

# BUUCTF 每日打卡 2021-4-28

原创

Σ2333! 于 2021-04-28 22:53:09 发布 189 收藏

分类专栏: [crypto](#) 文章标签: [密码学](#) [加密解密](#)

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/weixin\\_52446095/article/details/116244826](https://blog.csdn.net/weixin_52446095/article/details/116244826)

版权



[crypto](#) 专栏收录该内容

79 篇文章 1 订阅

订阅专栏

## 引言

数分考完了

明天蓝帽杯

## 可怜的RSA

附件给了公钥:

```
-----BEGIN PUBLIC KEY-----
MIIBJDANBgkqhkiG9w0BAQEFAAOCAQIIBDAKCAQMIsYv184kJfRcjeGa7Uc/4
3pIkU3SevEA7CZXJfA44bUbBYcr-f93xphg2uR5HCFM+Eh6qqnybpIK13g0kGA4rv
tcMIJ9/PP8nrdpVE+U4Hzf4IcgOaOmJiEWZ4smH7LWudMl0ekqFTs2dWkbqz1C59
NeMPfu9avxxQ15fQzIjhvcz9GhLqb373XDcn298ueA80KK6Pek+3qJ8YSjZQMrFT
+EJehFdQ6yt6vALcFc4CB1B6qVCG07hICngCjdYpeZRNBGM/r6ED5Nsozof1oMbt
Si8mZEJ/V1x3gathkUVtlxx/+j1ScjdM7AFV5fkRidt0LkwosDoPoRz/sDFz0qTM
5q5TAgMBAAE=
-----END PUBLIC KEY-----
```

提取公钥代码如下:

```
from Crypto.PublicKey import RSA

with open("public.key", "r") as f:
    key = RSA.import_key(f.read())
    print(key.n)
    print(key.e)
```

得到

```
e = 65537
n = 798321817573328185527646107613495929846147444322791353283989998016278802836109003612812499731758050699162101
7956050649707513252490208688112037221362664187946849193686097668693363086967382697261993832195159914674480765330
1076026577949579618331502776303983485566046485431039541708467141408260220098592761245010678592347501894176269580
5104597296336734680684671441997445637318263621026088110334008878137547802826280994434901700160878386069980174904
5660131580244856777241162382628174724566095424541378151979429533619755568854353799219714225805322045375766653784
0276416475602759374950715283890232230741542737319569819793988431443
```

然后爆破就可以求出 p,q

Search Sequences Report results Factor tables Status Downloads Login

798321817573328185527646107613495929846147444322791353283989998016278802836109003612812499 Factorize!

| Result:    |                            |   |
|------------|----------------------------|---|
| status (2) | digits                     | number  |
| FF         | 623 <a href="#">(show)</a> | <a href="#">7983218175...43</a> <623> = <a href="#">3133337</a> · <a href="#">2547832606...39</a> <617> |

More information [↗](#)

ECM [↗](#)

factordb.com - 11 queries to generate this page (0.01 seconds) [\(limits\)](#) [\(imprint\)](#) [\(Privacy Policy\)](#)

[https://blog.csdn.net/weixin\\_52448095](https://blog.csdn.net/weixin_52448095)

```
p = 3133337
q = 254783260649374192922001721363994977190818429145282283164559062116931183219713999360047291348411629741442462
7148643969578603658811742461188195595099621964680737882227828563826158209910833943894957303410121514115615640874
2843820048066830863814362379885720395082318462850002901605689761876319151147352730090957556940842144299887394678
7436077669378280944783364011594490358783068537162165483742734623865083073677131120730040113834189678949305540675
8245324898102201192288337444273684804592067634136187123178716344146753307689008172188217936916878728772476964266
5399992556052144845878600126283968890273067575342061776244939
```

附件密文如下:

```
Gvd1d3viIXFfcHapEYuo5fAvIiUS83adrtMW/MgPwxVBSl46joFCQ1plcnlDGfL19K/3PvChV6n5QGohzfvVyz2Z5GdTLaknxvHDUGf5HCukokyPw
K/1EYU7NzrhGE7J5jPdi0Aj7xi/Odxy0hGMgpaBLd/nL3N806i9pc4Gg308so0lciBG/6/xdfn3SzsStMYIN8nfZZMSq3xDDvz4YB7TcTBh4ik4w
YhuC77gmT+HW0v5gLTNq3EkZs5N3EAopy11zHNYU80yv1jtFGclunPyXYttU5qU33jcp0Wuznac+t+AZHeSQy5vk8DyWorSGMiS+J4KNqSV1Ds12
EqXEqqJ0uA==
```

