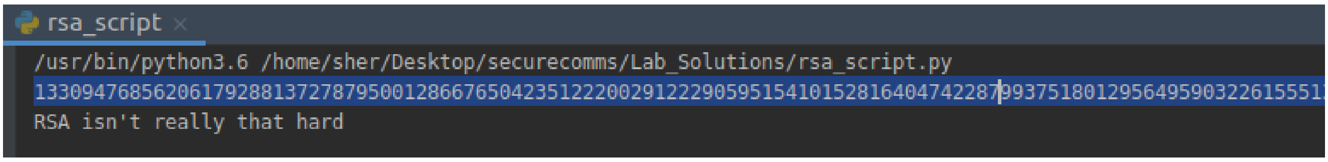
Zero-Days Challenges

Level 1

I did following steps to soluve this challenge:

1. downloaded rsa-script.py file from moodle
2. downloaded rsa1.txt file from zero days
3. substituated values of n, e, p, q in rsa-script.py
4. run rsa-script.py in pycharm and it encrypted the message



```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/rsa_script.py
133094768562061792881372787950012866765042351222002912229059515410152816404742287993751801295649590322615551
RSA isn't really that hard
```

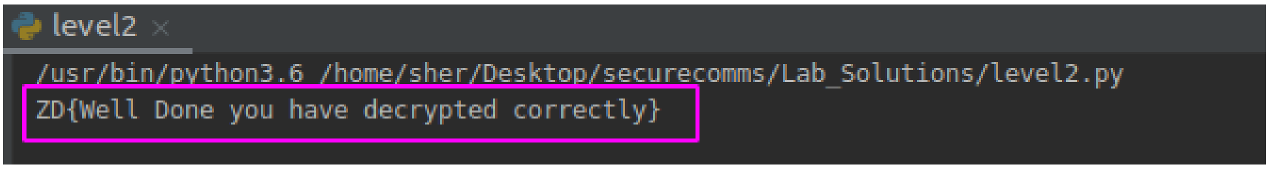
Script

As I did not made any changes to script provided (rsa_script.py), therefore not including code for this leve.

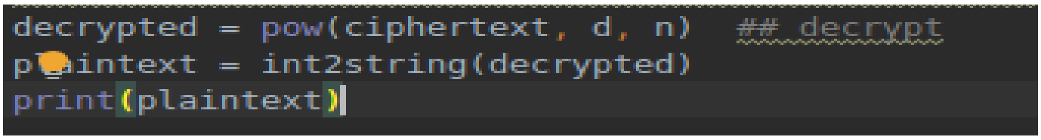
Level 2

Steps taken to soluve this challenge:

1. copied the cypher text from zero-days
2. substituted in rsa-script.py file and run it
3. it decrypted the cypher text



```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/level2.py
ZD{Well Done you have decrypted correctly}
```

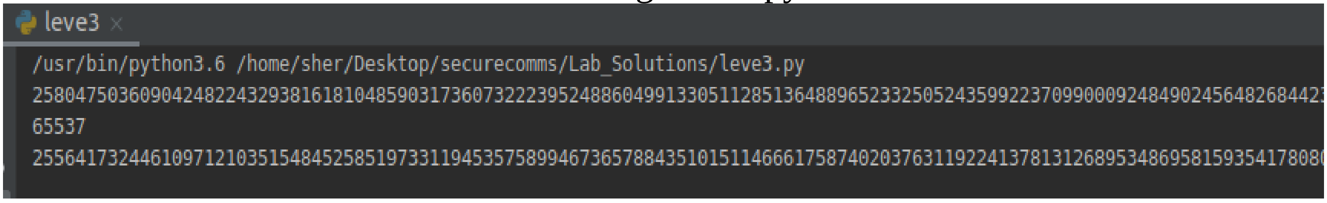


```
decrypted = pow(ciphertext, d, n)  ## decrypt
plaintext = int2string(decrypted)
print(plaintext)
```

