Assignment six – Python for Programmers

1. The full plaintext (decrypted message)

https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Attacks_against_plain_RSA

2. How I approached the problem

In the assignment the public key was given along with the encrypted message. By studying the code in rsa.py I saw that the *generate_keypair* method created the keys. This method returned the public key as (e,n) and the private key as (d,n). Since I had the public key I therefore *n*, I only had to find *d*.

To do this I had to factor *n*. I did this by using the method *PrimeGen()* with the arguments *math.sqrt(n)*. Then I found two numbers *p* and *q*, so *n=p*q*. I also needed *phi = (p-1) * (q-1)*. Then I had to generate the private key by using the Extended Euclid's Algorithm with the *e* from the public key and *phi*.

Then I used the *decrypt()* method with the private key and the encrypted message.

3. What are the problems with this simple implementation of RSA?

If plain text is used, an attacker can encrypt any likely plain-text, and see if it matches the ciphertext. Therefore we need randomized padding on our plain text.

Another problem with this simple implementation is the encryption with small prime numbers. The biggest problem with RSA is the factoring the public key. This is why we should use large prime numbers as public keys.

4. Multiple solutions

In this assignment I did not find multiple solutions. I could have tried encrypting likely plain-text to see if I could get a match with the ciphertext.