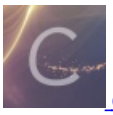
攻防世界-CRYPTO-新手练习区WP

原创

T2cer 于 2021-06-15 17:02:00 发布 5535 收藏 2

分类专栏： crypto

本文链接：https://blog.csdn.net/sinrier/article/details/117828842



攻防世界-CRYPTO做题记录

base64

题目

直接用base64工具.

Caesar

题目

方法一：直接上工具，根据flag格式简单算一下位移多少位时第一位会出现c字母，最后得出位移数位12

方法二：暴力解密

```
#include <stdio.h>
#include <stdlib.h>
/* 穷举法解密 */
int main()
{
    char word_1[1000] = "\0"; //解密前
    char word_2[1000] = "\0"; //解密后
    int i = 0;
    int j = 0;
    printf("输入密文：");
    scanf("%s", word_1);
    for(i = 0; i < 26; i++)
    {
        for(j = 0; word_1[j] != '\0'; j++)
        {
            if(word_1[j] >= 'A' && word_1[j] <= 'Z')
                word_2[j] = (word_1[j] - 'A' + i) % 26 + 'A';   //将密钥key换成i进行尝试
            else if(word_1[j] >= 'a' && word_1[j] <= 'z')
                word_2[j] = (word_1[j] - 'a' + i) % 26 + 'a';
        }
        printf("第%d次尝试：%s\n", i + 1, word_2);
    }
    return 0;
}
```

ps：使用前删掉密文中的符号

Morse

题目

打开题时一串0和1

1为-，0为.，空格直接空得到标准的摩斯密码然后直接用工具解

题目要求为小写再转换一下就可以了

```
11 111 010 000 0 1010 111 100 0 00 000 000 111 00 10 1 0 010 0 000 1 00 10 110
-- --- .-. ... . -.-. --- -.. . .. ... ... --- .. -. - . .-. . ... - .. -. --.
MORSECODEISSOINTERESTING
morsecodeissointeresting
```

幂数加密

题目

幂数加密介绍

以0为界限隔断数字,每项数字相加得到的结果即为字母表中对应的字母顺序

```
88421 0 122 0 48 0 2244 0 4 0 142242 0 248 0 122
8+8+4+2+1=23->W
1+2+2=5->E
4+8=12->L
2+2+4+4=12->L
4->D
1+4+2+2+4+2=15->O
2+4+8=14->N
1+2+2=5->E
```


或者写个脚本解

```python
# /usr/bin/env python
# coding=utf-8
a = "884210122048022440401422420248012 2"
a = a.split("0")
flag = ''
for i in range(0, len(a)):
    str = a[i]
    list = []
    sum = 0
    for j in str:
        list.append(j)
        length = len(list)
    for k in range(0, length):
        sum += int(list[k])
    flag += chr(sum + 64)
print(flag)
```

