

## 《现代密码学》实验报告

实验名称：古典密码分析	实验时间：2025 年 9 月 17 日
学生姓名：谢润文	学号：23336262
学生班级：23 计科	成绩评定：

### 一、实验目的

通过实现对维吉尼亚密码的唯密文攻击和对仿射希尔密码的已知明密文攻击进行密码分析，理解古典密码中代换密码的设计方式和分析方法，提高对字符串的编码处理能力和对简单解密过程的分析综合能力。

### 二、实验内容

1. C/C++实现维吉尼亚密码的密文分析，使用重合指数法找出密钥和明文。

参数要求：输入的文本是密文，只包含 ASCII 可打印字符，明文的内容来自已有的英文文本，密文和密钥的长度至少为 80:1。

2. C/C++实现仿射希尔密码的密文分析，在已知明密文的情况下求出密钥。

参数要求： $m$  为正整数，明文和密文空间均属于  $Z_{26}$ ，密钥由  $(L, b)$  构成，其中  $L$  为可逆矩阵， $b$  是  $m$  维定义在  $Z_{26}$  上的向量。对长度为  $m$  的明文片段  $x$ ，有密文段  $y = xL + b$ 。

### 三、实验原理

对于 1，首先用重合指数法找出周期：设  $x = x_1x_2\cdots x_n$  是含  $n$  个字母的串，则  $x$  中随机选择两元素且两元素相同的概率为  $f_i(f_i-1)/n(n-1)$ ，把 26 个字母的概率加和即为重合指数，当文本串足够长时，由中心极限定理可知该概率近似为  $I_c(x) \approx \sum_{i=0}^{25} p_i^2 \approx 0.065$ ，其中  $p_i$  为第  $i$  个字母在英文文本中出现的概率。假设密钥长度为  $d$ ，提取相同密钥字加密的密文，测试其重合指数。如果猜测正确，则重合指数接近 0.065。否则字符串表现得更加随机，一般在  $0.038 (26 \cdot 1/26^2)$  至 0.065 之间且更接近 0.038。可以采取 kasiski 测试法测算相同明文串被加密成密文的间隔长度，当  $d$  较小时亦可以直接枚举，得到周期。其次需要确定密钥的相对位移，即密钥的每一位是何值。依然采用重合指数法，统计在每一个子串中对应的频数/子串长度，应该近似英文文本的统计概率。定义数值  $M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n}$ ，则若  $g = k_i$  是密钥的第  $k$  位时，这一数值应接近  $I_c(x)$ ，否则小于 0.065。枚举每位可以得到密钥，再进行反加密过程即可得到明文，完成维吉尼亚密码的密文分析。

对于 2，注意到  $y_i = x_iL + b$ ， $y_{m+1} = x_{m+1}L + b$ ，做差值可以得到  $y_{m+1} - y_i = (x_{m+1} - x_i)L$ ，得到线性关系。同理可得  $y_{2m+1} - y_{m+1} = (x_{2m+1} - x_{m+1})L$ ，……整理可以得到一个矩阵乘法式  $Y_{diff} = X_{diff}L (*)$ ，这里  $diff$  表示差值， $Y$  和  $X$  矩阵可以通过明密文串得到，转化为在  $Z_{26}$  意义下的矩阵运算。对

于(\*)式,左右同乘  $X_{diff}^{-1}$ , 可得  $X_{diff}^{-1}Y_{diff}=L$ , 从而得出  $L$ 。关键在于  $X_{diff}^{-1}$  的计算: 在模 26 意义下, 要计算矩阵的逆普通的增广矩阵-高斯消元法不适用。参考模意义下的矩阵求逆 (<https://www.cnblogs.com/Higgerw/p/14464825.html>), 得知需要计算行列式、伴随矩阵。行列式可以使用代数余子式进行运算:  $\Delta = \sum_{i=1}^m (-1)^{1+i} a_{1i} M_{1i}$ , 其中  $M_{1i}$  是把第一行和第  $i$  列除去后剩余元素组成的  $m-1 \times m-1$  矩阵的行列式。伴随矩阵亦可以用代数余子式运算得到:  $C_{ij}=(-1)^{i+j} M_{ji}$ ,  $\text{adj}(A)=C^T$  即余子矩阵的转置矩阵。在求出  $L$  后, 分析第一对明密文  $y_1=x_1L+b$ ,  $y_1$  向量、 $x_1$  向量、 $L$  均已知, 可以直接求出  $b$  向量= $y_1-x_1L$ , 从而求出密钥, 完成对仿射希尔密码的分析。

