Flag:

```
ALEXCTF{HERE_GOES_THE_KEY}
```

CR4: Poor RSA (200)

Description:

```
This time Fady decided to go for modern cryptography implementations, He is fascinated with choosing his own

poor_rsa.tar.gz
```

This was a challenge where a little crypto education would've helped. Started this one off by looking up previous write-ups for RSA based challenges on a CTF. This one turned out to be great! - https://0x90r00t.com/2015/09/20/ekoparty-pre-ctf-2015-cry100-rsa-2070-write-up/

Extracting the tar.gz gave us two files, a encrypted flag and a public key:

```
-rw-r--r--@  1 user  staff    69B Dec 11 01:08 flag.b64
-rw-r--r--@  1 user  staff   162B Dec 11 00:59 key.pub
```

Walking through that writeup made this process very simple, starting off by identifying how many bits are used on the public key:

```
Modulus (399 bit):
    52:a9:9e:24:9e:e7:cf:3c:0c:bf:96:3a:00:96:61:
    77:2b:c9:cd:f6:e1:e3:fb:fc:6e:44:a0:7a:5e:0f:
    89:44:57:a9:f8:1c:3a:e1:32:ac:56:83:d3:5b:28:
    ba:5c:32:42:43
Exponent: 65537 (0x10001)
```

Looks like we also got an odd amount of bits (no pun intended).
Now we need to format the hex values to get the integer product:

```
openssl rsa -noout -text -inform PEM -in key.pub -pubin | grep -Evi 'mod|exp' | tr -d ':\n '
```

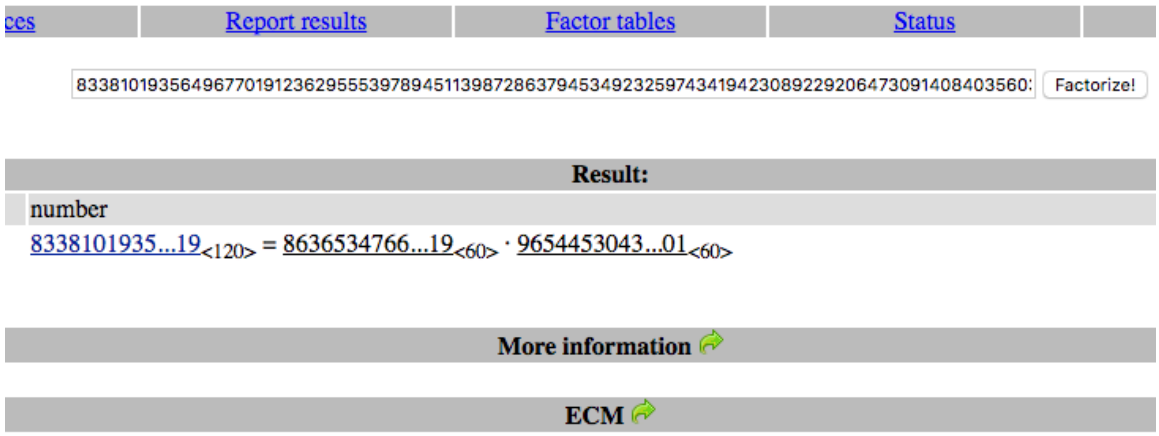
Then to get the int value, pass it into python:

```
$ openssl rsa -noout -text -inform PEM -in key.pub -pubin | grep -Evi 'mod|exp' | tr -d ':\n ' | xargs pytho

8338101935649677019123629555397894511398728637945349232597434194230892292064730914084035603111915457642213 10
```

Now we can query factordb for this value: http://www.factordb.com/index.php?query=8338101935649677019123629555397894511398728637945349232597434194230892292064730914084035603111915457642213106663388780 19

We end up seeing there is a match!



This turns out to be:
8636534766163765753088663449845764666449425722469000131569 19
* 9654453043269981947982822288424847324384571705959995234269 01

Now that we have p & q, we can generate the private key using RSATool - https://github.com/ius/rsatool

```
$ python ./rsatool/rsatool.py -p 8636534766163765753088663449845764666449425722469000131569 19 -q 96544530432
```

Finally we just need to decrypt the flag using openssl:

```
$ openssl rsautl -decrypt -in flag.raw -inkey priv.key
```

This drops the Flag:

```
ALEXCTF{SMALL_PRIMES_ARE_BAD}
```