Level 3

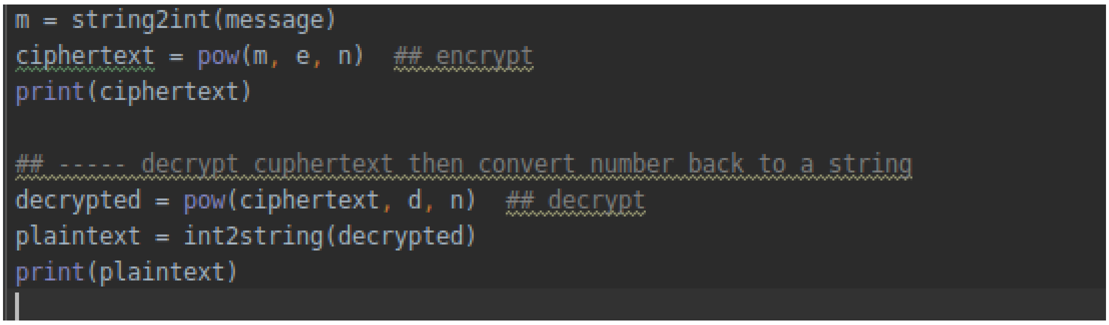
Steps taken to soluve this challenge:

1. downloaded mykey2 file
2. run “openssl asn1parse -in mykey2 -i” command on it
3. it gave me values of n, e, d but in hexadecimal
4. converted into decimal values using level3.py



```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/leve3.py
25804750360904248224329381618104859031736073222395248860499133051128513648896523325052435992237099000924849024564826844
65537
2556417324461097121035154845258519733119453575899467365788435101511466617587402037631192241378131268953486958159354178080
```

Script is available here:
https://github.com/AhmedPak/securecomms/blob/master/Lab_Solutions/leve3.py

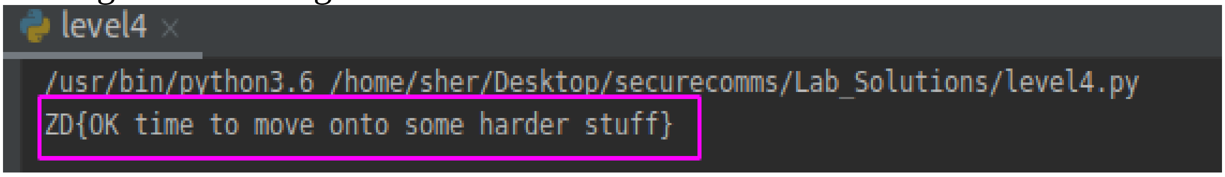


```
m = string2int(message)
ciphertext = pow(m, e, n)  ## encrypt
print(ciphertext)

## ----- decrypt cuphertext then convert number back to a string
decrypted = pow(ciphertext, d, n)  ## decrypt
plaintext = int2string(decrypted)
print(plaintext)
```

Level 4

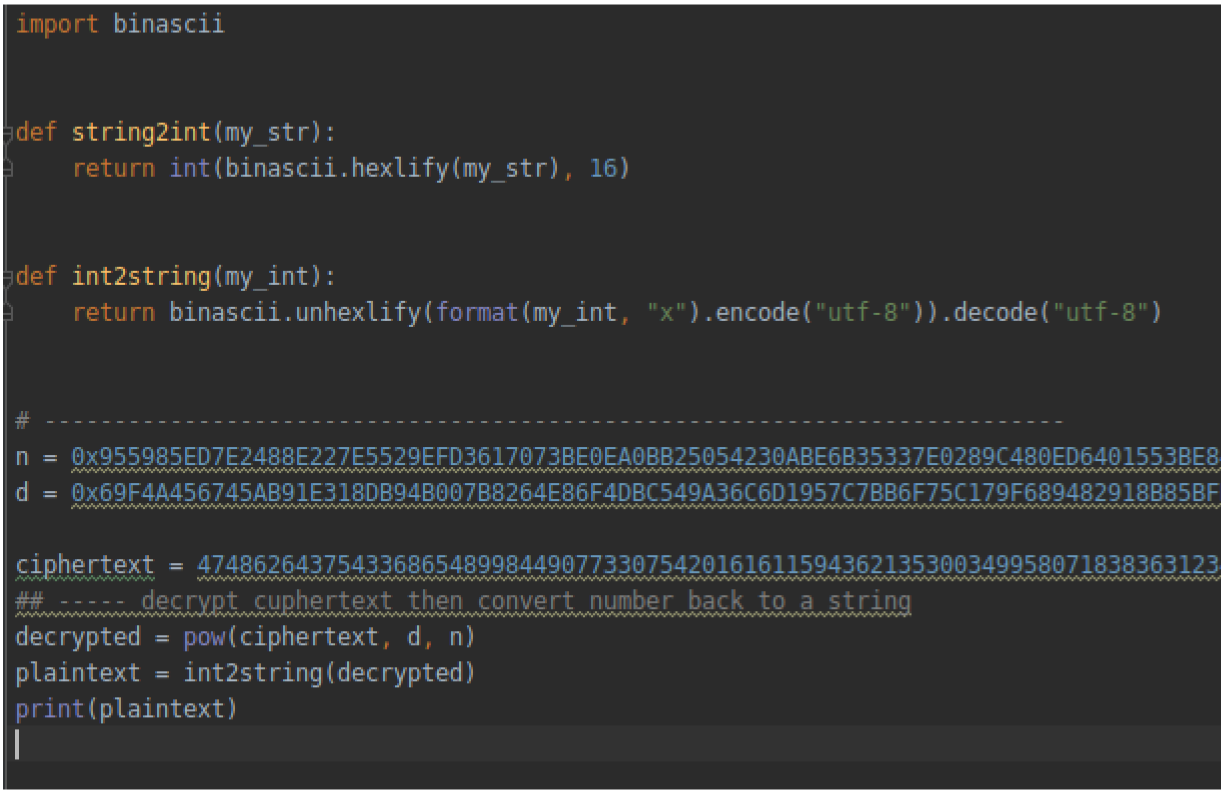
Steps taken to soluve this challenge:

1. copied cipher text from zero-days site and substituted in level4.py
2. downloaded mykey3 file and ran “openssl asn1parse -in mykey2 -i”
3. copied values hex values of n, d and substituted in level4.py
4. it gave us the flage



```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/level4.py
ZD{OK time to move onto some harder stuff}
```

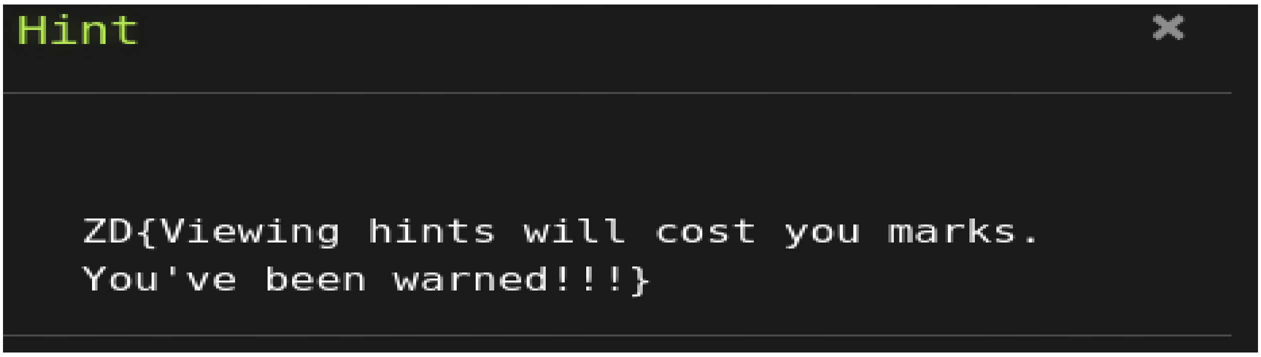
Script is available here:
https://github.com/AhmedPak/securecomms/blob/master/Lab_Solutions/level4.py



```
import binascii


def string2int(my_str):
    return int(binascii.hexlify(my_str), 16)


def int2string(my_int):
    return binascii.unhexlify(format(my_int, "x").encode("utf-8")).decode("utf-8")


# ------------------------------------------------------------------------
n = 0x955985ED7E2488E227E5529EFD3617073BE0EA0BB25054230ABE6B35337E0289C480ED6401553BE8
d = 0x69F4A456745AB91E318DB94B007B8264E86F4DBC549A36C6D1957C7BB6F75C179F689482918B85BF

ciphertext = 474862643754336865489984490773307542016161159436213530034995807183836312
## ----- decrypt cuphertext then convert number back to a string
decrypted = pow(ciphertext, d, n)
plaintext = int2string(decrypted)
print(plaintext)
```

Level 4.5

This level was self-explanatory.

