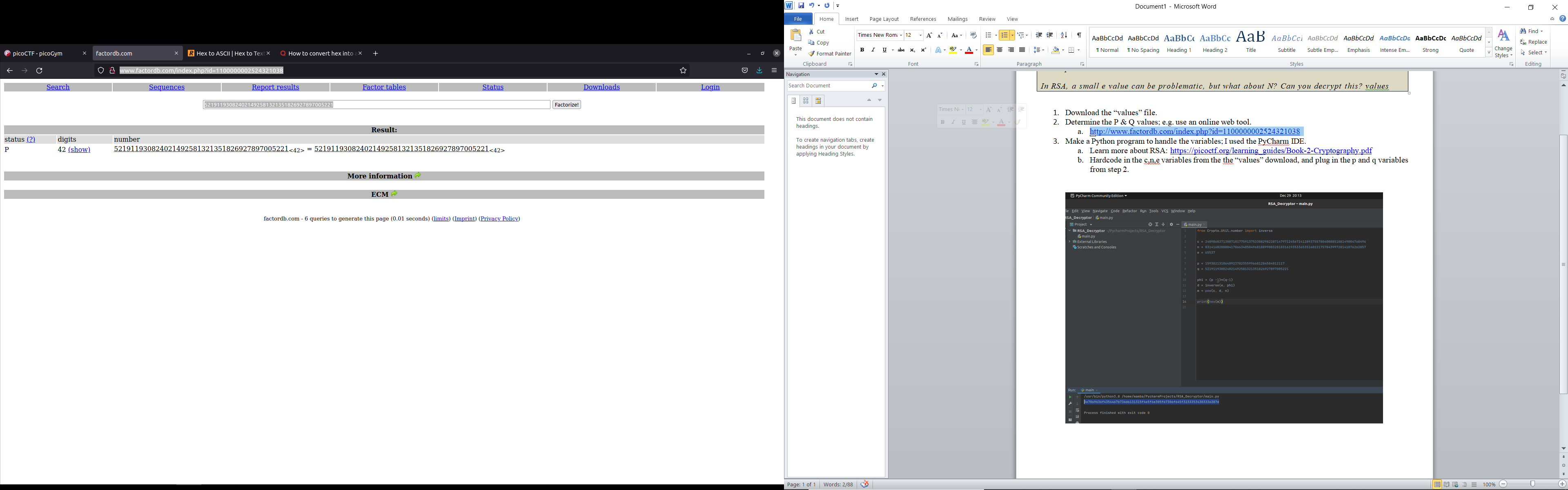
|  |
| --- |
| Mind Your Ps and Qs *20pts cryptography* |
| Description:  In RSA, a small e value can be problematic, but what about N? Can you decrypt this? values |

1. This challenge will deal with RSA encryption.
   1. Learn more about RSA on page 6: <https://picoctf.org/learning_guides/Book-2-Cryptography.pdf>
2. Download the “values” file.
3. Determine the P & Q values; e.g. use an online web tool.
   1. http://www.factordb.com
      1. Plug in the n value from “values” and click the factorize button. The two numbers that pop up are the p and q values in step 3.



1. Make a Python program to handle the variables; I used the PyCharm IDE with Python3.
   1. Hardcode in the c,n,e variables from the “values” download, and plug in the p and q variables from step 2.
   2. To use the Crypto library the “pip installer” and “pycrypto” library will be needed. Type in the following commands in order if neither are installed, or as necessary.
      1. *sudo apt install python3-pip*
      2. *sudo apt-get install python-dev*
      3. *pip install pycrypto*
   3. Code:

from Crypto.Util.number import inverse, long\_to\_bytes

c = 240986837130071017759137533082982207147971245672412893755780400885108149004760496

n = 831416828080417866340504968188990032810316193533653516022175784399720141076262857

e = 65537

p = 1593021310640923782355996681284584012117

q = 521911930824021492581321351826927897005221

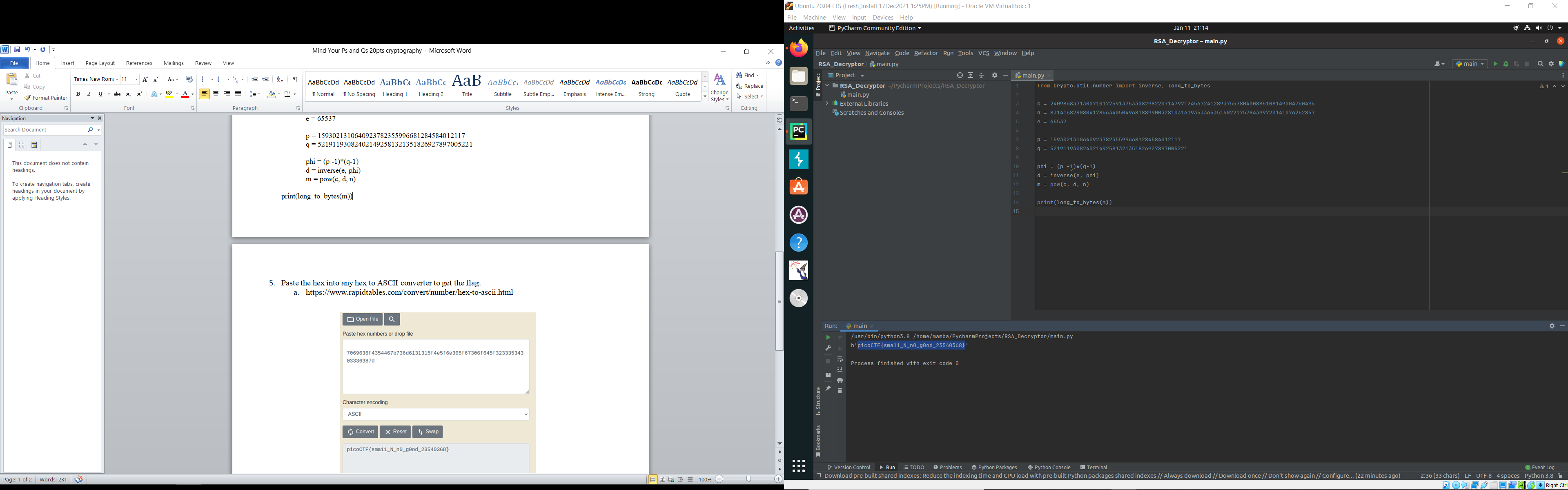
phi = (p -1)\*(q-1)

d = inverse(e, phi)

m = pow(c, d, n)

print(long\_to\_bytes(m))

4c



5

1. The plaintext flag will be outputted.