CG-CTF Crypto WriteUp

原创

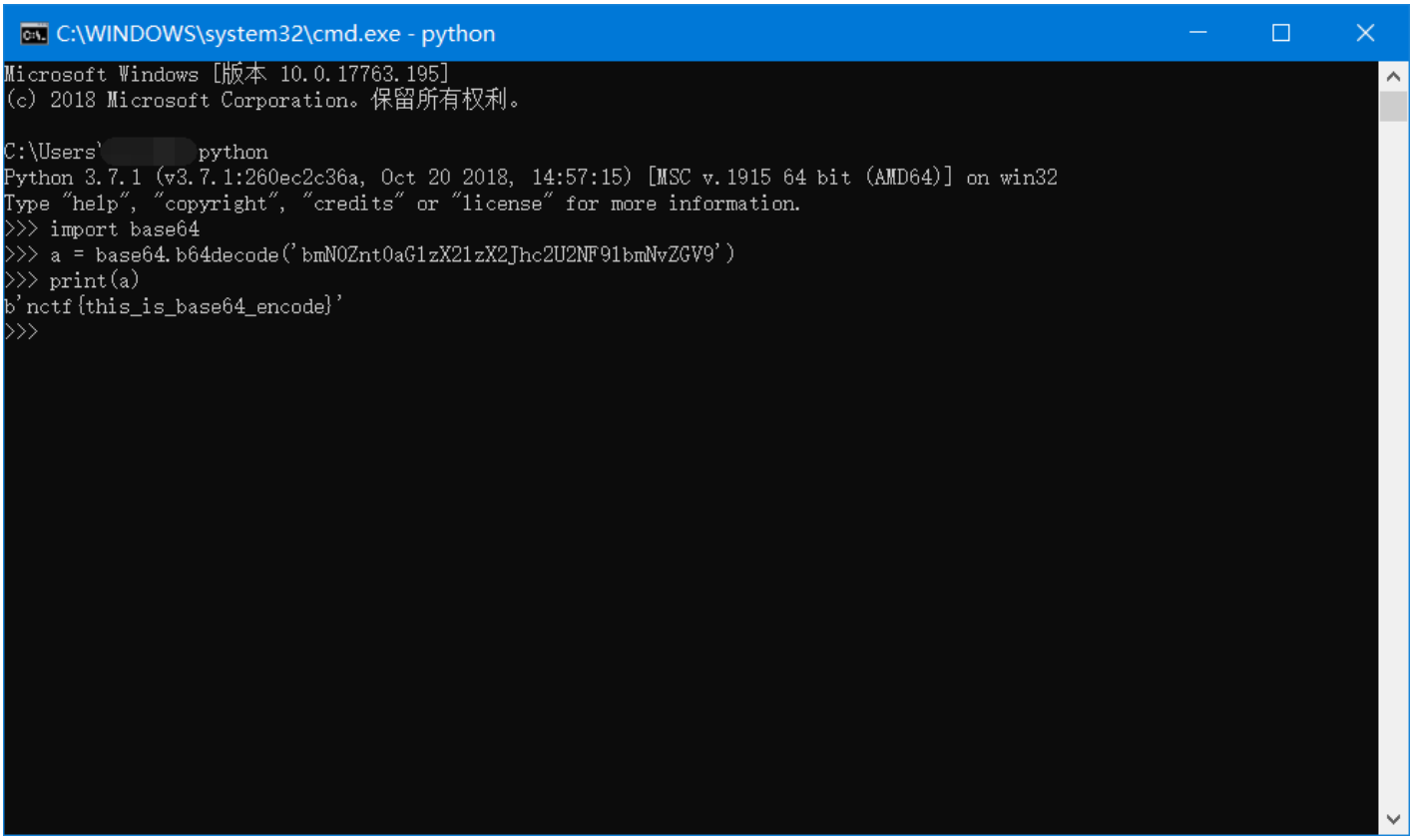
旗木家的卡卡西 于 2019-01-01 02:02:16 发布

第一次写WriteUp，好兴奋。

大一新生，萌新一枚，求关照...