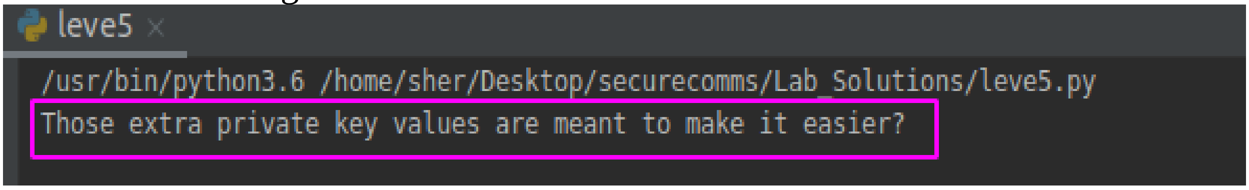


ZD{Viewing hints will cost you marks.
You've been warned!!!}

Level 5

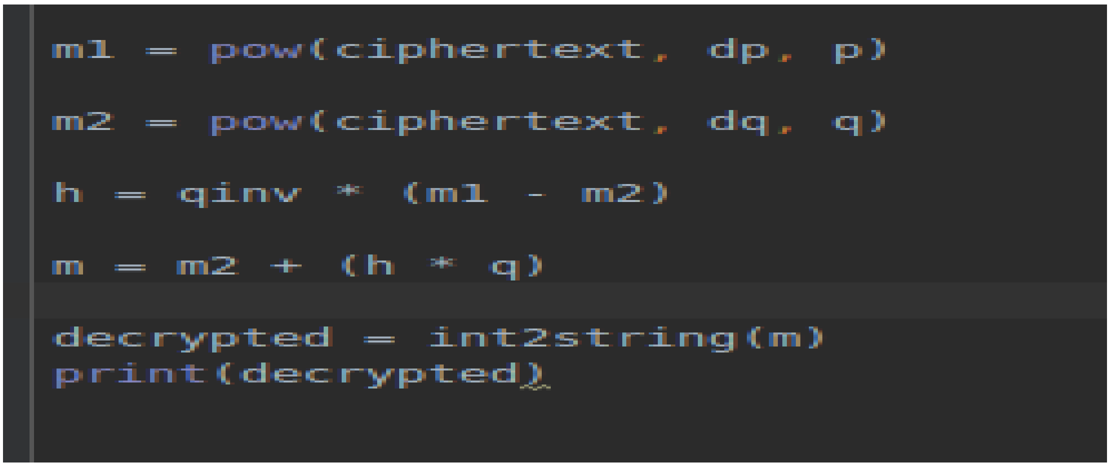
Steps taken to soluve this challenge:

1. downloaded key.txt file and substituted the values of p, q, dp, dq, pinv, qinv and ciphertext
2. used Chinese remaider algorithm to calculate the m (used hint-wiki rsa page)
3. used int2string function on m



```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/leve5.py
Those extra private key values are meant to make it easier?
```

Script is available here:
https://github.com/AhmedPak/securecomms/blob/master/Lab_Solutions/level5.py

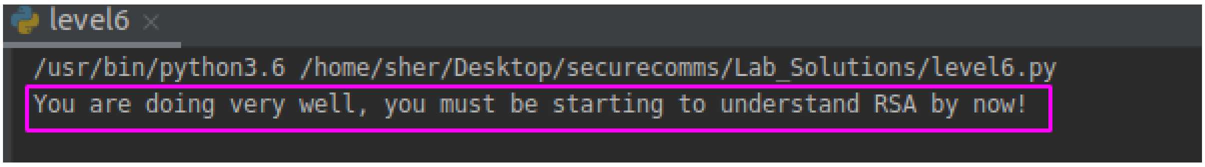


```
m1 = pow(ciphertext, dp, p)

m2 = pow(ciphertext, dq, q)

h = qinv * (m1 - m2)

m = m2 + (h * q)

decrypted = int2string(m)
print(decrypted)
```

