**实验一 古典密码算法及攻击方法**

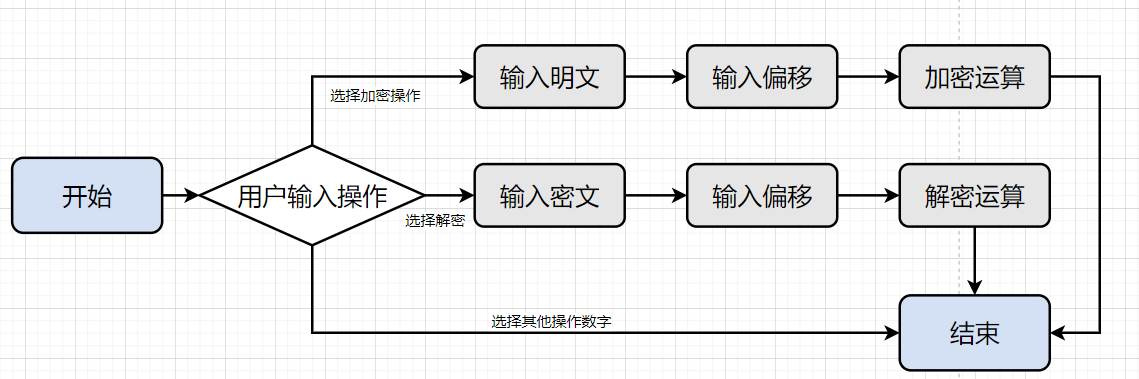
姓名：范毓哲 学号：1910378

1. **实验目的**

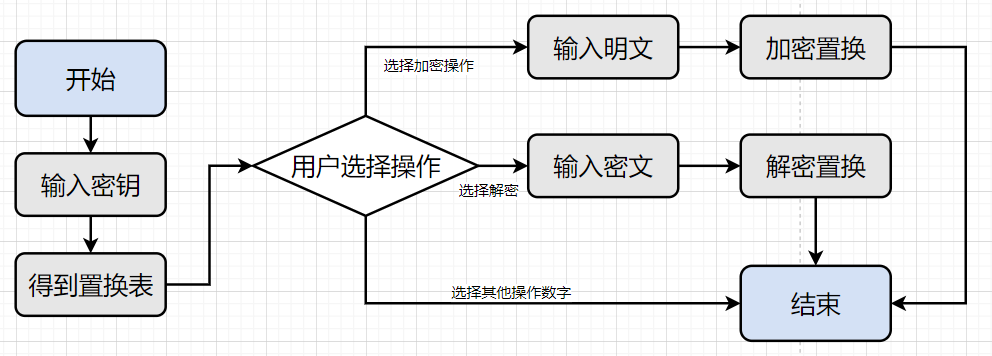
通过C＋＋编程实现移位密码和单表置换密码算法，加深对经典密码体制的了解。并通过对这两种密码实施攻击，了解对古典密码体制的攻击方法。