第一题：

这个题，一看字母数字base64跑起，百度或者python一跑，出结果。



Flag：nctf{this_is_base64_encode}

10分到手

base64，base32，base16的区别如下

base64：包含大写字母（A-Z）,小写字母（a-z），数字（0-9）以及+和/

base32：大写字母（A-Z）和数字234567

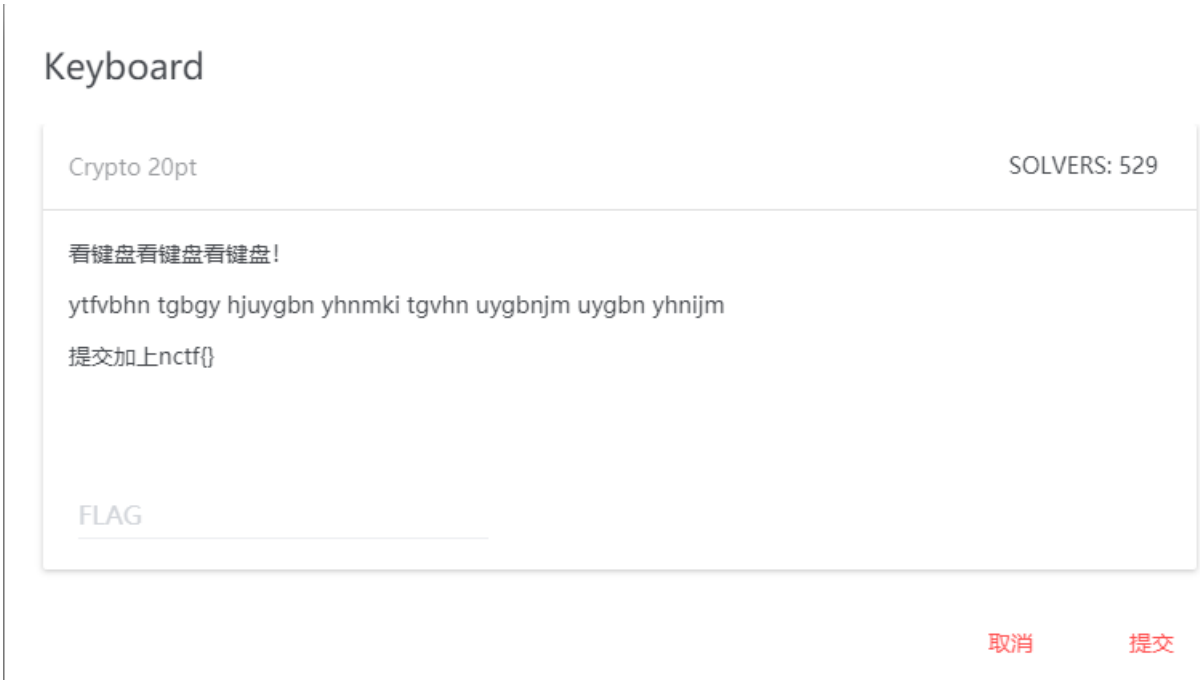
base16：数字(0-9)，字母（ABCDEF）

当ASCII用Base加密达不到所对应的位数的时候用=号补齐。

所以看到字母数字都有，就确定是base编码；有小写字母，base64，有只大写字母，base32，大写字母A-F和数字，base16。

嗯，记住了。

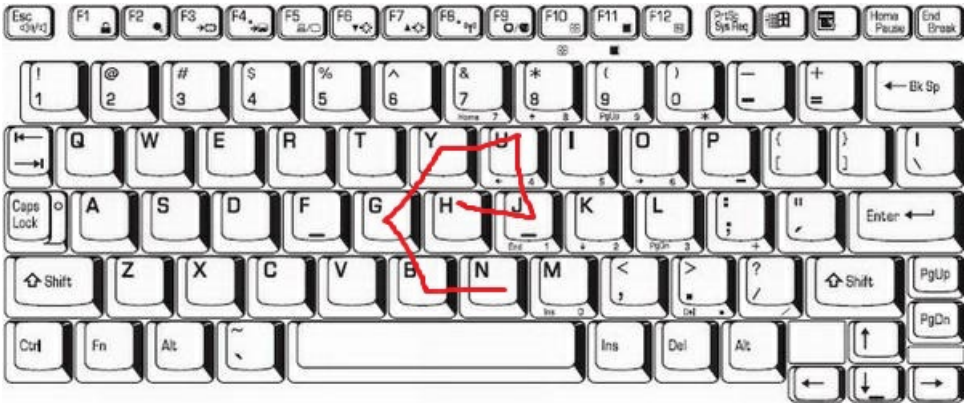
第二题：

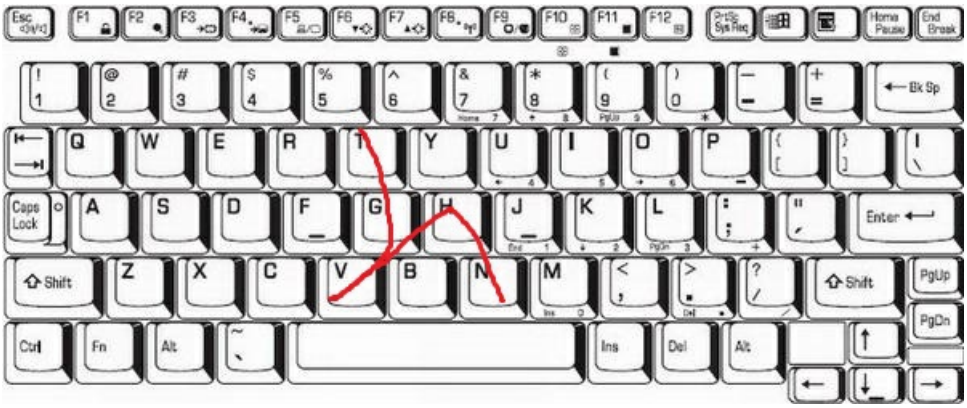


