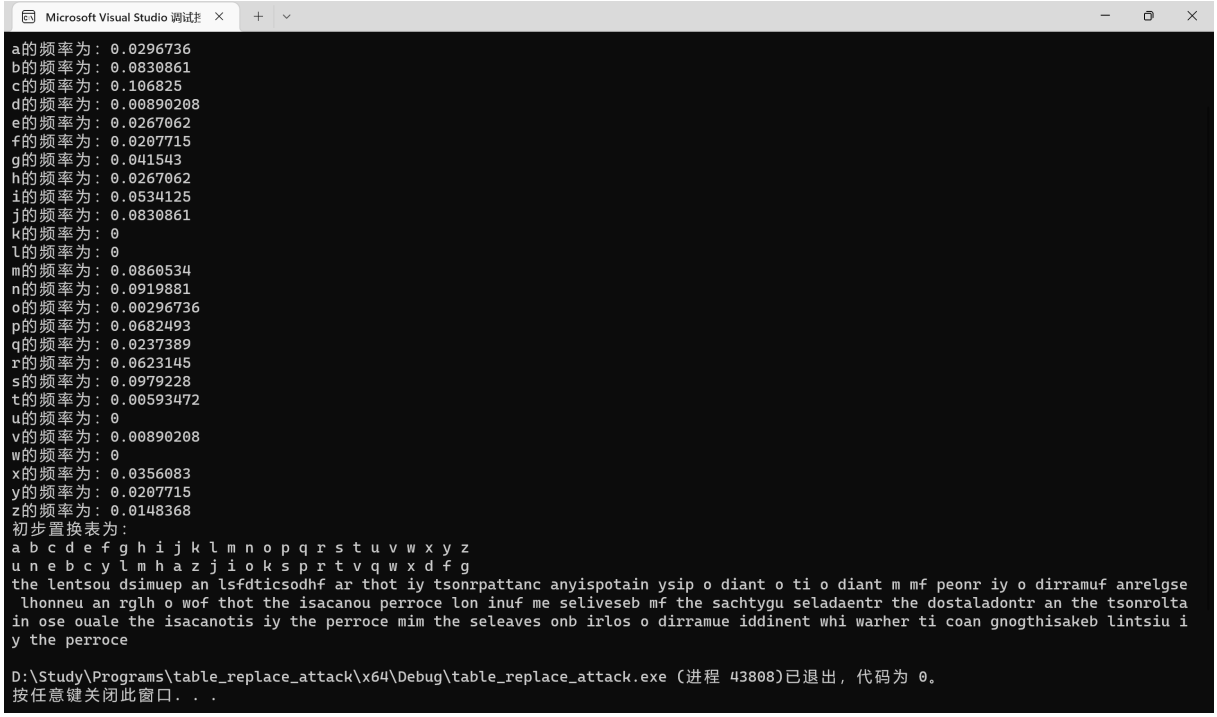
图 3.1: 字母频率统计攻击方法流程

3.2　字母频率统计攻击方法程序代码

```
#include <iostream>
using namespace std;
int main()
{
        char* ciphertext = new char[1024];
        cin.getline(ciphertext, 1024);
        int len = strlen(ciphertext);

        //统计字母频率
        int num[26], total = 0;
        for (int i = 0; i < 26; i++)
        num[i] = 0;
        for (int i = 0; i < len; i++)
        {
                if (ciphertext[i] >= 'a' && ciphertext[i] <= 'z')
                {
                        int n = ciphertext[i] - 'a';
                        num[n]++;
                        total++;
                }
                else if (ciphertext[i] >= 'A' && ciphertext[i] <= 'Z')
                {
                        int n = ciphertext[i] - 'A';
                        num[n]++;
                        total++;
                }
        }

        int n[26][2];
        for (int i = 0; i < 26; i++)
        {
                n[i][0] = i;
                cout << char('a' + i) << "的频率为：" << float(num[i]) / total << endl;
        }

        //将字母按频率大小排序
        for (int i = 0; i < 26; i++)
        {
                for (int j = i; j < 26; j++)
                {
                        if (num[i] < num[j])
                        {
                                swap(num[i], num[j]);
                                int temp = n[i][0];
                                n[i][0] = n[j][0];
                                n[j][0] = temp;
                        }
                }
```

```
        }
        //for (int i = 0; i < 26; i++)
        //{
                //    //cout << num[i] << endl;
                //    cout << char(n[i][0] + 'a') << endl;
                //}

        //创建n[26][2]二维数组，将排好序的字母与近似字母频率一一对应起来
        //其中n[i][0]储存排好序的字母，n[i][1]储存对应的近似字母
        //然后创建temp[26]创建置换表，将字母按顺序排好
        string str = "etoiansrhlducmpyfgwbvkxjqz";
        for (int i = 0; i < 26; i++)
        {
                n[i][1] = int(str[i] - 'a');
                //cout << n[i][1] << endl;
        }
        int temp[26];
        for (int i = 0; i < 26; i++)
        {
                temp[n[i][0]] = n[i][1];
        }

        //校正
        swap(temp['n' - 'a'], temp['j' - 'a']);
        swap(temp['y' - 'a'], temp['f' - 'a']);
        swap(temp['d' - 'a'], temp['x' - 'a']);
        swap(temp['m' - 'a'], temp['j' - 'a']);
        swap(temp['p' - 'a'], temp['r' - 'a']);
        swap(temp['q' - 'a'], temp['h' - 'a']);
        swap(temp['z' - 'a'], temp['a' - 'a']);
        swap(temp['e' - 'a'], temp['g' - 'a']);
        swap(temp['h' - 'a'], temp['x' - 'a']);
        swap(temp['a' - 'a'], temp['e' - 'a']);
        swap(temp['o' - 'a'], temp['k' - 'a']);

        cout << "置换表为：" << endl;
        for (int i = 0; i < 26; i++)
        {
                cout << char(i + 'a') << " ";
        }
        cout << endl;
        for (int i = 0; i < 26; i++)
        {
                cout << char(temp[i] + 'a') << " ";
        }
        cout << endl;

        for (int i = 0; i < len; i++)
        {
                if (ciphertext[i] >= 'A' && ciphertext[i] <= 'Z')
                ciphertext[i] = char(temp[ciphertext[i] - 'A'] + 'a');
                else if (ciphertext[i] >= 'a' && ciphertext[i] <= 'z')
                ciphertext[i] = char(temp[ciphertext[i] - 'a'] + 'a');
        }
        cout << ciphertext << endl;
        return 0;
}
```

