

# 信息安全技术作业（一）

## 维吉尼亚密码的破译

中山大学计算机学院 计算机科学与技术

19335174 施天予

### 目录

<b>1 实验原理</b>	<b>2</b>
<b>2 解决方案</b>	<b>2</b>
<b>3 分析过程</b>	<b>3</b>
3.1 Kasiski 测试法 . . . . .	3
3.2 重合指数法确定密钥长度 . . . . .	3
3.3 重合指数法确定密钥分量 . . . . .	4
3.4 运用密钥破译密文 . . . . .	5
<b>4 实验结果</b>	<b>5</b>
4.1 程序结果 . . . . .	5
4.2 翻译结果 . . . . .	6
<b>A 维吉尼亚密码的破译完整代码</b>	<b>6</b>

## 一、实验原理

维吉尼亚密码是使用一系列凯撒密码组成密码字母表的加密算法，属于多表密码的一种简单形式，一个字母可以被映射为  $m$  个字母中的某一个，即有  $m$  种可能。

	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
a	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	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	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	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

图 1: 维吉尼亚密码字母矩阵

## 二、解决方案

1. (Kasiski 测试法) 搜索长度至少为 3 的相同的密文段，记录这些相同密文段到起始点之间的距离，猜测这些距离的最大公约数为密匙字长度  $m$ 。
2. (重合指数法确定密匙长度) 把密文字符串  $y$  分割成  $m$  个长度相等的子串，分别计算它们各自的重合指数。它们的重合指数应该均为标准应与文本串的重合指数，即大约为 0.065，否则密匙字长度不等于  $m$ 。
3. (重合指数法确定密匙分量) 猜测第  $i$  个密匙分量为  $g$ ，即  $k_i = g$ 。如果是正确的，那么应该有  $M_g = \sum_{i=0}^{25} \frac{p_i f_{g+i}}{n'} \approx 0.065$ ，否则  $M_g$  应该不等于 (小于) 0.065。
4. 重复利用步骤 (3) 的方法获得其余的密匙分量。

### 三、分析过程

#### 1. Kasiski 测试法

按照书上的方法，先搜索长度至少为 3 的密文段，记录他们的距离。密文串 CHR 出现在五个地方。在密文中，从位置 1、166、236、276 和 286 开始。从第一次出现到其他四次出现（分别）是 165、235、275 和 285。这四个整数的最大公约数是 5，所以很可能是密钥长度。

```

1  /*用kasiski测试法获取key可能的长度*/
2  vector<int> Find_same(string cipher) {
3      vector<int> distance;
4      string p = cipher.substr(0, 3);
5      for (int i = 3; i < cipher.length()-3; ++i){
6          string tmp = cipher.substr(i, 3);
7          if (tmp == p)
8              distance.push_back(i);
9      }
10     return distance;
11 }

```

#### 2. 重合指数法确定密钥长度

接下来看重合指数的计算是否给出了相同的结论。当  $m = 1$  时，重合指数为 0.045。当  $m = 2$  时，两个指数是 0.046 和 0.041。当  $m = 3$  时，我们得到 0.043、0.050、0.047。当  $m = 4$  时，我们有指数 0.042、0.039、0.045、0.040。然后，尝试  $m = 5$ ，我们得到值 0.063、0.068、0.069、0.061 和 0.072。这也提供了强有力的证据表明密钥的长度为 5。

```

1  /*确定密钥key的长度*/
2  int Get_keyLength(string cipher) {
3      int key_length;
4      vector<int> distance = Find_same(cipher);
5      int m = distance[0];
6      for (int i = 0; i < distance.size()-1; ++i)
7          for (int j = i+1; j < distance.size(); ++j)
8              m = min(m, gcd(distance[i], distance[j]));
9      for (int i = 1; i <= m; ++i) {
10         double sum = 0.000;
11         for(int j = 0; j < i; ++j) {
12             double temp = IC(cipher, j, i);
13             sum += temp;
14             cout << temp << " ";
15         }
16         cout << endl;
17         int value = (double)fabs(0.065-(double)(sum/(double)i));
18         if (fabs(sum/i - 0.065) < 0.01)