Railfence

题目

试了挺久普通的栅栏解码都没解才想到可能是用了W型栅栏加密

工具

或者用脚本解

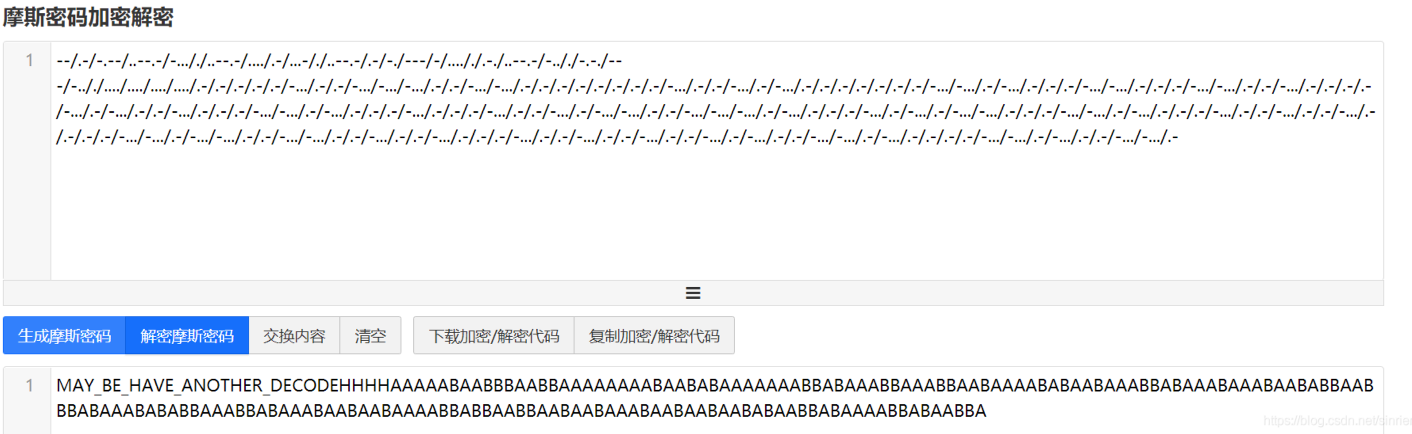
```
'''
若知道栏数，则使用decode解密，若不知道，则使用crack_cipher遍历所有可能性
'''

def generate_w(string, n):
    '''将字符排列成w型'''
    array = [['.'] * len(string) for i in range(n)]  # 生成初始矩阵
    row = 0
    upflag = False
    for col in range(len(string)):  # 在矩阵上按w型画出string
        array[row][col] = string[col]
        if row == n - 1:
            upflag = True
        if row == 0:
            upflag = False
        if upflag:
            row -= 1
        else:
            row += 1
    return array


def encode(string, n):
    '''加密'''
    array = generate_w(string, n)
    msg = []
    for row in range(n):  # 将每行的字符连起来
        for col in range(len(string)):
            if array[row][col] != '.':
                msg.append(array[row][col])
    return array, msg


def decode(string, n):
    '''解密'''
    array = generate_w(string, n)
    sub = 0
    for row in range(n):  # 将w型字符按行的顺序依次替换为string
        for col in range(len(string)):
            if array[row][col] != '.':
                array[row][col] = string[sub]
                sub += 1
    msg = []
    for col in range(len(string)):  # 以列的顺序依次连接各字符
        for row in range(n):
            if array[row][col] != '.':
                msg.append(array[row][col])
    return array, msg


def crack_cipher(string):
    '''破解密码'''
```

```python
    for n in range(2, len(string)):  # 遍历所有可能的栏数
        print(str(n) + '栏: ' + ''.join(decode(string, n)[1]))


if __name__ == "__main__":
    string = "ccehgyaefnpeoobe{lcirg}epriec_ora_g"
    n = 5  # 栏数

    # 若不知道栏数，则遍历所有可能
    # crack_cipher(string)

    # 若知道栏数
    array, msg = decode(string, n)
    # array,msg = encode(string, n)
    for i in array: print(i)
    print(''.join(msg))
```

