Nikolai Dokken

Python for Programmers Assignment 7

Plaintext – decrypted message:
https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Attacks_against_plain_RSA

How I approached the problem
Since the public key was given, I looked at the *generate_keypair* method to see how the private key was generated. Since the private key consists of two integers d and n, and n is also in the public key, I only had to find d.

The first thing I did was to factor n. I did this by using PrimeGen(math.sqrt(n) and then finding two integers p and q so that p*q=n. I then found phi = (p-1) * (q-1). The last step to generate the integer d is to Use Extended Euclid's Algorithm with an integer e, and phi. Since e was given in the public key, this was easy. Finally, I used the *decrypt* method with the private key I found.

What are the problems with this simple implementation of RSA?
If plaintext is used, an attacker can encrypt any likely plain-text, and see if it matches the ciphertext. This is why we need randomized padding on our plain-text.

Another problem with this simple implementation is encrypting with small prime numbers. This makes it much easier to find p and q. Usually a large prime number is used, which should help. Factoring the public key, as we did above, is considered one of the biggest problems with RSA. This is why we use large prime numbers as public key. However, many computers use the same systems to generate the prime number. This means that several keys have overlapping p's or q's. This means that you could try the most common p, and then easily find the q.

Multiple solutions?
I did not find multiple solutions, but I could have tried encrypting likely plain-text to see if I could get a match with the ciphertext.