**齐鲁工业大学实验报告** 成绩

课程名称 密码分析 指导教师 孔志刚 实验日期 2021.10

院（系） 网络空间安全学院 专业班级 网安19-1 实验地点 南213214

学生姓名 焦翔宇 学号 201903150004 同组人 徐弘硕、陈宇隆

# 未知古典密码分析

## 待解决问题

**分析未知密码密文：**

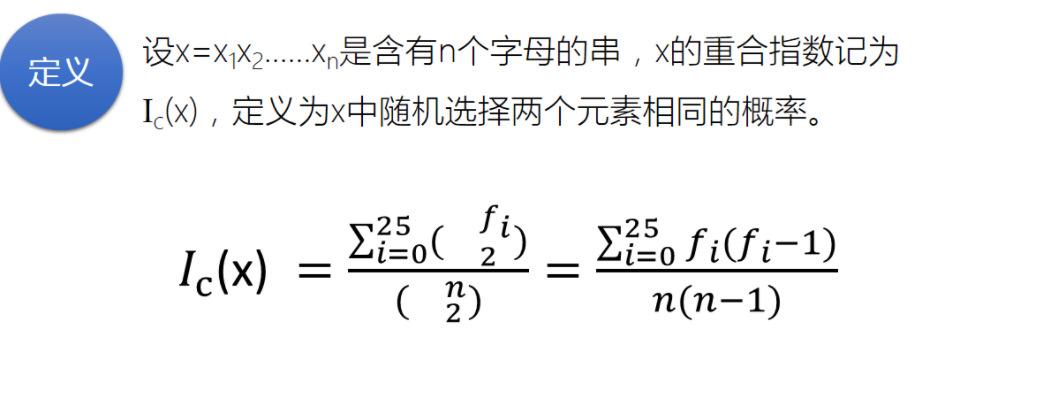
**BNVSNSIHQCEELSSKKYERIFJKXUMBGYKAMQLJTYAVFBKVTDVBPVVRJYYLAOKYMPQSCGDLFSRLLPROYGESEBUUALRWXMMASAZLGLEDFJBZAVVPXWICGJXASCBYEHOSNMULKCEAHTQOKMFLEBKFXLRREDTZXCIWBJSICBGAWDVYDHAVEJXZIBKCGJIWEAHTTOEWTUHKRQVVRGZBXYIREMMASCSPBHLHJMBLRFFJELHWEYLWISTFVVYEJCMHYUYRUFSFMGESIGRLWALSWMNUHSIMYYITCCQPZSICEHBCCMZFEGVJYOCDEMMPGHVAAUMELCMOEHVLTIPSUYILVGFLMVWDVYDBTHFRAYISYSGKVSUUHYHGGCKTMBLRX**

