BMZCTF Crypto

原创

Crazy198410 于 2021-01-07 08:37:57 发布 260 收藏 1

分类专栏： BMZCTF-Crypto 文章标签： 安全

本文链接：https://blog.csdn.net/Crazy198410/article/details/112301809

2018 HEBTUCTF Sudoku&Viginere

下载附件，为一个doc文件，打开后发现是一个数独游戏，和一个密码。

数独，每个块内可填：1,3,5,a,e,r,t,y,_

After solving Sudoku, you will find Viginere's secret:45 34 57 74 15 35 26 86 47 39

	_		a				5	
r							1	
e	3				1			
			_		y		3	t
					e			
					5			
						3	e	
1		e					_	
t				r				a

完成数独后：

y	_	1	a	t	r	e	5	3
r	t	5	y	e	3	a	1	_
e	3	a	5	_	1	r	t	y
a	e	r	_	1	y	5	3	t
5	y	t	r	3	e	_	a	1
3	1	_	t	a	5	y	r	e
_	a	y	1	5	t	3	e	r
1	r	e	3	y	a	t	_	5
t	5	3	e	r	_	1	y	a

看数独表很像Viginere的密码表，根据`45 34 57 74 15 35 26 86 47 39`，在密码表上选取：
密文 先横后竖 先竖后横
45 r 1
34 r 5
57 y _
74 5 1
15 5 t
35 t _
26 1 3
86 r a
47 1 5
39 3 y

得到两个解密后的字符串：
rry55t1r13
15_1t_3a5y
分别尝试提交，最后确定flag为：HEBTUCTF{15_1t_3a5y}

4进制
下载附件，为一个文本文件，打开后，是一串数字：
`1212 1230 1201 1213 1323 1012 1233 1311 1302 1202 1201 1303 1211 301 302 303 1331`
根据题目可知是4进制。我们要转为十进制，再查ASCII表。以第一个数字为例：
1212=1*43*+2*42*+1*41*+240=102，查表为f
编写程序：

```
s='1212 1230 1201 1213 1323 1012 1233 1311 1302 1202 1201 1303 1211 301 302 303 1331'
l=s.split(' ')
for i in l:
    tmp=0
    for j in range(len(i)):
        tmp+=(4**j)*int(i[-j-1])
print (chr(int(tmp)),end='')
```

运行后就可以得到flag

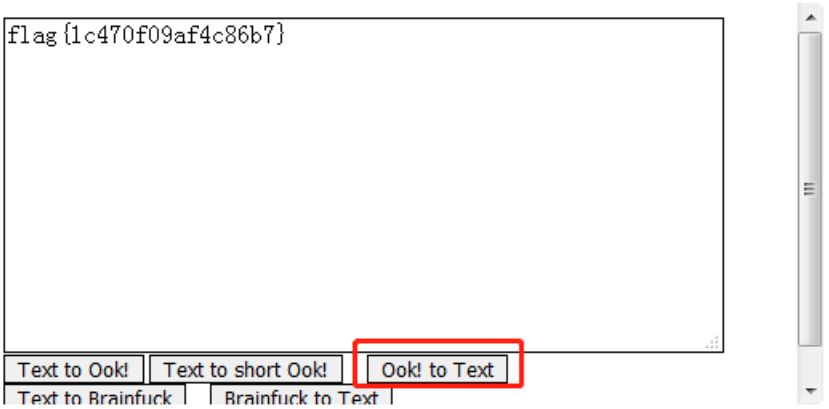
栅栏密码

打开题目，给出一串字符串：`fa{660cb679d7866ffalg7d27e041cfbd18ed}`
根据题目，进行栅栏密码解密，使用密码机器：

得到flag

Ook

下载附件，是一个文件文件，打开后都是OoK的编码，
打开https://www.splitbrain.org/services/ook，输入密文，选择Ook to text



就可以得到flag

CRC32 BOOM!

