# 目录

# 简单题

## Morse

### *解题思路*

摩尔斯电码转字符，观察会发现是十六进制数，转ASCII码得flag

### *flag*

```
afctf{1s't_s0_345y}
```

## BASE

### *解题思路*

现在放出加密代码

```
import base64
import random
import os
import sys
with open('flag.txt', 'r') as file:
    flag = bytes(file.read(),'ascii')
    file.close()

for i in range(0, 30):
    print(i)
    a = random.randint(0, 2)
    if a == 2:
        flag = base64.b64encode(flag)
    elif a == 1:
        flag = base64.b32encode(flag)
    else:
        flag = base64.b16encode(flag)
```

```
with open('flag_encode.txt', 'w') as file:
    file.write(str(flag)[2:-1])
    file.close()
```

这个问题的关键在于，经过30重加密的文件不可能会小，直接用文本编辑器打开的都完蛋啦。正确姿势是使用十六进制编辑器或者word打开，或者用程序读入一部分后查看内容。

此外根据题目可以大抵知道这是Base编码。

解密代码如下，可以在出题人博客找到主要部分

```
# -*- coding: utf-8 -*-
# Python3
# Solution.py
from base64 import *
s = ""
with open('flag_encode.txt', 'r') as file:
    s = bytes(file.read(),'ascii')
    file.close()

lis1 = [s]
lis2 = []
lis3 = []
lis4 = []
while(1):
    for a in lis1:
        ok = 0
        try:
            lis2.append(b64decode(a).decode('ascii'))
            ok = 1
        except:
            pass
        try:
            lis2.append(b32decode(a).decode('ascii'))
            ok = 1
        except:
            pass
        try:
            lis2.append(b16decode(a).decode('ascii'))
            ok = 1
        except:
            pass
        if not ok:
            lis3.append(a)
    if not len(lis2):
        break
    lis1=lis2.copy()
    lis2.clear()
for a in range(0,len(lis3)):
    ok = 1
    for b in lis3[a]:
        if ord(b)>126 or ord(b)<32:
            ok = 0
            break
    if ok:
```

```
        lis4.append(lis3[a])
print(lis4)
```

### *flag*

afctf{U_5h0u1d_Us3_T00l5}

## MagicNum

### *解题思路*

这个题可能有点脑洞的成分了，结果导致做出来的人很少。其实提示已经很明显了。

加密代码如下：

```
#include <stdio.h>
char flag[]="afctf{sec_is_everywhere}";

int main()
{
    for(int i=0;i<6;++i){
        printf("%20f\n",*(float*)(flag+i*4));
    }
    return 0;
}
```

### *flag*

afctf{sec_is_everywhere}

## Single

### *解题思路*

这是古典密码学中经典的加密方式，单表替换。将a-z映射到a-z排列。

虽然密钥空间达到了 $26!$ 但是可以被频率分析很轻松地解决。

此外为了降低难度，让新人也能靠自己体验一把解密的感觉，将flag以原样放了进去，并且放出了加密代码，即便不了解频率分析也可以手动推算出来。

工具的话用https://quipqiup.com/，可以秒解

### *flag*

```
afctf{Oh_U_found_it_nice_tRy}
```

### *非预期解*

C++标准库里的random_shuffle十分腊鸡……以至于我只放出加密代码的情况下，在另一台机器另一个编译器上……运行得到的结果是一样的……导致直接用加密代码改改就出flag的愚蠢情况……

