RSA Key Analysis & Challenges and Finding Large Primes

Project 1: RSA Key Format
Original Project: https://samsclass.info/141/proj/C403.htm

Overview
The RSA Key Analysis project focuses on understanding how RSA keys are stored, parsed, and analyzed. It involves working with private keys, public keys, and cryptographic operations using OpenSSL and Python.

Key Concepts Covered
RSA Key Structure – Understanding how RSA keys are made up of:
1. n (modulus) – The product of two large prime numbers (p * q).
2. e (public exponent) – Often set to 65537 for encryption.
3. d (private exponent) – Used for decryption.
4. p & q (prime factors) – The secret prime numbers used to generate n.

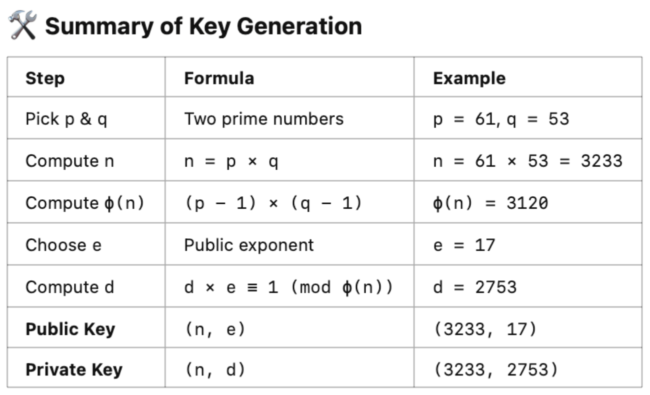
Extracting and Parsing RSA Keys
1. Using OpenSSL to generate and inspect RSA private keys.
2. Parsing ASN.1 formatted keys to extract n, e, d, p, q values.

Recovering RSA Key Components
1. Using OpenSSL and Python to find missing values like p and q from a given private key.
2. Converting hex values to decimal for computations.

Before starting the project, the first step is to downgrade OpenSSL to version 1.1.1, as newer versions (such as OpenSSL 3.x) may not be compatible with certain commands used in this challenge. In terminal run the following:

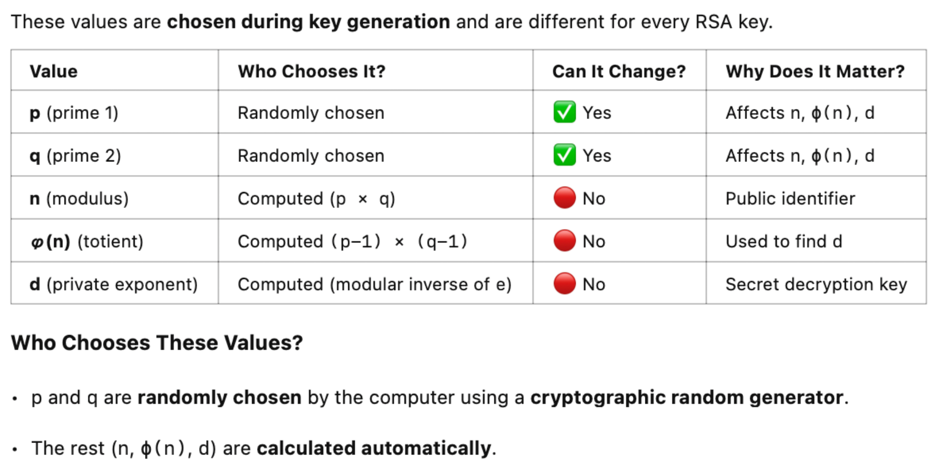
```
wget https://robohub.eng.uwaterloo.ca/mirror/ubuntu/pool/main/o/openssl/libssl1.1_1.1.1f-1ubuntu2.20_amd64.deb
wget https://robohub.eng.uwaterloo.ca/mirror/ubuntu/pool/main/o/openssl/libssl-dev_1.1.1f-1ubuntu2.20_amd64.deb
wget https://robohub.eng.uwaterloo.ca/mirror/ubuntu/pool/main/o/openssl/openssl_1.1.1f-1ubuntu2.20_amd64.deb

sudo dpkg -i libssl1.1_1.1.1f-1ubuntu2.20_amd64.deb
sudo dpkg -i libssl-dev_1.1.1f-1ubuntu2.20_amd64.deb
sudo dpkg -i openssl_1.1.1f-1ubuntu2.20_amd64.deb
```

This table outlines the key steps in RSA key generation, including choosing prime numbers, computing the modulus, selecting the public exponent, and deriving the private exponent. These steps ensure secure encryption and decryption in RSA cryptography.



🛠️ Summary of Key Generation

Step	Formula	Example
Pick p & q	Two prime numbers	`p = 61, q = 53`
Compute n	`n = p × q`	`n = 61 × 53 = 3233`
Compute `ϕ(n)`	`(p − 1) × (q − 1)`	`ϕ(n) = 3120`
Choose e	Public exponent	`e = 17`
Compute d	`d × e ≡ 1 (mod ϕ(n))`	`d = 2753`
Public Key	`(n, e)`	`(3233, 17)`
Private Key	`(n, d)`	`(3233, 2753)`