下载附件，为一个压缩包

名称	大小	压缩后大小	类型	修改时间	CRC32
..			文件夹		
1.txt *	6	18	文本文档	2019/4/11 22...	8E234AE0
2.txt *	6	18	文本文档	2019/4/11 22...	8115A277
flag.jpg *	72	80	JPG 图片文件	2019/3/27 23...	26834817

看到1,.txt和2.txt的文件大小很小，只有6个字节，尝试使用crc爆破：
使用CRC32.py进行爆破：

```
λ python2 crc32.py reverse 0x8E234AE0
4 bytes: {0x0e, 0xf5, 0x08, 0x69}
verification checksum: 0x8e234ae0 (OK)
alternative: 17LE1V (OK)
alternative: 53QD05 (OK)
alternative: 5Cm55e (OK)
alternative: 6cfFvu (OK)
alternative: 9l8GQb (OK)
alternative: BzTRNZ (OK)
alternative: EcSldq (OK)
alternative: HqVcs7 (OK)
alternative: JpB1Br (OK)
alternative: LuKbrT (OK)
alternative: TRyOcs (OK)
alternative: _5IbAC (OK)
alternative: bugku_ (OK)
alternative: dpn8Ey (OK)
alternative: mz9jRH (OK)
alternative: q5ekFH (OK)
alternative: w0l8vn (OK)
alternative: yOOyO0 (OK)
alternative: zRUFdx (OK)
```



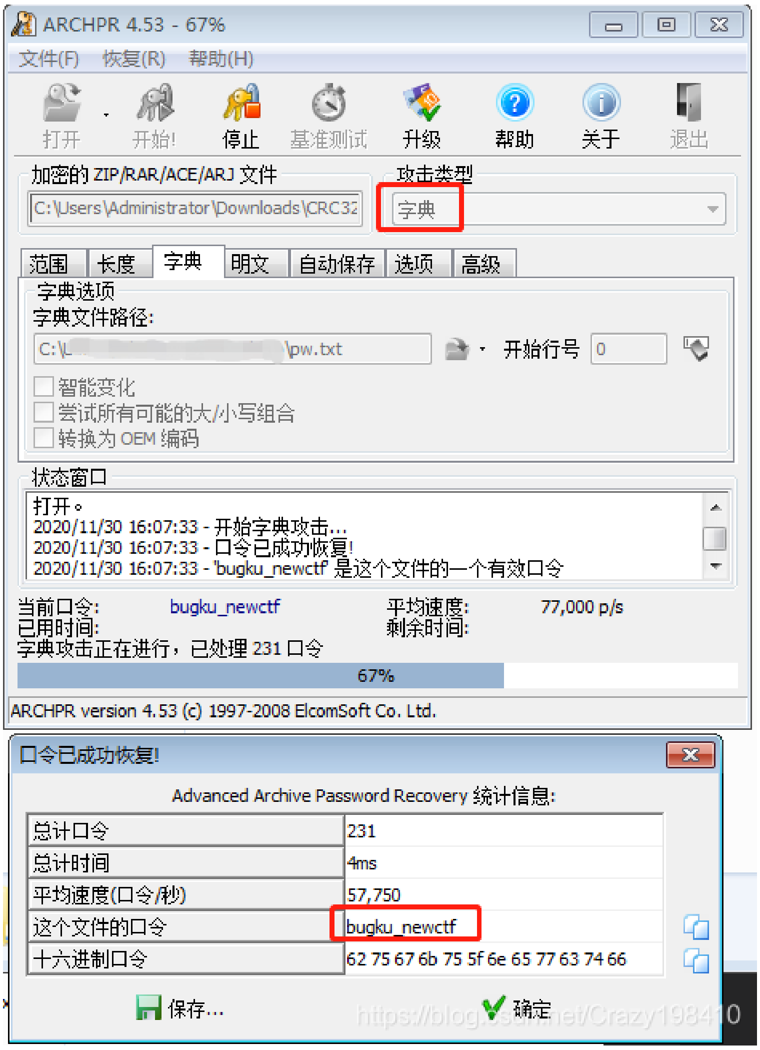
```
λ python2 crc32.py reverse 0x8115A277
4 bytes: {0x76, 0xa2, 0x0b, 0xe1}
verification checksum: 0x8115a277 (OK)
alternative: 1x5NQ8 (OK)
alternative: 9O2a5H (OK)
alternative: G1qi4N (OK)
alternative: IsCdeH (OK)
alternative: NjDZOc (OK)
alternative: OvJ7Un (OK)
alternative: P8CUk2 (OK)
alternative: PT0xov (OK)
alternative: QTqIto (OK)
alternative: Rh5f6k (OK)
alternative: XBiGbJ (OK)
alternative: Z3AdV_ (OK)
alternative: gsombC (OK)
alternative: mY3L6b (OK)
alternative: newctf (OK)
alternative: p7plP7 (OK)
alternative: t3mmQT (OK)
alternative: vBENeA (OK)
```


