BUUCTF笔记之Crypto部分WriteUp（二）

原创

KogRow 于 2021-05-17 22:50:49 发布 447 收藏 2
分类专栏： Crypto CTF 文章标签： Crypto CTF

1、凯撒？替换？呵呵！

凯撒密码一般就是26个字母经过单纯的按字母顺序来位移的加密方法（一般）
如：abc=def
进阶版的凯撒就不按照字母顺序的加密，等于是一个字母打乱顺序，使用类似密码本的形式对应另一个字母。
所以就要经过暴力破解出每一种可能的对应加密。

代码不会写，这里使用工具：

quipqiup - cryptoquip and cryptogram solver

2、[MRCTF2020]天干地支+甲子

得到得字符串用MRCTF{}包裹
一天Eki收到了一封来自Sndav的信，但是他有点迷希望您来解决一下
甲戌
甲寅
甲寅
癸卯
己酉
甲寅
辛丑

百度天干地支表：

干支次序表

编辑 讨论 上传视频

收藏 46 3

天干地支，简称“干支”。在中国古代的历法中，甲、乙、丙、丁、戊、己、庚、辛、壬、癸被称为“十天干”，子、丑、寅、卯、辰、巳、午、未、申、酉、戌、亥叫作“十二地支”。十干和十二支依次相配，组成六十个基本单位，两者按固定的顺序互相配合，组成了干支纪法，

中文名 干支次序表 与之相关 地支

干 [1] 支次序表如下：

01甲子	02乙丑	03丙寅	04丁卯	05戊辰	06己巳	07庚午	08辛未	09壬申	10癸酉
11甲戌	12乙亥	13丙子	14丁丑	15戊寅	16己卯	17庚辰	18辛巳	19壬午	20癸未
21甲申	22乙酉	23丙戌	24丁亥	25戊子	26己丑	27庚寅	28辛卯	29壬辰	30癸巳
31甲午	32乙未	33丙申	34丁酉	35戊戌	36己亥	37庚子	38辛丑	39壬寅	40癸卯
41甲辰	42乙巳	43丙午	44丁未	45戊申	46己酉	47庚戌	48辛亥	49壬子	50癸丑
51甲寅	52乙卯	53丙辰	54丁巳	55戊午	56己未	57庚申	58辛酉	59壬戌	60癸亥

根据顺序得到：

11
51
51
40
46
51
38

再加一甲子也就是加60得到：

71
111
111
100
106
111
98

这里我还以为是16进制转字符串，谁知道直接对着ascii码查就行。。。。

flag{Goodjob}

3、达芬奇密码

古典密码不精，现代密码不会，唉，我密码学实在是拉跨

题：达芬奇隐藏在蒙娜丽莎中的数字列:1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 317811 46368 4181 1 832040 2 28657 75025 34 13 17711
记录在达芬奇窗台口的神秘数字串:369688538821167255473421769522 86

打印斐波那契数列前32项，这里直接使用我以前写的C++代码：

```
#include<iostream>
using namespace std;
int main(){
    int count = 32;
    int * list = (int *)malloc(count);
    void fib(int num,int * list);
    fib(count,list);
    for(int i=0;i<count;i++) cout<<list[i]<<endl;
    getchar();
}
/**
 * 计算fib数列前n项
 * */
void fib(int num,int * list){
    list[0] = 1;    //定义数列第一项
    list[1] = 1;    //定义数列第二项
    if(num<=2) return ;
    for(int i=2;i<num;i++){
        list[i] = list[i-1]+list[i-2];
    }
}
```

结果：

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309

和题目的数列进行比对：

1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 317811 46368 4181 1 832040 2 28657 75025 34 13 17711

得到移位顺序：

(0)(13)(4)(31)(12).....

太麻烦了还是上python脚本吧：（

```
# -*- coding: UTF-8 -*-
lisb = [1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,12
ts=[1,233,3,2584,1346269,144,5,196418,21,1597,610,377,10946,89,514229,987,8,55,6765,2178309,121393,317811,4
shunxu = [0]*32
for i in range(0,32):
 for j in range(0,32):
  if(ts[i]==lisb[j]):
 shunxu[i] = j
 break
print shunxu
```

结果：

[0, 12, 3, 17, 30, 11, 4, 26, 7, 16, 14, 13, 20, 10, 28, 15, 5, 9, 19, 31, 25, 27, 23, 18, 0, 29, 2, 22, 24, 8, 6, 21]

根据该移位顺序从题目的密文恢复明文：

假设明文为m，则根据该移位顺序，m[12] = c[1]，以此类推继续上代码：

```
# -*- coding: UTF-8 -*-
lisb = [1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,12
ts=[1,233,3,2584,1346269,144,5,196418,21,1597,610,377,10946,89,514229,987,8,55,6765,2178309,121393,317811,4
shunxu = [0]*32
for i in range(0,32):
 for j in range(0,32):
  if(ts[i]==lisb[j]):
 shunxu[i] = j
 break
c = '3696885388211672554734217695228 6'
m = ['a']*32
for i in range(0,32):
 m[shunxu[i]] = c[i]
print m
```

得到['7', 'a', '9', '9', '5', '5', '8', '8', '2', '5', '6', '8', '6', '1', '2', '2', '8', '6', '1', '4', '1', '6', '5', '2', '2', '3', '3', '4', '7', '6', '8', '7']

这里因为题目的1不知道是第0还是第1个1，因此还有另一种可能：

['a', '7', '9', '9', '5', '5', '8', '8', '2', '5', '6', '8', '6', '1', '2', '2', '8', '6', '1', '4', '1', '6', '5', '2', '2', '3', '3', '4', '7', '6', '8', '7']

输出结果中还存在a，是因为斐波那契数列中存在两个1，而在index()找位置的时候，是从前往后找的，因此两次的1会覆盖掉。所以要将m中t的第二次出现1的位置上的数替换给a，然后复原被覆盖的值。

最后得到flag{37995588256861228614165223347687}

4.rot

根据题目名，猜测是移位密码。

rot5,rot13等等先分别试个遍看看最后发现是13.

```
a = [83,89,78,84,45,86,96,45,115,121,110,116,136,132,132,132,108,128,117,118,134,110,123,111,110,127,108,11
s = ""
for i in range(0,len(a)):
    s+=chr(a[i]-13)
print s
```

```
FLAG IS flag{www_shiyanbar_com_is_very_good_????}
MD5:38e4c352809e150186920aac37190cbc
```

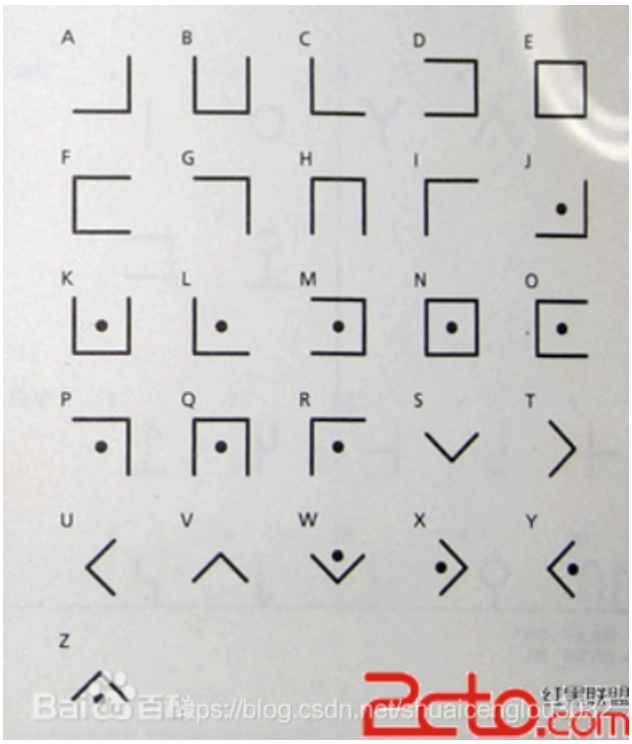
然后对其进行爆破，得到flag.