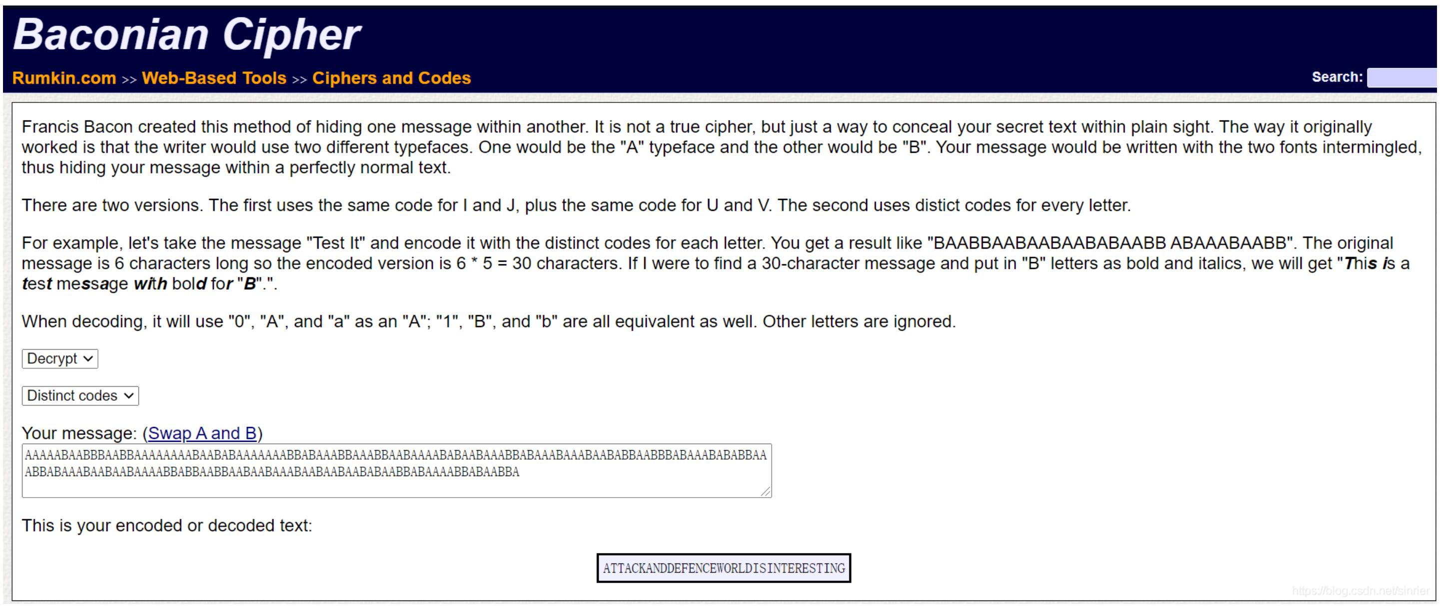
不仅仅是Morse

题目

看见题目首先进行摩斯解密，出来一堆A，B，然后我们就可以想到这大概率是培根加密了



培根解密



混合编码

题目

首先看到base64明显的标志我们先base64一下

请输入要进行 Base64 编码或解码的字符

xMjI7JiM2OTsmIzEyMDsmIzc4OyYjMTA1OyYjNTY7JiM1MzsmIzc4OyYjMTIxOyYjNTY7JiM1MzsmIzc5OyYjODM7JiM1NjsmIzEy
MDsmIzc3OyYjNjg7JiM5OTsmIzExODsmIzc5OyYjODQ7JiM5OTsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzExOTsmIzc2OyYjMTIyO
yYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NzsmIzY4OyYjNjU7JiMxMTg7JiM3NzsmIzg0OyYjNjU7JiMxMjA7JiM3
NjsmIzEyMjsmIzY5OyYjMTE5OyYjNzc7JiMxMDU7JiM1NjsmIzEyMDsmIzc3OyYjODQ7JiM2OTsmIzExOTsmIzc2OyYjMTIyOyYjNjk7
JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7
JiMxMjA7JiM3NzsmIzY4OyYjNjU7JiMxMjA7JiM3ODsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEy
MDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2
OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIy
OyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7
JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7
JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3
OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMx
MjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3
NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEw
NzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzEx
ODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3
OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7
JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsm
IzEyMDsmIzc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsm
Izc2OyYjMTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYj
MTIyOyYjNjk7JiMxMTk7JiM3NzsmIzY3OyYjNTY7JiMxMjA7JiM3NDsmIzEwNzsmIzExODsmIzc3OyYjODQ7JiM2OTsmIzEyMDsmIzc2OyYjMTIyOyYj
NjA7

编码 (Encode) 解码 (Decode) ↕ 交换 (编码快捷键: Ctrl + Enter)

Base64 编码或解码的结果:

☐ 编/解码后自动全选

LzExOS8xMDEvMTA4Lzk5LzExMS8xMDkvMTAxLzExNi8xMTEvOTcvMTE2LzExNi85Ny85OS8xMDcvOTcvMTEwLzEwMC8xMDAvMTAxLzEwMi8xMDEvMTEw&

出来的东西也很明显是unicode

Native:

LzExOS8xMDEvMTA4Lzk5LzExMS8xMDkvMTAxLzExNi8xMTEvOTcvMTE2LzExNi85Ny85OS8xMDcvOTcvMTEwLzEwMC8xMDAvMTAxLzEwMi8xMDEvMTEwLzk5LzEwMS8xMTkvMTExLzExNC8xMDgvMTAw