Level 6

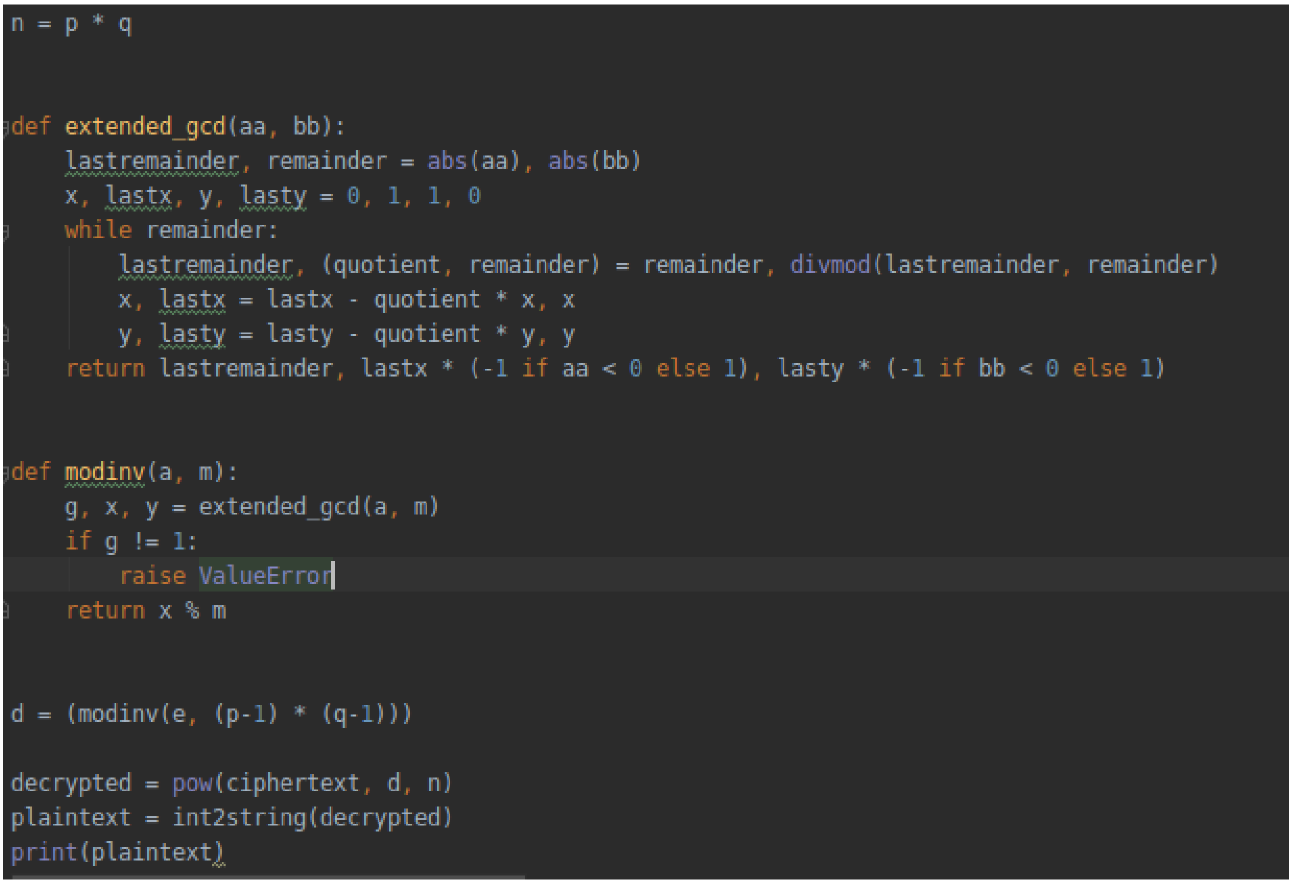
Steps taken to soluve this challenge:

1. downloaded key.txt file and substituted values of p, q, e and ciphertext into level6.py
2. calculated value of n using n = p * q
3. to calculate the value of d, we needed mod inverse of (p-1 * q-1) used a script from http://rosettacode.org/wiki/Modular_inverse
4. substituted the value of d in pow(ciphertxt, d, n)
5. used int2string function to get plaintext



```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/level6.py
You are doing very well, you must be starting to understand RSA by now!
```