容易发现经过了 base64 加密

尝试解密:

```
import base64
from Crypto.Util.number import *
# p, q, n, e
c = bytes_to_long(base64.b64decode(b'Gvd1d3viIXFfcHapEYuo5fAvIiUS83adrtMW/MgPwxVBSl46joFCQ1plcnlDGfL19K/3PvChV6n5QGohzfvVyz2Z5GdTLaknxvHDUGf5HCukokyPwK/1EYU7NzrhGE7J5jPdi0Aj7xi/Odxy0hGMgpaBLd/nL3N806i9pc4Gg308so0lciBG/6/xdfn3SzsStMYIN8nfZZMSq3xDDvz4YB7TcTBh4ik4wYhuC77gmT+HW0v5gLTNq3EkZs5N3EAopy11zHNYU80yv1jtFGclunPyXYttU5qU33jcp0Wuznac+t+AZHeSQy5vk8DyWorSGMiS+J4KNqSV1Ds12EqXEqqJ0uA=='))

phi = (p-1)*(q-1)
d = inverse(e, phi)
m = pow(c, d, n)

print(long_to_bytes(m))
```

发现是一堆乱码

```
798321817573328185527646107613495929846147444322791353283989998016278802836109003612812499731758050699162101795605064970751325249020868811203722136266
65537
b'\xb2u~\xc4\xd0\xbeV\x1c\x80\x0b\x16F\xee[\xe9\xf4\xbfJ\xd2\x17p\xc2c\xfcbv\x14\xe4\x8c`A\xd4\`Iy=\x11\xeah:\xb9n\x0b\x01\xd2\x03V\xaa\xb2C{\xc3yP\x8
```

嗯?

最后求助 wp

代码如下:

```

from Crypto.Cipher import PKCS1_OAEP
import base64
from Crypto.Util.number import *
# p,q,n,e
c = base64.b64decode(b'GVd1d3viIXFfcHapEYuo5fAvIiUS83adrMtMW/MgPwxVBS146joFCQ1p1cn1DGfL19K/3PvChV6n5QGohzfVyz2Z5G
dTLaknxvHDUGf5HCukokyPwK/1EYU7NzrhGE7J5jPd10Aj7xi/Odxy0hgMgpaBLd/nL3N806i9pc4Gg308so01ciBG/6/xdFN3SzSStMYIN8nfZZ
MSq3xDDvz4YB7TcTBh4ik4wYhuC77gmT+HW0v5gLTNQ3EkZs5N3EAopy11zHNYU80yv1jtFGcluNPYXYttU5qU33jcp0Wuznac+t+AZHeSQy5vk8
DyWorSGMiS+J4KNqSV1Ds12EqXEqqJ0uA==')

phi = (p-1)*(q-1)
d = inverse(e, phi)
key_info = RSA.construct((n, e, d, p, q))
key = RSA.importKey(key_info.exportKey())
key = PKCS1_OAEP.new(key)
print(key.decrypt(c))

```

结果为: afctf{R54\_|5\_\$0\_B0rin9}

啊这

多西得?

我们知道,

Crypto.Util 包中 long\_to\_bytes 方法是将字符串转化成二进制然后转化成十进制

示例如下:

```

>>> from Crypto.Util.number import *
>>> print(bytes_to_long(b'AB'))
16706
>>> print(bin(ord('A')))
0b1000001
>>> print(bin(ord('B')))
0b1000010
>>> print(hex(ord('A')))
0x41
>>> print(hex(ord('B')))
0x42
>>> print(int('4142',16))
16706
>>> print(int('100000101000010',2))
16706