Unicode > < Native

Unicode:

LzExOS8xMDEvMTA4Lzk5LzExMS8xMDkvMTAxLzExNi8xMTEvOTcvMTE2LzExNi85Ny85OS8xMDcvOTcvMTEwLzEwMC8xMDAvMTAxLzEwMi8xMDEvMTEwLzk5LzEwMS8xMTkvMTExLzExNC8xMDgvMTAw

然后再进行base64

请输入要进行 Base64 编码或解码的字符

LzExOS8xMDEvMTA4Lzk5LzExMS8xMDkvMTAxLzExNi8xMTEvOTcvMTE2LzExNi85Ny85OS8xMDcvOTcvMTEwLzEwMC8xMDAvMTAxLzEwMi8xMDEvMTEwLzk5LzEwMS8xMTkvMTExLzExNC8xMDgvMTAw

编码 (Encode) 解码 (Decode) ↕ 交换 (编码快捷键: Ctrl + Enter)

Base64 编码或解码的结果:

☐ 编/解码后自动全选

/119/101/108/99/111/109/101/116/111/97/116/116/97/99/107/97/110/100/100/101/102/101/110/99/101/119/111/114/108/100

最后可以对照ascii表手工写也可以写个脚本跑
脚本如下：
第一种：

```
#include <bits/stdc++.h>
using namespace std;

signed main()
{
    vector<int> a{119, 101, 108, 99, 111, 109, 101, 116, 111, 97, 116, 116, 97, 99, 107, 97, 110, 100, 100, 101,
 102, 101, 110, 99, 101, 119, 111, 114, 108, 10};
    for(int i=0;i<a.size();i++)cout<<(char)a[i]<<" ";
    return 0;
}
```

第二种:

```
//自动去除'\'
#include <bits/stdc++.h>

using namespace std;

signed main()
{
    char c;
    while ((c = getchar()) != EOF)
    {
        if (c == '\\')
            cout << " ";
        else
            cout << c;
    }
    return 0;
}
}
```

easy_RSA

题目

这题蛮简单的写脚本直接求就行
题目信息有p,q,e,那就可以直接求phi_n，然后求d

```
import gmpy2
from Crypto.Util.number import long_to_bytes

e = 17
p = 473398607161
q = 4511491
phi_n = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi_n)
print(d)
```

easychallenge

题目

首先我们拿到pyc文件，第一件事情就是反编译它，当然有的pyc文件会加密操作，让我们无法成功反编译，但是这个还是可以的直接反编译，在线网站和python都可以，这里使用的是在线网站

网站链接

紧接着，我们拿到逆向出来的源代码

```
#!/usr/bin/env python
# visit http://tool.lu/pyc/ for more information
import base64


def encode1(ans):
    s = ''
    for i in ans:
        x = ord(i) ^ 36
        x = x + 25
        s += chr(x)

    return s


def encode2(ans):
    s = ''
    for i in ans:
        x = ord(i) + 36
        x = x ^ 36
        s += chr(x)

    return s


def encode3(ans):
    return base64.b32encode(ans)


flag = ' '
print
'Please Input your flag:'
flag = raw_input()
final = 'UC7KOWVXWVNKNIC2XCXKHKK2W5NLBKNOUOSK3LNNVWW3E==='
if encode3(encode2(encode1(flag))) == final:
    print
    'correct'
else:
    print
    'wrong'
```

这个代码主函数部分很好理解

就是输入一个flag然后我们进行三层加密后，我们查看是否和我们最后的final一致，如果是一致的话，那么我们就是得到了正确的一个flag

那么我们的思路就是逆向推导，首先我们的最后一次加密encode3，我们可以直接base64.b32decode()这个返回的类型是int类型，用python测的话是byte

然后我们第一层解码成功

```
def decode3(ans):
    return base64.b32decode(ans)
```

然后我们进行第二层的一个解密操作，观察代码

```
def encode2(ans):
    s = ''
    for i in ans:
        x = ord(i) + 36
        x = x ^ 36
        s += chr(x)

    return s
```

