

Lab3

Joseph Potapenko

October 2025

1 Part 1 Screenshot

Reference Figure 1 below

2 Part 2 Answer

I did what the slides said to do step by step, just in python. But on a real note, I took n (the modulus) from the public key, and then I factored it into p and q, the two prime factors, and then used it to find the lamda of n by finding the lowest common multiple of p-1 and q-1. Then, I solved for d using the provided modinv function with e and lamda n, which gives us d as the modular multiplicative inverse of e modulo lamda n. And then we solve for the plaintext integer using the dcipher function $M = C$ to the power of $d \bmod n$, and turning that from the byte integer to text gives us the message: b'YouDonePassed!!'

3 Part 3 Answer and Figures

Reference Figures 2 and 3 below. Figure 2 is the data, and Figure 3 is the graph.

Since the amount every 10 steps increases exponentially, I cannot give a concrete answer, but I can hazard a guess. Since 104 was an average of approximately half an hour, 114 could be a few hours, 124 could be around half a day, and 134 could be around more than a day. I would estimate 144 is possible, but anything above 150-160 would be unreasonable in my opinion, but it is hard to have an accurate estimate without having a crap ton of time to actually test the program.

```

#I used    Windows PowerShell
#I could
PS C:\Users\joepo\Desktop\lab3> python .\practiceRSA.py 64
#I used    Windows PowerShell
#I could
from Crypto.PublicKey import RSA
import base64
private_key = (000002942810473142882697446887181688829, 65537)
public_key = (000002942810473142882697446887181688829, 78311631325974232869962803104372089033)

def main():
    message = b'test,message'
    print(int.from_bytes(message))
    pub_ciphered = 130523935161279606537284286196378293900
    pri_deciphered = b'test,message'

    #print(pub_ciphered)
    #print(pri_deciphered)
    #print(message)
    #print("")

    #decipher with private key
    y = cipher.decrypt(pub_ciphered, private[0])
    print("deciphered: ", y.to_bytes(len(message)))

    print("")

    #encrypt with private key
    ciph = cipher.encrypt(message, private[1], private[0])
    print("encrypted: ", ciph)

Ln 36, Col 43  2,346 characters          Plain text

```

Figure 1: Enter Caption

```

RSA key cracking
Testing 64-bit primes
Test 1: 1.6619 seconds
Test 2: 5.9161 seconds
Test 3: 2.7210 seconds
Test 4: 1.9920 seconds
Average for 64-bit
Time: 3.0728 seconds

Testing 74-bit primes
Test 1: 2.6672 seconds
Test 2: 30.6381 seconds
Test 3: 8.9725 seconds
Test 4: 1.1987 seconds
Average for 74-bit
Time: 11.0991 seconds

Testing 84-bit primes
Test 1: 80.8913 seconds
Test 2: 133.6597 seconds
Test 3: 133.6595 seconds
Test 4: 198.1897 seconds
Average for 84-bit
Time: 114.7905 seconds

Testing 94-bit primes
Test 1: 1946.6329 seconds
Test 2: 633.1093 seconds
Test 3: 507.5300 seconds
Test 4: 155.7500 seconds
Average for 94-bit
Time: 810.7380 seconds

Testing 104-bit primes
Test 1: 4840.0352 seconds
Test 2: 1456.9287 seconds
Test 3: 577.7086 seconds
Test 4: 115.0320 seconds
Average for 104-bit
Time: 1745.9261 seconds

```

Figure 2: Enter Caption

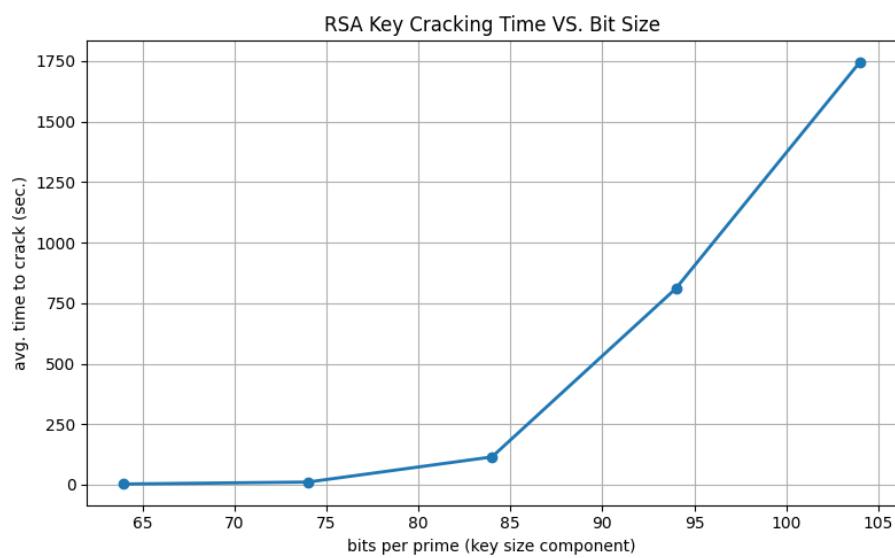


Figure 3: Enter Caption