5.萌萌哒的八戒

提示：萌萌哒的八戒原来曾经是猪村的村长，从远古时期，猪村就有一种神秘的代码。请从附件中找出代码，看看萌萌哒的猪八戒到底想说啥 注意：得到的 flag 请包上 flag{} 提交。

根据提示和图片知道是猪圈密码。

直接翻译得到flag。



6.old-fashion

蒙蔽，词频分析：

quipqiup - cryptoquip and cryptogram solver

```
0   -3.426  X1 fogkvryoeksg, e hjdhvxvjvxr1 fxksao xh e zavsrb rc alfrbxly dg wsxfs jlxvh rc knex1vaiv eoa oaknefab wxvs fxksaovaiv, effrobxly vr e oayjneo hghvaz; the units may
            be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an
            inverse substitution. So the flag is n1_2hen-d3_hu1-mi-ma_a

1   -3.473  K1 fogzvryoezsg, e hqdhvkvqvkr1 fkzsao kh e javsrb rc alfrbkly dg wskfs qlkvh rc zneklvaiv eoa oaznefab wkvs fkzsaovaiv, effrobkly vr e oayqneo hghvaj; the units may
            be single letters (the most common), pairs of letters, triplets of letters, miktures of the above, and so forth. The receiver deciphers the tekt by performing an
            inverse substitution. So the flag is n1_2hen-d3_hu1-mi-ma_a
```

7.RSA2

这题是dp泄露：

```
import gmpy2
import binascii

def getd(n,e,dp):
    for i in range(1,e):
        if (dp*e-1)%i == 0:
            if n%(((dp*e-1)/i)+1)==0:
                p=((dp*e-1)/i)+1
                q=n/(((dp*e-1)/i)+1)
                phi = (p-1)*(q-1)
                d = gmpy2.invert(e,phi)%phi
                return d

e = 65537
n = 24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869760
c = 14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906740
dp = 90507449805234690464302513287951833069192517457305400462187725331868267505542197094355201669552856036 4

d=getd(n,e,dp)
m=pow(c,d,n)
print binascii.unhexlify(hex(m)[2:])
```

运行得到flag。

8.RSA3

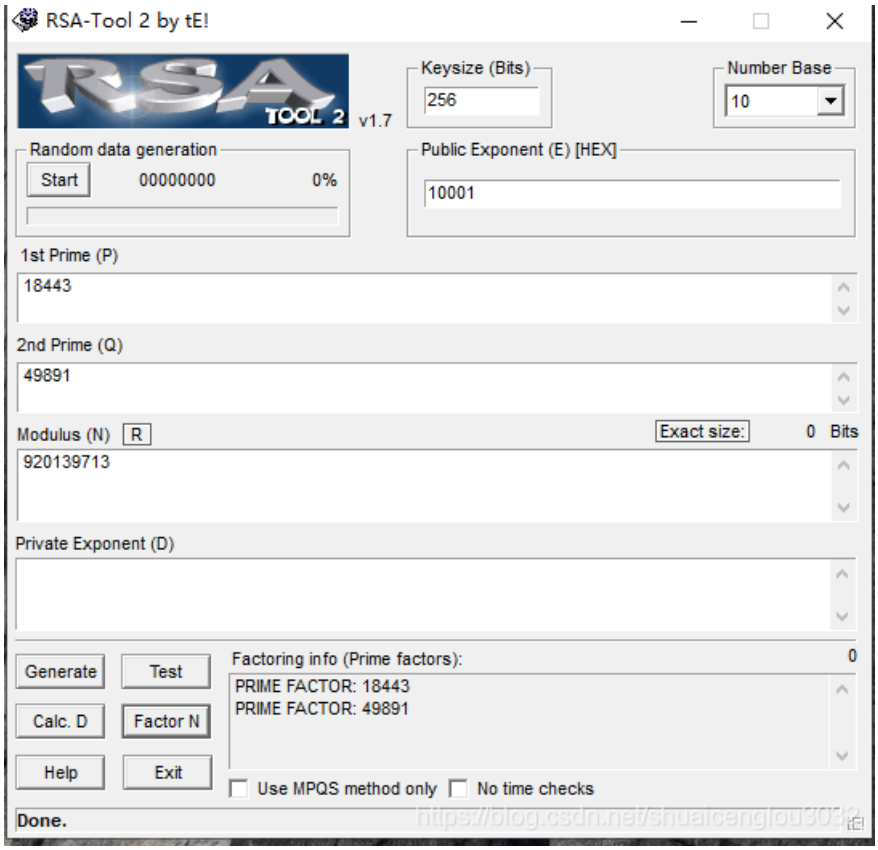
这题是共模攻击

```
#-*- coding:utf-8 -*-
import binascii
import gmpy2
e1 = 11187289
e2 = 9647291
a = gmpy2.gcdext(e1,e2)   #拓展欧里几德算法
r = int(a[1])
s = int(a[2])
c2 = 18702010045187015556548691642394982835669262147230212731309938675226458555210425972429418449273410535 3
c1 = 22322035275663237041646893770451933509324701913484303338076210603542612758956262869640822486470121149 4
N = 0xb3e1f44e2eed80c17893baed4054d7ab15bf456d9c7d3eb45e250e44bbae19b2b1dfc5e69b8b5adc423c88e392ed73a2db50f
m= (gmpy2.powmod(c1,r,N)*gmpy2.powmod(c2,s,N))%N      #计算明文，计算出的明文为16进制形式
flag = hex(m)
flag = binascii.a2b_hex(flag[2:]) # 将十六进制数字字符串转换为二进制数据（字符切片截取0x之后的）
print flag
```

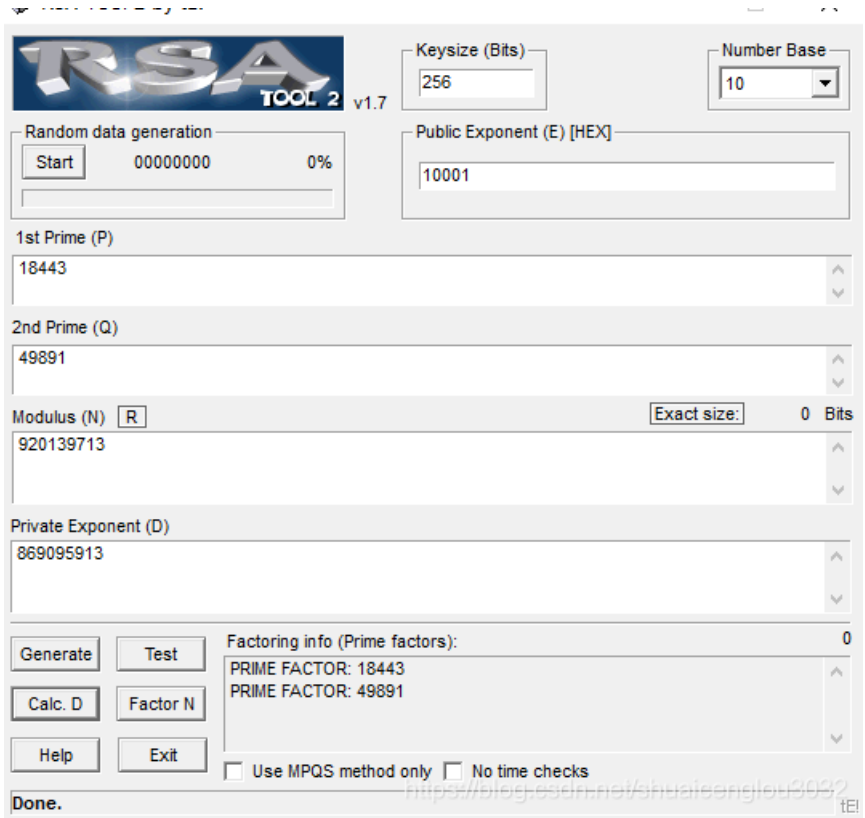
运行得到flag。

9.RSAROLL

这题先把N：920139713分解一下：



然后计算一下私钥d：



然后根据私钥解密：

```
import gmpy2 as gp
import binascii
p = 18443
q = 49891
e = 19
c =[704796792,752211152,274704164,18414022,368270835,483295235,263072905,459788476,483295235,459788476,6635
n = p*q
phi = (p-1) * (q-1)
d = gp.invert(e, phi)
flag = ""
for i in range(0,len(c)):
 m = pow(c[i], d, n)
 flag+=chr(m)
print flag
```