3.3 程序运行结果



```
a的频率为: 0.0296736
b的频率为: 0.0830861
c的频率为: 0.106825
d的频率为: 0.00890208
e的频率为: 0.0267062
f的频率为: 0.0207715
g的频率为: 0.041543
h的频率为: 0.0267062
i的频率为: 0.0534125
j的频率为: 0.0830861
k的频率为: 0
l的频率为: 0
m的频率为: 0.0860534
n的频率为: 0.0919881
o的频率为: 0.00296736
p的频率为: 0.0682493
q的频率为: 0.0237389
r的频率为: 0.0623145
s的频率为: 0.0979228
t的频率为: 0.00593472
u的频率为: 0
v的频率为: 0.00890208
w的频率为: 0
x的频率为: 0.0356083
y的频率为: 0.0207715
z的频率为: 0.0148368
初步置换表为:
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
u n e b c y l m h a z j i o k s p r t v q w x d f g
the lentsou dsimuep an lsfdticsodhf ar thot iy tsonrpattanc anyispotain ysip o diant o ti o diant m mf peonr iy o dirramuf anrelgse
 lhonneu an rglh o wof thot the isacanou perroce lon inuf me seliveseb mf the sachtygu seladaentr the dostaladontr an the tsonrolta
in ose ouale the isacanotis iy the perroce mim the seleaves onb irlos o dirramue iddinent whi warher ti coan gnogthisakeb lintsiu i
y the perroce

D:\Study\Programs\table_replace_attack\x64\Debug\table_replace_attack.exe (进程 43808)已退出, 代码为 0。
按任意键关闭此窗口. . .
```

图 3.2: 字母频率统计攻击程序运行结果 01

3.3.1 校正

可以发现只依据单个字母的频率攻击得到的明文语义并不通顺，根据常用单词的频率

将置换表中 o、a 互换

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

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

the lentsau dsimuep on lsfdticsadhf or that iy tsanrpottonc onyispatoin ysip a diont a ti a diont m mf peanr iy a dirromuf onrelgse lhanneu on rglh a waf that the isoconau perrace lan inuf me seliveseb mf the sochtygu selodoentr the dastolodantr on the tsanraltoin ase auole the isoconatis iy the perrace mim the seleoves anb irlas a dirromue iddinent whi worher ti caon gnagthisokeb lintsiu iy the perrace

发现得到的明文中有"iy"，应该是"if"，将置换表中 y、f 互换

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

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

the lentsau dsimuep on lsydticsadhy or that if tsanrpottonc onfispatoin fsip a diont a ti a diont m my peanr if a dirromuy onrelgse lhanneu on rglh a way that the isoconau perrace lan inuy me seliveseb my the sochtfgu selodoentr the dastolodantr on the tsanraltoin ase auole the isoconatis if the perrace mim the seleoves anb irlas a dirromue iddinent whi worher ti caon gnagthisokeb lintsiu if the perrace

发现得到的明文中有"anb"，应该是"and"，将置换表中 b、d 互换

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

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

the lentsau bsimuep on lsybticsabhy or that if tsanrpottonc onfispatoin fsip a biont a ti a biont m my peanr if a birromuy onrelgse lhanneu on rglh a way that the isoconau perrace lan inuy me selivesed my the sochtfgu seloboentr the bastolobantr on the tsanraltoin ase auole the isoconatis if the perrace mim the seleoves and irlas a birromue ibbinent whi worher ti caon gnagthisoked lintsiu if the perrace