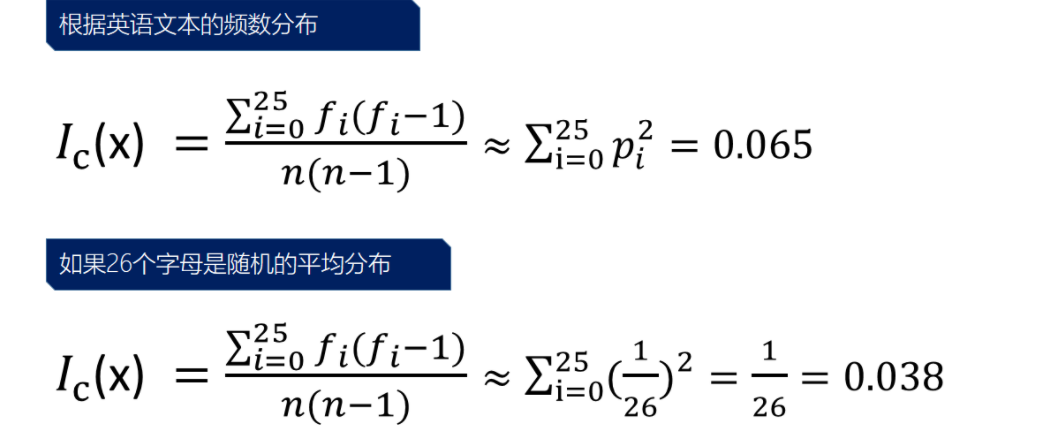
## 题目分析



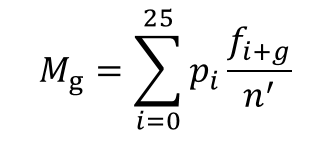
根据上图，分析前4种常见的古典密码可知，移位、仿射、代换都是单表代换，维吉尼亚密码是多表代换。单表代换不会改变文本的重合指数，多表代换会使重合指数发生变化，由原英文文本的0.065向完全随机文本的重合指数0.038变化。通过验证密文的重合指数可以区分出单表代换和多表代换。其中单表代换中的移位密码和仿射密码又可以通过穷举密钥加明文指数测试的方法求解出明文，如果不能给出明文解。说明不是移位密码或仿射密码，大概率是代换密码，需要通过频数分析的方法来求解。

## 重合指数Ic





## 明文指数Mg



Mg是英语文本的关于频数分布的度量。

Ic用于判断密文是否是单表单换的结果。

Mg是判断英语文本是否不是明文(有意义的文本)。

可以用Mg来判断是否是明文。

## 实验过程

### 1、求密文的重合指数，判断是哪一种加密方式

经计算密文的重合指数mg=0.041814,基本上可以确定加密方式是维吉尼亚密码。

下表为密文中各个字母出现次数和对应频率。



### 2、分析维吉尼亚密码

#### (1)确定分组数m。

#### 方法一：Kasiski测试法

根据重复出现的密文串的位置来确定m的值，找出重复出现的长度为3的密文串，计算它们之间的距离，并求解最大公约数来确定分组长度。由于密文长度太短，又或者是该密文是特殊构造的，用Kasiski测试法没有找到重复出现3次以上的长度为3的字符串。

**Kasiski代码**

1. **int** Kasiski(string s)
2. {
3. **int** m = 0;
4. **int** len = s.length();
5. **for** (**int** I = 0; I < len – 2; i++)
6. {
7. string chr = “”;
8. chr = chr + s[i] + s[I + 1] +s[I + 2];
9. **int** c = 0;//记录出现次数
10. **int** index[100];
11. memset(index, 0, **sizeof**(index));
12. **for** (**int** j = 0; j < len – 2; j++)
13. {
14. string tmp = “”;
15. tmp = tmp + s[j] + s[j + 1] +s[j + 2];
17. **if** (chr == tmp)
18. {
19. index[c++] = j;
20. }
21. }
22. **if** (c > 2)
23. {
24. **for** (**int** k = 1; k < c; k++)//计算距离
25. {
26. index[k] = index[k] – index[0];
27. }
28. **int** g = gcd(index[1], index[2]);
29. **for** (**int** k = 3; k < c; k++)
30. {
31. g = gcd(g, index[k]);
32. }
33. **if** (m == 0)m = g;
34. **else** {
35. **if** (m == g)m = g;
36. **else** m = gcd(m, g);
37. }
38. }
39. }
40. **return** m;
41. }