出题人表示背锅，并下次手写随机数生成来解决。

## Vigenère

### *解题思路*

多表替换，密钥是csuwangjiang

因为我的昵称和大学都是可见信息，所以又算脑洞了？

用工具可以秒解，随便放几个吧。

一个维吉尼亚：

- Vigenère and Gronsfeld Cipher

另一个维吉尼亚：https://atomcated.github.io/Vigenere/

另一个维吉尼亚：http://www.mygeocachingprofile.com/codebreaker.vigenerecipher.aspx

另一个维吉尼亚：https://f00l.de/hacking/vigenere.php

具体原理就还是频率分析、常见词的替换，比如中间那句可以推测是`flag is afctf{....}`

### *flag*

```
afctf{Whooooooo_U_Gotcha!}
```

# 你能看出这是什么加密么？

## *解题思路*

裸RSA

```
!/usr/bin/python

import libnum

p=int('0x928fb6aa9d813b6c3270131818a7c54edb18e3806942b88670106c1821e0326364194a8c49392849432b37632f0abe3f3c52e909b939c91c50e41a7b8cd00c67d6743b4f',16)

q=int('0xec301417ccdffa679a8dcc4027dd0d75baf9d441625ed8930472165717f4732884c33f25d4ee6a6c9ae6c44aedad039b0b72cf42cab7f80d32b74061',16)

e=int('0x10001', 16)

c=int('0x70c9133e1647e95c3cb99bd998a9028b5bf492929725a9e8e6d2e277fa0f37205580b196e5f121a2e83bc80a8204c99f5036a07c8cf6f96c420369b4161d2654a7eccbdaf583204b645e137b3bd15c5ce865298416fd5831cba0d947113ed5be5426b708b89451934d11f9aed9085b48b729449e461ff0863552149b965e22b6',16)

n=p*q

phi=(p-1)*(q-1)

d = libnum.modular.invmod(e, phi)

print libnum.n2s(pow(c, d, n))
```

## *flag*

afctf{R54_|5_$0_$imp13}

# 可怜的RSA

## *解题思路*

1. 用OpenSSL查看公钥，尝试将其分解成私钥

```
openssl rsa -noout -text -inform PEM -in public.key -pubin
```

1. 将十六进制转为十进制

```
python -c "print
int('25b18bf5f389097d17237866bb51cff8de922453749ebc403b0995c97c0e386d46c161cadff77c69860dae4791c
214cf8487aaaa9f26e920a977834906038aefb5c30827dfcf3fc9e9769544f94e07cdfe0872039a3a6262116678b261f
b2d6b9d32539e92a153b3675629bab3942e7d35e30f7eef5abf1c50d797d0cc88e1bdccfd1a12ea6f7ef75c3727dbdf2
e780f3428ae8f7a4fb7a89f184a365032b153f8425e845750eb2b7abc02dc15ce0207507aa950863bb8480a78028dd62
979944d6c633fafa103e4db28ce87f5a0c6ed4a2f2664427f565c7781ab6191456d971c7ffa395272374cec0155e5f91
189db742e4c28b03a0fa11cffb03173d2a4cce6ae53', 16)"
```

1. 用网站对其分解（http://www.factordb.com/index.php?query=798321817573328185527646107613495929846147444322791353283989998016278802836109003612812499731758050699162101795605064970751325249020868811203722136266418794684919368609766869336308696738269726199383219515991467448076533010760265779495796183315027763039834855660464854310395417084671414082602200985927612450106785923475018941762695805104597296336734680684671441997445637318263621026088110334008878137547802826280994434901700160878386069980174904566013158024485677724116238262817472456609542454137815197942953361975556885435379921971422580532204537576665378402764164756027593749507152838902322307415427373195698197939884314 43）

得到p、q

1. 然后用rsatool生成私钥，发现flag.enc还经过base64加密，所以最后

```
def decrypt_RSA(privkey, message):
    from Crypto.PublicKey import RSA
    from Crypto.Cipher import PKCS1_OAEP
    from base64 import b64decode
    key = open(privkey, "r").read()
    rsakey = RSA.importKey(key)
    rsakey = PKCS1_OAEP.new(rsakey)
    decrypted = rsakey.decrypt(b64decode(message))
    return decrypted

flag =
"GVd1d3viIXFfcHapEYuo5fAvIiUS83adrtMW/MgPwxVBSl46joFCQ1plcnlDGfL19K/3PvChV6n5QGohzfVyz2Z5GdTlakn
xvHDUGf5HCukokyPwK/1EYU7NzrhGE7J5jPdi0Aj7xi/Odxy0hGMgpaBLd/nL3N8O6i9pc4Gg3O8soOlciBG/6/xdfN3SzSS
tMYIN8nfZZMSq3xDDvz4YB7TcTBh4ik4wYhuC77gmT+HWOv5gLTNQ3EkZs5N3EAopy11zHNYU80yv1jtFGcluNPyXYttU5qU
33jcp0Wuznac+t+AZHeSQy5vk8DyWorSGMiS+J4KNqSVlDs12EqXEqqJ0uA=="
print decrypt_RSA('priv.key', flag)
```