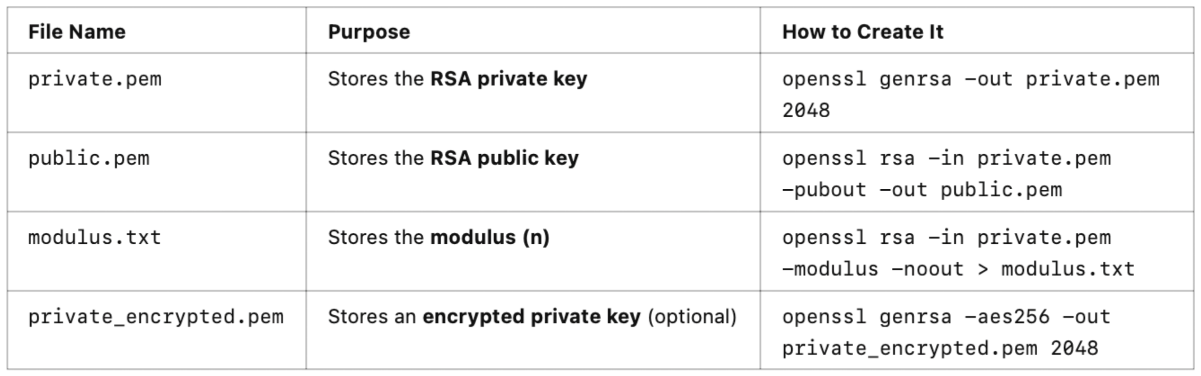
These values are **chosen during key generation** and are different for every RSA key.

Value	Who Chooses It?	Can It Change?	Why Does It Matter?
p (prime 1)	Randomly chosen	✅ Yes	Affects n, `ϕ(n)`, d
q (prime 2)	Randomly chosen	✅ Yes	Affects n, `ϕ(n)`, d
n (modulus)	Computed (p × q)	🔴 No	Public identifier
***φ*(n)** (totient)	Computed `(p−1) × (q−1)`	🔴 No	Used to find d
d (private exponent)	Computed (modular inverse of e)	🔴 No	Secret decryption key

Who Chooses These Values?

- p and q are **randomly chosen** by the computer using a **cryptographic random generator**.
- The rest (n, `ϕ(n)`, d) are **calculated automatically**.

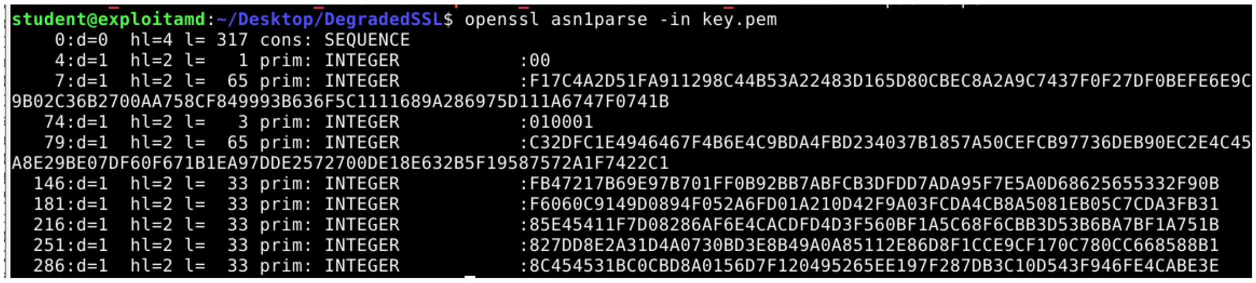
The following table provides key OpenSSL commands used for generating, extracting, and managing RSA keys. These commands help create private keys, public keys, modulus values, and encrypted private keys for secure cryptographic operations.



File Name	Purpose	How to Create It
`private.pem`	Stores the **RSA private key**	`openssl genrsa -out private.pem 2048`
`public.pem`	Stores the **RSA public key**	`openssl rsa -in private.pem -pubout -out public.pem`
`modulus.txt`	Stores the **modulus (n)**	`openssl rsa -in private.pem -modulus -noout > modulus.txt`
`private_encrypted.pem`	Stores an **encrypted private key** (optional)	`openssl genrsa -aes256 -out private_encrypted.pem 2048`

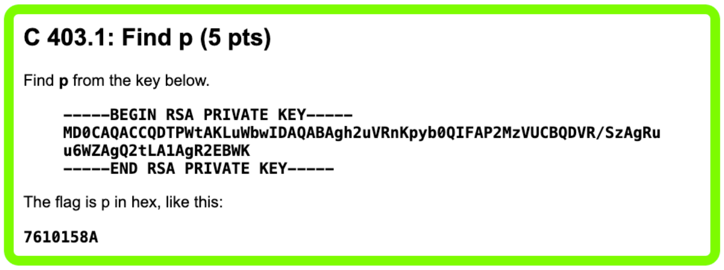
To see the values of need, p and q:

```
openssl asn1parse -in private.pem
```



```
student@exploitamd:~/Desktop/DegradedSSL$ openssl asn1parse -in key.pem
    0:d=0  hl=4 l= 317 cons: SEQUENCE
    4:d=1  hl=2 l=   1 prim: INTEGER           :00
    7:d=1  hl=2 l=  65 prim: INTEGER           :F17C4A2D51FA911298C44B53A22483D165D80CBEC8A2A9C7437F0F27DF0BEFE6E9C
9B02C36B2700AA758CF849993B636F5C1111689A286975D111A6747F0741B
   74:d=1  hl=2 l=   3 prim: INTEGER           :010001
   79:d=1  hl=2 l=  65 prim: INTEGER           :C32DFC1E4946467F4B6E4C9BDA4FBD234037B1857A50CEFCB97736DEB90EC2E4C45
A8E29BE07DF60F671B1EA97DDE2572700DE18E632B5F19587572A1F7422C1
  146:d=1  hl=2 l=  33 prim: INTEGER           :FB47217B69E97B701FF0B92BB7ABFCB3DFDD7ADA95F7E5A0D68625655332F90B
  181:d=1  hl=2 l=  33 prim: INTEGER           :F6060C9149D0894F052A6FD01A210D42F9A03FCDA4CB8A5081EB05C7CDA3FB31
  216:d=1  hl=2 l=  33 prim: INTEGER           :85E45411F7D08286AF6E4CACDFD4D3F560BF1A5C68F6CBB3D53B6BA7BF1A751B
  251:d=1  hl=2 l=  33 prim: INTEGER           :827DD8E2A31D4A0730BD3E8B49A0A85112E86D8F1CCE9CF170C780CC668588B1
  286:d=1  hl=2 l=  33 prim: INTEGER           :8C454531BC0CBD8A0156D7F120495265EE197F287DB3C10D543F946FE4CABE3E
```