10.权限获得第一步

Administrator:500:806EDC27AA52E314AAD3B435B51404EE:F4AD50F57683D4260DFD48AA351A17A8:::

这个一看就是windows密码。

把F4AD50F57683D4260DFD48AA351A17A8md5解密一下：

11.还原大师

我们得到了一串神秘字符串：TASC?O3RJMV?WDJKX?ZM,问号部分是未知大写字母，为了确定这个神秘字符串，我们通过了其他途径获得了这个字串的32位MD5码。但是我们获得它的32位MD5码也是残缺不全，E903???4DAB????08?????51?80??8A?,请猜出神秘字符串的原本模样，并且提交这个字串的32位MD5码作为答案

只有3个问号而且是大写字母，直接爆破了：
上python3脚本：

```
import hashlib
str1 = "TASC"
str2 = "O3RJMV"
str3 = "WDJKX"
str4 = "ZM"
dic = ["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",
for i in range(0,26):
 for j in range(0,26):
  for k in range(0,26):
   s = str1+dic[i]+str2+dic[j]+str3+dic[k]+str4
   # print(s)
   md5s = hashlib.md5(s.encode('utf8')).hexdigest().upper()
   if(md5s[:4]=="E903"):
    print(md5s)
```

运行得到flag。

12.世上无难事

以下是某国现任总统外发的一段指令，经过一种奇异的加密方式，毫无规律，看来只能分析了。请将这段语句还原成通顺语句，并从中找到key作为答案提交，答案是32位.

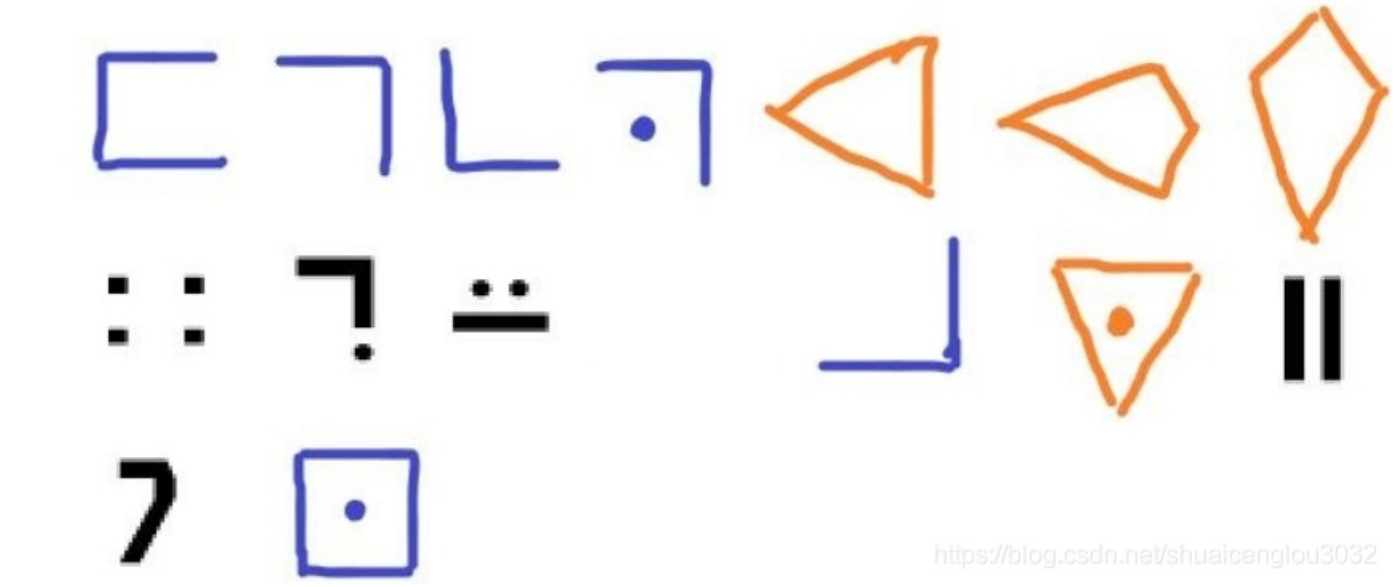
词频分析。

automatically selected statistics mode; you can override by using the drop down menu next to the solve button.

0	-1.534	HELLO EVERYBODY THANK YOU ALL RIGHT EVERYBODY GO AHEAD AND HAVE A SEAT HOW IS EVERYBODY DOING TODAY HOW ABOUT TIM SPICER WE ARE HERE WITH STUDENTS AT WAKEFIELD HIGH SCHOOL IN ARLINGTON VIRGINIA AND WE HAVE GOT STUDENTS TUNING IN FROM ALL ACROSS AMERICA FROM KINDERGARTEN THROUGH 12TH GRADE AND WE ARE JUST SO GLAD THAT ALL COULD JOIN US TODAY AND WE WANT TO THANK WAKEFIELD FOR BEING SUCH AN OUTSTANDING HOST GIVE YOURSELVES A BIG ROUND OF APPLAUSE AND THE KEY IS 640E11012805F211B0AB24FF02A1ED09
1	-4.124	HIDDO IFILMPOEM THANK MOR ADD LUGHT IFILMPOEM GO AHIAE ANE HAFI A SIAT HOW US IFILMPOEM EOUNG TOEAM HOW APORT TUB SZUCIL WI ALI HILI WUTH STREINTS AT WAKIVUIDE HUGH SCHOOD UN ALDUNGTON FULGUNUA ANE WI HAFI GOT STREINTS TRNUNG UN VLOB ADD ACLOSS ABILUCA VLOB KUNEILGALTIN THLORGH 12TH GLAEI ANE WI ALI YRST SO GDAE THAT ADD CORDE YOUN RS TOEAM ANE WI WANT TO THANK WAKIVUIDE VOL PIUNG SRCH AN ORTSTANEUNG HOST GUFI MORLSIDFIS A PUG LORNE OV AZZDARSI ANE THI KIM US 640I11012805V211P0AP24VV02A1IE09
2	-4.127	HIDDO IFILMPOEM THANJ MOR ADD LUGHT IFILMPOEM GO AHIAE ANE HAFI A SIAT HOW US IFILMPOEM EOUNG TOEAM HOW APORT TUK SZUBIL WI ALI HILI WUTH STREINTS AT WAJICUIDE HUGH SBHOOD UN ALDUNGTON FULGUNUA ANE WI HAFI GOT STREINTS TRNUNG UN CLOK ADD ABLOSS AKILUBA CLOK JUNEILGALTIN THLORGH 12TH GLAEI ANE WI ALI VRST SO GDAE THAT ADD BORDE VOUN RS TOEAM ANE WI WANT TO THANJ WAJICUIDE COL PIUNG SRBH AN ORTSTANEUNG HOST GUFI MORLSIDFIS A PUG LORNE OC AZZDARSI ANE THI JIM US 640I11012805C211P0AP24CC02A1IE09