## *flag*

afctf{R54_|5_$0_B0rin9}

# One Secret, Two encryption

## *解题思路*

素数复用，求gcd即可得到一个素数，随便求一对，然后解密得到flag

### *flag*

```
flag is afctf{You_Know_0p3u55I}
```

# 中等题

## Tiny LFSR

### *解题思路*

LFSR的下一位只由当前决定，通过一对明文密文异或获得初始密钥，然后进行解密即可

### *flag*

```
afctf{read_is_hard_but_worthy}
```

## MyOwnCBC

### *解题思路*

读了代码应该知道，其实并不是CBC而是愚蠢的自创加密模式，使用上一步的密文作为新一步的密钥。

所以直接读密文然后解密就行，flag也放在了末尾。

### *flag*

```
afctf{Don't_be_fooled_by_yourself}
```

# 中难题

# 你听过一次一密么？

## *解题思路*

Many-Time-Pad了解一下？

按这个名词去谷歌能搜到很多分析

解密代码如下：

```
#!/usr/bin/python
## OTP - Recovering the private key from a set of messages that were encrypted w/ the same
private key (Many time pad attack) - crypto100-many_time_secret @ alexctf 2017
# Original code by jwomers: https://github.com/Jwomers/many-time-pad-
attack/blob/master/attack.py)

import string
import collections
import sets, sys

# 11 unknown ciphertexts (in hex format), all encrpyted with the same key

c1='25030206463d3d393131555f7f1d061d4052111a19544e2e5d'
c2='0f020606150f203f307f5c0a7f24070747130e16545000035d'
c3='1203075429152a7020365c167f390f1013170b1006481e1314'
c4='0f4610170e1e2235787f7853372c0f065752111b15454e0e09'
c5='081543000e1e6f3f3a3348533a270d064a02111a1b5f4e0a18'
c6='0909075412132e247436425332281a1c561f04071d520f0b11'
c7='4116111b101e2170203011113a69001b475206011552050219'
c8='0410060646122970203754533 42c17545a01451811411a470e'
c9='021311114a5b0335207f7c167f22001b44520c15544801125d'
c10='06140611460c26243c7f5c167f3d0154460100530059071 45d'
c11='0f05110d160f263f3a7f4210372c03111313090415481d49'
ciphers = [c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11]
# The target ciphertext we want to crack
#target_cipher = "0529242a631234122d2b36697f13272c207f2021283a6b0c7908"

# XORs two string
def strxor(a, b):     # xor two strings (trims the longer input)
    return "".join([chr(ord(x) ^ ord(y)) for (x, y) in zip(a, b)])

def target_fix(target_cipher):
    # To store the final key
    final_key = [None]*150
    # To store the positions we know are broken
    known_key_positions = set()

    # For each ciphertext
    for current_index, ciphertext in enumerate(ciphers):
        counter = collections.Counter()
        # for each other ciphertext
        for index, ciphertext2 in enumerate(ciphers):
            if current_index != index: # don't xor a ciphertext with itself

                for indexOfChar, char in enumerate(strxor(ciphertext.decode('hex'),
```

```
ciphertext2.decode('hex'))): # Xor the two ciphertexts
                    # If a character in the xored result is a alphanumeric character, it means
there was probably a space character in one of the plaintexts (we don't know which one)
                    if char in string.printable and char.isalpha(): counter[indexOfChar] += 1 #
Increment the counter at this index
        knownSpaceIndexes = []

        # Loop through all positions where a space character was possible in the current_index
cipher
        for ind, val in counter.items():
            # If a space was found at least 7 times at this index out of the 9 possible XORS,
then the space character was likely from the current_index cipher!
            if val >= 7: knownSpaceIndexes.append(ind)
        #print knownSpaceIndexes # Shows all the positions where we now know the key!

        # Now Xor the current_index with spaces, and at the knownSpaceIndexes positions we get
the key back!
        xor_with_spaces = strxor(ciphertext.decode('hex'),' '*150)
        for index in knownSpaceIndexes:
            # Store the key's value at the correct position
            final_key[index] = xor_with_spaces[index].encode('hex')
            # Record that we known the key at this position
            known_key_positions.add(index)

    # Construct a hex key from the currently known key, adding in '00' hex chars where we do not
know (to make a complete hex string)
    final_key_hex = ''.join([val if val is not None else '00' for val in final_key])
    # Xor the currently known key with the target cipher
    output = strxor(target_cipher.decode('hex'),final_key_hex.decode('hex'))

    print "Fix this sentence:"
    print ''.join([char if index in known_key_positions else '*' for index, char in
enumerate(output)])+"\n"

    # WAIT.. MANUAL STEP HERE
    # This output are printing a * if that character is not known yet
    # fix the missing characters like this: "Let*M**k*ow if *o{*a" = "cure, Let Me know if you
a"
    # if is too hard, change the target_cipher to another one and try again
    # and we have our key to fix the entire text!

    #sys.exit(0) #comment and continue if u got a good key

    target_plaintext = "cure, Let Me know if you a"
    print "Fixed:"
    print target_plaintext+"\n"

    key = strxor(target_cipher.decode('hex'),target_plaintext)

    print "Decrypted msg:"
    for cipher in ciphers:
        print strxor(cipher.decode('hex'),key)

    print "\nPrivate key recovered: "+key+"\n"

for i in ciphers:
    target_fix(i)
```