#### 四、实验步骤 (源代码)

```
#include<iostream>
#include<algorithm>
#include<vector>
#include<cmath>
#include<map>
using namespace std;
double fre[26]={0.082,0.015,0.028,0.043,0.127,
               0.022,0.020,0.061,0.070,0.002,
               0.008,0.040,0.024,0.067,0.075,
               0.019,0.001,0.060,0.063,0.091,
               0.028,0.010,0.023,0.001,0.020,
               0.001}; //frequency in normal English sentences
string str,all_big="";
string key="";
int decrypt(int cycle){
    string substr[cycle+1];
    key="";
    int n=all_big.length();
    int subcnt[cycle+1][26]={0}; //count of each substrings
    int subtot[cycle+1]={0};
    double M[cycle+1][26]={0},I[cycle+1]={0};
    for(int i=0;i<cycle;++i) substr[i]="";
    for(int i=0;i<n;++i){
        substr[i%cycle]+=all_big[i];
        subcnt[i%cycle][all_big[i]-'A']++;
    }
}
```

```

        subtot[i%cycle]++;
    }
    int fl1=0;
    double sum_I=0;
    for(int i=0;i<cycle;++i){
        int sublen=substr[i].length();
        for(int k=0;k<26;++k){

I[i]+=((double)subcnt[i][k])/sublen*(subcnt[i][k]-1)/(sublen-1);
        }
        double i_dis=I[i]-0.065;
        sum_I+=i_dis;
        if(i_dis<0) i_dis=-i_dis;
        //cout<<I[i]<<endl;
        if(i_dis<=0.005) fl1++;
    }
    //cout<<fabs(sum_I/cycle)<<endl;
    if(fl1<cycle/6 || fabs(sum_I/cycle)>0.008) return 1;
    for(int i=0;i<cycle;++i){
        int sublen=substr[i].length();
        int mov_k=0;
        double eps=0.5;
        for(int k=0;k<26;++k){
            for(int j=0;j<26;++j){
                M[i][k]+=fre[j]*subcnt[i][(j+k)%26]/sublen;
            }
            //cout<<M[i][k]<<" ";
            double d_dis=M[i][k]-0.065;
            if(d_dis<0) d_dis=-d_dis;
            if(d_dis<eps){
                //cout<<d_dis<<endl;
                eps=d_dis;
                mov_k=k;
            }
        }
    }

```

```

        }
        key+='A'+mov_k;
    }
    cout<<key<<endl;
    return 0;
}
string line_str[100001];
int main(){
    int l_tot=0;//total num of letters
    int line_cnt=0;
    while(getline(cin,str)){
        int n=str.length(),flag=1;
        line_str[line_cnt++]=str;
        for(int i=0;i<n;++i){
            if(str[i]>='A' && str[i]<='Z'){
                flag=1;
                all_big+=str[i];
                l_tot++;
            }
            else if(str[i]>='a' && str[i]<='z'){
                flag=1;
                all_big+='A'+str[i]-'a';
                l_tot++;
            }
            else{
                if(flag){//first time not letter
                    flag=0;
                }
            }
        }
    }
    int cycle,y;
    for(int i=2;i<=5000;++i){
        y=decrypt(i);
    }
}