将两张图内的字符串组成密码字典：

```
s1=["1x5NQ8","9O2a5H","G1qi4N","IsCdeH","NjDZOc","OvJ7Un","P8CUk2","PT0xov","QTqIto","Rh5f6k","XBiGbJ","Z3AdV_",
"gsombC","mY3L6b","newctf","p7plP7","t3mmQT","vBENeA"]
s2=["17LE1V","53QD05","5Cm55e","6cfFvu","9l8GQb","BzTRNZ","EcSldq","HqVcs7","JpB1Br","LuKbrT","TRyOcs","_5IbAC",
"bugku_","dpn8Ey","mz9jRH","q5ekFH","w0l8vn","yOOyO0","zRUFdx"]
l=[]

for i in s2:
    for j in s1:
        l.append(i+j)
with open ('pw.txt', 'w') as fo:
    for k in l:
        fo.write(k+'\n')
```

使用Advanced Archive Password Recovery进行字典爆破：



得到密码为`bugku_newctf`
解压出flag.jpg。结果无法打开。
用文本打开，可以看到flag就在文件尾部：

2018 AFCTF Morse

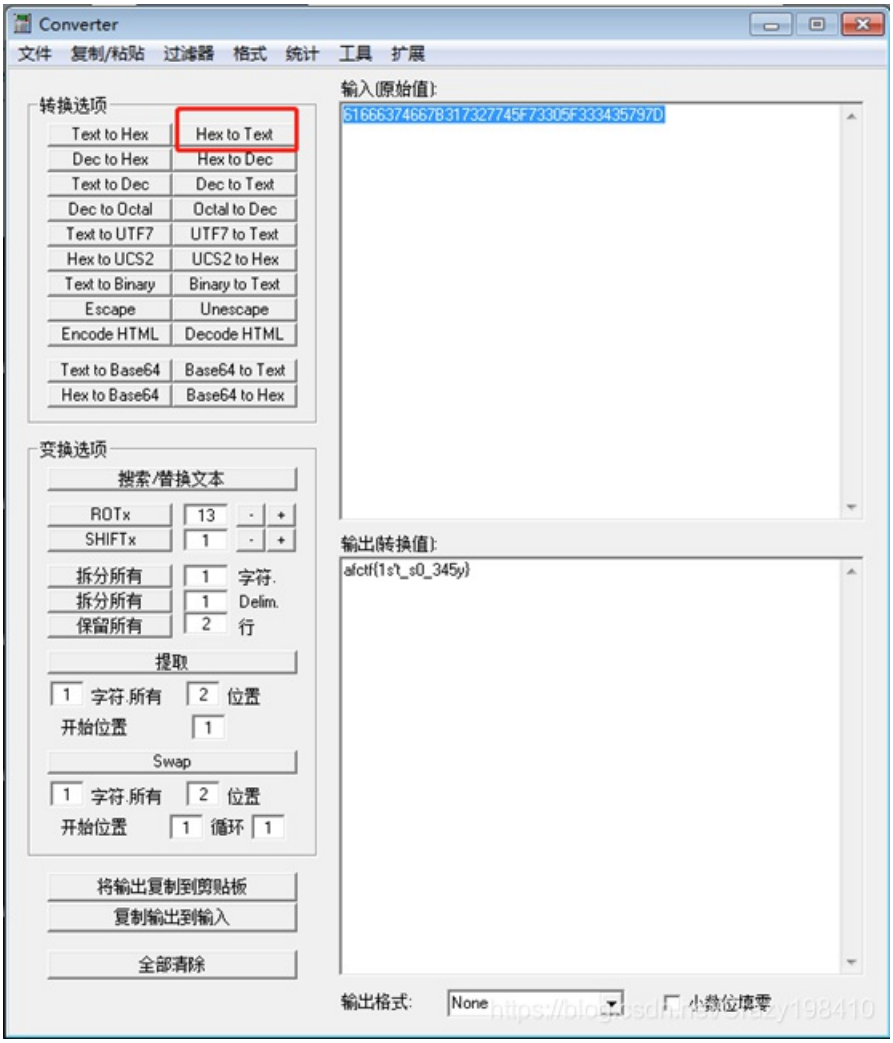
下载附件为一个压缩包，打开后为一个文本，内容如下：

```
-..../.----/-..../-..../-..../...--/--.../....-/-..../-..../--.../-.../...--/.----/--.../...--/..---/--.../--...
/....-/...../..-./--.../...--/...--/-----/...../..-./...--/...--/...--/....-/...--/...../--.../----./--.../-..
```

