2021年10月广东强网杯，CRYPTO的RSA AND BASE?

原创

沐一 · 林 于 2021-10-14 08:41:58 发布 114 收藏 1

分类专栏： CTF 密码学 文章标签： ctf

本文链接：https://blog.csdn.net/xiao__1bai/article/details/120756433

CTF 同时被 2 个专栏收录

167 篇文章 6 订阅

订阅专栏

密码学

51 篇文章 1 订阅

订阅专栏

2021年10月广东强网杯，CRYPTO的RSA AND BASE?

下载附件，是一个 `txt` 文件，打开，发现 `RSA` 密文和类似 `base32` 的变码表，也符合题目暗示：

```
RSA:
n=566612435194265632999200581340928623707373979499472103948430218564774209596151325536108
e=362697880447032674261773409928261721401744043905777362814788913816122942076668915290199
c=137954301101369152742229874240507191901061563449586247819350394387527789763579249250710

BASE:
"GHI45FQRSCX****UVWJK67DELMNOPAB3"
```

.

.

RSA题目照例先用CTF-RSA-TOOL工具跑一下，发现跑得出来：

```
└─$ python2 solve.py --verbose  -i /home/wdnmd/桌面/1.txt
DEBUG: factor N: try past ctf primes
DEBUG: factor N: try Gimmicky Primes method
DEBUG: factor N: try Wiener's attack
DEBUG: d = 0x1678d3c2f5a164fffffffffffffffffffffffffffffffffffffffffffffffffdL
INFO: flag{TCMDIEOH2MJFBLKHT2J7BLYZ2WUE5NYR2HNG====}
```

.

.

这应该是一层的flag，而且看起来像base32的四个等号，联想题目和TXT文件中后面的BASE码，可以猜出是BASE32变码的加密，而且还有4位未知：

RSA:
n=566612435194265632999200581340928623707373979499472103948430218564774209596151325536108 3
e=362697880447032674261773409928261721401744043905777362814788913816122942076668915290199 3
c=137954301101369152742229874240507191901061563449586247819350394387527789763579249250710 6

BASE:
'GHI45FQRSCX****UVWJK67DELMNOPAB3"

BASE32变码表

.

.

.

（这里积累第一个经验）

首先通过和密文以及传统base32的码表对比可以发现缺了2 T Y Z四个字母，这四个字母共有24种排列组合，可以用下面代码求出排列组合：

```
list1=['2','T','Z','Y']
len1=len(list1)
list2=[]
w=""
for i in list1:
 for j in list1:
  if j!=i:
   for k in list1:
    if k!=j and k!=i:
     for l in list1:
      if l !=k and l !=j and l !=i:
       w=i+j+k+l
       list2.append(w)
print(list2)
print(len(list2))
```

.

结果：

```
└─$ python 4.py                                                                                          1 ×
['2TZY', '2TYZ', '2ZTY', '2ZYT', '2YTZ', '2YZT', 'T2ZY', 'T2YZ', 'TZ2Y', 'TZY2', 'TY2Z', 'T
YZ2', 'Z2TY', 'Z2YT', 'ZT2Y', 'ZTY2', 'ZY2T', 'ZYT2', 'Y2TZ', 'Y2ZT', 'YT2Z', 'YTZ2', 'YZ2T
', 'YZT2']
24
```

.

.