这题，看键盘。

脑洞很重要！！！！看手指的轨迹！！！

百度找个键盘布局画图画一下，然后：





Flag：nctf{areuhack}

20分到手。

脑洞很重要，嗯，记住了。

第三题：

异性相吸

Crypto 30pt SOLVERS: 217

同性真爱，异性相吸都是假的！ （题目要求，我是直的）

解密压缩文件里的内容

提取密码：assm

TIPS: 1.xor 2.hex2binary 3.len(bin(miwen))==len(bin(mingwen))

题目地址

FLAG

取消 提交

看提示：

1.xor 异或

2.hex2binary 十六进制转二进制

3.len(bin(miwen))==len(bin(mingwen)) 密文的二进制长度和明文的二进制长度相等

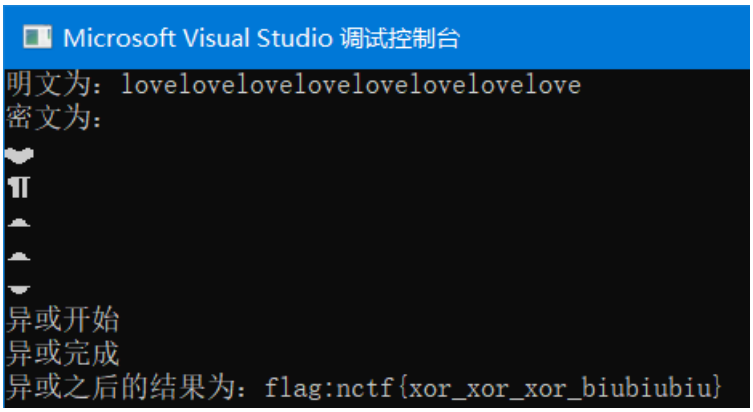
第一步，异或。

直接C++或者Python，我选择C++。

看一下文件大小， 32 字节 (32 字节)

