

Cryptohack刷题记录(一) General部分 WP

原创

[Mr_AgNO3](#)  已于 2022-01-24 00:46:00 修改  284  收藏 1

文章标签: [python](#) [密码学](#)

于 2022-01-24 00:43:44 首次发布

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/u010883831/article/details/122659878>

版权

[cryptohack](#), 很不错的一个密码学习平台。

很适合没有基础的beginner系统进行学习

文章目录

[注册平台账号](#)

[General](#)

[Encoding](#)

[第一题](#)

[第二题](#)

[第三题](#)

[第四题](#)

[第五题](#)

[XOR](#)

[1. XOR Starter](#)

[2. XOR Properties](#)

[3.Favourite byte](#)

[4.You either know, XOR you don't](#)

[5. Lemur XOR](#)

[MATHEMATICS](#)

[1. Greatest Common Divisor](#)

[2. Extended GCD](#)

[3. Modular Arithmetic 1](#)

[4. Modular Arithmetic 2](#)

[5. Modular Inverting](#)

[DATA FORMATS](#)

[1. Privaty-Enhanced Mail?](#)

[2. CERTainly not](#)

[3. SSH KEY](#)

[4. Transparency](#)

注册平台账号

Solve this Roman emperor's cipher: EJEXA JVAN KTGXDC ITCI

Enter four-word solution here

EJEXA JVAN

KTGXDC ITCI

罗马皇帝密码，那就是凯撒了

```
for i in range(26):
for j in c:
if j == ' ':
print(' ',end=")
continue
t = ss.index(j)
print(ss[(t+i)%26],end=")
print()
```

EJEXA JVAN KTGXDC ITCI
FKFYB KWBO LUHIYED JUDJ
GLGZC LXCP MVIJZFE KVEK
HMHAD MYDQ NWJKAGF LWFL
INIBE NZER OXKLBHG MXGM
JOJCF OAFS PYLMCIH NYHN
KPKDG PBGT QZMNDJI OZIO
LQLEH QCHU RANOEKJ PAJP
MRMFI RDIV SBOPFLK QBKQ
NSNGJ SEJW TCPQGML RCLR
OTOHK TFKX UDQRHNM SDMS
PUPIL UGLY VERSION TENT
QVQJM VHMZ WFSTJPO UFOU
RWRKN WINA XGTUKQP VGPV
SXSLO XJOB YHUVLRQ WHQW
TYTMP YKPC ZIVWMSR XIRX
UZUNQ ZLQD AJWXNTS YJSY
VAVOR AMRE BKXYOUT ZKTZ
WBWPS BNSF CLYZPVU ALUA
XCXQT COTG DMZAQWV BMVB
YDYRU DPUH ENABRXW CNWC
ZEZSV EQVI FOBCSYX DOXD
AFATW FRWJ GPCDTZY EPYE
BGBUX GSXK HQDEUAZ FQZF
CHCVY HTYL IREFVBA GRAG
DIDWZ IUZM JSFGWCB HSBH

General

Encoding

第一题

☆ ASCII

5 pts · 16765 Solves

ASCII is a 7-bit encoding standard which allows the representation of text using the integers 0-127.

Using the below integer array, convert the numbers to their corresponding ASCII characters to obtain a flag.

```
[99, 114, 121, 112, 116, 111, 123, 65, 83, 67, 73, 73, 95, 112, 114, 49, 110, 116, 52, 98, 108, 51, 125]
```



In Python, the `chr()` function can be used to convert an ASCII ordinal number to a character (the `ord()` function does the opposite).

Enter flag here: crypto{FLAG}

SUBMIT

```
>>> a = [99, 114, 121, 112, 116, 111, 123, 65, 83, 67, 73, 73, 95, 112, 114, 49, 110, 116, 52, 98, 108, 51, 125]
>>> "".join(chr(i) for i in a)
'crypto{'
```

第二题

When we encrypt something the resulting ciphertext commonly has bytes which are not printable ASCII characters. If we want to share our encrypted data, it's common to encode it into something more user-friendly and portable across different systems.