大写转小写之后得到flag。

13.[MRCTF2020]古典密码知多少



蓝色是猪圈密码，黄色是圣堂武士密码，黑色是标准银河字母。

查表解密得FGCPFLIRTUASYON

放进栅栏解密一下得到flag：

14.异性相吸

这题就是个简单的异或。

把密文用winhex打开，提取二进制数据，然后把二进制数据逐位和key异或，再二进制转字符串得到flag

15. Unencode

89FQA9WMD<V1A<V1S83DY.#<W3$Q,2TM]

补知识：

uuencode是以前unix下常用编码方式应用于UUCP(unix to unix copy),通过串行通讯传输二进制文件。base64属于MIME(多用途国际互联网邮件扩展)编码,与uuencode不是同一个范畴的,MIME主要应用于邮件,Uuencode主要应用在邮件和新闻组。

php直接解码：

```
<?php
/*
convert_uudecode() 函数对 uuencode 编码的字符串进行解码。
该函数常与 convert_uuencode() 函数一起使用。
*/

$str = "Hello world!";
// 对字符串进行编码
$encodeString = convert_uuencode($str);
echo $encodeString . "<br>";

// 对字符串进行解码
$decodeString = convert_uudecode($encodeString);
echo $decodeString;
?>
```

```php
// 对字符串进行编码
$encodeString = convert_uuencode($str);
echo $encodeString . "<br>";

// 对字符串进行解码
$decodeString = convert_uudecode("89FQA9WMD<V1A<V1S83DY.#<W3\$Q,2TM]");
echo $decodeString;
?>
```

也可以用python3的uu模块：

```
import uu
in2 = open("E:\\2.txt","wb")
uu.decode("E:\\1.txt",in2)
```

但这里对输入的源文件有特殊格式要求：

```
begin 666 1.txt
89FQA9WMD<V1A<V1S83DY.#<W3$Q,2TM]

end
```

第一行声明文件开始，第二行放编码，第三行空行，第四行声明文件结束。

如果不按这个格式来python会抛出错误。所以还是用php吧。

16.鸡藕椒盐味

公司食堂最新出了一种小吃，叫鸡藕椒盐味汉堡，售价八块钱，为了促销，上面有一个验证码，输入后可以再换取一个汉堡。但是问题是每个验证码几乎都有错误,而且打印的时候倒了一下。小明买到了一个汉堡，准备还原验证码，因为一个吃不饱啊~~验证码如下：1100 1010 0000 ,而且打印的时候倒了一下。把答案哈希一下就可以提交了~~。(答案为正确值(不包括数字之间的空格)的32位md5值的小写形式) 注意：得到的 flag 请包上 flag{} 提交

鸡（奇）藕（偶）椒（校）盐（验）味（位）。

奇偶校验位。。。。

这题有点脑洞大开了。

目（百）测（度）得到是海明码。

详见海明码

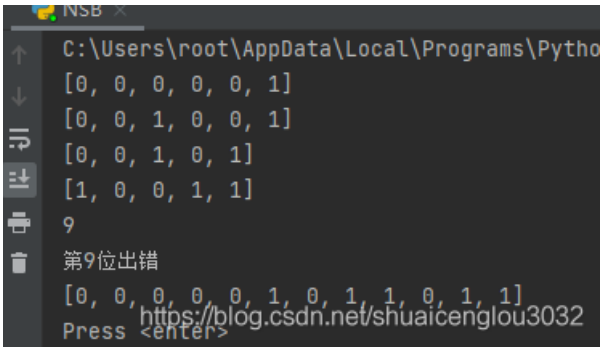
校验及恢复数据的python代码：

```
import math

userin = '110010100000' #这里填要校验的海明码
userin = userin[::-1]
haiming = [];
# 输入的海明码是字符串，下面的代码是将字符串转化成整型列表
haiming = list(haiming)
for x in userin:
    haiming.append(int(x))

# 计算校验码的个数
for i in range(0, len(haiming)):
    if 2 ** i > len(haiming):
        break
# 校验码个数
num = i
# 标记错位
flag = 0
for i in range(0, num):
    b = []
    if (i == 0):
        # 利用python的步长。
        a = haiming[(2 ** i) - 1:len(haiming):(2 ** i) + 1]
        if (a.count(1) % 2 == 1):
            flag = flag + 2 ** i
        print(a)
    else:
        for j in range(2 ** i, len(haiming) + 1):
            if ((j / (2 ** i)) % 2 == 1):
                for k in range(j, j + (2 ** i)):
                    if (k > len(haiming)):
                        break
                    b.append(haiming[k - 1])
        if (b.count(1) % 2 == 1):
            flag = flag + 2 ** i
        print(b)
    del (b)
print(flag)
if flag != 0:
    if haiming[flag - 1] == 0:
        haiming[flag - 1] = 1
    elif haiming[flag - 1] == 1:
        haiming[flag - 1] = 0
    print("第%d位出错" % (flag))
    print(haiming)
else:
    print("没有错误")
input("Press <enter>")
```