32个字节，对应32个字符，char类型，嗯，应该没错。

```
#include <fstream>
using std::ios;
using std::ifstream;
int main() {
 ifstream fr;
 char mingwen[33] = {}, miwen[33] = {};
 fr.open("明文.txt", ios::binary);
 fr.read(mingwen, 32);
 fr.close();
 fr.open("密文.txt", ios::binary);
 fr.read(miwen, 32);
 fr.close();
 printf("明文为：%s\n", mingwen);
 printf("密文为：%s\n", miwen);
 printf("异或开始\n");
 int i;
 char cxor[33] = {};
 for (i = 0; i < 32; i++)
  cxor[i] = mingwen[i] ^ miwen[i];
 printf("异或完成\n");
 printf("异或之后的结果为：%s", cxor);
 return 0;
}
```



Flag：nctf{xor_xor_xor_biubiubiu}

30分到手，开心！

后面有一个假的第四题：

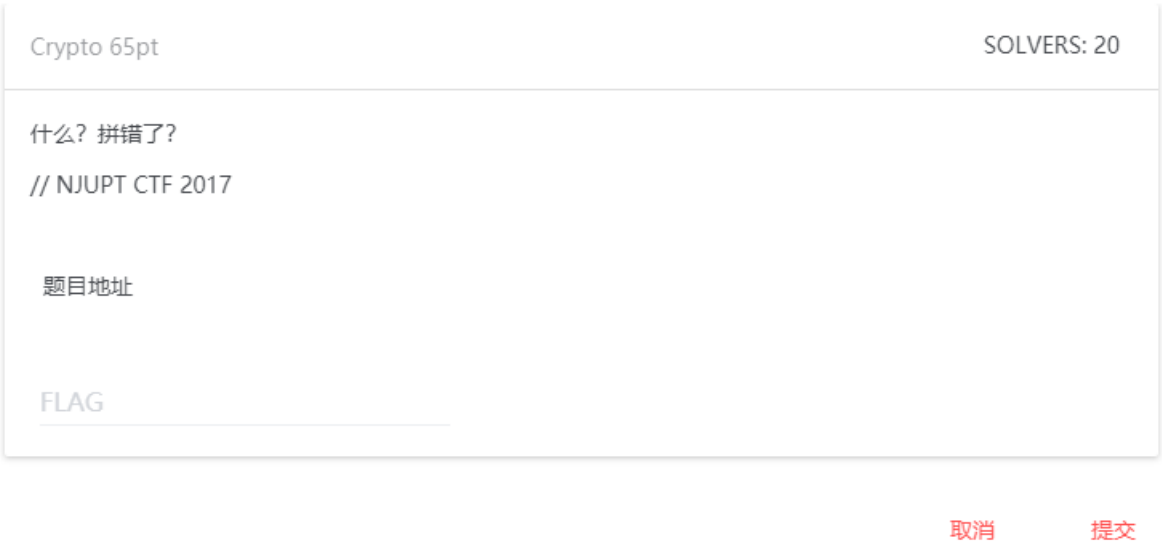
注意！！

Flag：nctf{zhaowomen}

一分到手，不说话了…

第四题：

Wiener Wiener Chicken Dinner



Wiener是啥？

百度搜半天找不到有关的，那就谷歌吧。

好像看到了什么不得了的东西…

下载下来看看…

看下Readme

```
rsa-wiener-attack
==================

A Python implementation of the Wiener attack on RSA public-key encryption scheme.

It uses some results about continued fractions approximations to infer the private key from public key in the cases the encryption exponent is too small or too large.
```

