密码学课程作业 Hw-01

武桐西　2112515

信息安全一班

2023 年 9 月 20 日

题目 1. 考虑下列定义在 $\mathbb{Z}_2$ 上的四级线性递归序列

$$z_{i+4} = (z_i + z_{i+1} + z_{i+2} + z_{i+3}) \bmod 2$$

$i \geqslant 0$。对其 16 种可能的初始向量 $(z_0, z_1, z_2, z_3) \in (\mathbb{Z}_2)^4$，分别求出其生成的密钥流的周期。

解答. 作出该 FSR 的状态转移表，如表1所示。

表 1: 状态转移表

Current State	Next State	Current State	Next State
(x_3, x_2, x_1, x_0)	(x_3, x_2, x_1, x_0)	(x_3, x_2, x_1, x_0)	(x_3, x_2, x_1, x_0)
0000	0000	1000	1100
0001	1000	1001	0100
0010	1001	1010	0101
0011	0001	1011	1101
0100	1010	1100	0110
0101	0010	1101	1110
0110	0011	1110	1111
0111	1011	1111	0111

根据状态转移表，作出状态转移图，如图1所示。

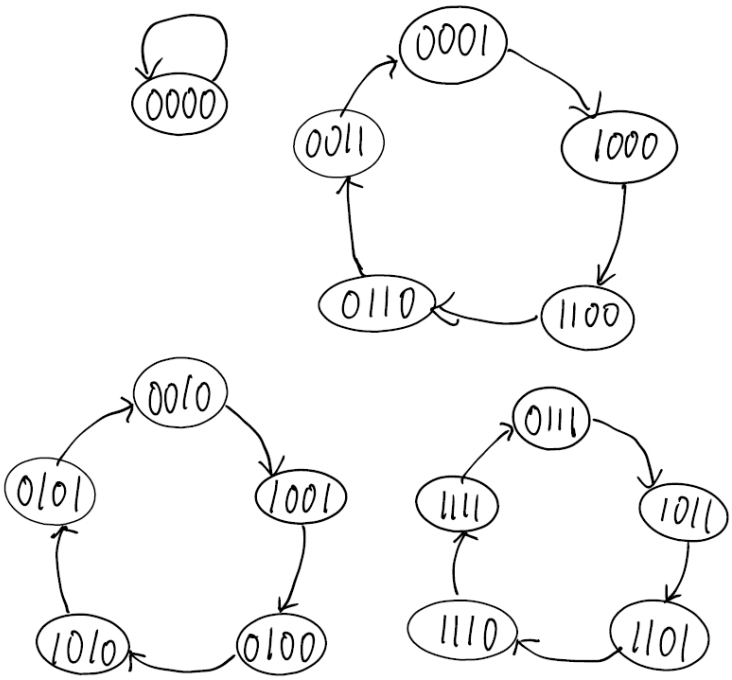

图 1: 状态转移图

由图1，当初始向量 $\mathbf{x} = (x_3, x_2, x_1, x_0)$ 为 0000 时，其生成的密钥流的周期为 1，当初始向量 $\mathbf{x} = (x_3, x_2, x_1, x_0)$ 取其他 15 个值时，其生成的密钥流的周期为 5.

题目 2. 以下为维吉尼亚密码加密的密文，请破译其明文。

K C C P K B G U F D P H Q T Y A V I N R R T M V G R K D N B V F D E T D G I L T X R G U D D K O T F M B P V G E G L T G C K Q R A C Q C W D N A W C R X I Z A K F T L E W R P T Y C Q K Y V X C H K F T P O N C Q Q R H J V A J U W E T M C M S P K Q D Y H J V D A H C T R L S V S K C G C Z Q Q D Z X G S F R L S W C W S J T B H A F S I A S P R J A H K J R J U M V G K M I T Z H F P D I S P Z L V L G W T F P L K K E B D P G C E B S H C T J R W X B A F S P E Z Q N R W X C V Y C G A O N W D D K A C K A W B B I K F T I O V K C G G H J V L N H I F F S Q E S V Y C L A C N V R W B B I R E P B B V F E X O S C D Y G Z W P F D T K F Q I Y C W H J V L N H I Q I B T K H J V N P I S T

解答.