将`/`替换为空格。进行莫斯解密：
得到：

```
61666374667B317327745F73305F333435797D
```

进行Hex to text：



得到flag

山东省大学生网络技术大赛-baby

下载附件,是一个脚本文件:

```
# -*- coding: utf-8 -*-

from Crypto.PublicKey import RSA
import libnum
import uuid

flag = "flag{***************}"
rsa = RSA.generate(4096,e=3)
p = rsa.p
d = rsa.d
e = rsa.e
N = rsa.n
m = libnum.s2n(flag)
c = pow(m, e, N)
print "[+]c:",c
print "[+]N:",N
'''
[+]c: 344246784248256132370323757453790755403533762276297110321055748005034935987304162433626178273150906891000336054704994248241503686290484460048497667442360486171016603355857692143806855595194896609965890260672529255195234519313297399628856624613870875481051164681136201776906304142511571230562974834120779230569459074206697120252340530156123334199103737410126562326533207078744933299179209709004476197370590921713711964909131345720658980347979789492440201727354371992484959207032839627676038150161293403965369113166771094366231134224937826657958579219178937599421230874628798840627205579064291831558595977568908961920440032408219063325752924761600720395057717945312555422441235169296712773063614670745457208237358063080030919834276783002877094695822824665722300665801952272782147762802137222159530977474534372897344694547124261079671881095489669072378778403160098284762003883273291447838770334912387099544738099911527273336160224065174431305427131672064217870385963129751531658486257219110805612426460922990168026629130170716857405486991638360074742247154265876095493722891819778300926771283688061131314598311823905209428926706964471286314856065799438858122606408057560353775841551357701559157821200251164860615401051393396557229047212946291490250330668235998239644446207792591061769134788393701008912130721000631012326351836365523609527628386563073006211952480592536147451188521635693884180862917488051001750086583878038782000348402155065167156406211656616421773718638745860695240222586429151006155960324431450348470315643566715591792127054661456096984755462109947489491213598530942479905330750043935345654217764687858212612913094632053140578820162660663656360180844991588067170369725908484588910191715832689201806912211684536120296985102 71
'''
```