Included below is a flag encoded as a hex string. Decode this back into bytes to get the flag.

```
63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f737472696e67735f615f6c6f747d
```

💡 In Python, the `bytes.fromhex()` function can be used to convert hex to bytes. The `.hex()` instance method can be called on byte strings to get the hex representation.

Enter flag here: crypto{FLAG}

SUBMIT

```
63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f737472696e67735f615f6c6f747d
```

```
>>> from binascii import *
>>> unhexlify('63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f737472696e67735f615f6c6f747d')
b'crypto{'
```

第三题

Another common encoding scheme is Base64, which allows us to represent binary data as an ASCII string using 64 characters. One character of a Base64 string encodes 6 bits, and so 4 characters of Base64 encode three 8-bit bytes.

Base64 is most commonly used online, so binary data such as images can be easily included into HTML or CSS files.

Take the below hex string, *decode* it into bytes and then *encode* it into Base64.

```
72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf
```



In Python, after importing the base64 module with `import base64`, you can use the `base64.b64encode()` function. Remember to decode the hex first as the challenge description states.

Enter flag here: crypto/FLAG/

SUBMIT

```
72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf
```

```
>>> c = '72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf'
>>> from base64 import b64encode as e
>>> unhexlify(c)
b'\xbcl\xa9\xb6\x8f\xc1j\xc7\xbe\xeb\x8f\x84\x9d\xca\x1d\x8a>\x8a\xcf\x96y\xbf\x92i\xf7\xbf'
>>> e(unhexlify(c))
b'crypto/'
```

有点意思

第四题

Cryptosystems like RSA works on numbers, but messages are made up of characters. How should we convert our messages into numbers so that mathematical operations can be applied?

The most common way is to take the ordinal bytes of the message, convert them into hexadecimal, and concatenate. This can be interpreted as a base-16 number, and also represented in base-10.

To illustrate:

```
message: HELLO
ascii bytes: [72, 69, 76, 76, 79]
hex bytes: [0x48, 0x45, 0x4c, 0x4c, 0x4f]
base-16: 0x48454c4c4f
base-10: 310400273487
```

💡 Python's PyCryptodome library implements this with the methods `Crypto.Util.number.bytes_to_long()` and `Crypto.Util.number.long_to_bytes()`. You may first have to install PyCryptodome and import it with `from Crypto.Util.number import *`. For more details check the [FAQ](#).

Convert the following integer back into a message:

```
11515195063862318899931685488813747395775516287289682636499965282714637259206269
```

Enter flag here: crypto{FLAG}

SUBMIT

```
>>> from number import *
>>> l2b(11515195063862318899931685488813747395775516287289682636499965282714637259206269)
b'crypto{'
```

第五题

Now you've got the hang of the various encodings you'll be encountering, let's have a look at automating it.

Can you pass all 100 levels to get the flag?

The `13377.py` file attached below is the source code for what's running on the server. The `pwntools_example.py` file provides the start of a solution using the incredibly convenient `pwntools` library, which we recommend. If you'd prefer to use Python's in-built `telnetlib`, `telnetlib_example.py` is also provided.

For more information about connecting to interactive challenges, see the [FAQ](#). Feel free to skip ahead to the cryptography if you aren't in the mood for a coding challenge!

Connect at `nc socket.cryptohack.org 13377`

`13377.py`

`pwntools_example.py`

`telnetlib_example.py`

Enter flag here: crypto{FLAG}

SUBMIT

nc到服务器后，会给你一个加密方式 `type` 和一个加密后的字符串 `c`

你需要解密c之后给服务器发送过去，重复100次

服务端代码：

```
if encoding == "base64":
    encoded = base64.b64encode(self.challenge_words.encode()).decode()
elif encoding == "hex":
    encoded = self.challenge_words.encode().hex()
elif encoding == "rot13":
    encoded = codecs.encode(self.challenge_words, 'rot_13')
elif encoding == "bigint":
    encoded = hex(bytes_to_long(self.challenge_words.encode()))
elif encoding == "utf-8":
    encoded = [ord(b) for b in self.challenge_words]
```