#### 方法二：穷举m值，用重合指数Ic验证

下表为分组为m时，对应各分组重合指数以及重合指数均值(部分表格)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m |  |  |  |  |  |  |  |  |  |  |  |  | 均值 |
| m=1 | 0.0418 |  |  |  |  |  |  |  |  |  |  |  | 0.0418 |
| m=2 | 0.0456 | 0.0464 |  |  |  |  |  |  |  |  |  |  | 0.046 |
| m=3 | 0.0446 | 0.0481 | 0.0484 |  |  |  |  |  |  |  |  |  | 0.0471 |
| m=4 | 0.0471 | 0.0575 | 0.0465 | 0.0475 |  |  |  |  |  |  |  |  | 0.0497 |
| m=5 | 0.045 | 0.0425 | 0.0418 | 0.0455 | 0.037 |  |  |  |  |  |  |  | 0.0424 |
| **m=6** | **0.0512** | **0.0635** | **0.055** | **0.0698** | **0.0576** | **0.0698** |  |  |  |  |  |  | **0.0612** |
| m=7 | 0.0405 | 0.0454 | 0.0443 | 0.0377 | 0.0493 | 0.0319 | 0.0464 |  |  |  |  |  | 0.0422 |
| m=8 | 0.0583 | 0.0592 | 0.0509 | 0.0481 | 0.0379 | 0.0647 | 0.0464 | 0.0502 |  |  |  |  | 0.052 |
| m=9 | 0.0488 | 0.0441 | 0.0418 | 0.036 | 0.0402 | 0.0402 | 0.0427 | 0.0451 | 0.0622 |  |  |  | 0.0446 |
| m=10 | 0.0654 | 0.0413 | 0.0256 | 0.0435 | 0.0405 | 0.0405 | 0.0541 | 0.0495 | 0.0601 | 0.0511 |  |  | 0.0472 |
| m=11 | 0.0499 | 0.041 | 0.0446 | 0.0374 | 0.0303 | 0.0553 | 0.0428 | 0.0357 | 0.0357 | 0.0517 | 0.0511 |  | 0.0432 |
| m=12 | 0.0544 | 0.0645 | 0.0516 | 0.0624 | 0.0667 | 0.1226 | 0.0452 | 0.071 | 0.0473 | 0.0817 | 0.0667 | 0.0559 | 0.0658 |

发现m=6和m=12时，重合指数均接近0.065，所以基本可以确定m=6。

**代码**

1. **double** mmg = 0;
2. fprintf(fp, "输出分组为m的各组重合指数\n");
3. **for** (**int** i = 1; i <= 20; i++)
4. {
5. string tmp[20];
6. **for** (**int** j = 0; j < s.length(); j++)
7. {
8. tmp[j % i] += s[j];
9. }
10. mmg = 0;
11. fprintf(fp,"m=%d,", i);
12. **for** (**int** j = 0; j < i; j++)
13. {
14. tmp[j][tmp[j].length()] = '\0';
15. fprintf(fp,"%.4f,", Mg(tmp[j], **true**));
16. mmg = mmg + Mg(tmp[j], **true**);
17. }
18. mmg = mmg / i;
19. **int** n = 20 - i;
20. **while** (n--)fprintf(fp, ",");
21. fprintf(fp,"%.4f\n", mmg);
22. **if** (mmg >= 0.06)
23. {
24. **if** (m == 0||m==1)m = i;
25. **else** m = gcd(m, i);
26. }
27. }
28. cout <<"m=" << m << endl;

#### (2)求解加密密钥

维吉尼亚密码的每一组是一个简单的移位密码，可以通过穷举密钥的方法来破解，并用明文指数来判断是否是明文。

经计算，密钥为**theory**

#### (3)根据密钥求解明文

明文：

**igrewupamongslowtalkersmeninparticularwhodroppedwordsafewatatimelikebeansinahillandwhenigottominneapoliswherepeopletookalakewobegoncommatomeantheendnfastoryicouldntspeakawhnlesentenceincompanyandwasconsiderednottoobriahtsoienrolledinaspeechcouqsetaughtbyorvillesandthefounderofreflexiverelaxologyaselfhypnotictechniquethatenabledapersontospeakuptothreehundredwordsperminute**

整理：

l grew up anong slow talkers,men in particular,who dropped words a few at a time like beansin a hill, and when I got to Minneapolis where people took a Lake Wobegon comma to mean theend of a story, I couldn't speak a whole sentence in company and was considered not too bright.So I enrolled in a speech course taught by Orville Sand, the founder of reflexive relaxology, aself-hypnotic technique that enabled a person to speak up to three hundred words per minute.

代码

1. string Vigenere\_Dncode(string s, string key)
2. {
3. **int** len1 = s.length(), len2 = key.length();
4. string m;
5. **for** (**int** i = 0; i < len1; i++)
6. {
7. m = m + **char**((**int**(s[i] - 'a') - (key[i % len2] - 'a') + 26) % 26 + 'a');
8. }
9. **return** m;
10. }