那么我们可以把每一位分离出来，然后我们再异或上一个36，异或的逆运算还是其本身，然后再减去一个36，最后chr()一下加到s中

代码如下：

```
def decode2(ans):
    s = ''
    for i in ans:
        x = i ^ 36
        x = x - 36
        s += chr(x)
    return s
```

这里有一点需要注意的就是，我们的返回类型上次是int了这里不需要ord了

然后最后一层代码也是同理，唯一的一个区别就是它这次返回类型是string，所以我们需要ord一下再进行操作

代码如下：

```
def decode1(ans):
    s = ''
    for i in ans:
        x = ord(i)
        x -= 25
        x = x ^ 36
        s += chr(x)

    return s
```

最后，我们的就成功的解密了我们的这个文件了，最后总代码如下：

```python
import base64
import chardet


def decode1(ans):
    s = ''
    for i in ans:
        x = ord(i)
        x -= 25
        x = x ^ 36
        s += chr(x)

    return s


def decode2(ans):
    s = ''
    for i in ans:
        x = i ^ 36
        x = x - 36
        s += chr(x)
    return s


def decode3(ans):
    return base64.b32decode(ans)


flag = 'UC7KOWVXWVNKNIC2XCXKHKK2W5NLBKNOUOSK3LNNVWW3E==='

flag = decode3(flag)

# response = chardet.detect(flag)
# print(flag.decode('ISO-8859-1'))

flag = decode2(flag)
flag = decode1(flag)

print(flag)
```

Normal_RSA

题目

提供文件

> flag.enc 后缀enc，分析是一个通过openssl加密后生成的文件

> pubkey.pem 打开时一个公钥加密文件

使用openssl提取pubkey.pem文件信息

命令如下：
1.进入openssl

```
openssl
```

2.提取信息

```
rsa -pubin -text -modulus -in warmup -in pubkey.pem
```

3.得到信息

Exponent即为e

Modulus即为n

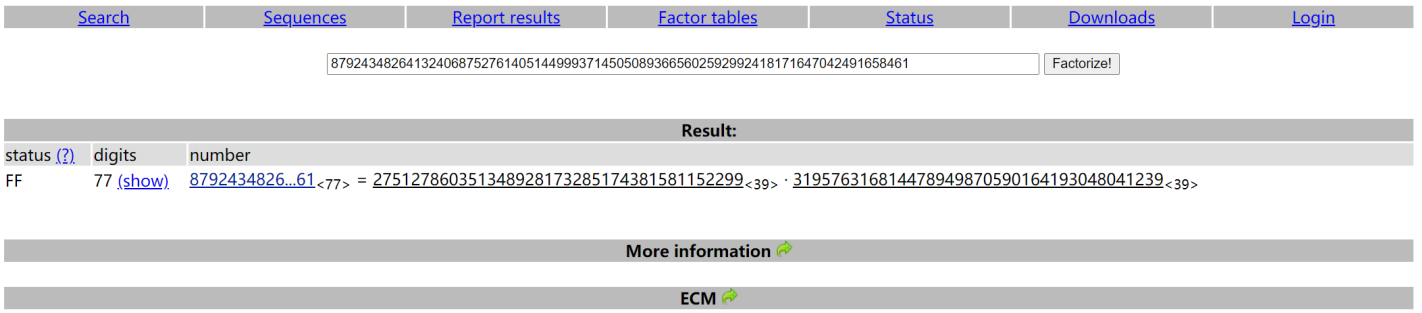
分解素数

可以看到modulus是个十六进制数，我们得先转换成十进制

转换得到：

```
879243482641324068752761405144999371450508936656025929924181716470424916584 61
```

分解得到：



现在已知信息：

p=275127860351348928173285174381581152299

q=319576316814478949870590164193048041239

e=65537

生成private.pem文件

安装工具rsatools

命令如下：

```
apt-get install libgmp-dev
apt-get install libmpfr-dev
apt-get install libmpc-dev
apt-get install python3-pip
pip install gmpy2
git clone https://github.com/Ganapati/RsaCtfTool.git
cd RsaCtfTool
pip install -r requirements.txt
```