根据题目，把结果反转之后放进去校验：



得到0000 0101 1011

计算哈希不对，把结果再反转之后计算哈希，正确。

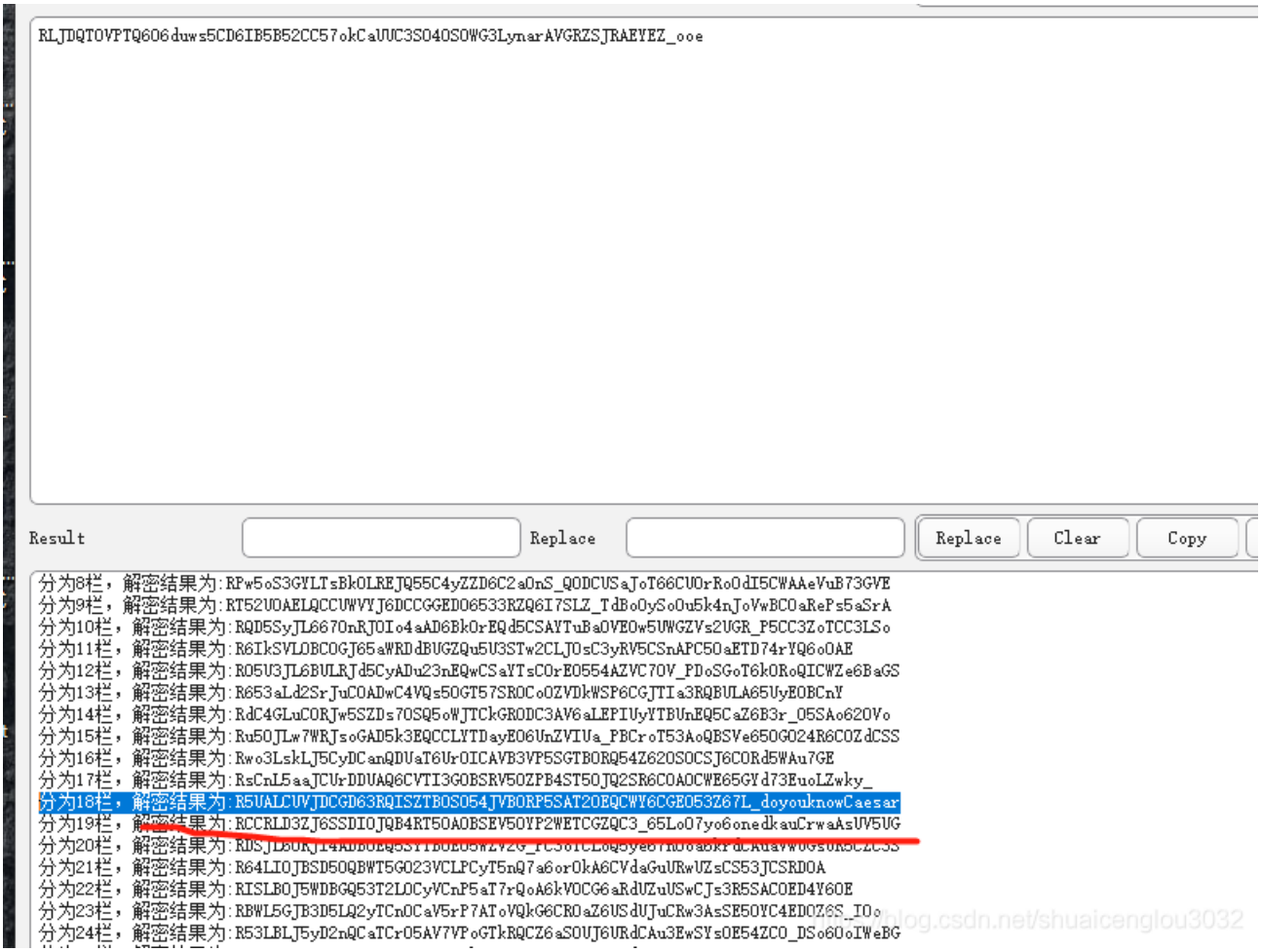
flag{d14084c7ceca6359eaac6df3c234dd3b}

17.[WUSTCTF2020]佛说：只能四天

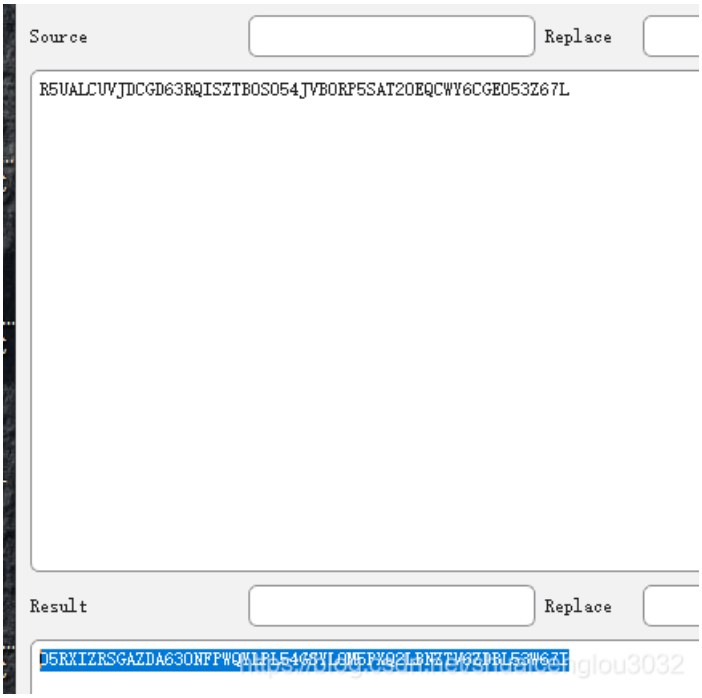
这题与佛论禅解不出来，百度说使用新约佛论禅解：

然后社会主义核心价值观编码器解码：

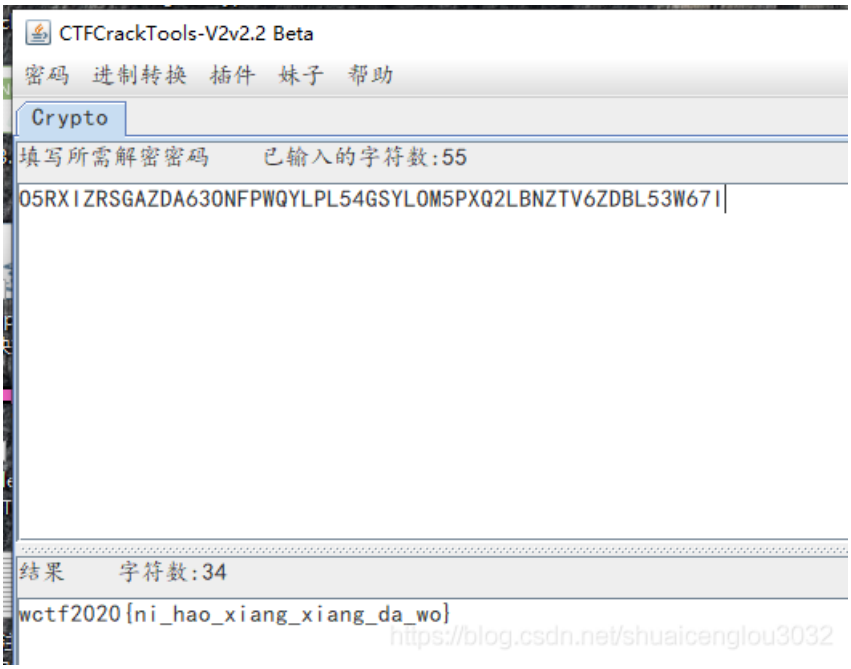
然后用栅栏解码，记得把后面的doyou*去掉。



凯撒密码：



然后逐个尝试发现是base32:



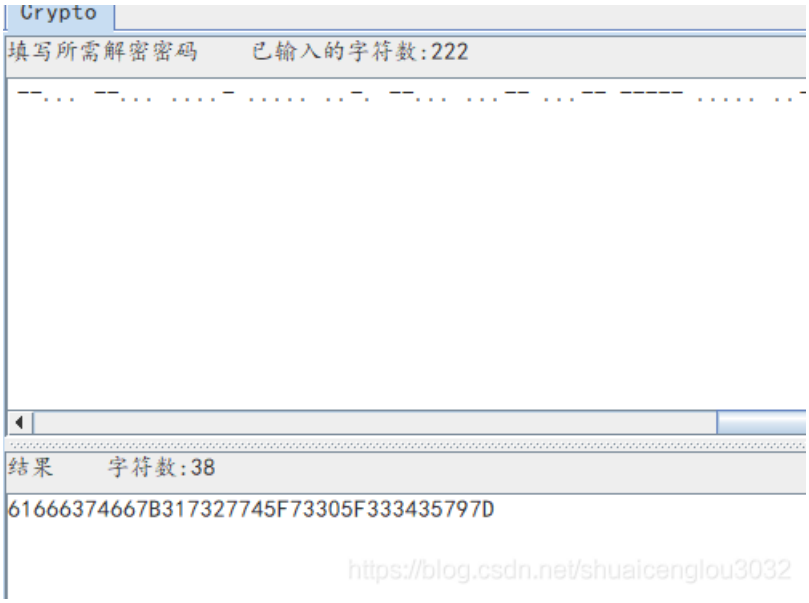
18.[AFCTF2018]Morse

这题就是摩斯密码+16进制。

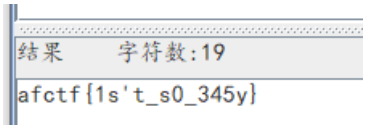
下载下来之后把/替换成空格得到：

-.... .---- -.... -.... -.... ...-- --... ....- -.... -.... --... -... ...-- .---- --... ...-- ..--- --... --... ....- ....-. --... ...-- ...-- ----- ....-. ...-- ...-- ...-- ....- ...-- ..... --... ----. --... -..