是一个RSA算法，给出了n，c和e，但e很小为3。可以确定是低指数攻击。写脚本如下：

```
from gmpy2 import iroot
import libnum
n = 691316677109436623113422493782665795857921917893759942123087462879884062720557906429183155859597756890896192
044003240821906332575292476160072039505771794531255542244123516929671277306361467074545720823735806308003091983 4
276783002877094695822824665722300665801952272782147762802137222159530977474534372897344694547124261079671881095 4
896690723787784031600982847620038832732914478387703349123870995447380999115272733361602240651744313054271316720 6
421787038596312975153165848625721911080561242646092299016802662913017071685740548699163836007474224715426587609 5
493722891819778300926771283688061131314598311823905209428926706964471286314856065799438858122606408057560353775 8
415513577015591578212002511648606154010513933965572290472129462914902503306682359982396444462077925910617691347 8
839370100891213072100063101232635183636552360952762838656307300621195248059253614745118852163569388418086291748 8
051001750086583878038782000348402155065167156406211656616421773718638745860695240222586429151006155960324431450 3
484703156435667155917921270546614560969847554621099474894912135985309424799053307500439353456542177646878582126 1
291309463205314057882016266066365636018084499158806717036972590848458891019171583268920180691221168453612029698 5
10271
c = 344246784248256132370323757453790755403533762276297110321055748005034935987304162433626178273150906891000336 0
547049942482415036862904844600484976674423604861710166033558576921438068555951948966099658902606725292551952345 1
931329739962885662461387087548105116468113620177690630414251157123056297483412077923056945907420669712025234053 0
156123334199103737410126562326533207078744933299179209709004476197370590921713711964909131345720658980347979789 4
924402017273543719924849592070328396276760381501612934039653

k = 0
while 1:
    res=iroot(c+k*n,3)
    if(res[1]==True):
        print(libnum.n2s(int(res[0])))
        break
    k=k+1
```

运行后，就可以得到flag

键盘之争

打开之后是看不懂的字符串，根据题目，可能是键盘布局不一样。
将QWER布局转为dvorak布局，使用脚本：

```
dic={r"'":"q",
r",":"w",
r".":"e",
"p":"4",
"y":"t",
"f":"y",
"g":"u",
"c":"i",
"r":"o",
"l":"p",
r"/":r"[",
r"=":r"]",
r'"':'Q',
r"<":"W",
r">":"E",
"P":"R",
"Y":"T",
"F":"Y",
"G":"U",
"C":"I",
"R":"O",
"L":"P",
r"?":r"{",
r"+":r"}",
```

```
"a":"a",
"A":"A",
"o":"s",
"O":"S",
"e":"d",
"E":"D",
"u":"f",
"U":"F",
"i":"g",
"I":"G",
"d":"h",
"D":"H",
"h":"j",
"H":"J",
"t":"k",
"T":"K",
"n":"l",
"N":"L",
"s":";",
"S":":",
r"-":r"'",
r'_':r'"',
r";":"z",
r":":"Z",
"q":"x",
"Q":"X",
"j":"c",
"J":"C",
"k":"v",
"K":"V",
"x":"b",
"X":"B",
"b":"n",
"B":"N",
"m":"m",
"M":"M",
"w":r",",
"W":r"<",
"v":r".",
"V":r">",
"z":r"/",
"Z":r"?",
r'!':"!",
r"@":r"@",
r"#":r"#",
r"$":r"$",
r"%":r"%",
r"^":r"^",
r"&":r"&",
r"*":r"*",
r"(":r"(",
r")":r")",
r"[":r"-",
r"]":r"=",
r"{":r"_",
r"}":r"+"}
s=r'ypau_kjg;"g;"ypau+'

for i in s:
print ("".join([key for key, value in dic.items() if value == i]),end='')
```