安装成功了之后把公钥（pubkey.pem)放到这个文件夹里面，然后我们在终端执行下边的命令

```
python3 RsaCtfTool.py --publickey pubkey.pem --private
```

这里就生成了私钥，然后创建一个叫private.pem的文件把私钥放里边和pubke.pem等文件放在同一个文件夹里
这里手动创建私钥文件的原因是下面这个命令执行后kail依旧无法自动生成private.pem文件，所以最后换成了上边的命令后能得到私钥的内容但是需要自己手动创建

```
python rsatool.py -f PEM -o private.pem -p 275127860351348928173285174381581152299 -q 3195763168144789498705901641930480412 39 -e 65537
```

然后在这个文件夹里面打开终端

执行如下命令：

```
openssl
```

```
rsautl -decrypt -in flag.enc -inkey private.pem -out flag.txt
```

最后flag就出来了

转轮机加密

题目
之前没遇到过，所以得先了解一下原理

换位后的密文为：

NACZDTRXMJQOYHGVSFUWIKPBEL
FHTEQGYXPLOCKBDMAIZVRNSJUW
QGWTHSPYBXIZULVKMRAFDCEONJ
KCPMNZQWXYIHFRLABEUOTSGJVD
SXCDERFVBGTYHNUMKILOPJZQAW
EIURYTASBKJDFHGLVNCMXZPQOW
VUBMCQWAOIKZGJXPLTDSRFHENY
OSFEZWAXJGDLUBVIQHKYPNTCRM
QNOZUTWDCVRJLXKISEFAPMYGHB
OWTGVRSCZQKELMXYIHPUDNAJFB
FCUKTEBSXQYIZMJWAORPLNDVHG
NBVCXZQWERTPOIUYALSKDJFHGM
PNYCJBFZDRUSLOQXVETAMKGHIW

接下来写脚本将每一列提出来并转成小写字母

```cpp
#include <bits/stdc++.h>

using namespace std;

signed main()
{
//    cout<<(int)'a'<<endl;
//    cout<<(int)'A'<<endl;
    freopen("out.txt","w",stdout);
    char s[1010][1010]=
    {
        "NACZDTRXMJQOYHGVSFUWIKPBEL",
        "FHTEQGYXPLOCKBDMAIZVRNSJUW",
        "QGWTHSPYBXIZULVKMRAFDCEONJ",
        "KCPMNZQWXYIHFRLABEUOTSGJVD",
        "SXCDERFVBGTYHNUMKILOPJZQAW",
        "EIURYTASBKJDFHGLVNCMXZPQOW",
        "VUBMCQWAOIKZGJXPLTDSRFHENY",
        "OSFEZWAXJGDLUBVIQHKYPNTCRM",
        "QNOZUTWDCVRJLXKISEFAPMYGHB",
        "OWTGVRSCZQKELMXYIHPUDNAJFB",
        "FCUKTEBSXQYIZMJWAORPLNDVHG",
        "NBVCXZQWERTPOIUYALSKDJFHGM",
        "PNYCJBFZDRUSLOQXVETAMKGHIW",
    };

    for(int i=0;i<26;i++)
    {
        for(int j=0;j<13;j++)
        {
            cout<<(char)(s[j][i]+32);
        }
        puts("");
    }
    return 0;
}
```

得到的结果是：

nfqksevoqofnp
ahgcxiusnwcbn
ctwpcubfotuvy
zetmdrmezgkcc
dqhneyczuvtxj
tgszrtqwtrezb
rypqfawawsbqf
xxywvsaxdcswz
mpbxbbojczxed
jlxygkigvqqrr
qoiitjkdrkytu
oczhydzljeips
ykufhfgullzol
hblrnhjbxmmio
gdvlugxvkxjuq
vmkamlpiiywyx
sambkvlqsiaav
fireinthehole
uzaulcdkfprst
wvfoomsyaupka
irdtpxrppdldm
kncsjzfnmnnjk
psegzphtyadfg
bjojqqecgjvhh
eunvaonrhfhgi
lwjdwwymbbgmw