Script is available here: https://github.com/AhmedPak/securecomms/blob/master/Lab_Solutions/level6.py



```
n = p * q


def extended_gcd(aa, bb):
    lastremainder, remainder = abs(aa), abs(bb)
    x, lastx, y, lasty = 0, 1, 1, 0
    while remainder:
        lastremainder, (quotient, remainder) = remainder, divmod(lastremainder, remainder)
        x, lastx = lastx - quotient * x, x
        y, lasty = lasty - quotient * y, y
    return lastremainder, lastx * (-1 if aa < 0 else 1), lasty * (-1 if bb < 0 else 1)


def modinv(a, m):
    g, x, y = extended_gcd(a, m)
    if g != 1:
        raise ValueError
    return x % m


d = (modinv(e, (p-1) * (q-1)))

decrypted = pow(ciphertext, d, n)
plaintext = int2string(decrypted)
print(plaintext)
```

Level 7

Steps taken to soluve this challenge:

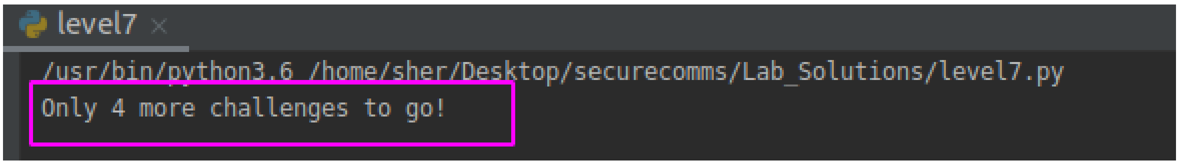
downloaded public.key file and substituted values of n, e and ciphertext into level7.py

to calculate values of p and q factorized n online using a website http://factordb.com/

substitued values of p and q in level7.py

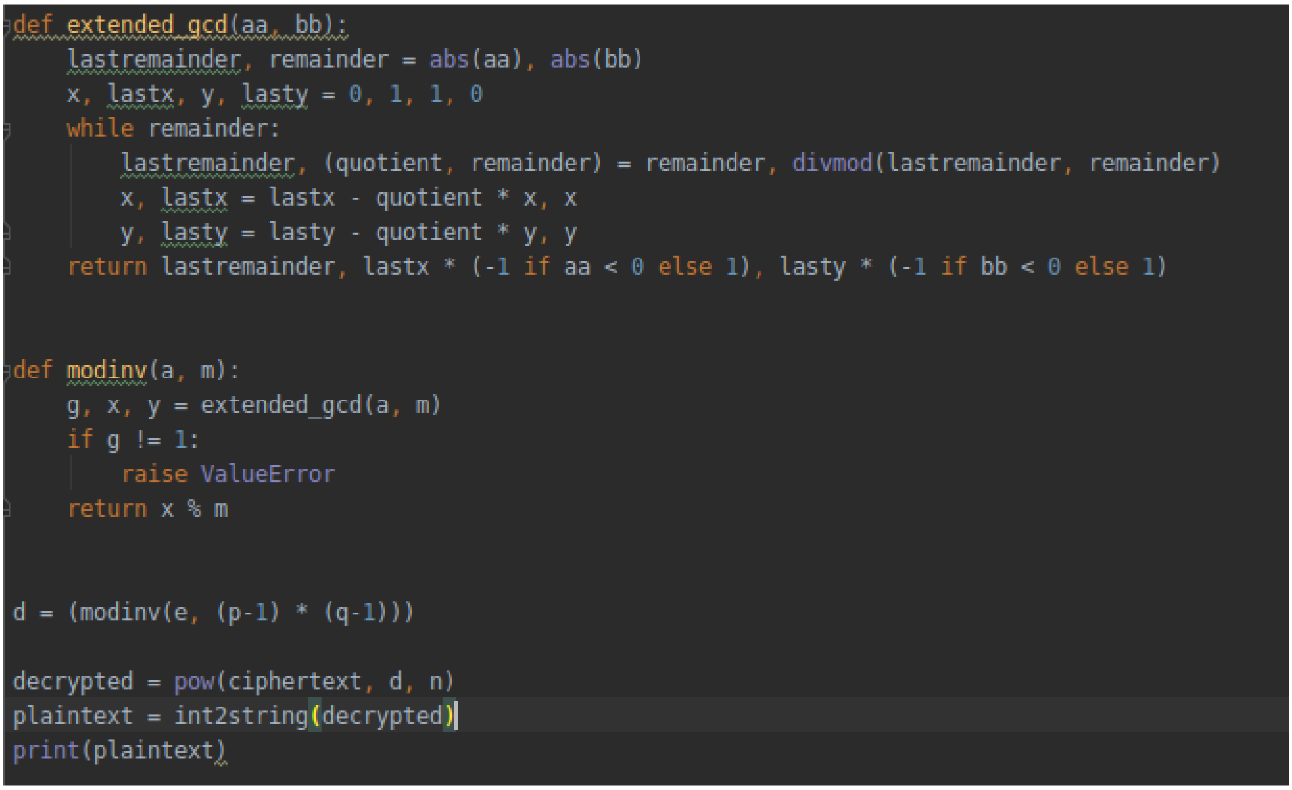
calculated value of d using code from http://rosettacode.org/wiki/Modular_inverse