要重复100次，所以肯定不能手动解，所以用 `pwntools` 的 `remote` 进行连续破解

```

from pwn import *
import json

r = remote('socket.crytohack.org', 13377, level = 'debug')

def json_rcv():
    line = r.recvline()
    return json.loads(line.decode())

def json_send(hsh):
    request = json.dumps(hsh).encode()
    r.sendline(request)

from binascii import *
from number import *
from base64 import b64decode as dd

from string import *
def rot13(x):
    ss = ascii_lowercase
    res = ""
    for i in x:
        if i in ss:
            res += ss[(13+ss.index(i))%26]
        else:res += i
    return res

def dec(tp,c):
    if tp == 'bigint':
        m = unhexlify(c[2:]).decode()
    elif tp == 'base64':
        m = dd(c.encode()).decode()
    elif tp == 'rot13':
        m = rot13(c)
    elif tp == 'hex':
        m = l2b(int(c,16)).decode()
    elif tp == 'utf-8':
        m = "".join(chr(i) for i in c)
    return m

def func(received):
    print("Received type: ",end="")
    print(received["type"])
    print("Received encoded value: ",end="")
    print(received["encoded"])
    c = received["encoded"]
    tp = received["type"]
    to_send = {"decoded": dec(tp,c)}
    print('to_send:',to_send)
    json_send(to_send)
    rrr = json_rcv()
    print(rrr,'=====',sep = '\n')
    return rrr

rcvd = json_rcv()
for i in range(100):
    rcvd = func(rcvd)

```

得到flag


```
b'{"decoded": "giving_journals_running"}\n'
[DEBUG] Received 0x4f bytes:
b'{"type": "hex", "encoded": "72657475726e656445f70656e64616e745f67726561746c79"}\n'
{'type': 'hex', 'encoded': '72657475726e656445f70656e64616e745f67726561746c79'}
=====
Received type: hex
Received encoded value: 72657475726e656445f70656e64616e745f67726561746c79
to sent: {'decoded': 'returned_pendant_greatly'}
[DEBUG] Sent 0x28 bytes:
b'{"decoded": "returned_pendant_greatly"}\n'
[DEBUG] Received 0x29 bytes:
b'{"flag": "crypto{3nc0d3_d3c0d3_3nc0d3}"}\n'
{'flag': 'crypto{3nc0d3_d3c0d3_3nc0d3}'}
=====
>>>
```

Ln: 072, Col: 1

XOR

1. XOR Starter

将 `label` 的每一位与 `13` 进行异或

题目说了可以用 `pwntools` 里面的 `xor` 函数

```
>>> from pwn import *
>>> xor('label',13)
b'aloha'
```

`crypto{aloha}`

2. XOR Properties

题目给了如下信息

```
k1
k2 ^ k1
k2 ^ k3
flag ^ k1 ^ k2 ^ k3
```

分别记为 `A`、`B`、`C`、`D`，则 `flag = D ^ A ^ C`

```
>>> a = 0xa6c8b6733c9b22de7bc0253266a3867df55acde8635e19c73313
>>> b = 0x37dcb292030faa90d07eec17e3b1c6d8daf94c35d4c9191a5e1e
>>> c = 0xc1545756687e7573db23aa1c3452a098b71a7fb0fddddd5fc1
>>> d = 0x04ee9855208a2cd59091d04767ae47963170d1660df7f56f5faf
>>> f = d^a^c
>>> from number import *
>>> l2b(f)
b'crypto{'
```

3.Favourite byte

```
73626960647f6b206821204f21254f7d694f7624662065622127234f726927756d
```

将数字转为byte型后为

```
>>> l2b(int('73626960647f6b206821204f21254f7d694f7624662065622127234f726927756d',16))
b"sbi`d\x7fk h! O!%O}iOv$f eb!#Ori'um"
```