运行后，就可以看到flag
将{}内的进行md5加密，就可以提交了。

2020sdnisc-ezRSA

下载附件，打开后为一个文本，内容为：

```
n = 0xa1d4d377001f1b8d5b2740514ce699b49dc8a02f12df9a960e80e2a6ee13b7a97d9f508721e3dd7a6842c24ab25ab87d1132358de7
c6c4cee3fb3ec9b7fd873626bd0251d16912de1f0f1a2bba52b082339113ad1a262121db31db9ee1bf9f26023182acce8f84612bfeb07580
3cf610f27b7b16147f7d29cc3fd463df7ea31ca860d59aae5506479c76206603de54044e7b778e21082c4c4da795d39dc2b9c0589e577a77
3133c89fa8e3a4bd047b8e7d6da0d9a0d8a3c1a3607ce983deb350e1c649725cccb0e9d756fc3107dd4352aa18c45a65bab7772a4c5aef70
20a1e67e6085cc125d9fc042d96489a08d885f448ece8f7f254067dfff0c4e72a63557L
e1 = 0xf4c1158fL
c1 = 120517963665240884892844451092955026863414984269652772300699152941591319762314737899772793642639650994222356
64772377527806056937807146913186636839939477289822416651808959334080391379832745196358999673432349794330181905171
87098075186558685696569412424491099808763976616052715174597166696849009202795974774466296076276937697387336231436
93170696779851882404994923673483971528314806130892416509854017091137325195201225617407959645788145876202882024723
10620418325709475500292470800913856034743255209090548913213515493298752123929957850900829061439870079967092880
56926097569248236280552452272902889406491588625764485378334 23L
e2 = 0xf493f7d1L
c2 = 16648382384980770705624348910895797622774711113202207693584907182552301186239613809347201161450012615995859
73841066145243849675635348553830561494921177666879386498442969679094475089469195779923426450853008402689461122851
36989633474023291098381096216097704069257005209833878114510748384703700446786340992020034809259032675087440061954
55234025325060817223813858985074720872124168142943926467694676717713503559007112874381750005406371400109962943508
34949715114844606484609653144503741617491391592305033224284340392613316581731027263388435826377851677028851559295
98321517624995263631318019451635019993378082080743812127 95L
```

发现只有一个n，两个e，两个c。推测为共模攻击。
使用脚本：

```
from libnum import n2s,s2n
from gmpy2 import invert

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def main():
    n = int('0xa1d4d377001f1b8d5b2740514ce699b49dc8a02f12df9a960e80e2a6ee13b7a97d9f508721e3dd7a6842c24ab25ab87d1
132358de7c6c4cee3fb3ec9b7fd873626bd0251d16912de1f0f1a2bba52b082339113ad1a262121db31db9ee1bf9f26023182acce8f84612
bfeb075803cf610f27b7b16147f7d29cc3fd463df7ea31ca860d59aae5506479c76206603de54044e7b778e21082c4c4da795d39dc2b9c05
89e577a773133c89fa8e3a4bd047b8e7d6da0d9a0d8a3c1a3607ce983deb350e1c649725cccb0e9d756fc3107dd4352aa18c45a65bab7772
a4c5aef7020a1e67e6085cc125d9fc042d96489a08d885f448ece8f7f254067dfff0c4e72a63557',16)
    c1 = 120517963665240884892844451092955026863414984269652772300699152941591319762314737899772793642639650994223564772377527806056937807146913186636839939477289822416651808959334080391379832745196358999673432349794330181905171870980751865586856965694124244910998087639766160527151745971666968490092027959747744662960762769376973873362314369317069677985188240499492367348397152831480613089241650985401709113732519520122561740795964578814587620288202472310620418325709475500292470800913856034743255209090548913213515493298752123929957850900829061439870079967092880569260975692482362805524522729028894064915886257644853783342 3
    c2 = 166483823849807707056243489108957976227747111132022076935849071825523011862396138093472011614500126159958597384106614524384967563534855383056149492117766687938649844296967909447508946919577992342645085300840268946112285136989633474023291098381096216097704069257005209833878114510748384703700446786340992020034809259032675087440061954552340253250608172238138589850747208721241681429439264676946767177135035590071128743817500054063714001099629435083494971511484460648460965314450374161749139159230503322428434039261331658173102726338843582637785167702885155929598321517624995263631318019451635019993378082080743812127 95
    e1 = int('0xf4c1158f',16)
    e2 = int('0xf493f7d1',16)
    s = egcd(e1, e2)
    s1 = s[1]
    s2 = s[2]

    if s1<0:
        s1 = - s1
        c1 = invert(c1, n)
    elif s2<0:
        s2 = - s2
        c2 = invert(c2, n)

    m = pow(c1,s1,n)*pow(c2,s2,n) % n
    print n2s(m)

if __name__ == '__main__':
  main()
```