C403.1: Getting prime values



C 403.1: Find p (5 pts)

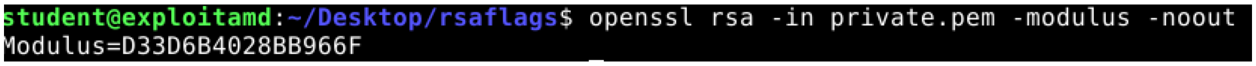
Find **p** from the key below.

```
-----BEGIN RSA PRIVATE KEY-----
MD0CAQACCQDTPWtAKLuWbwIDAQABAgh2uVRnKpyb0QIFAP2MzVUCBQDVR/SzAgRu
u6WZAgQ2tLA1AgR2EBWK
-----END RSA PRIVATE KEY-----
```

The flag is p in hex, like this:

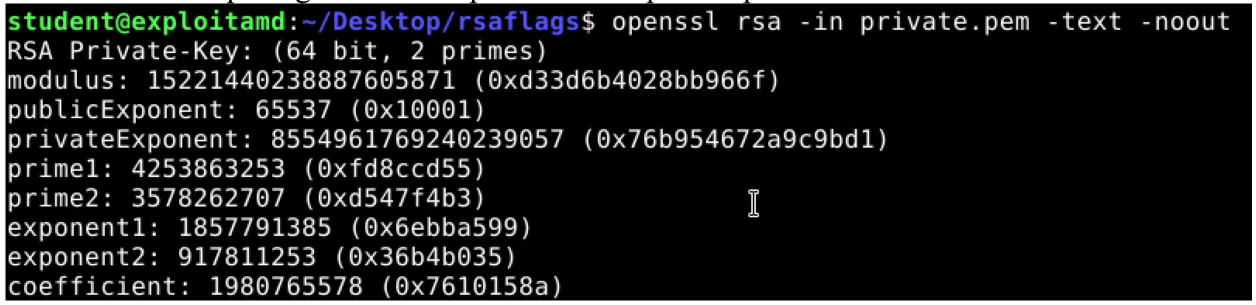
7610158A

This challenge provides the private key that contains n (modulus) and d (private exponent). We are tasked to find the prime in hex. First, we extract n from private key using command openssl rsa -in private.pem -modulus -noout



```
student@exploitamd:~/Desktop/rsaflags$ openssl rsa -in private.pem -modulus -noout
Modulus=D33D6B4028BB966F
```

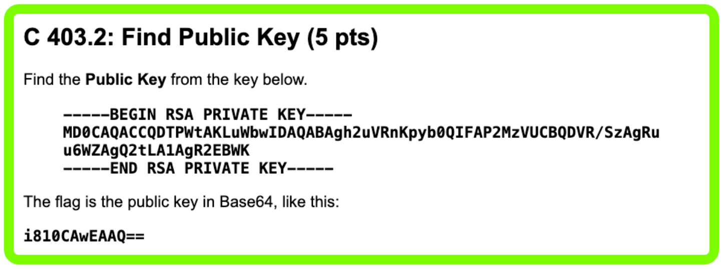
Then we extract q using command openssl rsa -in private.pem -text -noout.



```
student@exploitamd:~/Desktop/rsaflags$ openssl rsa -in private.pem -text -noout
RSA Private-Key: (64 bit, 2 primes)
modulus: 15221440238887605871 (0xd33d6b4028bb966f)
publicExponent: 65537 (0x10001)
privateExponent: 8554961769240239057 (0x76b954672a9c9bd1)
prime1: 4253863253 (0xfd8ccd55)
prime2: 3578262707 (0xd547f4b3)
exponent1: 1857791385 (0x6ebba599)
exponent2: 917811253 (0x36b4b035)
coefficient: 1980765578 (0x7610158a)
```

The value of q in hex is fd8ccd55

C 403.2: Find public key



C 403.2: Find Public Key (5 pts)

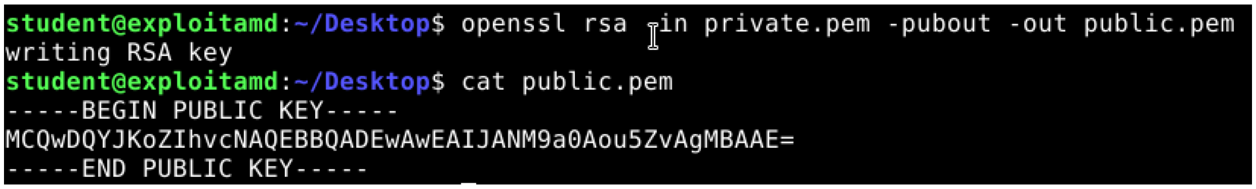
Find the **Public Key** from the key below.

```
-----BEGIN RSA PRIVATE KEY-----
MD0CAQACCQDTPWtAKLuWbwIDAQABAgh2uVRnKpyb0QIFAP2MzVUCBQDVR/SzAgRu
u6WZAgQ2tLA1AgR2EBWK
-----END RSA PRIVATE KEY-----
```

