crackWithFreq

发表于 2021-01-16 分类于 Challenge， 2019， 安洵杯， Crypto
Challenge | 2019 | 安洵杯 | Crypto | crackWithFreq

点击此处获得更好的阅读体验

WriteUp来源

https://xz.aliyun.com/t/6912

题目考点

解题思路

利用字母频率破解密文。

首先使用重合指数法猜接触密钥长度，得到长度为12。这里解出出来的长度其实是key1 key2 长度的最小公倍数。然后，将密文中的 每个字母以 12 为间隔分 12 组（假如密文是: ABCDEFGHIJKLMN，以 3 为间隔分一 组，那么 ADGJM 就是一组）。

这样每组既可以看作一个仿射密码的破解，这时密钥空间只有256，可以爆破利用字母频率进行破解。

```
# -*- coding: utf-8 -*-
from pycipher import Affine
import string

table = string.ascii_lowercase

englishExpectedFrequencies = {
    'a': 0.08167, 'b': 0.01492, 'c': 0.02782, 'd': 0.04253,
    'e': 0.12702, 'f': 0.02228, 'g': 0.02015, 'h': 0.06094,
    'i': 0.06966, 'j': 0.00153, 'k': 0.00772, 'l': 0.04025,
    'm': 0.02406, 'n': 0.06749, 'o': 0.07507, 'p': 0.01929,
    'q': 0.00095, 'r': 0.05987, 's': 0.06327, 't': 0.09056,
    'u': 0.02758, 'v': 0.00978, 'w': 0.02361, 'x': 0.00150,
    'y': 0.01974, 'z': 0.00074
    }

dic = {1: 1, 3: 9, 5: 21, 7: 15, 9: 3, 11: 19, 15: 7, 17: 23, 19: 11, 21: 5, 23: 17, 25: 25}


# 找出假定秘钥长度内的最可能长度
# 所用的方法：重合指数法
def decryptFirstStage(toDecrypt):   # 将密文传入。
    min_len = 3
    max_len = 15
    toDecrypt = toDecrypt.lower()   # 将密文转为小写

    best_len = 0
    best_aver = 0

    best_rate = 0.65
    min_rate = 100

    for i in range(0, len(toDecrypt)):  # 每次循环测试一个密钥长度。
        lengthOfKey = i + 1
        averageIC = 0.0             # 重置 averageIC
        sequenceDictionary = {}     # 序列字典或用于统计分组。
        hadZeroError = False  # hadZeroError 或用于预防某种错误的计算。

        # 此循环用于生成分组字典 sequenceDictionary
        for index in range(0, len(toDecrypt)):
            sequenceNumber = index % lengthOfKey    # 密文中的第 index 个字符应该属于那个分组。
            if sequenceNumber in sequenceDictionary:    # 分组若存在，则将 toDecrypt[index] 加入分组字符串，如不存在，则先创建再添加。
                sequenceDictionary[sequenceNumber] += toDecrypt[index]
            else:
                sequenceDictionary[sequenceNumber] = toDecrypt[index]

        # 此循环用于生成各个分组的重合指数和 averageIC
        for stringSequence in sequenceDictionary.values():
            try:
                averageIC += calculateIC(stringSequence)    # 统计各个分组的重合指数，并求和。最后储存在 averageIC 中。引入了自定义函数 calculateIC()
            except ZeroDivisionError:
                hadZeroError = True
                break

        if hadZeroError == True:
            averageIC = 0
        else:
            averageIC /= len(sequenceDictionary.keys())     # averageIC 求平均值。

        rate = abs(1 - (averageIC / best_rate))

        # 这个判断用于选出最佳长度
        if (min_len <= lengthOfKey <= max_len) and (rate < min_rate):     # 判断条件为： averageIC 最大的一组 & 密钥长度区间在[3,5]
            min_rate = rate                                               # 考虑是否可以修改循环次数？
            best_len = lengthOfKey\

    # 找出指定秘钥长度范围内averageIC最大的那个秘钥长度
    # print('最佳长度：', best_len)
    return best_len


# 用于计算重合指数，输入类型为 str
def calculateIC(inputText):
    inputText = "".join(inputText.lower().split())  # 是否可以省略这一步？
    frequency = getFrequencyOfText(inputText)       # 获取字母-次数字典。
    ic = 0.0

    # 循环26个小写字母
    for letter in table:
        if letter in frequency:
            ic += frequency[letter] * (frequency[letter] - 1)

    ic /= len(inputText) * (len(inputText) - 1)     # 重合指数计算公式。
    return ic


def getFrequencyOfText(inputText):
    frequency = {}
    for letter in inputText:
        if letter in frequency:
            frequency[letter] += 1
        else:
            frequency[letter] = 1
    return frequency
```

```
95
96
97 def getGroups(raw, block):
98     groups = []
99     for i in range(block):
100         k = i
101         part = ""
102         while True:
103             try:
104                 part += raw[k]
105                 k += block
106             except:
107                 break
108         groups.append(part)
109     return groups
110
111
112 def getEnglishScore(inputText):
113     """计算英文字符串的“评分”，计算方法为：
114     Score = (字符串中各个字母的数量 * 其对应的字母频率)的总和 / 字符串去掉空格后的长度
115
116     :param inputText: 英文字符串 string
117     :return: 评分 int
118     """
119     inputText = inputText.lower().replace(" ", "")
120     score = sum([englishExpectedFrequencies.get(char, 0) for char in inputText]) / len(inputText)
121
122     return score
123
124

125 def crack(cipher):
126     fitness = float("-inf")
127     bestResult = ""
128     key_a = None
129     key_b = None
130     for i in dic.keys():
131         for j in range(0, 26):
132             af = Affine(a=i, b=j)
133             result = af.decipher(cipher)
134             bestFitness = getEnglishScore(result)
135             if bestFitness > fitness:
136                 key_a = i
137                 key_b = j
138                 fitness = bestFitness
139                 bestResult = result
140
141     return bestResult, key_a, key_b
142
143
144 en = "ltpflwfkqnyfmbjbchqnadkaykyhgpzaezjfrfkdonetcvcrkaaronhdnvghmyzwshrhefgqjfbrphqmgvglgvlfonzzqngxqfsessrhphupnvlfxsxotmzccnqfvmfdlhujqvezonbhsnsgykffzmbhefxtrrfjqsx
145 length = decryptFirstStage(en)
146 print "Length is: %d" % length
147 gp = getGroups(en, length)
148 key1 = ""
149 key2 = ""
150 for p in gp:
151     res, a, b = crack(p)
152     key1 += table[a]
153     key2 += table[b]
154 print key1 + '\n' + key2
```

Flag

- ***本文作者：*** *CTFHub*
- ***本文链接：*** *https://writeup.ctfhub.com/Challenge/2019/安洵杯/Crypto/2nvkMcxxF1GtxviD1rNBGv.html*
- ***版权声明：*** *本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！*

 # Challenge # 2019 # Crypto # 安洵杯
This_is_not_a_standard_AES256
Blindpwn-32位有偏移