运行后得到flag：

```
C:\                                    ython2.e
flag{8c16c91be3f3287ff5a10167e922b33b}
```

baby_dsa

DSA完全不懂，现学，相关知识如下链接：
https://www.jarvisw.com/?p=169
用大佬的脚本：

```
from Crypto.Util.number import *
from hashlib import sha512,md5

def hash(message):
    return int(sha512(message).hexdigest(), 16)


def sign(message, pubkey, privkey, random1):
    p, q, g, y = pubkey
    x = privkey
    k = pow(y, x, g) * random1 % q
    r = pow(g, k, p) % q
    s = inverse(k, q) * (hash(message) + x * r) % q
    return r, s

public = (329722646303732445800883728449896337264903888939068505184968017501650564600176122010985892162426604403
513313413540256123563583342820688688830802777235303076740092107834686837729840121381205325031600203394169227219 2
644613252296579884516731560436501073253924457646558698855484781747029397755111633297587215976579633451933658235 3
853865395180065700696535751460600168119111406146064719303273413685829798360425854068113522363260652926364845508 0
721375648215308442771454969426468569597753153742568221215555356884866608857693288823465935521366490975378175391 7
401161977762663658097504411914908081677033980915039079517766159760522261279115347385813009437510156898969769563 6
878694957219772654447995856340198809515320802179604569017889184392657881699100628228895801993664174551865954899 7
300035177020048509500849422882930014503969593694637958562505140255303497120747476246314774446736015884759335603 0
745194143276254949463650698210515569533, 823028354421121371258914033681512499102687068248547861266003904136223022
196443, 115623326429934097110649837149549569522588059235437403414219551847254052191169950639131132467659068536523
348871703457221350600098850023347488364771698061661698061802317949189612145206983618748391104546102663883419779 8
490275656983859461625511266160046681887013743277280036885946144585470095629188557685506940518377190307627714492 7
769029433730710613058788277691211698675287829143272152835171859480781061918556840079857761203012054552142555673 0
718653103553319862886064227115257908775913767708341806184927942653621786031112366154952256121012503444219325880 3
824480419922944973867508256051206256436547303509726388925793714077899338930589337851434403235280652197236799102 7
459721160744835688761657797398841523104074451793557924512992305640697344011520550723893828185707635141404445213 4
459355869652894502820242220644889658787699915663671151536197615838435615795317050579559332880085561659520661733 0
489139137510034631277653953044861100 5, 290999623787731812697719691852061290246619413463636312382146969900546384 5
147107828431539627048519160916016790288308661763323315195151563014015371730699081815090284643226473522566324246 8
480934912102426259700691370748381111764419748195905378547508340647258309914050650507130019335600244300775022052 4
932219191932969202270343323955035291396808472686684787610559114702054784699365490860392737061056233160308943296 4
785407983531348789370883366729281628943329617622775593458604799162480868211178119903910251871251930740590010864 4
130597713325277469899665312229712344783744916865734730801627003088139567406642114400295975193683916693572620083 3
785132736328859710351871352567511516142170956091885352178579302299634322254818383978585773136692588922976043617 3
379045453961467556092841637434762977726865484751701976054128476895871715224532290559327127141548699894548085614 5
885203176911948923559840206692408277837608149463225844843404856205 3)
p, q, g, y = public

tmp1 = (b'0234e7971889def7e60348f77db94b7a', (108592362699597657352363937799363052173055743318392345021902267089
29991582386, 137075573238956952604710531378285238377458956832183313433600273803109801088 19))
tmp2 = (b'16c5ac270b72f70319657b4410d985d4', (419606422463790676405247094160015360582928173191097643173697772244
26218746518, 746767253225155935023462754688434115637469821493736700210826863413690767190 88))
message1, r1, s1 = tmp1[0], tmp1[1][0], tmp1[1][1]
message2, r2, s2 = tmp2[0], tmp2[1][0], tmp2[1][1]
hm1, hm2 = hash(message1), hash(message2)
#print(hm1, hm2)


for random1 in range(1, 512):
    for random2 in range(1, 512):
        random1_inv = inverse(random1, q)
        random_mul = random1_inv * random2
        x = (s1 * hm2 - s2 * hm1 * random_mul) * inverse(s2 * r1 * random_mul - s1 * r2, q)
        x = x % q
        #print(x)
```

```
        #print(x)
        #print(sign(message1, public, x, random1))
        #print(tmp1[1])
        if sign(message1, public, x, random1) == tmp1[1]:
            print(x)
            flag = 'flag{'+md5(long_to_bytes(x)).hexdigest()+'}'
            print(flag)
            print(random2)
            exit()
    print(random1)
```