然后解码：



把解码结果拿去16进制转字符串：



19.RSA5

低加密指数广播攻击

```
import gmpy2
import libnum

e = 65537

n0 = 204749188940517785333052623456018809280882844711218237540497253540724771558737788480550738433458206978
c0 = 974463908243330865728978769213595400782053398596897741316275722596415018912929508637393850919224969271

n1 = 209188199606488913494382630469549022109591464078609807421659302537813187592856924925114752632342420025
c1 = 158196362019711855386948805051204693325821518567140708245218031218482923875568641771962297189237708100

n2 = 250332546259067572723696091192142020331621286251712464366395706152639491573632732131215568258787379232
c2 = 418530852941687400583123078101409240719845138595567739966850183390262347839566927940488399072518433270

n3 = 212069680973141310071834279444868019535831511514436279431137369967767871811110639579606980926968005550
c3 = 452103801104475844189112846846723308849388575085058898570851991115477809059713612615028904189345412667

n4 = 228220397330493881109367781730147656636633038117912832343612306497758059239021734385539278054074631061
c4 = 154064985807617801086258918780085268151453720962340839366814422251550972992648086243588266869065355948

n5 = 215741398553414329084740647843184620184752968093272855323377069401269425753495076682892140780261026822
c5 = 203668561507103051245830653752976618197952422383764852649511853369960837446045934189833362851854911974

n6 = 253602274126666124901021611311745848192409318031964484812243052505838414395810085285359308141673383819
c6 = 198927725246514523410275956194827343562434356715923981726803799815027596957840879006690899199877056758

n7 = 227268552446323560291596917534518221633315192375476399387795177514964987131745889355665761673295764947
c7 = 604011979517585640754108236002353220461472385868863672482271271757275979396024634180030814973980987123

n8 = 232973337914430532973630007868353360952522908184619500545426583274845074065946327857127674599589179430
c8 = 541812030120837871311588946557996425787181411451504609609096015973785907682925851692036157785390392595

n9 = 288736679047156827229872342934932003069769478987112550641251159336669686787425988587224314262189144629
c9 = 991988046378683668498795797909152747747144499639237524407552784186550916018166654301631763496351243751

n10 = 223246859475396537224999324694096075330654191573478139619580756890476904652664043841994836839085947 87
c10 = 149152705020329498988282924856039518480497727774712614310395721916462418752844104783735126358044068 64

n11 = 276467464237590201110078286532640279992578476456661299077890260545943936488002361170467691127626417 78
c11 = 219915241289572605360437712848549203931058081267001282221258567755068857219711931093613159611291908 14

n12 = 205454874058169287317389883744750126868279337097897843918557068351362702709334012030193291369376508 78
c12 = 142274391881910294612504766927905396546191998884873194291144145579753763086889080281408171572055798 04

n13 = 273597277115842772348971577240558527940192168452297989386558142694600463843535681385985677553925596 53
c13 = 378852978424825502708167454087701637280784822277688792045348887824713793057829679743764792249451048 37

n14 = 275459376037517372487852208917357964689733297380762091440799214499672925723494245390105022875640301 16
c14 = 140691129706088957324170399775427326657966018937624015008787868716806457987547833156935112617400597 25

n15 = 257461620756979115602631817912164330625741785724246003368562781761127330544314632539034331282327090 54
c15 = 173442848602754894774915258199228553267922751287197094012925456081228598298274620883900446122349675 51

n16 = 232884869341171203150369194185881362270284854941379301963237153362088493278339656938946705672179717 27
c16 = 107382544181140765480714488449640464681416217406032143849863541891052369770710014292715606364280759 70

n17 = 195914413839585294355987291139363466570013525783579093476572572397775404248117498177830612332358179 16
c17 = 383491709888720293198196870465911934162443229475936191955393755105349960744033323401818914197024630 22
```

```
n18 = 19254242571588430171308191757871261075358521158624745702744057556054652332495961196795369630484782930
c18 = 67905535339912972058045619912254931053123988251876822507801975107847652264296632842204004805630393419

n19 = 26809700251171279102974962949184411136459372267620535198421449833298448092580497485301953796619185339
c19 = 38621355660843401376986472712387941204199127152899052854850745121069261898665287042463221942460167752

n=[n0,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18]
c=[c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18]

for i in range(len(n)):
    for j in range(len(n)):
        if(i!=j):
            if(gmpy2.gcd(n[i],n[j])!=1):   #对不同的n进行 欧几里得算法，以求出最大公约数(p)
                print(i,j)                 #输出对应的n的序号
                p = gmpy2.gcd(n[i],n[j])
                print("p = ",p)
                q = n[i] // p
                print("q = ",q)
                d = gmpy2.invert(e , (p-1)*(q-1))
                print("d = ",d)
                m = pow(c[i],d,n[i])
                print("m = ",m)

print(libnum.n2s(int(m)))
```

20.rsa2

看到e很大，判断是低解密指数攻击。

```
#该代码在python2.7下运行
import gmpy2
from Crypto.Util.number import long_to_bytes


def continuedFra(x, y):
    cF = []
    while y:
        cF += [x // y]
        x, y = y, x % y
    return cF


def Simplify(ctnf):
    numerator = 0
    denominator = 1
    for x in ctnf[::-1]:
        numerator, denominator = denominator, x * denominator + numerator
    return (numerator, denominator)


def calculateFrac(x, y):
    cF = continuedFra(x, y)
    cF = list(map(Simplify, (cF[0:i] for i in range(1, len(cF)))))
    return cF


def solve_pq(a, b, c):
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) / (2 * a), (-b - par) / (2 * a)


def wienerAttack(e, n):
    for (d, k) in calculateFrac(e, n):
        print(e)
        print(d)
        print(k)
        if k == 0:
            continue
        if (e * d - 1) % k != 0:
            continue
        phi = (e * d - 1) / k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return abs(int(p)), abs(int(q))
    print('[!]not find!')
n = 10199180977755325347027675139926474013115768232925267350179215450700615843443200914199536724196252570591
e = 46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378631
p, q = wienerAttack(e, n)
print('[+]Found!')
print('[-]p =', p)
print('[-]q =', q)
d = gmpy2.invert(e, (p-1)*(q-1))
import hashlib
flag = "flag{" + hashlib.md5(hex(int(gmpy2.digits(d)))).hexdigest() + "}"
print (flag)
```

21.Dangerous RSA

```
import gmpy2
import os
from functools import reduce

from Crypto.Util.number import long_to_bytes


def CRT(items):
    N = reduce(lambda x, y: x * y, (i[1] for i in items))
    result = 0
    for a, n in items:
        m = N // n
        d, r, s = gmpy2.gcdext(n, m)
        if d != 1:
            raise Exception("Input not pairwise co-prime")
        result += a * s * m
    return result % N, N


# e, n, c
e = 0x3
n=[0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43
c=[0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d

data = list(zip(c, n))
x, n = CRT(data)
m = gmpy2.iroot(gmpy2.mpz(x), e)[0].digits()
print('m is: ' + long_to_bytes(m))
```

22.[BJDCTF2020]这是base??