The flag is the public key in Base64, like this:

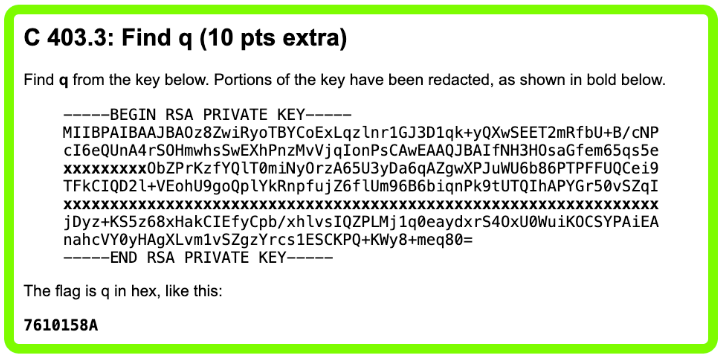
i81OCAwEAAQ==

This challenge requires us to get the public key from private key. The command to use is openssl rsa -in private.pem -pubout -out public.pem


```
student@exploitamd:~/Desktop$ openssl rsa -in private.pem -pubout -out public.pem
writing RSA key
student@exploitamd:~/Desktop$ cat public.pem
-----BEGIN PUBLIC KEY-----
MCQwDQYJKoZIhvcNAQEBBQADEwAwEAIJANM9a0Aou5ZvAgMBAAE=
-----END PUBLIC KEY-----
```

The public key in base 64 is
LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUNRd0RRWUpLb1pJaHZjTkFRRUJCUU
FERXdBd0VB
SUpBTk05YTBBb3U1WnZBZ01CQUFFPQotLS0tLUVORCBQVUJMSUMgS0VZLS0tLS0K

C 403.3: Find q from a partially redacted private key


C 403.3: Find q (10 pts extra)

Find **q** from the key below. Portions of the key have been redacted, as shown in bold below.

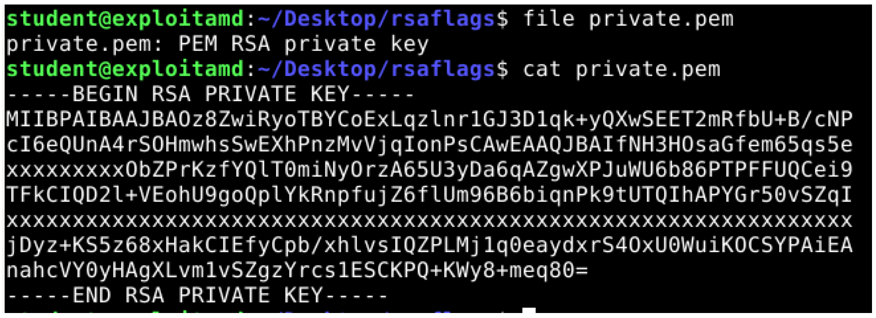
```
-----BEGIN RSA PRIVATE KEY-----
MIIBPAIBAAJBAOz8ZwiRyoTBYCoExLqzlnr1GJ3D1qk+yQXwSEET2mRfbU+B/cNP
cI6eQUnA4rSOHmwhsSwEXhPnzMvVjqIonPsCAwEAAQJBAIfNH3HOsaGfem65qs5e
xxxxxxxxxObZPrKzfYQlT0miNyOrzA65U3yDa6qAZgwXPJuWU6b86PTPFFUQCei9
TFkCIQD2l+VEohU9goQplYkRnpfujZ6flUm96B6biqnPk9tUTQIhAPYGr50vSZqI
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
jDyz+KS5z68xHakCIEfyCpb/xhlvsIQZPLMj1q0eaydxrS4OxU0WuiKOCSYPAiEA
nahcVY0yHAgXLvm1vSZgzYrcs1ESCKPQ+KWy8+meq80=
-----END RSA PRIVATE KEY-----
```

The flag is q in hex, like this:

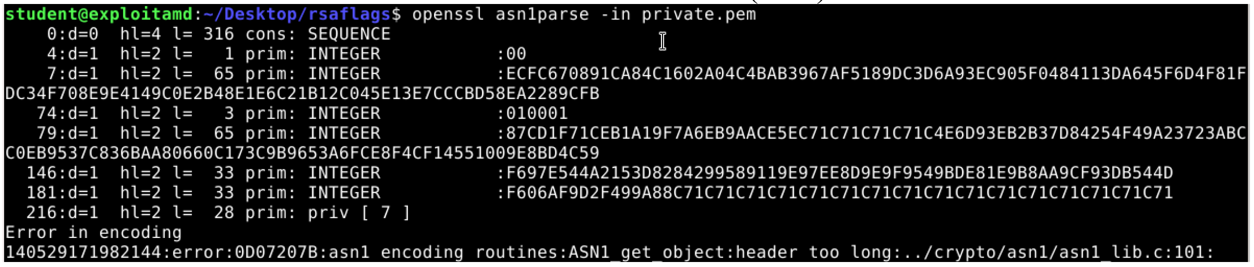
7610158A

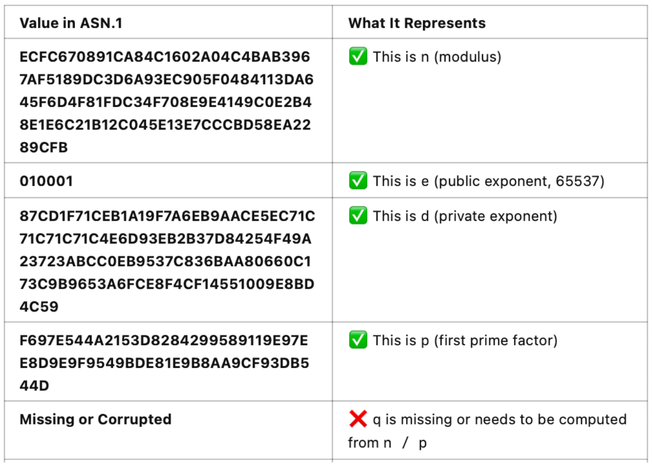
The given private key values can be read as such:

```
-----BEGIN RSA PRIVATE KEY-----
MIIBPAIBAAJBAOz8ZwiRyoTBYCoExLqzlnr1GJ3D1qk+yQXwSEET2mRfbU+B/cNP  <-- Public Key Info (n & e)
cI6eQUnA4rSOHmwhsSwEXhPnzMvVjqIonPsCAwEAAQJBAIfNH3HOsaGfem65qs5e  <-- Private Exponent (d)
xxxxxxxxxObZPrKzfYQlT0miNyOrzA65U3yDa6qAZgwXPJuWU6b86PTPFFUQCei9  <-- Redacted value (likely `p`)
TFkCIQD2l+VEohU9goQplYkRnpfujZ6flUm96B6biqnPk9tUTQIhAPYGr50vSZqI  <-- Redacted value (likely `q`)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  <-- Redacted value (possibly `d`, `q`, or `n`)
jDyz+KS5z68xHakCIEfyCpb/xhlvsIQZPLMj1q0eaydxrS4OxU0WuiKOCSYPAiEA  <-- Redacted value (possibly `dp`, `dq`, or `qi`)
nahcVY0yHAgXLvm1vSZgzYrcs1ESCKPQ+KWy8+meq80=
-----END RSA PRIVATE KEY-----
```


```
student@exploitamd:~/Desktop/rsaflags$ file private.pem
private.pem: PEM RSA private key
student@exploitamd:~/Desktop/rsaflags$ cat private.pem
-----BEGIN RSA PRIVATE KEY-----
MIIBPAIBAAJBAOz8ZwiRyoTBYCoExLqzlnr1GJ3D1qk+yQXwSEET2mRfbU+B/cNP
cI6eQUnA4rSOHmwhsSwEXhPnzMvVjqIonPsCAwEAAQJBAIfNH3HOsaGfem65qs5e
xxxxxxxxxObZPrKzfYQlT0miNyOrzA65U3yDa6qAZgwXPJuWU6b86PTPFFUQCei9
TFkCIQD2l+VEohU9goQplYkRnpfujZ6flUm96B6biqnPk9tUTQIhAPYGr50vSZqI
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
jDyz+KS5z68xHakCIEfyCpb/xhlvsIQZPLMj1q0eaydxrS4OxU0WuiKOCSYPAiEA
nahcVY0yHAgXLvm1vSZgzYrcs1ESCKPQ+KWy8+meq80=
-----END RSA PRIVATE KEY-----
```

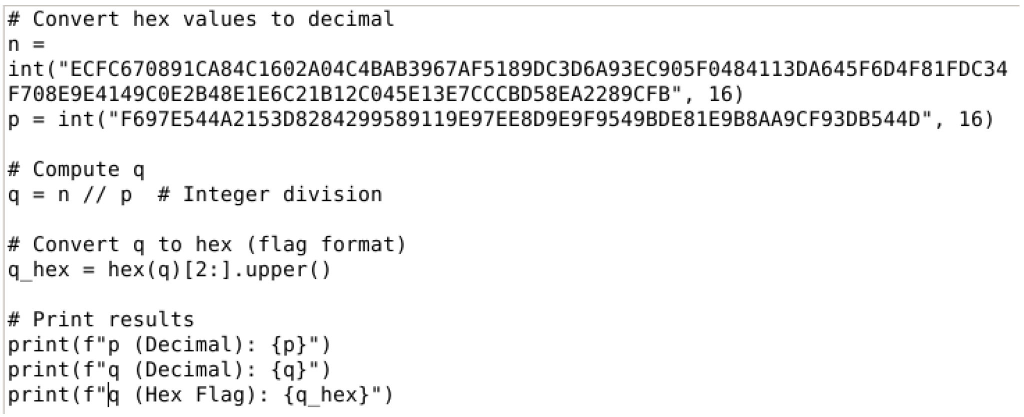
Here we can utilize the command to extracts n if readable (asn1)


```
student@exploitamd:~/Desktop/rsaflags$ openssl asn1parse -in private.pem
    0:d=0  hl=4 l= 316 cons: SEQUENCE
    4:d=1  hl=2 l=   1 prim: INTEGER           :00
    7:d=1  hl=2 l=  65 prim: INTEGER           :ECFC670891CA84C1602A04C4BAB3967AF5189DC3D6A93EC905F0484113DA645F6D4F81F
DC34F708E9E4149C0E2B48E1E6C21B12C045E13E7CCCBD58EA2289CFB
   74:d=1  hl=2 l=   3 prim: INTEGER           :010001
   79:d=1  hl=2 l=  65 prim: INTEGER           :87CD1F71CEB1A19F7A6EB9AACE5EC71C71C71C71C4E6D93EB2B37D84254F49A23723ABC
C0EB9537C836BAA80660C173C9B9653A6FCE8F4CF14551009E8BD4C59
  146:d=1  hl=2 l=  33 prim: INTEGER           :F697E544A2153D8284299589119E97EE8D9E9F9549BDE81E9B8AA9CF93DB544D
  181:d=1  hl=2 l=  33 prim: INTEGER           :F606AF9D2F499A88C71C71C71C71C71C71C71C71C71C71C71C71C71C71C71C71
  216:d=1  hl=2 l=  28 prim: priv [ 7 ]
Error in encoding
140529171982144:error:0D07207B:asn1 encoding routines:ASN1_get_object:header too long:../crypto/asn1/asn1_lib.c:101:
```