得到flag:

```
23497022217148412421132818343765031481439042037015488649255576621612254005403
flag{448018d725636e5ca1ed9675a8b23df4}
```

2018 AFCTF BASE

下载附件，是一个文本，很大，打开看了下，是一个很长的字符串，有0_9, $_AZ$，怀疑是16进制，

解码后，看到后面有个“=”，怀疑是base64。

```
54 6B 52 56 4D 30 35 45 55 6B 64 4F 56 45 6B 79   TkRVM05EUkdOVEky
55 58 70 57 51 6B 35 46 56 54 46 4F 56 46 70 43   UXpWQk5FVTFOVFpC
54 6B 56 46 4D 46 46 36 56 58 70 4F 56 46 56 36   TkVFMFF6VXpOVFV6
54 6C 52 52 4D 55 35 55 53 54 46 4F 56 45 30 31   TlRRMU5USTFOVE01
54 6C 52 6A 4D 55 31 55 54 58 64 4F 65 6C 45 77   TlRjMU1UTXdOelEw
55 58 70 56 4D 45 35 72 53 54 46 4F 61 6C 55 7A   UXpVME5rSTFOalUz
54 6C 52 4E 65 6B 31 55 55 58 6C 4F 52 55 6B 77   TlRNek1UUXlORUkw
55 6C 52 56 4D 55 35 71 5A 7A 46 4F 61 6C 4A 47   UlRVMU5qZzFOalJG
54 6D 74 4E 4D 55 31 55 59 7A 56 4F 56 46 55 77   TmtNMU1UYzVOVFUw
54 6C 52 61 51 30 31 36 61 7A 30 3D               TlRaQ016az0=
```

多次base64后，看到结尾是“=”，但解码后是乱码，看到题目是base。怀疑不是base32，就是base16。
写脚本：

```
import base64
with open (r'C: \flag_encode.txt','r',encoding='utf-8') as f:
    for a in f:
        while 1:
            try :
                a=base64.b64decode(a).decode("utf-8")
            except:
                pass
            try:
                a=base64.b32decode(a).decode("utf-8")
            except:
                pass
            try:
                a=base64.b16decode(a).decode('utf-8')
            except:
                pass
            if "{" in a:
                print (a)
                break
```

运行后得到flag

2018 AFCTF MagicNum

下载附件，是一个文本，里面是好几个小数，

```
720659105101771380000000000000000.000000
71863209670811371000000.000000
184896826254127600000000000000000.000000
72723257588050687000000.000000
46746591674697662000000000.000000
190616988374992920000000000000000000000.000000
```

不明白有什么用，先进行进制转换
将浮点型转为16进制：
运行脚本，就可看到flag:

```
import struct
import binascii

s=[720659105101771380000000000000000.000000,71863209670811371000000.000000,184896826254127600000000000000000.000000,72723257588050687000000.000000,46746591674697662000000000.000000,190616988374992920000000000000000000000.000000]
a=''
b=''
for i in s:
    i=float(i)
    tmp=struct.pack('<f', i).hex()#小端
    a+=tmp
for j in s:
    j=float(j)
    tmp=struct.pack('>f', j).hex()#大端
    b+=tmp

print (binascii.a2b_hex(a))
print (binascii.a2b_hex(b))
```