就是自定义编码而已，没啥特别的，直接上java代码：

```
public class Base64 {

//    private static final char S_BASE64CHAR[] = {
//        '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', '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', '0', '1', '2', '3', '4', '5', '6', '7',
//        '8', '9', '+', '/'
//    };

  private static final char S_BASE64CHAR[] = {
  'J', 'K','L', 'M', 'N', 'O', 'x', 'y', 'U', 'V', 'z',
  'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', '7', '8',
  '9', 'P', 'Q', 'I', 'a', 'b', 'c', 'd', 'e', 'f',
  'g', 'h', 'i', 'j', 'k', 'l', 'm', 'W', 'X', 'Y',
  'Z', '0', '1', '2', '3', '4', '5', '6', 'R', 'S',
```

```
  'T', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
  'w', '+', '/'
};




//    private static final char S_BASE64PAD = 61;

    private static final byte S_DECODETABLE[];

    static {
        S_DECODETABLE = new byte[128];
        for(int i = 0; i < S_DECODETABLE.length; i++)
            S_DECODETABLE[i] = 127;

        for(int j = 0; j < S_BASE64CHAR.length; j++)
            S_DECODETABLE[S_BASE64CHAR[j]] = (byte)j;
    }


    private Base64() { }


    public static byte[] decode(String s) {
        char ac[] = new char[4];
        int i = 0;
        byte abyte0[] = new byte[(s.length() / 4) * 3 + 3];
        int j = 0;
        for(int k = 0; k < s.length(); k++) {
            char c = s.charAt(k);
            if(c == '=' ||
               c < S_DECODETABLE.length &&
               S_DECODETABLE[c] != 127)
            {
                ac[i++] = c;
                if(i == ac.length) {
                    i = 0;
                    j += decode0(ac, abyte0, j);
                }
            }
        }

        if(j == abyte0.length) {
            return abyte0;
        } else {
            byte abyte1[] = new byte[j];
            System.arraycopy(abyte0, 0, abyte1, 0, j);
            return abyte1;
        }
    }


    public static String encode(byte abyte0[]) {
        return encode(abyte0, 0, abyte0.length);
    }


    public static String encode(byte abyte0[], int i, int j) {
```

```
        if(j <= 0)
            return "";

        char ac[] = new char[(j / 3) * 4 + 4];
        int k = i;
        int l = 0;
        int i1;
        for(i1 = j - i; i1 >= 3; i1 -= 3) {
            int j1 = ((abyte0[k] & 0xff) << 16) + ((abyte0[k + 1] & 0xff) << 8) + (abyte0[k + 2] & 0xff);
            ac[l++] = S_BASE64CHAR[j1 >> 18];
            ac[l++] = S_BASE64CHAR[j1 >> 12 & 0x3f];
            ac[l++] = S_BASE64CHAR[j1 >> 6 & 0x3f];
            ac[l++] = S_BASE64CHAR[j1 & 0x3f];
            k += 3;
        }

        if(i1 == 1) {
            int k1 = abyte0[k] & 0xff;
            ac[l++] = S_BASE64CHAR[k1 >> 2];
            ac[l++] = S_BASE64CHAR[k1 << 4 & 0x3f];
            ac[l++] = '=';
            ac[l++] = '=';
        } else if(i1 == 2) {
            int l1 = ((abyte0[k] & 0xff) << 8) + (abyte0[k + 1] & 0xff);
            ac[l++] = S_BASE64CHAR[l1 >> 10];
            ac[l++] = S_BASE64CHAR[l1 >> 4 & 0x3f];
            ac[l++] = S_BASE64CHAR[l1 << 2 & 0x3f];
            ac[l++] = '=';
        }
        return new String(ac, 0, l);
    }


    private static int decode0(char ac[], byte abyte0[], int i) {
        byte byte0 = 3;
        if(ac[3] == '=')
            byte0 = 2;
        if(ac[2] == '=')
            byte0 = 1;
        byte byte1 = S_DECODETABLE[ac[0]];
        byte byte2 = S_DECODETABLE[ac[1]];
        byte byte3 = S_DECODETABLE[ac[2]];
        byte byte4 = S_DECODETABLE[ac[3]];
        switch(byte0) {
        case 1: // '\001'
            abyte0[i] = (byte)(byte1 << 2 & 0xfc | byte2 >> 4 & 3);
            return 1;

        case 2: // '\002'
            abyte0[i++] = (byte)(byte1 << 2 & 0xfc | byte2 >> 4 & 3);
            abyte0[i] = (byte)(byte2 << 4 & 0xf0 | byte3 >> 2 & 0xf);
            return 2;

        case 3: // '\003'
            abyte0[i++] = (byte)(byte1 << 2 & 0xfc | byte2 >> 4 & 3);
            abyte0[i++] = (byte)(byte2 << 4 & 0xf0 | byte3 >> 2 & 0xf);
            abyte0[i] = (byte)(byte3 << 6 & 0xc0 | byte4 & 0x3f);
            return 3;
        }
        throw new RuntimeException("Internal Error");
```

```
        throw new RuntimeException("Internal Error");
    }


    public static void main(String[] args) {

     String  a="FlZNfnF6Qol6e9w17WwQQoGYBQCgIkGTa9w3IQKw";

     byte [] b=null;

     b=a.getBytes();


     String encodeString=Base64.encode(b);

      System.out.println(encodeString);

     byte[] decodeByte=Base64.decode(a);



      System.out.println(new String(decodeByte));


//        if(args.length != 2) {
//            System.out.println("ERROR: use -encode <string> OR -decode <string>");
//            return;
//        }
//        if(args[0].equals("-encode")) {
//            System.out.println(Base64.encode(args[1].getBytes()));
//        } else if (args[0].equals("-decode")) {
//            System.out.println(new String(Base64.decode(args[1])));
//        } else {
//            System.out.println("ERROR: use -encode <string> OR -decode <string>");
//            return;
//        }
//



    }

}
```

运行得到BJD{D0_Y0u_kNoW_Th1s_b4se_map}

23.[MRCTF2020]keyboard

```
得到的flag用
MRCTF{xxxxxx}形式上叫
都为小写字母

6
666
22
444
555
33
7
44
666
66
3
```

手机键盘九宫格

6->m

666->o

22->b

444->i

555->l

33->e

以此类推得到{mobilephone}

24.[MRCTF2020]vigenere

维吉尼亚密码无密钥破解：

代码来自Vigenere的加密和解密、破解

```
from string import ascii_lowercase as lowercase

# Vigenere加密
def VigenereEncrypto(message,key):
    cipher = ''
    non_alpha_count = 0
    for i in range (len(message)):#遍历
        if message[i].isalpha():#判断是否为字母
            if message[i].islower():##判断是否为小写
                offset = ord(key[(i - non_alpha_count) % len(key)]) - ord('a')
                cipher += chr((ord (message[i]) - ord('a') + offset) % 26 + ord('a'))
            else:#大写字母
                offset = ord(key[(i - non_alpha_count) % len(key)]) - ord('a')
                cipher += chr((ord (message[i]) - ord('A') + offset) % 26 + ord('A'))
        else:#非字母，就记下
            cipher += message[i]
            non_alpha_count += 1
    return cipher

# Vigenere解密
```

```
def VigenereDecrypto(cipher,key):
    message = ''
    non_alpha_count = 0
    for i in range (len(cipher)):#遍历
        if cipher[i].isalpha():
            if cipher[i].islower():
                offset = ord(key[(i - non_alpha_count) % len(key)]) - ord('a')
                message += chr((ord (cipher[i]) - ord('a') - offset) % 26 + ord('a'))
            else:
                offset = ord(key[(i - non_alpha_count) % len(key)]) - ord('a')
                message += chr((ord (cipher[i]) - ord('A') - offset) % 26 + ord('A'))
        else:
            message += cipher[i]
            non_alpha_count += 1
    return message

def get_trim_text(text):
    text = text.lower()
    trim_text = ''
    for l in text:
        if lowercase.find(l) >= 0:
            trim_text += l
    return trim_text

# 计算重合指数
def get_coincidence_index(text):
    text = get_trim_text(text)
    length = len(text)
    letter_stats = []
    for l in lowercase:
        lt = {}
        count = text.count(l)
        lt[l] = count
        letter_stats.append(lt)

    index = 0
    for d in letter_stats:
        v = list(d.values())[0]
        index += (float(v)/length) ** 2

    return index

# 计算和0.067的差距大小
def get_var(data, mean=0.067):
    if not data:
        return 0
    var_sum = 0
    for d in data:
        var_sum += (d - mean) ** 2

    return float(var_sum) / len(data)

# 求秘钥长度
def get_key_length(text):
    # assume text length less than 26
    text = get_trim_text(text)
    group = []
    for n in range(1, len(text)+1):
        group_str = ['' for i in range(n)]
        for i in range(len(text)):
```

```
        for i in range(len(text)):
            l = text[i]
            for j in range(n):
                if i % n == j:
                    group_str[j] += l
        group.append(group_str)

    var_list = []
    length = 1
    for tex in group:
        data = []
        for t in tex:
            index = get_coincidence_index(t)
            data.append(index)
        var_list.append([length, get_var(data)])
        length += 1
    var_list = sorted(var_list, key=lambda x: x[1])
    print(var_list)
    return [v[0] for v in var_list[:int(n/2)+1]]  #var_list[0][0]

# 统计字母频度
def countList(lis):
    li = []
    alphabet = [chr(i) for i in range(97,123)]
    for c in alphabet:
        count = 0
        for ch in lis:
            if ch == c:
                count+=1
        li.append(float(count)/len(lis))
    return li

def openfile(fileName): # 读文件
    file = open(fileName,'r')
    text = file.read()
    file.close();
    text = text.replace('\n','')
    return text

# 根据密钥长度将密文分组
def textToList(text,length):
    text = get_trim_text(text)
    textMatrix = []
    row = []
    index = 0
    for ch in text:
        row.append(ch)
        index += 1
        if index % length ==0:
            textMatrix.append(row)
            row = []
    textMatrix.append(row)
    return textMatrix

# 获取密钥
def getKey(text,length):
    text = get_trim_text(text)
    key = [] # 定义空白列表用来存密钥
    alphaRate =[0.08167,0.01492,0.02782,0.04253,0.12705,0.02228,0.02015,0.06094,\
                0.06996,0.00153,0.00772,0.04025,0.02406,0.06749,0.07507,0.01929,\
```

```
                0.0009,0.05987,0.06327,0.09056,0.02758,0.00978,0.02360,0.0015,0.01974,0.00074]
    matrix = textToList(text,length)
    for i in range(length):
        w = [row[i] for row in matrix if len(row) > i] #获取每组密文
        li = countList(w)
        powLi = [] #算乘积
        for j in range(26):
            Sum = 0.0
            for k in range(26):
                Sum += alphaRate[k]*li[k]
            powLi.append(Sum)
            li = li[1:]+li[:1]#循环移位
        Abs = 100
        ch = ''
        for j in range(len(powLi)):
             if abs(powLi[j] -0.065546)<Abs: # 找出最接近英文字母重合指数的项
                 Abs = abs(powLi[j] -0.065546) # 保存最接近的距离，作为下次比较的基准
                 ch = chr(j+97)
        key.append(ch)
    return key

if __name__ == '__main__':
    key_lengths = []
    c = openfile(r'cipher.txt')
    key_lengths = get_key_length(c)
    print(key_lengths)
    for i in range(len(key_lengths)):
        key = getKey(c, key_lengths[i])
        print("the plaintext is %s, the length of key is %d, key is %s" \
              % (VigenereDecrypto (c , key), key_lengths[i], key))
```