```

然后我又找了 Crypto.Cipher 包中的 decrypt 方法[源码](#)

```

def decrypt(self, ciphertext):
    """Decrypt a message with PKCS#1 OAEP.
    :param ciphertext: The encrypted message.
    :type ciphertext: bytes/bytearray/memoryview
    :returns: The original message (plaintext).
    :rtype: bytes
    :raises ValueError:
        if the ciphertext has the wrong length, or if decryption
        fails the integrity check (in which case, the decryption
        key is probably wrong).
    :raises TypeError:
        if the RSA key has no private half (i.e. you are trying
        to decrypt using a public key).
    """

    # See 7.1.2 in RFC3447
    modBits = Crypto.Util.number.size(self._key.n)
    k = ceil_div(modBits,8) # Convert from bits to bytes
    hLen = self._hashObj.digest_size

    # Step 1b and 1c
    if len(ciphertext) != k or k<hLen+2:
        raise ValueError("Ciphertext with incorrect length.")
    # Step 2a (O2SIP)
    ct_int = bytes_to_long(ciphertext)
    # Step 2b (RSADP)
    m_int = self._key._decrypt(ct_int)
    # Complete step 2c (I2OSP)
    em = long_to_bytes(m_int, k)
    # Step 3a
    lHash = self._hashObj.new(self._label).digest()
    # Step 3b
    y = em[0]
    # y must be 0, but we MUST NOT check it here in order not to
    # allow attacks like Manger's (http://dl.acm.org/citation.cfm?id=704143)
    maskedSeed = em[1:hLen+1]
    maskedDB = em[hLen+1:]
    # Step 3c
    seedMask = self._mgf(maskedDB, hLen)
    # Step 3d
    seed = strxor(maskedSeed, seedMask)
    # Step 3e
    dbMask = self._mgf(seed, k-hLen-1)
    # Step 3f
    db = strxor(maskedDB, dbMask)
    # Step 3g
    one_pos = hLen + db[hLen:].find(b'\x01')
    lHash1 = db[:hLen]
    invalid = bord(y) | int(one_pos < hLen)
    hash_compare = strxor(lHash1, lHash)
    for x in hash_compare:
        invalid |= bord(x)
    for x in db[hLen:one_pos]:
        invalid |= bord(x)
    if invalid != 0:
        raise ValueError("Incorrect decryption.")
    # Step 4
    return db[one_pos + 1:]

```

其中也有 `long_to_bytes` 和 `bytes_to_long` 方法，原理类似（大概）  
因为举例比较麻烦，所以就没有尝试  
浏览了一下源码（因为不怎么看得懂），发现里面有 `hash`, `xor` 之类的字眼  
而且看注释内容（也可见于[官方文档](#)）  
发现它对密文长度也做了要求，如果密文长度错误会报错 `ValueError`  
就很迷。。。

那么如果碰到了该怎么办？  
那就先尝试第一种代码，再尝试第二种代码咯[扶额]