1. **设计流程**

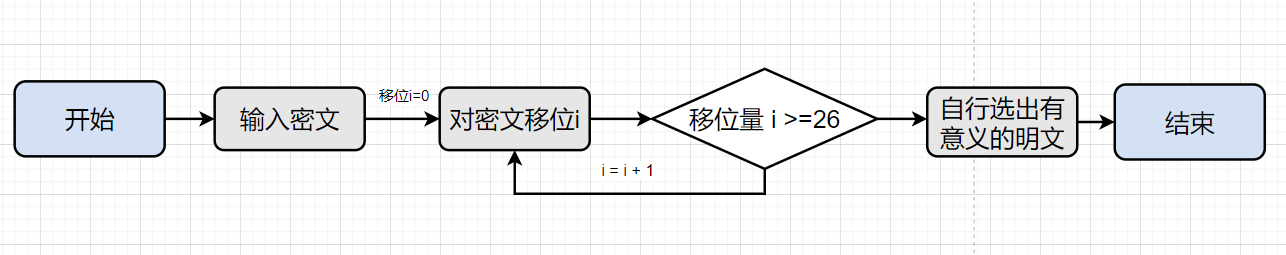
移位密码的加解密：



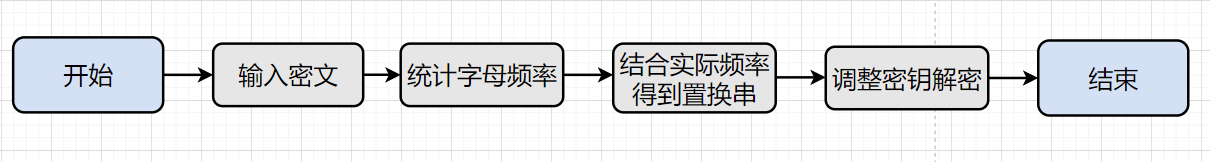
单表置换密码的加解密：



移位密码的攻击：



单表置换密码的攻击：



1. **实验环境**

Windows10操作系统+VS2019(VC17)。

同级目录下，Ccaesar.cpp为移位密码的实现，Ctable.cpp为单表置换密码，tableCrack.cpp为单表置换密码的密钥破解，对应的exe分别为其可执行程序。

1. **程序实现**

1. 移位密码

移位密码的实现较为简单，其加密、解密和攻击破解的代码如下：

#include<string>

#include<iostream>

**using** **namespace** std;

**char** res[100];    //记录程序结果

**void** JieMi(**char** \*txt, **int** offset, **int** tag)    //加解密函数，tag为1时加密，-1解密

{

**int** real\_offset = (offset % 26)\*tag;     //计算实际偏移，范围0-25，加密加，解密减

**int** length = strlen(txt);

**for** (**int** i = 0; i < length; i++)

    {

**if** (txt[i] >= 65 && txt[i] <= 90)     //区分大写和小写的情况，只处理字母

        {

**int** temp = txt[i] + real\_offset;

**if** (temp < 65) temp += 26;

**if** (temp > 90) temp -= 26;

            res[i] = (**char**)temp; **continue**;

        }

**if** (txt[i] >= 97 && txt[i] <= 122)

        {

**int** temp = txt[i] + real\_offset;

**if** (temp < 97) temp += 26;

**if** (temp > 122) temp -= 26;

            res[i] = (**char**)temp; **continue**;

        }

        res[i] = txt[i];

    }

    res[length] = '\0';     //结束字符

    cout << "结果为： " << res<< endl;

}

**int** main()

{

**int** flag, offset = 0;

    cout << "请输入一个数字，以选择您要进行的操作。" << endl;

    cout << "(操作提示：0加密，1解密，2攻击，其余退出)" << endl;

**while** (1)

    {

**char** txt[100]; cout << "选择操作数： "; cin >> flag;

**switch** (flag)

        {

**case** 0: cout << "【加密】请输入明文和偏移量： "; cin >> txt >> offset; JieMi(txt, offset, 1); **break**;

**case** 1: cout << "【解密】请输入密文和偏移量： "; cin >> txt >> offset; JieMi(txt, offset, -1); **break**;

**case** 2: cout << "【攻击】请输入待破解的密文： ";

cin >> txt; cout << "可能的明文情况如下。" << endl;

**for** (**int** i = 0; i < 26; i++)   //攻击时尝试26种可能即可

           {cout << "移位为  " << i << "  时的明文  "; JieMi(txt, i, -1);}**break**;

**default**: **return** 0;

        }

    }

**return** 0;

}

2. 单表置换密码

首先实现的是单表置换密码的加解密，程序实现如下：

#include<iostream>

#include<map>

#include<string>

#include<stdio.h>

**using** **namespace** std;

map<**char**, **char**> p2cList,c2pList;  //两个字典，分别用于加密和解密

**bool** isk[26];**char** res[1000];  //isk用于在转化置换表时，标记密钥字母是否出现过

**void** getKey(**char** \* CKey)  //得到导出表，CKey为用户输入的密钥串

{

**int** t = 0;

**for** (**int** i = 0; i < strlen(CKey); i++)  //密钥串的部分，依次对应进表中

    {

        CKey[i] = tolower(CKey[i]);