```

```

        if(y==0){
            cycle=key.length();
            break;
        }
    }
    l_tot=0;
    for(int i=0;i<line_cnt;++i){
        int str_len=line_str[i].length();
        for(int j=0;j<str_len;++j){
            if(line_str[i][j]>='a' && line_str[i][j]<='z'){
                if(line_str[i][j]-'a'-(key[l_tot%cycle]-'A')>=0){
                    cout<<char(line_str[i][j]-key[l_tot%cycle]+'A');
                }
                else
                    cout<<char(26+line_str[i][j]-key[l_tot%cycle]+'A');
                l_tot++;
            }
            else if(line_str[i][j]>='A' && line_str[i][j]<='Z'){
                if(line_str[i][j]-key[l_tot%cycle]>=0){
                    cout<<char(line_str[i][j]-key[l_tot%cycle]+'A');
                }
                else
                    cout<<char('A'+26+line_str[i][j]-key[l_tot%cycle]);
                l_tot++;
            }
            else cout<<line_str[i][j];
        }
        cout<<endl;
    }
}

```

上框内为实验 1-1 的代码。

```

#include<iostream>

#include<cmath>

using namespace std;

```

```

const int MAXM=30;

int
y_diff[MAXM][MAXM],mat[MAXM][MAXM],adj[MAXM][MAXM],key[MAXM][MAXM];

int rev(int x){//mod 26,x^-1
    x%=26;
    for(int i=1;i<26;i+=2) if(x*i%26==1) return i;
    return -1;
}

int det(int mat[MAXM][MAXM],int m){
    int ans=0;
    if(m==1) return mat[0][0];
    else{
        for(int i=0;i<m;++i){
            int tmp[MAXM][MAXM];
            for(int j=0;j<m;++j){
                for(int k=0;k<m;++k){
                    if(k==i) continue;
                    int col=k<i?k:k-1;
                    tmp[j-1][col]=mat[j][k];
                }
            }
            ans+=((i%2==1)?-1:1)*mat[0][i]*det(tmp,m-1);
        }
    }
    return ans;
}

void com_adj(int mat[MAXM][MAXM],int m){//compute the adjacency matrix
    for(int i=0;i<m;++i){
        for(int j=0;j<m;++j){
            int tmp[MAXM][MAXM];
            for(int k=0;k<m;++k){
                if(k==i) continue;
                int row=k<i?k:k-1;
                for(int l=0;l<m;++l){

```

```

        if(l==j) continue;
        int col=l<j?l:l-1;
        tmp[row][col]=mat[k][l];
    }
}
adj[j][i]=pow(-1,i+j)*det(tmp,m-1);
}
}

int main(){
    int m;
    string pl,ci;//plaintext & ciphertext
    cin>>m;
    cin>>pl>>ci;
    for(int i=0;i<m*m;++i){
        mat[i/m][i%m]=pl[i+m]-pl[i];
        y_diff[i/m][i%m]=ci[i+m]-ci[i];
    }
    com_adj(mat,m);
    int _rev=rev(det(mat,m));
    for(int i=0;i<m;++i){
        for(int j=0;j<m;++j){
            adj[i][j]=((adj[i][j]*_rev)%26+26)%26;
            //cout<<adj[i][j]<<" ";
        }
        //cout<<endl;
    }
    /*
    int test[MAXM][MAXM]={0};
    for(int i=0;i<m;++i){
        for(int j=0;j<m;++j){
            for(int k=0;k<m;++k){
                test[i][j]+=adj[i][k]*mat[k][j];
            }
        }
    }
    */
}

```

```

        }
        test[i][j]=(test[i][j]%26+26)%26;
        cout<<test[i][j]<<" ";
    }
    cout<<endl;
}
*/
for(int i=0;i<m;++i){
    for(int j=0;j<m;++j){
        for(int k=0;k<m;++k){
            key[i][j]+=adj[i][k]*y_diff[k][j];
        }
        key[i][j]=(key[i][j]%26+26)%26;
        cout<<key[i][j]<<" ";
    }
    cout<<endl;
}
int dif[MAXM]={0};
for(int i=0;i<m;++i){
    for(int j=0;j<m;++j){
        dif[i]+=(pl[j]-'A')*key[j][i];
    }
    cout<<((ci[i]-'A'-dif[i])%26+26)%26<<" ";
}
}

```

上框内为实验 1-2 的代码。

## 五、实验结果

实验 1-1:

样例 1:

```
C:\Users\Ferro\Desktop\现代密码学 × + ▾

K ccpkbgv fdp hq tyavinrrt mvg rkdnbv fd etdgi ltxrgu ddk o
tfmb pvge G ltg ck qracqc. Wdn awcrxiza kft lewrpt ycqkyvx ch
kft poncq qr hjv ajuwe tmcmspkq dy hjv dahct rls vskcgcz
qqdzxgsf rls wcwsjt bh. Afs iasp rjahk jrju mvg kmitz hfp
dispzlvl gwtf pl kkebdpg ceb shctj. Rwx b afs pezqn rwx cvycg
aonw ddk ackawbbi kft iovkcg. Hjvl nhi ffsqes vyc lacnv rwbbi
repbb vf exos c dygzwp fd tkfqi. Ycw hjvl nhi qibtk hjv npist.
^Z
CRYPTO
I learned how to calculate the amount of paper needed for a
room when I was at school. You multiply the square footage of
the walls by the cubic contents of the floor and ceiling
combined and double it. You then allow half the total for
openings such as windows and doors. Then you allow the other
half for matching the pattern. Then you double the whole thing
again to give a margin of error. And then you order the paper.

-----
Process exited after 3.418 seconds with return value 0
请按任意键继续. . . |
```

样例 2:

```
C:\Users\Ferro\Desktop\现代密码学 × + ▾

S lotabgy kqmasbaew qy n eizuwuggakgy kknrem lbj dkeanevfo zuw
iagzmtgakogq wl qaoogst srkagtw ue vmihemtgc. I bndqj qaoogst
yvyvggmzk bf i srkagtw ooiwa g ewkocamtg uwsalkaum zusb zuw
ukfkimr uisr xzuz s akavmx xfwca lw zuw zkpaxorfb.

Jvyqznd aotfizhjmy njm g flitqszy rdmsrfb us ewyg uzeclwmesxnvu
xxblwibd aavlmy, nfl gew kuzewtyq cyrv nue kwlgoixr
vqygjghhlqua, xqtnfkond bxnfaoplquak, kualzgpl ugasokzwvz
fgnzjszk, nfl oa gbnrj kgfwa cuwzk vl qy vexuelitg lw jrlmig
xwxwtwe bj bgzhmxvfo.
^Z
SIGN
A digital signature is a mathematical scheme for verifying the
authenticity of digital messages or documents. A valid digital
signature on a message gives a recipient confidence that the
message came from a sender known to the recipient.

Digital signatures are a standard element of most cryptographic
protocol suites, and are commonly used for software
distribution, financial transactions, contract management
software, and in other cases where it is important to detect
forgery or tampering.

-----
Process exited after 3.162 seconds with return value 0
请按任意键继续. . . |
```

例 1.12:

```
C:\Users\Ferro\Desktop\现代密码学 × + ▾

CHREEVOAHMAERATBIAXXWTNXXBEEOPHBSBQNMQEERBW RVXUOAKXAOSXXWEAHBWGMNOMNKGFRVGVXWTRZXWIAK LXFPSKAUTERNDCNGTSXMXBTUADNGMGPSR
ELXNJELX VRVPTULHDNQWTDYGBPHXTFALJHASVBFXNGLLCHR ZBWELEKMSJIKNBHWRJGNMGJSLXFEYPHAGNRBIEQJT AMRVLCRREMNDGLXRRINGNSNRW
CHRHAEYEVTAQEBSI PEWEVKAHOEWADRENMTBHHCHRTKDNVRZCHRCLOOHP WQAIIXNRMGWOIFKEE
^Z
JANET
THEALMONDTREEMASINTENTATIVEBLOSSOMTHEDAYSW ERELONGEROFTENENDINGWITFMAGNIFICENTEVENING SOFCORRUGATEDPINKSKIESTHEHUNTINGSE
ASONWASO VERWITHHOUNDSANDGUNSPUTAWAYFORSIXMONTHSTHE VINEYARDSWEREBUSYAGAINASTHEWELLORGANIZEDFA RMERSTREATEDTHEIRVINESAND
THEMORELACKADAISI CALNEIGHBORSHURRIEDTODOTHEPRUNINGTHEYSFOUL DHAVEDONEINNOVEMBER

-----
Process exited after 2.759 seconds with return value 0
请按任意键继续. . . |
```