嗯，就知道是个RSA加密，看了还是不会用…

那就直接看主程序吧…

这主程序看着像python3的，看print函数就能看出来

```
def test_hack_RSA():
    print("Testing Wiener Attack")
    times = 5

    while(times>0):
        e,n,d = RSAvulnerableKeyGenerator.generateKeys(1024)
        print("(e,n) is (", e, ", ", n, ")")
        print("d = ", d)

        hacked_d = hack_RSA(e, n)

        if d == hacked_d:
            print("Hack WORKED!")
        else:
            print("Hack FAILED")

        print("d = ", d, ", hacked_d = ", hacked_d)
        print("----------------------------")
        times -= 1

if __name__ == "__main__":
    #test_is_perfect_square()
    #print("----------------------------")
    test_hack_RSA()
```

看见用法了。

已知e,n，出来d，大概…懂了？

看一下下载下来的那个py。

逆着走一遍算法，构造一个密钥，然后加密，大概就这样。

RSA中，e,n组成公钥，e,d组成私钥，然后

```
106304532128384446834453116899277852065119216210094853399153909744703144009006819183583893846080726086687237983479686291611878527106220928126766706964000050169814269338920927537684338286357965011997705976837502858632649005508739463152824198363146247170991375872859145947679911505097749397961354505673616286804 9L,
837165022918376318972691589160491375229372195625940137121740685432530132860541010174727630660292946071507342455170833392895060048564125597915757582027572284342507277083636059558106672685400173531425920294781499112027917636249795495843766035757540022269297984487337210580199821084528577514626311739919118537934 7L))
```

这两个长整数中有一个是d，有一个是n，于是…

改一下主程序的那个py

```
if __name__ == "__main__":
    #test_is_perfect_square()
    #print("----------------------------")
    #test_hack_RSA()
    e=
    106304532128384446834453116899277852065112
    843382863579650119977059768375028586326492
    n=
    837165022918376318972691589160491375229372
    066726854001735314259202947814991120279172
    print(hack_RSA(e,n))
```

试试…

于是挂了…

```
C:\Users\        ownloads\rsa-wiener-attack-master>python rsawienerhacker.py
None
```

那就交换一下n,e再来一次…

```
C:\Users\      Downloads\rsa-wiener-attack-master>python rsawienerhacker.py
Hacked!
5789976380172226106289129050355983590457194655725876115442254610482409467084 3
```

得到d

查一下RSA.construct在python中的用法

https://www.dlitz.net/software/pycrypto/api/2.6/

文档如上

- tup (tuple) - A tuple of long integers, with at least 2 and no more than 6 items. The items come in the following order:
    1. RSA modulus (n).
    2. Public exponent (e).
    3. Private exponent (d). Only required if the key is private.
    4. First factor of n (p). Optional.
    5. Second factor of n (q). Optional.
    6. CRT coefficient, (1/p) mod q (u). Optional.

挑重点！！！construct私钥要n,e,d

所以，改一下CTF下载下来的那个py，那个脚本看着像py2写的

```
#coding:utf-8
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_v1_5 as Cipher_pkcs1_v1_5
import base64
cipher_text = 'AGgt1h6dudnkeoCr7SFclkYYsYa65KZ8V29bbgbf+BDyjnyx5stCYjcyktat73aHs2EOaMgwGUwj3HwPTvT+T5LHIxM4
key=RSA.construct((106304532128384446834453116899277852065119216210094853399153909744703144009006819183583 8
cipher = Cipher_pkcs1_v1_5.new(key)
cipher_text = cipher.decrypt(base64.b64decode(cipher_text))
print cipher_text
```

改成这样大概，然后跑一下，崩了

```
Traceback (most recent call last):
  File "wiener_chicken.py", line 16, in <module>
    cipher_text = cipher.decrypt(base64.b64decode(cipher_text))
TypeError: decrypt() takes exactly 3 arguments (2 given)
```

decrypt()需要3个参数只提供了两个，于是百度…

```
# 伪随机数生成器
random_generator = Random.new().read

text = cipher.decrypt(base64.b64decode(encrypt_text), random_generator)
```