(1) 首先采用 *Kasiski* **测试法**猜测该维吉尼亚密码使用的密钥的长度 m。

首先搜索长度为 3 的相同的密文段，并计算相同密文段之间的间距 δ_i，得到对应的结果如下：

密文段：['MVG', 'BVF', 'DDK', 'KFT', 'KFT', 'HJV', 'HJV', 'HCT', 'RLS', 'KCG', 'AFS', 'RWX', 'VYC', 'WBB', 'BBI', 'HJV', 'JVL', 'VLN', 'LNH', 'NHI', 'HJV']；其对应的间距：[156, 264, 198, 18, 156, 18, 138, 84, 18, 121, 60, 12, 42, 36, 36, 54, 54, 54, 54, 54, 12]。

容易发现，密文段中出现次数最多的是 HJV，其在密文中共出现了 5 次，其对应的间距分别为 18、138、54、12，因此猜测密钥长度 m 为它们的最大公因子 $\gcd(18, 138, 54, 12) = 6$，因此猜测密钥长度 $m = 6$。

上述分析过程的计算如代码1所示。

代码 1: guess_m.py: *Kasiski* 测试法猜测密钥长度 m

```python
# 利用 Kasiski 检测法猜测 Vigenere 密码的密钥长度

import re # 正则表达式
from collections import Counter

def get_gcd(a, b):
    # 求 a 和 b 的最大公因子

    if a < b:
        a, b = b, a
    while b != 0:
        a, b = b, a % b
    return a

def kasiski_guess(cipher_text):
    # 搜索长度为 3 的相同的密文，计算其间距

    # 去除密文中的空格
    cipher_text = re.sub(r'\s', '', cipher_text)

    # 搜索长度为 3 的相同的密文，计算其间距
    same_cipher_text = []
    same_cipher_text_length = []
```

```
    for i in  range( len(cipher_text) - 3):
        pos = cipher_text.find(cipher_text[i:i + 3], i + 3)
        if pos != -1:
            same_cipher_text.append(cipher_text[i:i + 3])
            same_cipher_text_length.append(pos - i)

    # 打印信息
    print('长度为 3 的相同的密文为: ')
    print(same_cipher_text)
    print('其间距为: ')
    print(same_cipher_text_length)

    # 找出重复最多的长度为 3 的密文
    same_cipher_text_most_common = Counter(same_cipher_text).most_common(1)[0][0]

    counter = []
    for i, s in  enumerate(same_cipher_text):
        if s == same_cipher_text_most_common:
            counter.append(same_cipher_text_length[i])

    # 求最大公因子
    gcd = counter[0]
    for i in  range(1,  len(counter)):
        gcd = get_gcd(gcd, counter[i])

    # 最大公因子很可能是密钥长度
    return gcd
```

为验证上述猜测的密钥长度的合理性，将密文分为 m 组，计算每组的重合指数 (Index of Coincidence) [1]，结果表2所示。

表 2: 密钥长度 $m = 6$ 时，每组的重合指数

分组	第 1 组	第 2 组	第 3 组	第 4 组	第 5 组	第 6 组
重合指数	0.063	0.084	0.049	0.065	0.043	0.073

可以看到每组的重合指数均比较接近 0.065，因此猜测密钥长度 $m = 6$ 很可能是正确的。

[1]该部分计算在代码2的前半部分。

(2) 接下来采用重合指数法（Index of Coincidence Method）来猜测密钥 $K = (k_1, k_2, k_3, k_4, k_5, k_6)$。

令 p_i 表示第 i 个英文字母出现的频率（$i = 0, 1, 2, \cdots, 25$，即 $i \in \mathbb{Z}_{26}$），f_i 表示当前分组中第 i 个英文字母出现的频数，n' 表示当前分组的密文长度。假设 $0 \leqslant g \leqslant 25$，定义数值

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'} \tag{1}$$

如果 $g = k_i$，则式 (1) 退化为重合指数，应该有 $M_g \approx \sum_{i=0}^{25} p_i^2 = 0.065$。如果 $g \neq k_i$，则 M_g 一般应该小于 0.065。[2]

由上述分析可知，可以取每组中 M_g 取最大值时对应的偏移 g 作为该组的密钥 k_i，即 $k_i = \arg\max_g M_g$。

综上，编程计算如代码2所示，得到解密后的明文如下：

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

将该明文进行单词划分并添加标点符号，可以解析出如下信息：

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.