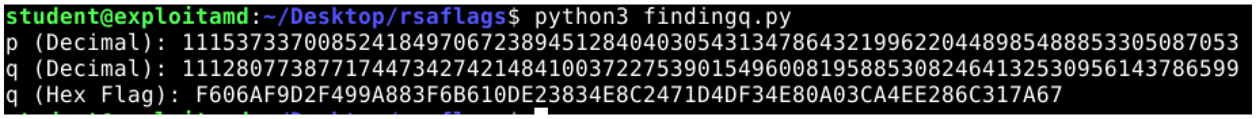


Value in ASN.1	What It Represents
ECFC670891CA84C1602A04C4BAB3967AF5189DC3D6A93EC905F0484113DA645F6D4F81FDC34F708E9E4149C0E2B48E1E6C21B12C045E13E7CCCBD58EA2289CFB	✅ This is n (modulus)
010001	✅ This is e (public exponent, 65537)
87CD1F71CEB1A19F7A6EB9AACE5EC71C71C71C71C4E6D93EB2B37D84254F49A23723ABCC0EB9537C836BAA80660C173C9B9653A6FCE8F4CF14551009E8BD4C59	✅ This is d (private exponent)
F697E544A2153D8284299589119E97EE8D9E9F9549BDE81E9B8AA9CF93DB544D	✅ This is p (first prime factor)
Missing or Corrupted	❌ q is missing or needs to be computed from n / p

We were able to find values n, e, d and p. From there, we can find q by converting hex values into decimal followed by dividing n with p.



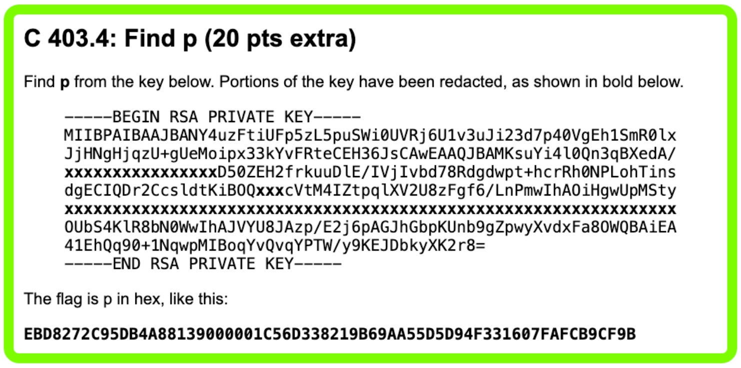
```
# Convert hex values to decimal
n =
int("ECFC670891CA84C1602A04C4BAB3967AF5189DC3D6A93EC905F0484113DA645F6D4F81FDC34
F708E9E4149C0E2B48E1E6C21B12C045E13E7CCCBD58EA2289CFB", 16)
p = int("F697E544A2153D8284299589119E97EE8D9E9F9549BDE81E9B8AA9CF93DB544D", 16)

# Compute q
q = n // p  # Integer division

# Convert q to hex (flag format)
q_hex = hex(q)[2:].upper()

# Print results
print(f"p (Decimal): {p}")
print(f"q (Decimal): {q}")
print(f"q (Hex Flag): {q_hex}")
```



```
student@exploitamd:~/Desktop/rsaflags$ python3 findingq.py
p (Decimal): 111537337008524184970672389451284040305431347864321996220448985488853305087053
q (Decimal): 111280773877174473427421484100372275390154960081958853082464132530956143786599
q (Hex Flag): F606AF9D2F499A883F6B610DE23834E8C2471D4DF34E80A03CA4EE286C317A67
```

The prime q value in hex is F606AF9D2F499A883F6B610DE23834E8C2471D4DF34E80A03CA4EE286C317A67

C403.4: Find p from a partially redacted private key



C 403.4: Find p (20 pts extra)