来源：https://www.jb51.net/article/86022.htm

后面那个参数加上

需要from Crypto import Random

加上后再跑一下…

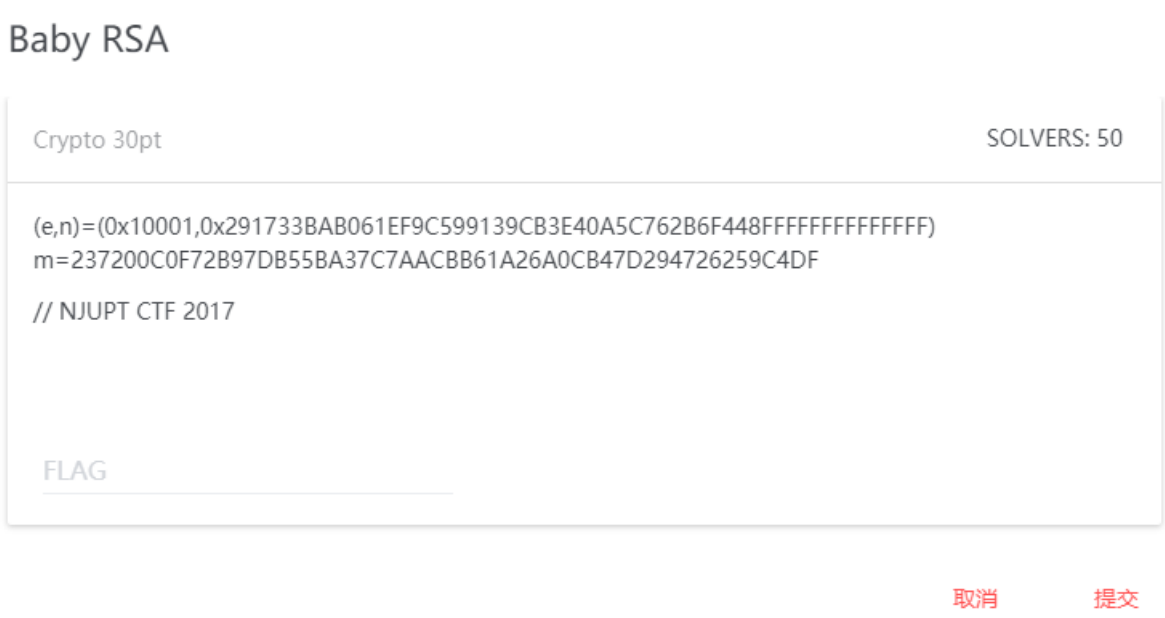
```
cipher_text = cipher.decrypt(base64.b64decode(cipher_text),Random.new().read)
```

```
(py2.7work) C:\Users\        )ownloads>python wiener_chicken.py
flag{nell_anima_ritrovo_la_speranza_che_nel_corpo_stanco_ormai}
```

Flag：flag{nell_anima_ritrovo_la_speranza_che_nel_corpo_stanco_ormai}

65分到手，兴奋。

第五题：



给了e,n,m(明文)，求密文

于是先拿上面的那个wiener跑一下，然后崩了

```
C:\Users\        )ownloads\rsa-wiener-attack-master>python RSAwienerHacker.py
None
```

然后开始RSA的日常，上factordb

http://factordb.com/

这是个分解n的网站，然后就知道p,q了，然后再按照RSA的算法写个脚本跑一下，大概就这样...