（这里积累第二个经验）

然后就是获取`BASE32`的编码实现来替换码表了，由于我在网上找不到`base32`的`python`编码实现，而且我也不会`GO`语言，没法直接替换封装函数码表，所以我用了我`广州羊城杯`的`BabySmc`技巧，通过下标对应来转为传统的`base32`密文。

下标对应法，这就要了解`base32`加密解密的本质了。

`base32`加密是`5*8`变`8*5`，获取的`5`个`8`位数对应着`0~32`内的范围，而`base32`的基本字符表**ABCDEFGHIJKLMNOPQRSTUVWXYZ234567**不过是`0~32`范围内对应的映射下标而已。

解密的时候也是用每个加密字符在`0 ~32`范围的数来拆分解密，关键就是这个解密时的`5`位数是怎么找的呢？是通过`base32.index('加密字符')`来找对应的`0 ~ 32`的下标。

所以加密字符表单的作用只是用来根据下标来映射`5`位数的`0~32`的范围而已，本质是`0 ~64`的下标。

那么我们就可以通过一一对应的方法来把题目中新的加密表单的下标来对应原生的`base32`加密下标了，因为`base32`在线解密工具只能通过`base32.index('加密字符')`来找`0 ~ 32`的下标。

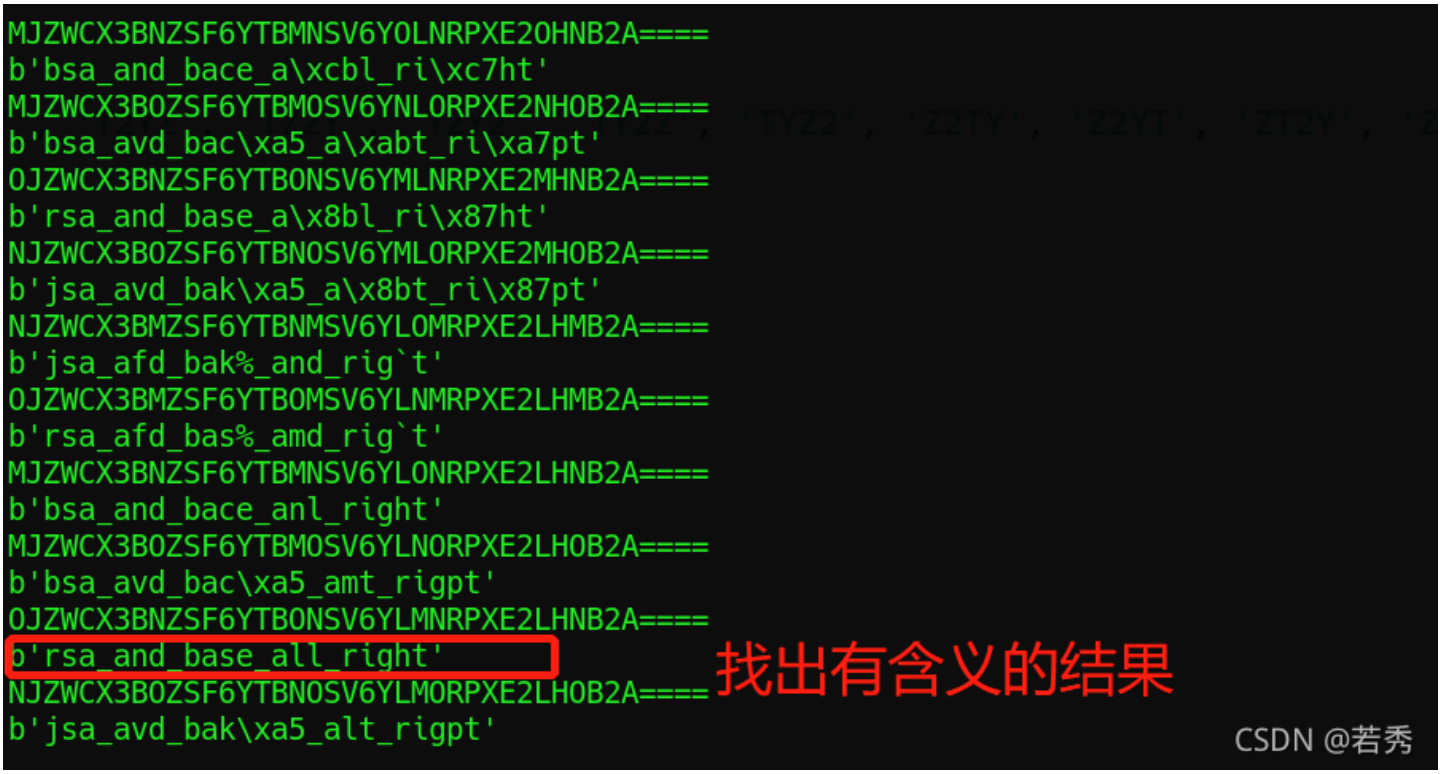
脚本如下，注意抽出`'='`来，并且在最后要把`'='`加上去，因为`base32`解密必须是`8`的倍数。

```python
import base64
#key1="GHI45FQRSCX****UVWJK67DELMNOPAB3" #少了2TYZ
key1="GHI45FQRSCX"  #变形表单1
key2="UVWJK67DELMNOPAB3" #变形表单2
base32="ABCDEFGHIJKLMNOPQRSTUVWXYZ234567" #传统表单
key3="TCMDIEOH2MJFBLKHT2J7BLYZ2WUE5NYR2HNG" #变形密文

list1=['2TZY', '2TYZ', '2ZTY', '2ZYT', '2YTZ', '2YZT', 'T2ZY', 'T2YZ', 'TZ2Y', 'TZY2', 'TY2Z', 'TYZ2', 'Z2TY', '
Z2YT', 'ZT2Y', 'ZTY2', 'ZY2T', 'ZYT2', 'Y2TZ', 'Y2ZT', 'YT2Z', 'YTZ2', 'YZ2T', 'YZT2']
secret=""

for i in list1:
 key4=key1+i+key2  #最终变形表单
 #print(key4)
 for m in key3:
  g=key4.index(m)
  secret+=base32[g]
 secret+='===='
 print(secret)
 print(base64.b32decode(secret))
 secret=""
```