25.[ACTF新生赛2020]crypto-classic0

提示是生日密码，那就ARCHPR暴力破解密码，纯数字，范围1970000—20201118

爆破得到解压密码19990306。解压得到C代码：

```
#include<stdio.h>
char flag[25] = ***
int main()
{
 int i;
 for(i=0;i<25;i++)
 {
  flag[i] -= 3;
  flag[i] ^= 0x7;
  printf("%c",flag[i]);
 }
 return 0;
}
```

明文先逐个字符减去3，再和0x7异或，写python运算一下：

```
#!/usr/bin/python
s = 'Ygvdmq[lYate[elghqvakl}'
sl = list(s)
m = ''
for i in range(0,len(sl)):
 a = (ord(sl[i])^0x7)+3
 m += chr(a)
print(m)
```

运行得到佛莱格：actf{my_naive_encrytion}

26.[ACTF新生赛2020]crypto-aes

27.[ACTF新生赛2020]crypto-rsa3

使用yafu分解大整数：

```
PS E:\> yafu-x64
factor(177606504836499246970959030226871608885969321778211051080524634084516973331441644993898029573612290095853069264036530459253652875586267946877831055147546910227100566496658142742231377276886895358235330467787931319021434444087356108211678387174888599022428636837727688689535823533046778793131902143444408735610821167838717488859902242863683)


fac: factoring 177606504836499246970959030226871608885969321778211051080524634084516973331441644993898029573612290095853069264036530459253652875586267946877831055147546910227100566
27422313772768868953582353304677879313190214344440873561082116783871748885990224286368
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
Total factoring time = 0.6360 seconds


***factors found***

P155 = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179293
P155 = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179231

ans = 1
```


然后py脚本：

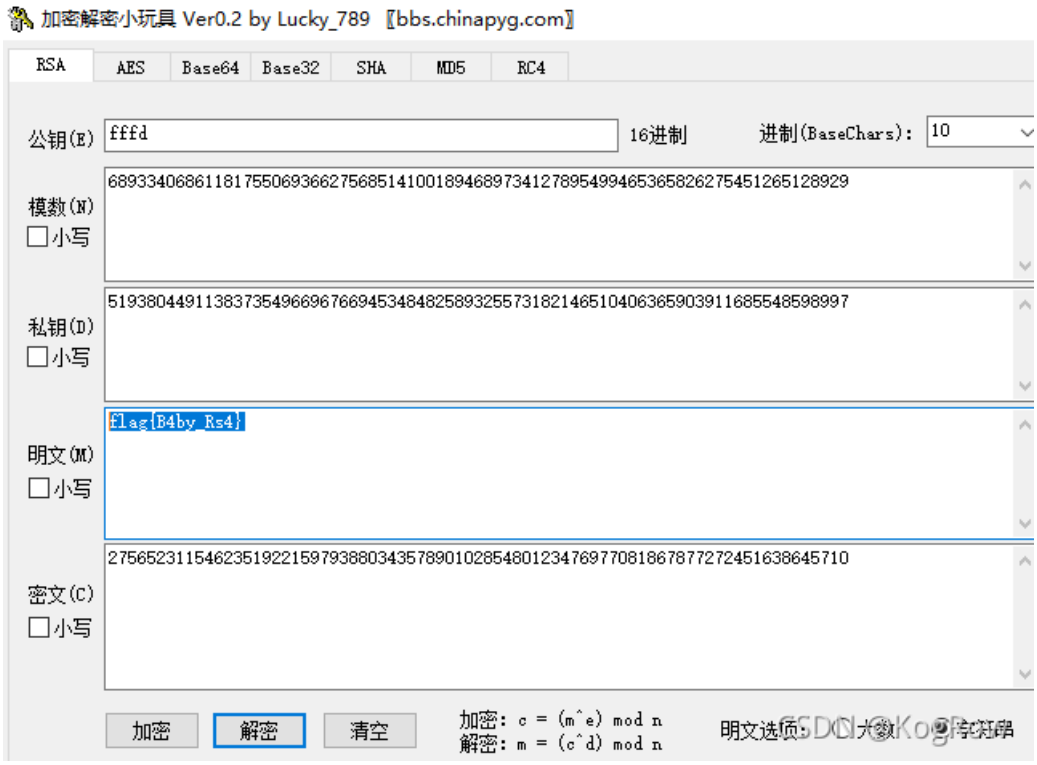
```
from Crypto.Util.number import long_to_bytes

c = 14573903785113823547710005409453611689847750526930736416823750714074908512897030709057495258304830359887371176539714284246123320209259266173955588681603806019124982
N = 17760650483649924697095903022687160888596932177821105108052463408451697333144164499389802957361229009585306926403653045925365287558626794687783105514754691022710056
d =11421103406975333421340765323223945381830555406901332890943482580615352127928157661508519703908198962112442972768902360002419925826186240372948581699098548699684783
enc = pow(c,d,N)
print long_to_bytes(enc)
```


运行得到佛莱格。

28.[HDCTF2019]basic rsa

送分题：


加密解密小玩具 Ver0.2 by Lucky_789 〖bbs.chinapyg.com〗
RSA
AES
Base64
Base32
SHA
MD5
RC4
公钥(E)
fffd
16进制
进制(BaseChars):
10
模数(N)
小写
68933406861181755069366275685141001894689734127895499465365826275451265128929
私钥(D)
小写
519380449113837354966967669453484825893255731821465104063659039116855485989 97
明文(M)
小写
flag{B4by_Rs4}
密文(C)
小写
27565231154623519221597938803435789010285480123476977081867877272451638645710
加密
解密
清空
加密: c = (m^e) mod n
解密: m = (c^d) mod n
明文选项:
大数
字符串