used valuses of d, n and c in m = pow (c, d, n) to get decrypted value
used int2string funtion to get plaintext



```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/level7.py
Only 4 more challenges to go!
```

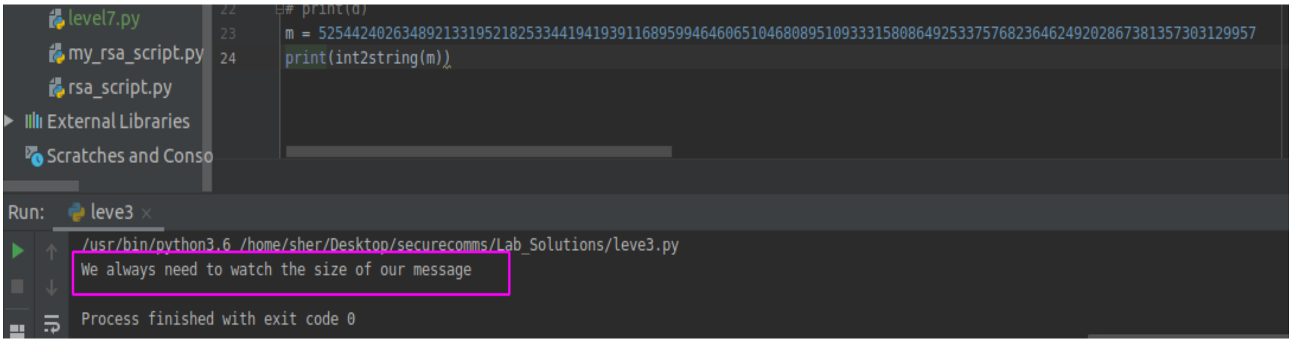
Script is available here:
https://github.com/AhmedPak/securecomms/blob/master/Lab_Solutions/level7.py



```python
def extended_gcd(aa, bb):
    lastremainder, remainder = abs(aa), abs(bb)
    x, lastx, y, lasty = 0, 1, 1, 0
    while remainder:
        lastremainder, (quotient, remainder) = remainder, divmod(lastremainder, remainder)
        x, lastx = lastx - quotient * x, x
        y, lasty = lasty - quotient * y, y
    return lastremainder, lastx * (-1 if aa < 0 else 1), lasty * (-1 if bb < 0 else 1)


def modinv(a, m):
    g, x, y = extended_gcd(a, m)
    if g != 1:
        raise ValueError
    return x % m


d = (modinv(e, (p-1) * (q-1)))

decrypted = pow(ciphertext, d, n)
plaintext = int2string(decrypted)
print(plaintext)
```

Level 8

Steps taken to soluve this challenge:
by looking at values.txt file realised that value of e is very small. We know that when value of e very small the value of m is the value of n. In other words simply factorizing the m gave us our flage. I used factordb.com to facterize m and than used int2string function on it to get our plain text.



```
# print(d)
m = 52544240263489213319521825334419419391168959946460651046808951093331580864925337576823646249202867381357303129957
print(int2string(m))
```

```
/usr/bin/python3.6 /home/sher/Desktop/securecomms/Lab_Solutions/leve3.py
We always need to watch the size of our message

Process finished with exit code 0
```

Scripts that were used to soluve these challenges are available at https://github.com/AhmedPak/securecomms/tree/master/Lab_Solutions