可以看到对于每一段明文，都能正确的分析出密钥和密文。

实验 1-2:

为了验证矩阵求逆的正确性，此处把输出 test 数组的代码取消注释，对于两个样例分别得到结果如下：

```
C:\Users\Ferro\Desktop\现代密码学 > 3
ADISPLAYEDEQUATION
DSRMSIOPLXLJBZULLM
1 0 0
0 1 0
0 0 1
3 6 4
5 15 18
17 8 5
8 13 1
-----
Process exited after 2.249 seconds with return value 0
请按任意键继续 . . .
```

```
C:\Users\Ferro\Desktop\现代密码学 > 4
THEQUICKLYBROWNFOXJUMPSOVERALAZYDOGE
IOWWEMXSWRIKPPHJHTJQFQLLBZSQLXGUVYAB
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
1 6 23 15
7 14 20 3
18 13 18 23
9 8 14 22
10 12 7 0
-----
Process exited after 2.964 seconds with return value 0
请按任意键继续 . . .
```

可以看到输出了单位矩阵，证明求逆过程无误，且 L 矩阵与 b 向量的输出均与标准输出无异。

## 六、思考题

1. 在 1-1 中还有什么实现找到周期的方法？

除了代码中的重合指数法，可以尝试采用 kasiski 测试法猜测维吉尼亚密码体系的密钥的长度。但是实际采用在 Onlinejudge（下称 OJ）上挂了（3/5），猜测原因有二：1）周期太长，导致完全没有重复出现的明-密文对，造成无法破解。2）可能有完全不同的明文加密成相同的密文这一情况，导致周期判断错误。比如 is 和 an 加密成相同的字符，影响最大公因数，进而周期误判。

## 2. 其他的问题？

1-1 中怎么合理量化“接近 0.065”？我的代码中采用了 `if(f11<cycle/6 || fabs(sum_I/cycle)>0.008) return 1;` 的形式，即首先分组频率在 0.065 左右（ $\pm 0.005$ ）的数量应该大于组数的  $1/6$ ，否则不能认为该分组“稳定”；其次分组频率的总体平均数应该也保持在 0.065 左右。通过对样例 1 和 2 的测试发现误差一个在 0.0034 一个在 0.0047，注意到越长的文本，统计性质应该越好，所以代码实际上取了 0.008 作为误差的判据，并成功通过了 OJ。

1-2 中存不存在某种情况使得行列式为 0 的情况？有可能。因为行列式为 0 意味着非满秩，即某些列向量线性有关，假如明文中恰好间隔  $m$  个字母的内容有重合（比如 Can the can be recycled）， $m=3$ ，那么  $x_1$  与  $x_7$  重合，导致前三组中出现行列式为 0 的情况。理论上应该加入特判，如果出现这种情况，那么跳过重复的明-密文对，直到没有重复的为止。实际上由于一般的英文文本出现这种情况的概率较低，在 OJ 上没有特判仍然通过了。

## 七、实验总结

本次实验主要考察古典密码的分析，包含算法设计、基本字符串操作、基础线性代数知识的应用，旨在理解代换密码的思想，为后面的实验做铺垫。

在 1-1 中，算法的瓶颈主要有以下几处：1. 读入与输出。改进方法：cout 对于大数据的处理较慢，输出可以采取 puts 函数加速。2. 枚举周期  $d$ ，每一次验证需要  $O(26*26*d)$  的复杂度。暂时无优化方法。

在 1-2 中，算法的瓶颈在于：1. 行列式的计算。由于想避免高斯消元法可能带来的误差问题，因此采用了较慢的递归方式，每次递归要计算  $m$  个  $(m-1)$  阶行列式，递归树深度为  $m$ ，复杂度为  $O(m!)$ 。优化方法：采用 eps 控制高斯消元法的误差，可以将计算行列式复杂度降至  $O(m^3)$ 。2. 计算伴随矩阵调用 det 导致的高复杂度  $O(m!*m^2)$ 。随着 1 的优化，2 自然也被优化，可以将总体复杂度降至  $O(m^5)$ 。