```

```

19         key_length = i;
20     }
21     return key_length;
22 }

```

从  $m = 1$  到  $m = 5$  的重合指数如图2所示。

图 2: 维吉尼亚密码字母矩阵

### 3. 重合指数法确定密钥分量

- 设  $f_0, f_1, \dots, f_{25}$  分别表示字母  $A, B, C, \dots, Z$  在字符串  $y_i$  中的出现次数。 $y_i$  的长度记为  $n' = n/m$ 。
- 字符串中的每个字母是对应的明文字母移动  $k_i$  个位置得到的，因此有

$$\frac{f_{k_i}}{n'} \approx p_0, \frac{f_{k_i+1}}{n'} \approx p_1, \dots, \frac{f_{k_i+25}}{n'} \approx p_{25}$$

- 猜测  $k_i = g$ ，如果是正确的，那么一定有

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{g+i}}{n'} \approx \sum_{i=0}^{25} p_i^2 \approx 0.065$$

否则  $M_g$  应不等于 0.065。

```

1  /*确定密钥*/
2  vector<int> Get_key(string cipher, int key_length){
3      vector<int> key(key_length, 0);
4      map<char, int> mp;
5      for (int i = 0; i < key_length; ++i) {
6          for (int j = 0; j < 26; ++j) {
7              mp.clear();
8              double pg = 0.000;
9              int sum = 0;
10             for (int k = i; k < cipher.length(); k += key_length) {
11                 char c = (char)((cipher[k] - 'a' + j) % 26 + 'a');
12                 mp[c]++;

```

```

13         sum++;
14     }
15     for (char k = 'a'; k <= 'z'; ++k)
16         pg += ((double)mp[k]/(double)sum)*table[k-'a'];
17     if (fabs(pg-0.065) < 0.01)
18         key[i] = j;
19 }
20 }
21 return key;
22 }

```

#### 4. 运用密钥破译密文

最后运用得到的密钥一位一位破解密文，就能得到最终的明文。

```

1     for (int i = 0; i < cipher.length(); ++i)
2         cout << (char)((cipher[i]-'a'+key[i%key_length])%26+'a');

```

### 四、实验结果

#### 1. 程序结果

运行代码，可以得到密钥的长度为 5，密钥字符串是“janet”。具体结果如图3所示。

```

0.0447284
0.0455659 0.040503
0.0432234 0.0487179 0.04705
0.0421941 0.0379747 0.045288 0.040293
0.06298 0.0681004 0.0686124 0.0588838 0.0724484

key_length: 5
KEY: janet
TEXT:
thealmondtreewasintentativeblossomthedayswere-longerof-
enendingwithmagnificenteveningsofcorrugatedpinkskiesth-
ehuntingseasonwasoverwithhoundsandgunsputawayforsixmon-
thsthevineyardswerebusyagainasthewellorganizedfarmerst-
reatedtheirvinesandthemorelackadaisicalneighborshurrie-
dtodothepruningtheyshouldhavedoneinnovember

```

图 3: 维吉尼亚密码字母矩阵

## 2. 翻译结果

将破译后的明文人为划分得到英文

```
1 the almond tree was intentative blossom the days were longer often ending
2 with magnificent evenings of corrugated pink skies the hunting season was
3 over with hounds and guns put away for six months the vineyards were busy
4 again as the well organized farmers treated their vinesand the more lackadaisical
5 neighbor shurried to do the pruning they should have done in november
```

使用 Google 翻译，得到中文翻译如图4所示。

杏树有意开花 日子更长 常常以波光粼粼的粉红色天空的壮丽夜晚结束  
狩猎季节结束，猎犬和枪支被收起六个月 葡萄园再次忙碌，因为井井  
有条的农民对待他们的葡萄树和更懒惰的邻居 匆匆忙忙地修剪他们本  
应在 11 月完成的修剪工作