**if** (CKey[i] >= 97 && CKey[i] <= 122 && !isk[CKey[i] - 'a'])

        {

            isk[CKey[i] - 'a'] = 1;

            p2cList[(**char**)('a' + t)] = CKey[i];  //在两个List当中添加映射

            c2pList[CKey[i]] = (**char**)('a' + t); t++;

        }

    }

**for** (**int** j = t; j < 26; j++)   //密钥串之外的部分，按顺序与剩下的字母对应

    {

**char** temp = '\0';

**for** (**int** k = 0; k < 26; k++)

        {  **if** (isk[k] == 0) { isk[k] = 1;temp = (**char**)('a' + k);**break**;} }

        p2cList[(**char**)('a' + j)] = temp;c2pList[temp] = (**char**)('a' + j);

    }

    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 <<p2cList[(**char**)('a'+j)]<<" "; cout << endl;

}

**void** getRes(**char**\* inTxt, **int** tag)  //加解密操作，inTxt为用户输入的明文或密文

{   //tag代表着加密或解密的操作，为0加密，为1解密

**int** len = strlen(inTxt);

**for** (**int** j = 0; j < len; j++)

    {

        inTxt[j] = tolower(inTxt[j]);

**if** (inTxt[j] < 97 || inTxt[j] > 122) res[j] = inTxt[j];

**else**

        {  //需要操作的字符是字母，则在映射表中进行查找置换

**if** (tag == 0) { res[j] = p2cList.at((**char**)(inTxt[j])); }

**else** { res[j] = c2pList.at((**char**)(inTxt[j])); }

        }

    }  res[len] = '\0';cout << "结果： " << res << endl<<endl;

}

**int** main()

{

    cout << "请输入一个数字，以选择您要进行的操作。" << endl;

    cout << "(操作提示：0加密，1解密，其余退出。两种操作均需先输入密钥)" << endl;

**while** (1)

    {

**int** flag = 0; cout << "选择操作数： "; cin >> flag;

**if**(flag>1) { cout << "【退出】程序结束。" << endl; **return** 0; }

**char** CKey[100]; **for** (**int** i = 0; i < 26; i++) isk[i] = 0;

        //这里一开始用成memset了，导致程序只能正常运行一次，需要注意

        cout << "请输入您的密钥： "; cin.ignore(); cin.getline(CKey, 100);

**if** (strlen(CKey) > 26) CKey[26] = '\0';

        getKey(CKey);//将密钥转换为密钥表

**char** plaintext[1000]; **char** ciphertext[1000];  //明密文

**if** (flag == 0)

        {

            cout << "【加密】请输入需加密的明文： ";

            cin.getline(plaintext, 1000);   getRes(plaintext, 0);

        }

**else** **if**(flag == 1)

        {

            cout << "【解密】请输入需解密的密文： ";

            cin.getline(ciphertext, 1000);   getRes(ciphertext, 1);

        }

    }

**return** 0;

}

实现单表置换的攻击破解。这里不能再使用穷举法，而需要利用字母频率和语言特性、结合一定的统计规律与猜测进行实现：

#include<iostream>

#include<string>

#include<algorithm>

**using** **namespace** std;

**struct** sig2freq{**char** sig='a'; **int** num=0;}txtList[26];   //结构体：字符-出现次数

**bool** cmp(sig2freq a, sig2freq b)

{  **return** a.num > b.num;  }  //结构体比较函数

**int** main()

{

**char** re[27];  //用来存储密钥结果

cout << "请输入待破译的密文串： " << endl;**char** text[1000];

cin.getline(text, 1000);   **int** len = strlen(text);

**double** num = 0;//所有字母出现的总次数

**for** (**int** i = 0; i < len; i++)

    {  //统计用户输入的密文串当中各字母出现次数

        text[i] = tolower(text[i]);

**if** (text[i] >= 97 && text[i] <= 122)

        {num++; txtList[text[i] - 'a'].num++;}

    }

    cout << "\n这段文本的字符频率信息统计如下： " << endl;

**for** (**int** i = 0; i < 26; i++)

    {

        txtList[i].sig='a' + i;

        cout << (**char**)('a' + i) << "   " << (**double**)(txtList[i].num/ num) << endl;