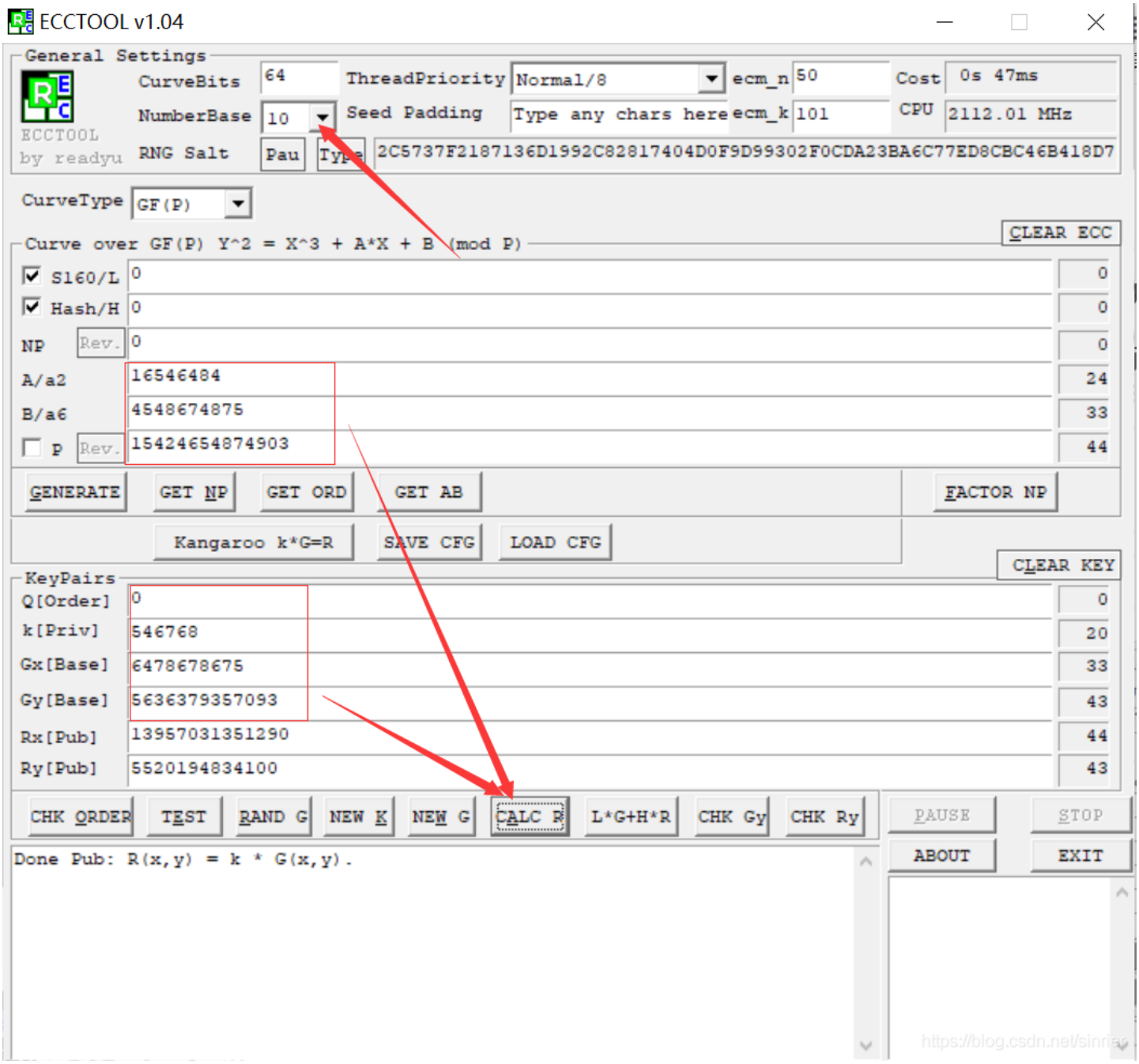
从中挑出通顺的句子交flag

ps：最后交flag的时候没有格式！直接交就行，不要写flag{}也不要写cyberpeace{}

easy_ECC

题目

使用工具Ecctool



x+y=13957031351290+5520194834100=19477226185390

得到flag

完结撒花！