## *flag*

```
afctf{OPT_1s_Int3rest1ng}
```

# 花开藏宝地

## *解题思路*

题目里的藏宝图 + 题面里提到只要集3份就能解密->门限方案

花开->bloom门限方案

也是一次尝试吧，尝试除了加密与编码外的考点

1. secret1 生日字典/脑洞 19260817
2. secret2 小写爆破 alice
3. secret3 大写爆破 AVADA
4. secret4 伪加密
5. secret5 NTFS隐写

任意取得三份后用bloom门限方案解，素数为题面的数字

代码（取1/2/3）：

```
a1
=16330503996300832270095867893842065503910858484859423647303655613020629222976196145963535510552
91199559507691190006478211663024099877261814566242338202380041305965825521430520858265627719386 5
33147222885839567947401828693369271410531107399812902378941121527208220142402309720118486835764 0
25359948253090298227618556239036113357520596666833775369200524286483023894266096721185220035103 9
8578217
d1
=34705155962246314453966995009665816342564641143579769197370151372570157510081044617584942400000
00758550704302405077327353934114938665405726796261727423013661465018626702724430709705119434858 6
58874942294874205037504579742628020537220939051262353403802618285935084556216673099463617055306 6
79574847319291518755274894784493611986483106847025746271993210929271111373983330296970684747628 2
0813413
a2
=15175810009332802475553436215715264491668955680040709163807726215205135637468742600269130833136
09116586816756211807840784643005577135976586687377552755783036835127636514244906966630466597622 0
94594010958034072340747931440347997989374630859893646588094894730168145642843742530471112853075 6
89380115714826137617217463386198799409283807413773673815174273416796418711260769912091769353390 5
8909863
```

```
d2
```

```
=34705155962246314453966995009665816342564641143579769197370151372570157510081044617584942400000
00758550704302405077327353934114938665405726796261727423013661465018626702724430709705119434858 6
588749422948742050375045797426280205372209390512623534038026182859350845562166730994636170553066
795748473192915187552748947844936119864831068470257462719932109292711113739833302969706847476282
0818553
a3 =
34607759206825939935008037976794198200379437373605809772372810402081480089768682869302621572369 5
173898771936691822530717642440410239211631306801809213192374695040232378965389612021366734818648
00727533232262106465919968084874524270075544020694946595344127786641961796123223420108371621603 1
999849609543380477085554544227121956015035672626500140341901966363694497881768843758979050832435
224875
d3
=34705155962246314453966995009665816342564641143579769197370151372570157510081044617584942400000
00758550704302405077327353934114938665405726796261727423013661465018626702724430709705119434858 6
588749422948742050375045797426280205372209390512623534038026182859350845562166730994636170553066
795748473192915187552748947844936119864831068470257462719932109292711113739833302969706847476282
0819351

dd = d1*d2*d3
t1 = pow(dd//d1,d1-2,d1)
assert(t1*d2*d3%d1 == 1)
t2 = pow(dd//d2,d2-2,d2)
assert(t2*d1*d3%d2 == 1)
t3 = pow(dd//d3,d3-2,d3)
assert(t3*d2*d1%d3 == 1)
s = a1*t1*d2*d3+a2*t2*d1*d3+a3*t3*d1*d2
p =
80804238007977405688648566160504278593148666302626415149704905628622876270862865768337953835725 8
019631426851825108129380721159963557823963183039270207056231206520140800328094211804009842420615
925207337102434839472309626319450451345401595174882887816666226353283169729791837619528420108063
047483133262156196950853805860525504430250745019719250050729992756285497109153574009464088 57
s %= dd
print(s)
s %= p
print(s)
```