先跑一下n的10进制

```
C:\Users\        ,Downloads\rsa-wiener-attack-master>python
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15) [MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(0x291733BAB061EF9C599139CB3E40A5C762B6F448FFFFFFFFFFFFFF)
16903705997349646195704375376941855414691523387719679999999999999
```

复制粘贴到factordb

status (?)	digits	number
FF	65 (show)	1690370599...99<65> = 157817387176484486971605217 1<28> · 107109275471951139731750470662151462 69<38>

跑出来了

按照RSA的算法，p= 157817387176484486971605217 1，q= 107109275471951139731750470662151462 69

然后就要求d了，已知p,q,e，求d，找个现成的脚本跑一下就行，我找的这个：

https://blog.csdn.net/zyxyzz/article/details/78205321

感谢楼上师傅提供的脚本

跑一下出来d了

然后再拿上面的那个脚本跑一下就出来明文了

```
C:\Users\        Downloads>workon py2.7work
(py2.7work) C:\Users        Downloads>python wiener_chicken.py
719613954447190333491237122110934109677157592876628602144619 93
```

```
from Crypto import Random
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_v1_5 as Cipher_pkcs1_v1_5
text = 0x237200C0F72B97DB55BA37C7AACBB61A26A0CB47D294726259C4DF
key = RSA.construct((169037059973496461957043753769418554146915233877196799999999999999L, 65537L,71961395444
cipher = Cipher_pkcs1_v1_5.new(key)
text = cipher.encrypt(str(text))
print text
```

然后崩了...

```
C:\Users\1       \Downloads>workon py2.7work
(py2.7work) C:\Users\1       \Downloads>python wiener_chicken.py
Traceback (most recent call last):
  File "wiener_chicken.py", line 17, in <module>
    text = cipher.encrypt(str(text))
  File "C:\Users\       \Envs\py2.7work\lib\site-packages\Crypto\Cipher\PKCS1_v1_5.py", line 128, in encrypt
    raise ValueError("Plaintext is too long.")
ValueError: Plaintext is too long.
```

百度一下，发现了

先去google下，没发现什么有用的信息，于是去找源码，在 源码错误处理地方 看到了一些信息

```
        :Parameters:
         message : byte string
                The message to encrypt, also known as plaintext. It can be of
                variable length, but not longer than the RSA modulus (in bytes) minus 11.
```

原来加密的 plaintext 最大长度是 `证书key位数/8 - 11`，例如1024 bit的证书，被加密的串最长 `1024/8 - 11=117`，顺着这个思路google下，解决办法是 分块 加密，然后分块解密就行了，因为 证书key固定的情况下，加密出来的串长度是固定的。

来自：https://blog.csdn.net/orangleliu/article/details/72964948

那就分开加密试试

```
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(0x237200C0F72B97DB55BA37C7AACBB61A26A0CB47D294726259C4DF)
145813605451065893613472357177296342272158759434412899385313702 07
>>>
```

先得到这个数字

然后分块，每次1个数字不算很长吧…

于是还是一样的崩，我也不知道为啥，也不想看库的文档了…

算了不用这个加密器了…

那就自力更生，丰衣足食

RSA的加密的算法不难，python大数运算没有位数限制，于是搜一下python的运算符

%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000

看到了重点

于是写了个脚本：

```
N=169037059973496461957043753769418554146915233877196799999999999999
e=65537
d=71961395444719033349123712211093410967715759287662860214461993
m=0x237200C0F72B97DB55BA37C7AACBB61A26A0CB47D294726259C4DF
print ((m**d)%N)
```

跑了一个小时还没出，笔记本都热了…

好像有哪里不对劲…

算了还是百度吧…

https://www.freebuf.com/articles/others-articles/166049.html

这个上面找了个脚本跑了一下，一秒出结果…

感谢楼上师傅提供的脚本

```
import math
N=169037059973496461957043753769418554146915233877196799999999999999
e=65537
d=71961395444719033349123712211093410967715759287662860214461993
m=0x237200C0F72B97DB55BA37C7AACBB61A26A0CB47D294726259C4DF
print (hex(pow(m,d,N))[2:-1].decode('hex'))
```

Flag：flag{Acdxvf5vD_15_W7f}

30分到手。

不说话了，还是自己太菜…

学习中...