    }

**char** 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' };  //实际的频率排序

    sort(txtList, txtList + 26,cmp);  //用户密文的频率排序

**for** (**int** i = 0; i < 26; i++)

re[freq[i] - 'a'] = txtList[i].sig;

//将密文字母频率与实际字母频率相对应，作为置换密钥

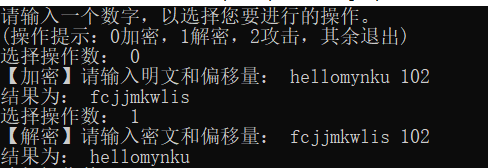
    re[26] = '\0';cout << "\n据此得到的密钥为： " << re << endl;

**return** 0;

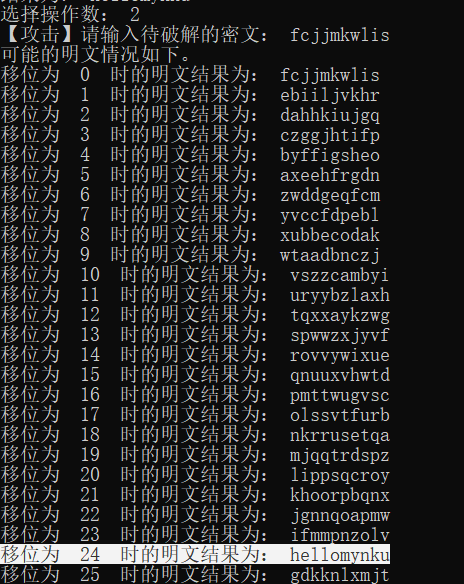
}

1. **实验结果**