已知flag前几位，为 `crypto{`，与解出的字符串对应前几位进行异或便可以得到secret

```
>>> xor(b"sbi`d\x7fk",b"crypto{")
b"\x10\x10\x10\x10\x10\x10\x10\x10"
```

得到flag

```
>>> xor(b"sbi`d\x7fk h! O!%O}iOv$f eb!#Ori'um","\x10")
b"crypto{'
```

4. You either know, XOR you don't

```
0e0b213f26041e480b26217f27342e175d0e070a3c5b103e2526217f27342e175d0e077e263451150104
```

尝试同同样的步骤，找出异或的字符串

```
>>> c = l2b(int('0e0b213f26041e480b26217f27342e175d0e070a3c5b103e2526217f27342e175d0e077e263451150104',16))
>>> c
b"\x0e\x0b!?\&\x04\x1eH\x0b&!\x7f4.\x17]\x0e\x07n<[\x10>%&!\x7f4.\x17]\x0e\x07~&4Q\x15\x01\x04"
>>> xor(c[:7],'crypto{')
b'myXORke'
>>> xor(c[-1,'],'')
b'y'
```

拼接起来可以得到一段有意义的语句 `myXORkey`

猜测这就是加密的字符串

```
>>> xor(c,'myXORkey')
b"crypto{'
```

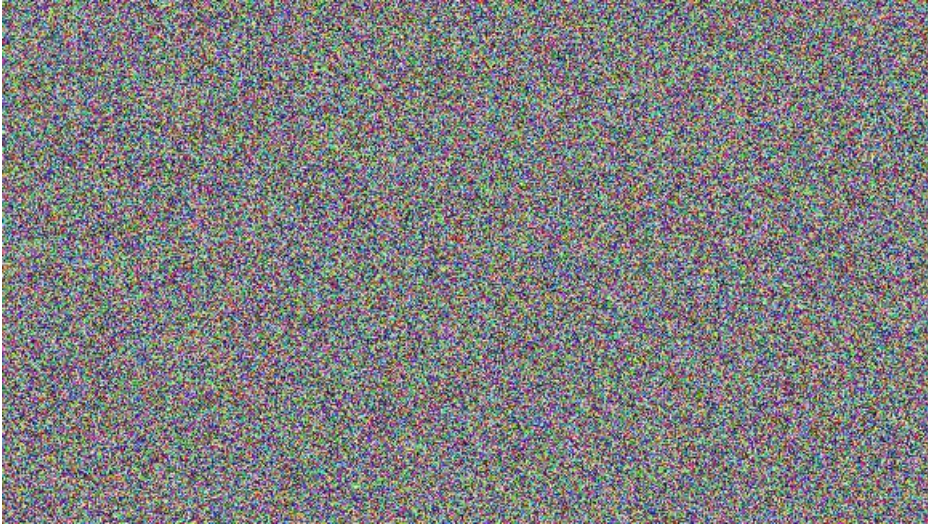
5. Lemur XOR

这是个图片题

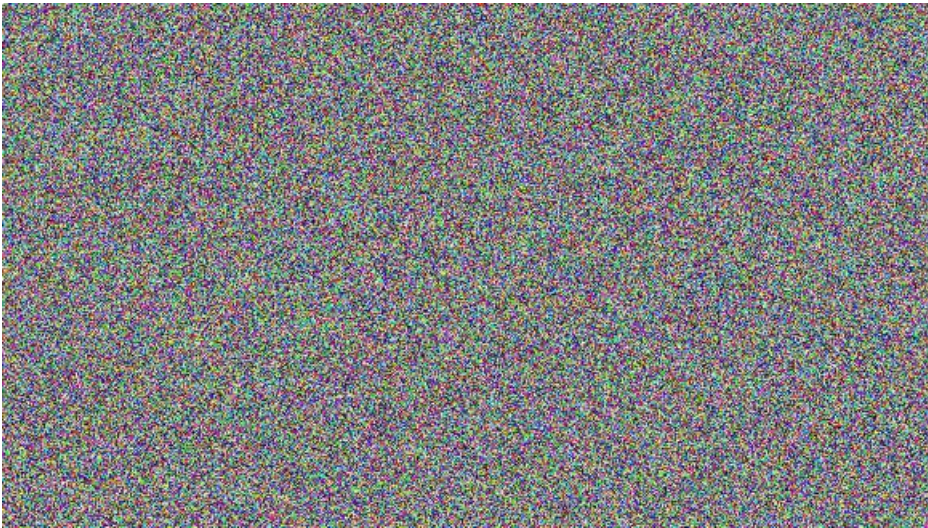
```
ps.
lemur n. 狐猴
```

两个图片

lemur.png:



flag.png:



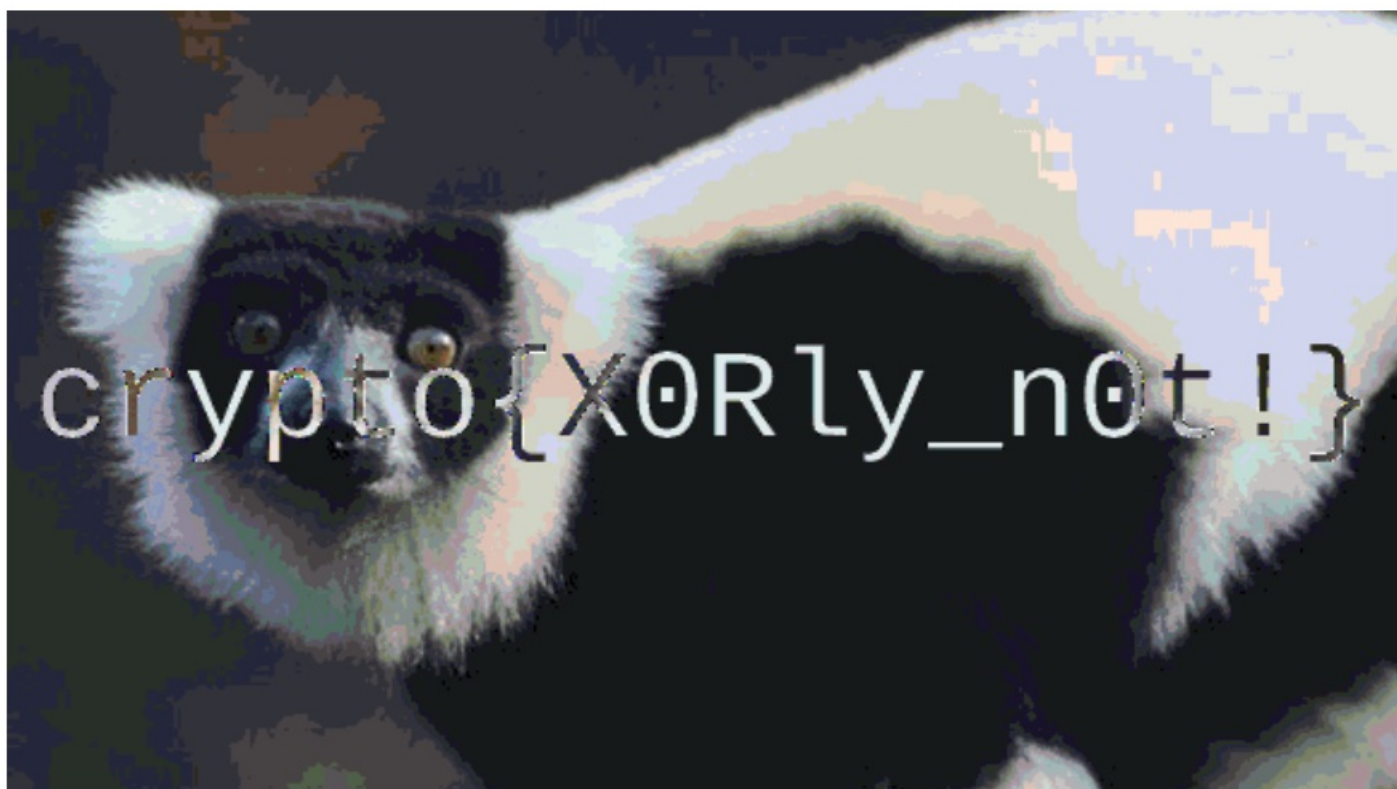
异或试试，使用 `PIL.Image` 模块和 `numpy` 模块

```
from PIL import Image as img
import numpy as np

ll = img.open(r".\lemur.png")
ff = img.open(r".\flag.png")

nl = np.array(ll)
nf = np.array(ff)
img.fromarray(nl^nf).show()
```