代码 2: index_of_coincidence_method.py

```
1 # 采用重合指数法，破译维吉尼亚密码
2
```

[2]可由以下不等式得到：若 $a > b > 0$，$c > d > 0$，则 $ac + bd > ad + bc$。

```python
import re # 正则表达式
from guess_m import * # 导入 guess_m.py 中的 kasiski_guess() 和 get_gcd() 函数

# 字母频率表
letter_frequency_table = {
    # 大写字母
    'A': 0.082,
    'B': 0.015,
    'C': 0.028,
    'D': 0.043,
    'E': 0.127,
    'F': 0.022,
    'G': 0.020,
    'H': 0.061,
    'I': 0.070,
    'J': 0.002,
    'K': 0.008,
    'L': 0.040,
    'M': 0.024,
    'N': 0.067,
    'O': 0.075,
    'P': 0.019,
    'Q': 0.001,
    'R': 0.060,
    'S': 0.063,
    'T': 0.091,
    'U': 0.028,
    'V': 0.010,
    'W': 0.023,
    'X': 0.001,
    'Y': 0.020,
    'Z': 0.001
}

def shift_letter(letter, shift):
    '''
    字母移位
    :param letter: 字母
    :param shift: 移位数
    :return 移位后的字母
    '''
```

```
44
45      # 大写字母
46      if letter.isupper():
47          letter =  chr(( ord(letter) -  ord('A') + shift) % 26 +  ord('A'))
48      # 小写字母
49      elif letter.islower():
50          letter =  chr(( ord(letter) -  ord('a') + shift) % 26 +  ord('a'))
51      # 非字母
52      else:
53          pass
54
55      return letter
56
57  def index_of_coincidence_method(m, cipher_text):
58      '''
59      重合指数法
60      :param m: 密钥长度
61      :param cipher_text: 密文
62      :return 明文, 密钥
63      '''
64
65      # 过滤密文空白
66      cipher_text = re.sub(r'\s', '', cipher_text)
67
68      # 分组
69      cipher_text_group = []
70      for i in  range(m):
71          cipher_text_group.append(cipher_text[i::m])
72
73      # 计算每一组的重合指数, 以进一步辅助判断密钥长度 m 的正确性
74      index_of_coincidence_group = []
75      letter_count_group = []
76      for i in  range(m):
77          # 计算每个字母出现的次数
78          letter_count = {}
79          for letter in cipher_text_group[i]:
80              if letter in letter_count:
81                  letter_count[letter] += 1
82              else:
83                  letter_count[letter] = 1
84          letter_count_group.append(letter_count)
```

```
85
86          # 计算重合指数
87          cipher_text_length =  len(cipher_text_group[i])
88          index_of_coincidence = 0
89          for letter in letter_count:
90              index_of_coincidence += letter_count[letter] * (letter_count[letter] - 1)
91          index_of_coincidence /= cipher_text_length * (cipher_text_length - 1)
92          index_of_coincidence_group.append(index_of_coincidence)
93
94      # 打印每一组的重合指数
95      print('每一组的重合指数为: ')
96      for i in  range(m):
97          print('第 %d 组: ' % (i + 1), end='')
98          print(cipher_text_group[i], end=' ')
99          print(index_of_coincidence_group[i])
100
101     # 猜测每一组的密钥
102     key_group = []
103     for i, letter_count in  enumerate(letter_count_group):
104         M_max = 0
105         key = None
106         for g in  range(26):
107             # 计算M_g