图 4: 翻译后的明文

## 附录 A. 维吉尼亚密码的破译完整代码

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*英文字母使用频率表table*/
5 double table[]={0.08167, 0.01492, 0.02782, 0.04253, 0.12702, 0.02228, 0.02015,
6                 0.06094, 0.06966, 0.00153, 0.00772, 0.04025, 0.02406, 0.06749,
7                 0.07507, 0.01929, 0.00095, 0.05987, 0.06327, 0.09056, 0.02758,
8                 0.00978, 0.02360, 0.00150,0.01974, 0.00074};
9
10 /*用kasiski测试法获取key可能的长度*/
11 vector<int> Find_same(string cipher) {
12     vector<int> distance;
13     string p = cipher.substr(0, 3);
14     for (int i = 3; i < cipher.length()-3; ++i){
15         string tmp = cipher.substr(i, 3);
16         if (tmp == p)
17             distance.push_back(i);
18     }
19     return distance;
20 }
21
22 /*计算最大公约数*/
23 int gcd (int a, int b) {
```

```

24     if (b == 0) return a;
25     else return gcd(b, a % b);
26 }
27
28 /*计算所选分组的重合指数*/
29 double IC(string cipher, int start, int len) {
30     double result = 0.000;
31     int s = 0;
32     int n[26] = {0};
33     while (start <= cipher.length()) {
34         n[cipher[start]-'a']++;
35         start += len;
36         s++;
37     }
38     for (int i = 0; i < 26; ++i) {
39         if (n[i] <= 1) continue;
40         result += (double)(n[i]*(n[i]-1))/(double)((s)*(s-1));
41     }
42     return result;
43 }
44
45 /*确定密钥key的长度*/
46 int Get_keyLength(string cipher) {
47     int key_length;
48     vector<int> distance = Find_same(cipher);
49     int m = distance[0];
50     for (int i = 0; i < distance.size()-1; ++i)
51         for (int j = i+1; j < distance.size(); ++j)
52             m = min(m, gcd(distance[i], distance[j]));
53     for (int i = 1; i <= m; ++i) {
54         double sum = 0.000;
55         for(int j = 0; j < i; ++j) {
56             double temp = IC(cipher, j, i);
57             sum += temp;
58             cout << temp << " ";
59         }
60         cout << endl;
61         int value = (double)fabs(0.065-(double)(sum/(double)i));
62         if (fabs(sum/i - 0.065) < 0.01)
63             key_length = i;
64     }
65     cout << endl;
66     return key_length;
67 }
68

```

```

69  /*确定密钥*/
70  vector<int> Get_key(string cipher, int key_length){
71      vector<int> key(key_length, 0);
72      map<char, int> mp;
73      for (int i = 0; i < key_length; ++i) {
74          for (int j = 0; j < 26; ++j) {
75              mp.clear();
76              double pg = 0.000;
77              int sum = 0;
78              for (int k = i; k < cipher.length(); k += key_length) {
79                  char c = (char)((cipher[k] - 'a' + j) % 26 + 'a');
80                  mp[c]++;
81                  sum++;
82              }
83              for (char k = 'a'; k <= 'z'; ++k)
84                  pg += ((double)mp[k]/(double)sum)*table[k-'a'];
85              if (fabs(pg-0.065) < 0.01)
86                  key[i] = j;
87          }
88      }
89      return key;
90  }
91
92  int main(){
93      char c[500];
94      ifstream inFile("cipher.txt");
95      inFile.getline(c, 500);
96      inFile.close();
97      string cipher = c;
98
99      transform(cipher.begin(), cipher.end(), cipher.begin(), ::tolower);
100     int key_length = Get_keyLength(cipher);
101     vector<int> key = Get_key(cipher, key_length);
102
103     cout << "key_length: " << key_length << endl << "KEY: ";
104     for (int i = 0; i < key_length; ++i)
105         cout << (char)((26-key[i])%26+'a');
106     cout << endl << "TEXT:" << endl;
107     for (int i = 0; i < cipher.length(); ++i)
108         cout << (char)((cipher[i]-'a'+key[i%key_length])%26+'a');
109     cout << endl;
110     return 0;
111 }

```