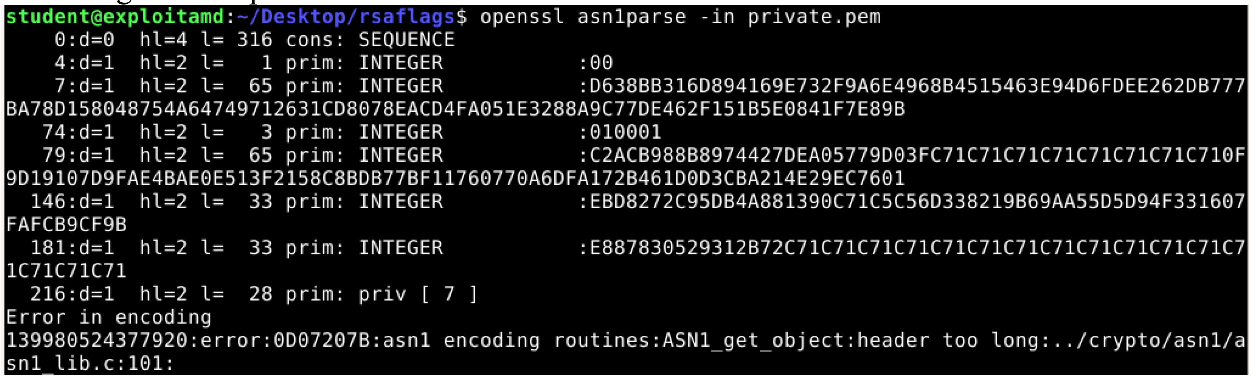
Find **p** from the key below. Portions of the key have been redacted, as shown in bold below.

```
-----BEGIN RSA PRIVATE KEY-----
MIIBPAIBAAJBANY4uzFtiUFp5zL5puSWi0UVRj6U1v3uJi23d7p40VgEh1SmR0lx
JjHNgHjqzU+gUeMoipx33kYvFRteCEH36JsCAwEAAQJBAMKsuYi4l0Qn3qBXedA/
xxxxxxxxxxxxxxxxD50ZEH2frkuuDlE/IVjIvbd78Rdgdwpt+hcrRh0NPLohTins
dgECIQDr2CcsldtKiBOQxxxcVtM4IZtpqlXV2U8zFgf6/LnPmwIhAOiHgwUpMSty
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
OUbS4KlR8bN0WwIhAJVYU8JAzp/E2j6pAGJhGbpKUnb9gZpwyXvdxFa8OWQBAiEA
41EhQq90+1NqwpMIBoqYvQvqYPTW/y9KEJDbkyXK2r8=
-----END RSA PRIVATE KEY-----
```

The flag is p in hex, like this:

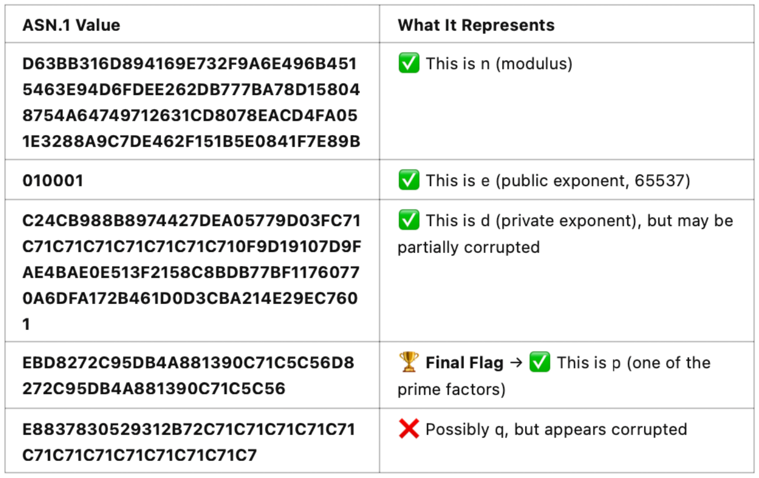
EBD8272C95DB4A88139000001C56D338219B69AA55D5D94F331607FAFCB9CF9B

Following same steps in C403.3:



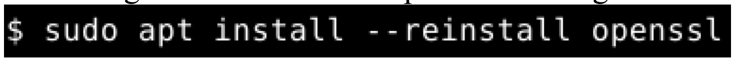
```
student@exploitamd:~/Desktop/rsaflags$ openssl asn1parse -in private.pem
    0:d=0  hl=4 l= 316 cons: SEQUENCE
    4:d=1  hl=2 l=   1 prim: INTEGER           :00
    7:d=1  hl=2 l=  65 prim: INTEGER           :D638BB316D894169E732F9A6E4968B4515463E94D6FDEE262DB777
BA78D158048754A64749712631CD8078EACD4FA051E3288A9C77DE462F151B5E0841F7E89B
   74:d=1  hl=2 l=   3 prim: INTEGER           :010001
   79:d=1  hl=2 l=  65 prim: INTEGER           :C2ACB988B8974427DEA05779D03FC71C71C71C71C71C71C710F
9D19107D9FAE4BAE0E513F2158C8BDB77BF11760770A6DFA172B461D0D3CBA214E29EC7601
  146:d=1  hl=2 l=  33 prim: INTEGER           :EBD8272C95DB4A881390C71C5C56D338219B69AA55D5D94F331607
FAFCB9CF9B
  181:d=1  hl=2 l=  33 prim: INTEGER           :E887830529312B72C71C71C71C71C71C71C71C71C71C71C71C71C7
1C71C71C71
  216:d=1  hl=2 l=  28 prim: priv [ 7 ]
Error in encoding
139980524377920:error:0D07207B:asn1 encoding routines:ASN1_get_object:header too long:../crypto/asn1/a
sn1_lib.c:101:
```

We were able to find values n, e, and p. The prime p value is EBD8272C95DB4A881390C71C5C56D338219B69AA55D5D94F331607FAFCB9CF9B



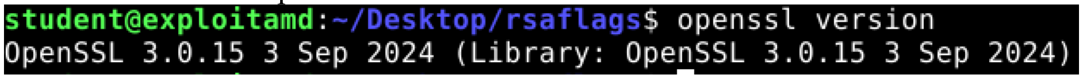
ASN.1 Value	What It Represents
D63BB316D894169E732F9A6E496B4515463E94D6FDEE262DB777BA78D158048754A64749712631CD8078EACD4FA051E3288A9C7DE462F151B5E0841F7E89B	✅ This is n (modulus)
010001	✅ This is e (public exponent, 65537)
C24CB988B8974427DEA05779D03FC71C71C71C71C71C71C710F9D19107D9FAE4BAE0E513F2158C8BDB77BF11760770A6DFA172B461D0D3CBA214E29EC7601	✅ This is d (private exponent), but may be partially corrupted
EBD8272C95DB4A881390C71C5C56D8272C95DB4A881390C71C5C56	🏆 **Final Flag** → ✅ This is p (one of the prime factors)
E8837830529312B72C71C71C71C71C71C71C71C71C71C71C71C7	❌ Possibly q, but appears corrupted