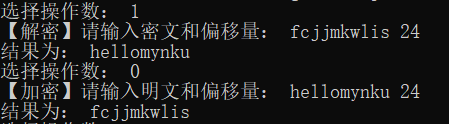
移位密码的加解密：



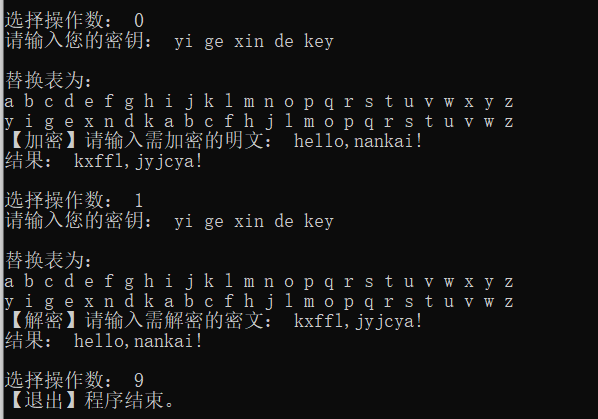
移位密码的攻击破解：



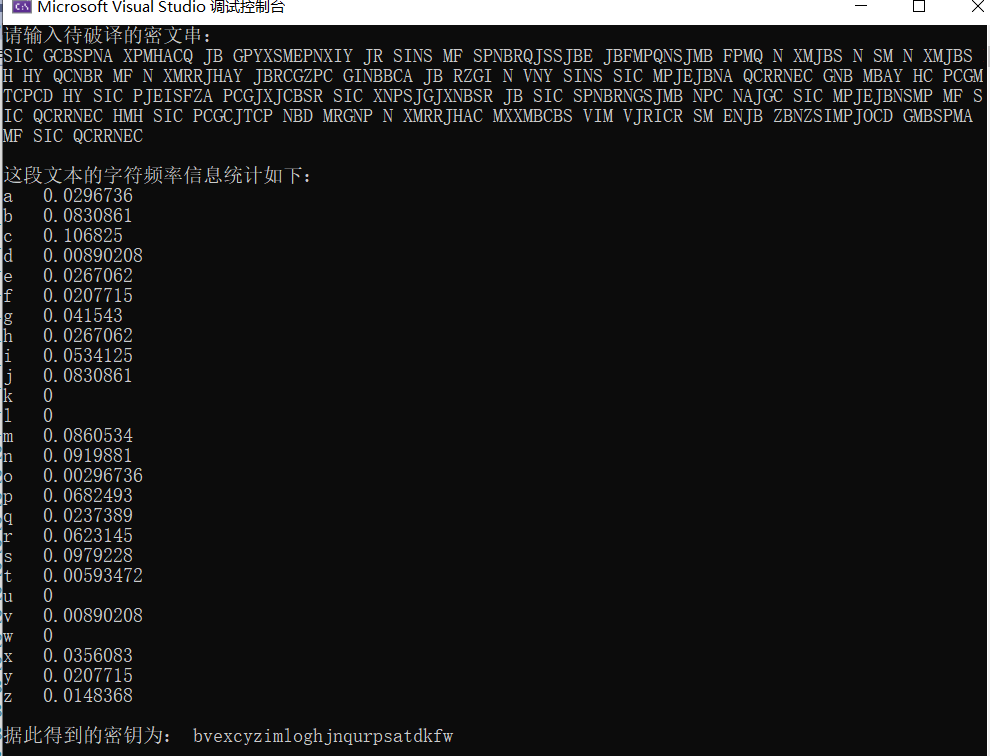
验证攻击结果，可以看到，攻击是成功的：



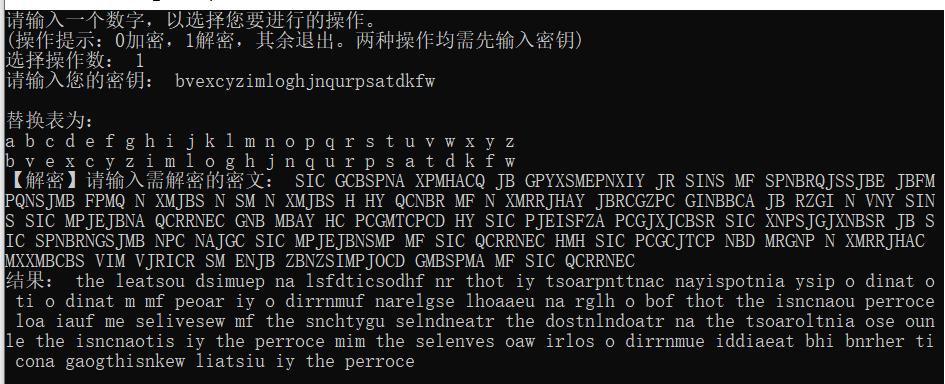
单表置换密码的加解密：



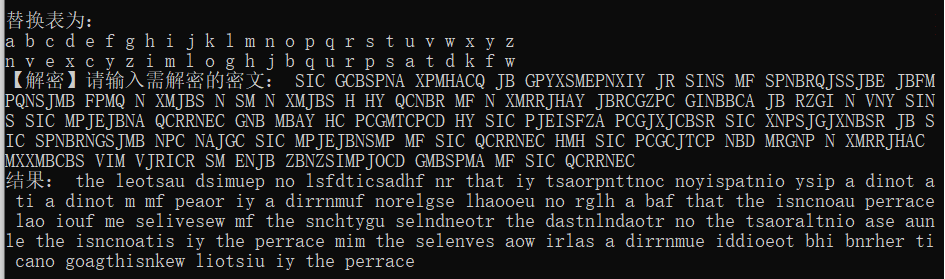
单表置换密码的攻击，输入实验提供的密文串作为样本：



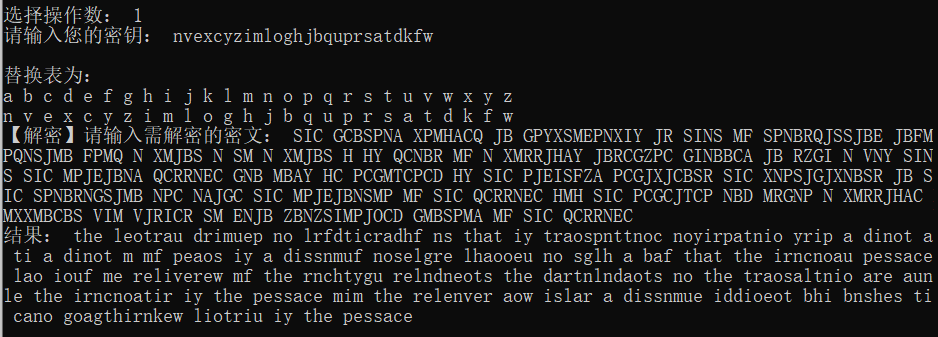
最初得到的密钥为bvexcyzimloghjnqurpsatdkfw，我们将它输入到之前完成的单表置换加解密程序进行解密：