发现得到的明文中有"ir"，应该是"or"，将置换表中 i、o 互换

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

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

the lentsau bsomuep in lsybtocsabhy ir that of tsanrpittinc infospation fsop a boint a to a boint m my peanr of a borrimuy inrelgse lhanneu in rglh a way that the osicinau perrace lan onuy me selovesed my the sichtfgu selibientr the bastilibantr in the tsanraltion ase auile the

osicinatos of the perrace mom the seleives and orlas a borrimue obbonent who wirher to cain gnagthosiked lontsou of the perrace

发现得到的明文中有"frop"，应该是"from"，将置换表中 p、m 互换
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
u n e d c f l p h i z j o a k r m s t v q w x b y g
the lentrau bropuem in lrybtocrabhy is that of transmittinc information from a boint a to a boint p py means of a bossipuy inselgre lhanneu in sglh a way that the oricinau messace lan onuy pe relovered py the richtfgu relibients the bartilibants in the transaltion are auile the oricinator of the messace pop the releiver and oslar a bossipue obbonent who wishes to cain gnagthoriked lontrou of the messace

发现得到的明文中有"py"，应该是"by"，将置换表中 b、p 互换
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
g n e d l f c b h i z j o a k r m s t v q w x p y u
the centrag probgem in cryptolraphy is that of transmittinl information from a point a to a point b by means of a possibgy insecure channeg in such a way that the orilinag messale can ongy be recovered by the rilhtfug recipients the participants in the transaction are agice the orilinator of the messale bob the receiver and oscar a possibge opponent who wishes to lain unauthoriked controg of the messale

发现得到的明文中有"messale"，应该是"message"，将置换表中 l、g 互换
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
l n e d g f c b h i z j o a k r m s t v q w x p y u
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 unauthoriked control of the message

发现得到的明文中有"unauthoriked"，应该是"unauthorized"，将置换表中 k、z 互换，得到语义通顺的明文，攻击成功。
置换表：
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
l n e d g f c b h i k j o a z r m s t v q w x p y u
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

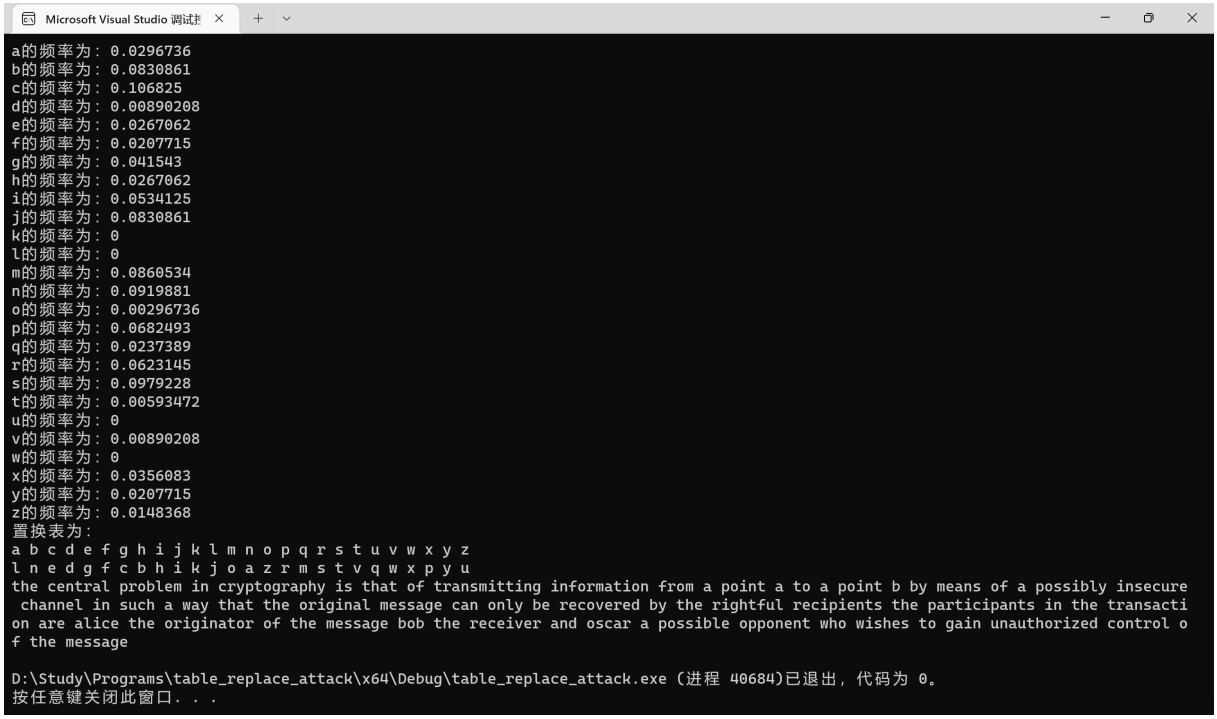


图 3.3: 字母频率统计攻击程序运行结果 02