结果：



.

.

.

总结：

> 1：
> （这里积累第一个经验）
> 首先通过和密文以及传统 `base32` 的码表对比可以发现缺了 `2 T Y Z` 四个字母，这四个字母共有 `24种` 排列组合，可以用下面代码求出排列组合

```
list1=['2','T','Z','Y']
len1=len(list1)
list2=[]
w=""
for i in list1:
 for j in list1:
  if j!=i:
   for k in list1:
    if k!=j and k!=i:
     for l in list1:
      if l !=k and l !=j and l !=i:
       w=i+j+k+l
       list2.append(w)
print(list2)
print(len(list2))
```

2：
（这里积累第二个经验）
然后就是获取 BASE32 的编码实现来替换码表了，由于我在网上找不到 base32 的 python 编码实现，而且我也不会 GO 语言，没法直接替换封装函数码表，所以我用了我 广州羊城杯 的 BabySmc 技巧，通过下标对应来转为传统的 base32 密文。

.

下标对应法，这就要了解 base32 加密解密的本质了。
base32 加密是 5*8 变 8*5 ，获取的 5 个 8 位数对应着 0~32 内的范围，而 base32 的基本字符表ABCDEFGHIJKLMNOPQRSTUVWXYZ234567不过是 0~32 范围内对应的映射下标而已。

.

解密的时候也是用每个加密字符在 0 ~32 范围的数来拆分解密，关键就是这个解密时的 5 位数是怎么找的呢？是通过 base32.index('加密字符') 来找对应的 0 ~ 32 的下标。

.

所以加密字符表单的作用只是用来根据下标来映射 5 位数的 0~32 的范围而已，本质是 0 ~64 的下标。

.

那么我们就可以通过一一对应的方法来把题目中新的加密表单的下标来对应原生的 base32 加密下标了，因为 base32 在线解密工具只能通过 base32.index('加密字符') 来找 0 ~ 32 的下标。

.

脚本如下，注意抽出 '=' 来，并且在最后要把 '=' 加上去，因为 base32 解密必须是 8 的倍数。

解毕！敬礼！