108             cipher_text_length =  len(cipher_text_group[i])
109             M = 0
110             for letter in letter_frequency_table:
111                 try:
112                     M += letter_frequency_table[letter] * letter_count[shift_letter(letter, g)]
113                 except:
114                     # 此时出现频数为零的情况
115                     pass
116             M /= cipher_text_length
117
118             # 找出最大的 M_g
119             if M > M_max:
120                 M_max = M
121                 key = g
122         key_group.append(key)
123
124     # 解密
125     plain_text = ''
```

```
126     for i in  range( len(cipher_text)):
127         plain_text += shift_letter(cipher_text[i], -key_group[i % m]) # 解密：移位数取反
128     plain_text = plain_text.lower()
129 
130     return plain_text, key_group
131 
132 
133 if __name__ == '__main__':
134     cipher_text = """
135     KCCPKBGUFDPHQTYAVINRRTMVGRKDNBVFDETDGILTXRGUD
136     DKOTFMBPVGEGLTGCKQRACQCWDNAWCRXIZAKFTLEWRPTYC
137     QKYVXCHKFTPONCQQRHJVAJUWETMCMSPKQDYHJVDAHCTRL
138     SVSKCGCZQQDZXGSFRLSWCWSJTBHAFSIASPRJAHKJRJUMV
139     GKMITZHFPDISPZLVLGWTFPLKKEBDPGCEBSHCTJRWXBAFS
140     PEZQNRWXCVYCGAONWDDKACKAWBBIKFTIOVKCGGHJVLNHI
141     FFSQESVYCLACNVRWBBIREPBBVFEXOSCDYGZWPFDTKFQIY
142     CWHJVLNHIQIBTKHJVNPIST
143     """
144 
145     # 猜测密钥长度
146     m = kasiski_guess(cipher_text)
147     print('密钥长度为：{}'. format(m))
148 
149     # 重合指数法破译密钥并解密
150     plain_text, key_group = index_of_coincidence_method(m, cipher_text)
151     print('密钥为：')
152     print(key_group)
153     print('明文为：')
154     print(plain_text)
155 
156 # ------Output------
157 '''
158 长度为 3 的相同的密文为：
159 ['MVG', 'BVF', 'DDK', 'KFT', 'KFT', 'HJV', 'HJV', 'HCT', 'RLS', 'KCG', 'AFS', 'RWX', 'VYC', 'WBB', '
        BBI', 'HJV', 'JVL', 'VLN', 'LNH', 'NHI', 'HJV']
160 其间距为：
161 [156, 264, 198, 18, 156, 18, 138, 84, 18, 121, 60, 12, 42, 36, 36, 54, 54, 54, 54, 54, 12]
162 密钥长度为：6
163 每一组的重合指数为：
164 第 1 组：KGQNGVGGTGCQWAWQHNJEPJTKQFWAPJGHPWKCTAQVNCIVJFVNIVCPQJQJT 0.06265664160401002
165 第 2 组：CUTRRFIUFEKCCKRKKCVTKVRCDRSFRRKFZTEEJFNYWKKKVFYVRFDFIVIV 0.08376623376623377
```

```
第 3 组: CFYRKDLDMGQWRFPYFQAMQDLGZLJSJJMPLFBBRSRCDAFCLSCREEYDYLBN 0.04935064935064935
第 4 组: PDATDETDBLRDXTTVTQJCDASCXSTIAUIDVPDSWPWGDWTGNQLWPXGTCNTP 0.06493506493506493
第 5 组: KPVMNTXKPTANILYXPRUMYHVZGWBAHMTILLPHXEXAKBIGHEABBOZKWHKI 0.04285714285714286
第 6 组: BHIVBDROVGCAZECCOHWSHCSQSCHSKVZSGKGCBZCOABOHISCBBSWFHIHS 0.07337662337662337
密钥为:
[2, 17, 24, 15, 19, 14]
明文为:
ilearnedhowtocalculatetheamountofpaperneededforaroomwheniwasatschoolyoumultiplythesquare
footageofthewallsbythecubiccontentsofthefloorandceilingcombinedanddoubleityouthenallowha
lfthetotalforopeningssuchaswindowsanddoorsthenyouallowtheotherhalfformatchingthepatternt
henyoudoublethewholethingagaintogiveamarginoferrorandthenyouorderthepaper
'''
```