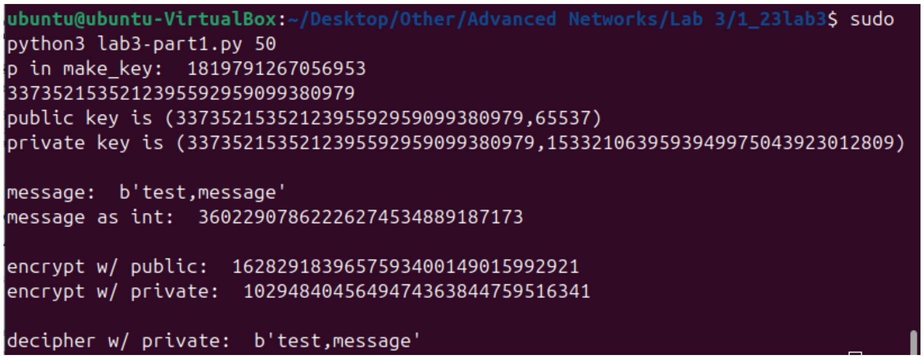
Lab 3 - RSA

Preston Lantzer

January 25, 2025

Part 1

1. To make the key I used prime.py's gen_prime function to create p and q, which make up n.
2. I then used pythons math.lcm function to find the least common multiple of (p-1) and (q-1), named 'lamda'.
3. To calculate d, I passed 'lamda', and e (65537), to the modinv(a, m) function that was given.
4. After n and d were calculated the public and private key tuples were returned to main.
5. To decipher the text, I simple had to follow the decipher formula of M = C^d % n (pow(c,d,n)).



```
ubuntu@ubuntu-VirtualBox:~/Desktop/Other/Advanced Networks/Lab 3/1_23lab3$ sudo
python3 lab3-part1.py 50
p in make_key:  1819791267056953
3373521535212395592959099380979
public key is (3373521535212395592959099380979,65537)
private key is (3373521535212395592959099380979,1533210639593949975043923012809)

message:  b'test,message'
message as int:  36022907862226274534889187173

encrypt w/ public:  1628291839657593400149015992921
encrypt w/ private:  1029484045649474363844759516341

decipher w/ private:  b'test,message'
```

Part 2

1. I hardcoded the given c, n, and e variables provided in the lab pdf.
2. I used WolframAlpha to factor n, and then hardcoded the p and q that were found (p = 40822754178477882469, q = 50374890465154864223) and calculated lambda with the math.lcm() function.
3. 'd' was found by passing e and 'lamda'(variable name for lambda) into the modinv(a, m) function that we got in part 1.

4. With d known, I just had to decipher the given C. I used the pow() function (pow(c,d,n)) to calculate M, the issue was it was an int.
5. I converted the int of M into bytes with the .to_bytes() function with a guess of 50 bytes required to represent the int of M as bytes.
6. The final result was that C deciphered into ‘ConGratSPassed!’.