得到flag，在图片中



MATHEMATICS

1. Greatest Common Divisor

计算GCD

属实是零基础入门题了

```
>>> a = 66528; b = 52920
>>> from gmpy2 import *
>>> gcd
<built-in function gcd>
>>> gcd(a,b)
mpz(1512)
```

2. Extended GCD

要求扩展欧几里得算法

使用gmpy2库自带的gcdext()函数

```
>>> gcdext
<built-in function gcdext>
>>> p = 26513;q = 32321
>>> gcdext(p,q)
(mpz(1), mpz(10245), mpz(-8404))
```

按照要求，flag是 -8404

3. Modular Arithmetic 1

解方程

$$11 \equiv x \pmod{6}$$

4. Modular Arithmetic 2

费马小定理，当p为素数时，有

$$a^{p-1} \equiv 1 \pmod{p}$$

5. Modular Inverting

求逆元

b是a对m的逆元，则在模m的情况下有 $ab \equiv 1 \pmod{m}$

使用gmpy2中的 `invert()` 函数

```
>>> invert(3,13)
mpz(9)
```

DATA FORMATS

1. Privaty-Enhanced Mail?

有点懵，啥意思

啊

给了一个pem文件，把里面的私钥信息解密出来即可

```
>>> c = open(r"./ppp").read()
>>> p = RSA.import_key(c)
>>> p.d
1568270028805633136478717104581997365499114994919795992986086122818002170731685192445620554366556581089267419005
9831330231436970914474774562714945620519144389785158908994181951348846017432506464163564960993784254153395406799
1013147600334450651934295925123499520209829322185244623410021020634354893188133164645116217369439384407104706949
1233623768021974620459512895916180059521636623753829644733537581887195252002699310214832889708354718428649324119
1505953601668858941129790966909236941127851370202421135897091086763569884760099112291072056970636380417349019579
768748054760104838790424708988260443926906673795975104689
```

私钥d即flag

2. CERTainly not

PEM和DER是证书编码的两个方式，可以互相转化

PEM是可读的文本格式的，以"-----BEGIN XXX-----"开头，同样格式END结尾的文件，内容经过base64加密

DER是二进制文件，不能直接看懂，直接二进制方式读取

同样使用Crypto.RSA.import_key函数

```
>>> ddd = open(r".\der.der", 'rb').read()
>>> p = RSA.import_key(ddd)
>>> p
RsaKey(n=2282537369201953080430621286460951277537417182399370851650989763154751363463585637562400373706803454904
7677999310941837454378829351398302382629658264078775456838626207507725494030600516872852306191255492926495965536
3792718753104573191079360207300504762352786715282658175714339195611756650961711897584061364539879662552369637826
6606696265467846495007592306032735869135663290860649823175596356738233901098522262320558692346640580921742667033
3410014429905146941652293366212903733630083016398810887356019977409467374742266276267137547021576874204809506045
914964491063393800499167416471949021995447722415959979785959569497, e=65537)
```

flag是整数内容的模，也就是n的值

3. SSH KEY

泪目了 他讲的好详细

主要是ssh,Secure Shell Protocol这个东西

明明简写是SSP为什么要叫SSH呢?????