## Single

加密代码如下：

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    freopen("Plain.txt", "r", stdin);
    freopen("Cipher.txt", "w", stdout);
    map<char, char> f;
    int arr[26];
    for(int i=0; i<26; ++i){
        arr[i]=i;
    }
    random_shuffle(arr, arr+26);
    for(int i=0; i<26; ++i){
        f['a'+i]='a'+arr[i];
        f['A'+i]='A'+arr[i];
    }
    char ch;
    while((ch=getchar())!=EOF){
        if(f.count(ch)){
            putchar(f[ch]);
        }else{
            putchar(ch);
        }
    }
    return 0;
}
```

这学期刚学 C 语言  
大致看了一下  
大概就是给明文随机移位了一下  
密文如下：

Jmqrida rva Lfmz (JRL) eu m uqajemf seny xl enlxdomrexn uajiderc jxoqarerexn. Rvada mda rvdaa jxooxn rcqau xl JRLu: Paxqmdyc, Mrrmjs-Yalanja mny oekay.

Paxqmdyc-urcfa JRLu vmu m jxiqfa xl giaurexnu (rmusu) en dmnza xl jmrazxdeau. Lxd akmoqfa, Wab, Lxdanuej, Jdcqrx, Benmdc xd uxoarvenz afua. Ramo jmn zmen uxoa qxenru lxd atadc uxftay rmus. Oxda qxenru lxd oxda jxoqfejmray rmusu iuimffc. Rva nakr rmus en jvmen jmn ba xqanay xnfc mlrad uxoa ramo uxfta qdatexiu rmus. Rvan rva zmoa reoa eu xtad uio xl qxenru uvxwu cxi m JRL wenad. Lmoxiu akmoqfa xl uijv JRL eu Yaljxn JRL gimfu.

Waff, mrrmjs-yalanja eu mnxrvad enradaurenz seny xl jxoqarerexn. Vada atadc ramo vmu xwn narwxds(xd xnfc xna vxur) werv tifnmdmbfa uadtejau. Cxid ramo vmu reoa lxd qmrjvenz cxid uadtejau mny yatafxqenz akqfxeru iuimffc. Ux, rvan xdzmnehadu jxnnajru qmdrejeqmru xl jxoqarerexn mny rva wmdzmoa urmdru! Cxi uvxify qdxrajr xwn uadtejau lxd yalanja qxenru mny vmjs xqqxnanru lxd mrrmjs qxenru. Veurxdejmffc rveu eu m ledur rcqa xl JRLu, atadcbxyc snxwu mbxir YAL JXN JRL - uxoarvenz fesa m Wxdy Jiq xl mff xrvad jxoqarerexn.

Oekay jxoqarerexn omc tmdc qxuebfalxdomru. Er omc ba uxoarvenz fesa wmdzmoa werv uqajemf reoa lxd rmus-bmuay afaoranru (a.z. IJUB eJRL).

JRL zmoau xlransxijv xn omnc xrvad muqajru xl enlxdomrexn uajiderc: jdcqrxzdmqvc, uraxz, benmdc mnmfcueu, datada anzanaadenz, oxbefa uajiderc mny xrvadu. Zxxy ramou zanadmffc vmta urdxnz useffu mny akqadeanja en mff rvau euuiou.

Iuimffc, lfmz eu uxoa urdenz xl dmnxyo ymm xd rakr en uxoa lxdomr. Akmoqfa mljrl{Xv\_I\_lxiny\_er\_neja\_rDc}

## 尝试爆破

**Puzzle:**

Oekay jxoqarerexn omc tmdc qxuebfalxdomru. Er omc ba uxoarvenz fesa wmdzmoa werv uqajemf reoa lxd rmus-bmuay afaoranru (a.z. IJUB eJRL).

JRL zmoau xlransxijv xn omnc xrvad muqajru xl enlxdomrexn uajiderc: jdcqrxzdmqvc, uraxz, benmdc mnmfcueu, datada anzanaadenz, oxbefa uajiderc mny xrvadu. Zxxy ramou zanadmffc vmta urdxnz useffu mny akqadeanja en mff rvau euuiou.

Iuimffc, lfmz eu uxoa urdenz xl dmnxyo ymm xd rakr en uxoa lxdomr. Akmoqfa mljrl{Xv\_I\_lxiny\_er\_neja\_rDc}

**Clues:** For example G=R QVW=THE

**Solve**

Ad closed by Google

0 -1.702 Capture the Flag (CTF) is a special kind of information security competitions. There are three common types of CTFs: Jeopardy, Attack-Defence and mixed. Jeopardy-style CTFs has a couple of questions (tasks) in range of categories. For example, Web, Forensic, Crypto, Binary or something else. Team can gain some points for every solved task. More points for more complicated tasks usually. The next task in chain can be opened only after some team solve previous task. Then the game time is over sum of points shows you a CTF winner. Famous example of such CTF is Defcon CTF quals. Well, attack-defence is another interesting kind of competitions. Here every team has own network(or only one host) with vulnerable services. Your team has time for patching your services and developing exploits usually. So, then organizers connects participants of competition and the wargame starts! You should protect own services for defence points and hack opponents for attack points. Historically this is a first type of CTFs, everybody knows about DEF CON CTF - something like a World Cup of all other competitions. Mixed competitions may vary possible formats. It may be something like wargame with special time for task-based elements (e.g. UE3B iCTF). CTF games often touch on many other aspects of information security: cryptography, stego, binary analysis, reverse engineering, mobile security and others. Good teams generally have strong skills and experience in all these issues. Usually, flag is some string of random data or text in some format. Example [hctf{0h\\_U\\_found\\_it\\_nice\\_tRy}](#)

1 -4.232 Kubrche rye Tlug (KKT) is u sbekiul find at intahmuran sekchiro kamberirians. Ryehe uhe ryhee kamman robes at KKTs: Peabuhdo, Urrukf-Detenke und mixed. Peabuhdo-srole KKTs yus u kacble at vcesrians (rusfs) in huge at kuregahies. Tah exumble, Wez, Tahensik, Khobra, Zimho ah samerying else. Reum kun guin same bainrs tah ejehe saljed rusf. Maho bainrs tah maho kambikured rusfs csullio. Rye nexr rusf in kyulin kun ze abened anlo utreh same reum salje bhejiacs rusf. Ryeen rye gume rime is ajeh scm at bainrs sywe oac u KKT winch. Tumacs exumble at socky KKT is Detkan KKT vculs. Well, urrukf-detenke is unaryeh inrewhering find at kamberirians. Yeha ejehe reum yus awn nerwahf(ah anlo ane yasr) wiry jclmahuzle sehjikcs. Oach reum yus rime tah burkying oach sehjikcs und dejelabing exblairs csullio. Sa, ryen ahguniqehs kannekrs buhrikibunrs at kamberirian und rye wuhgume sruhrs! Oac syacld bharekr awn sehjikcs tah detenke bainrs und yukf abbanenrs tah urrukf bainrs. Yisrahikullo [https://blog.csdn.net/weixin\\_52440088](https://blog.csdn.net/weixin_52440088)

啊这

完事了

## 结语

这几天的博客属实水

主要是数分考试的原因

(但是差不多凉了, 求求给我及格吧)

希望继续坚持