最后结果转HEX转ASCII即可

题目生成代码有需要的邮件我叭

## *flag*

afctf{1sn't_s0_int3Resting}

## *非预期解*

后来发现自己又犯蠢了。。。

因为选的素数……是用nextprime()选的，所以相差……不大……

你可以直接把那个素数转成字符串……和明文只有……最后一个字符不一样……

**ZZ点数 + 10**

# 一道有趣的题目

## *解题思路*

这道题来自dctf2015

```
#求出加密的比特位
def decrypt(cipherText):
    guessed_bits = ['?'] * len(cipherText)
    length = len(cipherText)
    i = 0
    orded_cipher = [ord(c) & 1 for c in cipherText]
    decrypt_r(orded_cipher, guessed_bits, i, length, 10)


def try_guess(orded_cipher, guessedbits, i, length, guess, space):
    guessedbits = list(guessedbits)
    guessedbits[i] = guess
    if i + space < length - 1:
        nextndx = i + space
    else:
        nextndx = space

    nextbit = orded_cipher[i] ^ guess
    if guess == 0:
        newspace = space + 1
    else:
        newspace = space - 1

    if guessedbits[nextndx] == '?' or guessedbits[nextndx] == nextbit:
        guessedbits[nextndx] = nextbit
        decrypt_r(orded_cipher, guessedbits, i + 1, length, newspace)

def decrypt_r(orded_cipher, guessedbits, i, length, space):
    if i >= length:
        print 'ok:', ''.join(str(c) for c in guessedbits)
        return
    if guessedbits[i] == '?':
        try_guess(orded_cipher, guessedbits, i, length, 0, space)
        try_guess(orded_cipher, guessedbits, i, length, 1, space)
    elif guessedbits[i] == 0:
        try_guess(orded_cipher, guessedbits, i, length, 0, space)
    elif guessedbits[i] == 1:
        try_guess(orded_cipher, guessedbits, i, length, 1, space)
    #print ''.join(str(c) for c in guessedbits)
```

```
s='15120d1a0a0810010a031d3e31000d1d170d173b0d173b0c07060206'
#print len(s)
s=s.decode('hex')
#print len(s)

decrypt(s)

#求出明文
sln='10100110100101011111111101001'
ciph=s
print sln

import string
slv = [None] * len(sln)
def filling_pass(slv):
    while True:
        any = False
        space = 10
        for i in range(len(sln)):
            if i + space < len(sln) - 1:
                nx = i + space
            else:
                nx = space
            if sln[i] == '0':
                space += 1
            else:
                space -= 1
            if slv[i] is not None:
                sn = ord(slv[i]) ^ ord(ciph[i])
                if slv[nx] is None:
                    slv[nx] = chr(sn)
                    if (sn >= 32 and sn < 127) or sn == 10 or sn == 13:
                        any = True
                    else:
                        return False
                else:
                    if slv[nx] != chr(sn):
                        return False
        if not any:
            return True

def tryit(slvo, start):
    while slvo[start] is not None:
        start += 1
        if start >= len(slvo):
            print ''.join(' ' if c is None else '.' if ord(c) < 32 else c for c in slvo)
            return
        continue
    for c in string.printable:
        slv = list(slvo)
        slv[start] = c
        possible = filling_pass(slv)
        if possible:
            tryit(slv, start)

print tryit(slv, 0)
```

### *flag*

```
afctf{cryptanalysis_is_hard}
```

# 难题

## Fast LFSR

### *解题思路*

使用LFSR生成流密钥，具体名称是Geffe Generator

攻击方式是快速相关攻击（Fast Correlation Attack）

原题出自强网杯StreamGame3，出题的时候还搜不到可用的WP

具体请查看论文以及网上WP如这篇

### *flag*

```
afctf{01abcd056789123456}
```