为什么要有ssh? 为了安全。

As the internet became increasingly hostile, people realised the need for both authentication and encryption for administrative network traffic

以及SSH为什么安全

依旧使用同样方法即可

```
>>> c = open(r"./bruce_rsa_6e7ecd53b443a97013397b1a1ea30e14.pub").read()
>>> p = RSA.import_key(c)
>>> p
RsaKey(n=3931406272922523448436194599820093016241472658151801552845094518579507815990600459669259603645261532927
6111529849428408898987565320608948570451753001457658006334990054517388720813812670040698655573956385500411142061
4308540360723410929328633639355275689398460521435298870525863897945473651499731422366907590078380671539888031069
5945945147755132919037973889075191785977797861557228678159538882153544717797100401096435062359474129755625453831
8824906035601344770432354332027089486152345369847158721133438127601028123231803915444960301636530469314147238513
7455487303658228238990483859766828654333742658168081779603871122840144324465516219930235201796499786667731716101
4083116730535875521286631858102768961098851209400973899393964931605067856005410998631842673030901078008408649613
5381437999598036850415669645144898092119629845343223483944280109089843189404116989611507312043166706466769763619
58828528229837610795843145048243492909, e=65537)
```

flag还是n

4. Transparency

有点懵

给了一个PEM格式的公钥文件，打开文件可以得到以下信息

```
>>> from Crypto.PublicKey.RSA import *
>>> c = open(r".\transparency.pem").read()
>>> p = import_key(c)
>>> p
RsaKey(n=2342162228564134140563361689015041377107149279166261923701553268927120925467525521418777283514380180903
9951016782376679973376782695533167272817148034946155291022588458116896449130547957859630601417029406537713697722
2164841265084046694925746517387007853236278038029670978141921557139882067656772559964537465702212036054646836981
3975906820174580564322660230964817772084236973742530766267452453075757062697023253754982400599839360902186177313
4215542450556839250804799098903483152012713520167414613141526302727512388972623173809195225592109964416682348203
058784103484962051844890398766510080562420295832329553237528041393, e=65537)
```

要求是找到 `cryptohack.org` 下的一个，用这些参数进行证书认证的，某个子域名，flag就在这个子域名中

没有什么思路，搜了下wp，是爆破的，但也只有一张图

使用 `Maltego` 分析网站域名

1. 新建一个graph，从左侧Entity栏中拖入一个 `Domain` 实体，并修改为 `cryptohack.org`

The screenshot shows the Maltego Community Edition 4.2.19 interface. The main workspace displays a single entity: a Domain named 'cryptohack.org'. The interface includes a top menu bar with options like Investigate, View, Entities, Collections, Transforms, Machines, Collaboration, and Import | Export. On the left, there is an Entity Palette with categories like Domain, IPv4 Address, and Machines. Below the main workspace is an Output - Transform Output window showing a list of transform results, including 'Transform Mirror: External links found returned with 12 entities (from entity "101.41.123.237")' and 'Running transform To Files (Office) [using Search Engine] on 1 entities (from entity "cryptohack.org")'. On the right, there are panels for Overview, Detail View, and Property View. The Detail View shows the domain 'cryptohack.org' with its WHOIS info and graph info.

2. 右键实体，点击第一行的 All Transforms 右侧的双箭头



开始了自动查询域名下的所有信息

包括该域名的子域名，IP地址，DNS服务器，联系人，地址，电话号等等

等待结果

使用 **Ctrl+F** 搜索功能，查询题目的 **Transparency** 关键字或者直接查询 **flag** 字样

The screenshot shows the Maltego Community Edition 4.2.19 interface. The main window displays a network graph with several DNS entities. The search bar at the bottom of the graph contains the text "Find: flag". The search results are displayed in the "Output - Transform Output" window at the bottom, showing a list of entities found, including "the transparency flagshere.cryptohack.org". The "Detail View" window on the right shows the properties of the selected entity, including its DNS Name and Type.

访问，得到flag



[创作打卡挑战赛](#)

[赢取流量/现金/CSDN周边激励大奖](#)