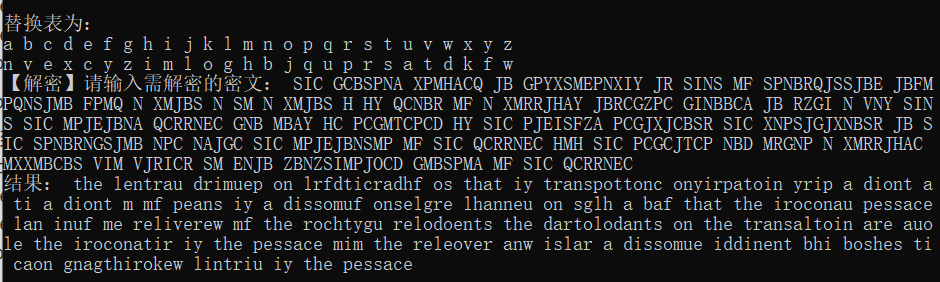
得到奇怪的明文串，显而易见，o和a应该对换：



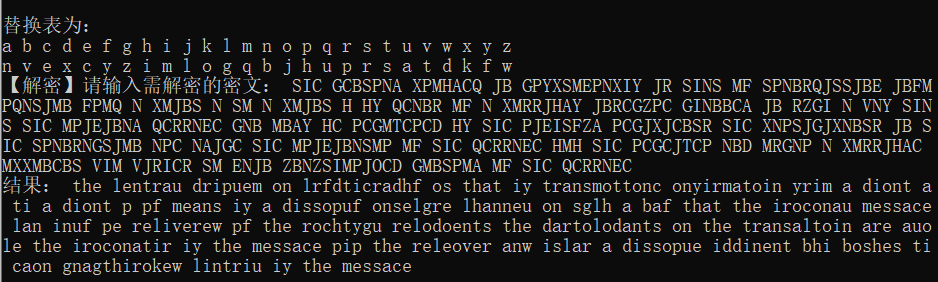
这当中，ase明显不是一个正常的单词，ase应该原本是are，所以s和r应该对换，新的结果如下：



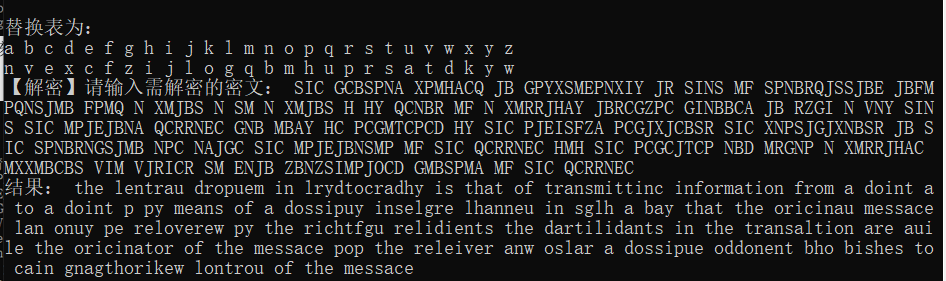
然后nf应该是单词of，n和o对换，密钥是nvexcyzimlogqbjhuprsatdkfw：