Don’t forget to revert back to updated ssl using command:



```
$ sudo apt install --reinstall openssl
```

Or sudo apt install openssl libssl-dev
Make sure to check openssl version:



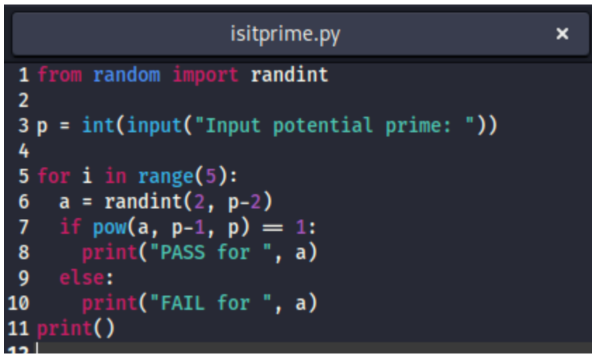
```
student@exploitamd:~/Desktop/rsaflags$ openssl version
OpenSSL 3.0.15 3 Sep 2024 (Library: OpenSSL 3.0.15 3 Sep 2024)
```

****

Project 2: Finding Large Primes:
Original Project: https://samsclass.info/141/proj/C503.htm

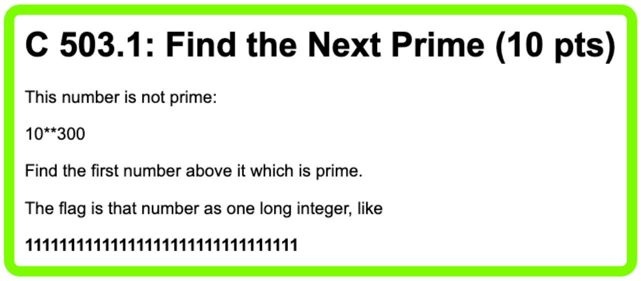
To determine whether a number is **prime**, we apply **Fermat’s Primality Test**:

1. If a number passes the test for multiple values of a, it is **probably prime**.
2. If it fails for any a, it is **not prime**.



isitprime.py

```python
from random import randint

p = int(input("Input potential prime: "))

for i in range(5):
  a = randint(2, p-2)
  if pow(a, p-1, p) == 1:
    print("PASS for ", a)
  else:
    print("FAIL for ", a)
print()
```



C 503.1: Find the Next Prime (10 pts)

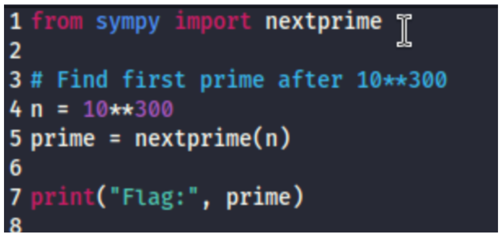
This number is not prime:

10**300

Find the first number above it which is prime.

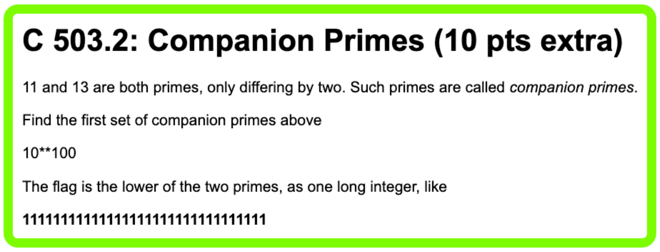
The flag is that number as one long integer, like

11111111111111111111111111111111



```python
from sympy import nextprime

# Find first prime after 10**300
n = 10**300
prime = nextprime(n)

print("Flag:", prime)
```

Answer:
100000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000331



C 503.2: Companion Primes (10 pts extra)

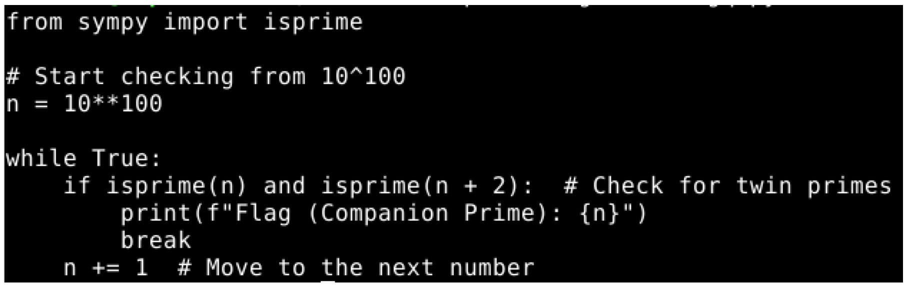
11 and 13 are both primes, only differing by two. Such primes are called *companion primes*.

Find the first set of companion primes above

10**100

The flag is the lower of the two primes, as one long integer, like

11111111111111111111111111111111



```python
from sympy import isprime

# Start checking from 10^100
n = 10**100

while True:
    if isprime(n) and isprime(n + 2):  # Check for twin primes
        print(f"Flag (Companion Prime): {n}")
        break
    n += 1  # Move to the next number
```

Answer:
100000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000035737