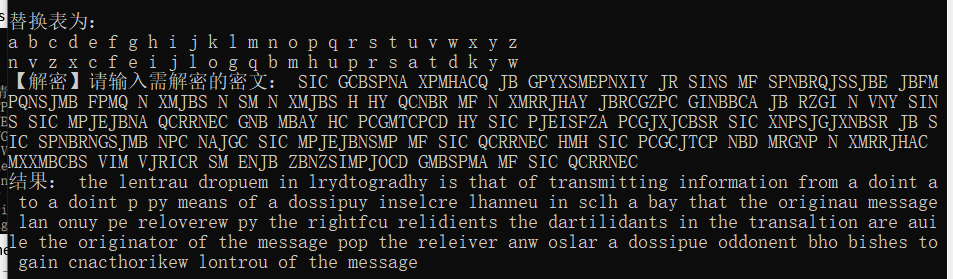
可以发现，peans应该是means，所以p和m也对换。



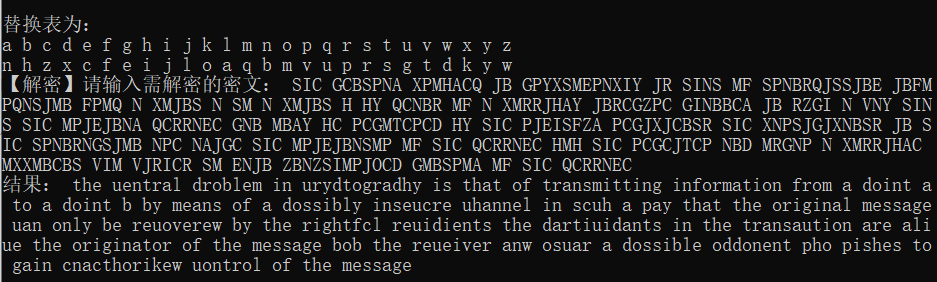
当中的os应该是is，iy应该是if，所以o和i对换，y和f对换。



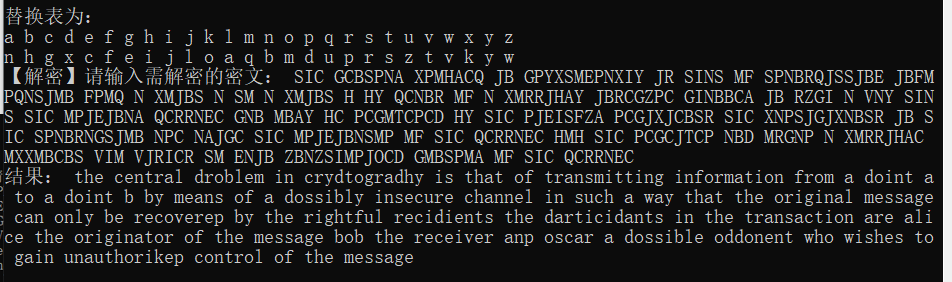
这段明文中的messace应该是message，c和g对换：



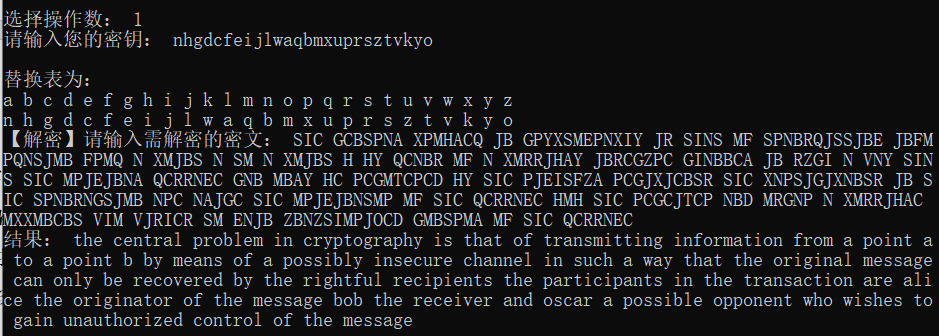
dropuem大概是problem，所以p和b 对换，u和l对换：



uontrol应该是control，pishes应该是wishes，所以u和c对换，p和w对换：



之前换好的表又被打乱了一部分，调整之（p和d对换），同时，unauthoriked应该是unauthorized，所以最终的密钥是nhgdcfeijlwaqbmxuprsztvkyo：



于是，实际得到的明文为：

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 opponent who wishes to gain unauthorized control of the message