Task2

In order to get the private key d, according to the RSA algorithm, we have to get the two distinct prime numbers p and q, such as N=p*q. The good thing is N here is not a very long number so that we can solve the p and q by using brute force. After several minutes computing, I got the number of p and q

N= 0xcd62f24bbda5ce3 = 924978608288193763

p= 0x39533b93 = 961756051

q= 0x39534b71 = 961760113

According to Euler's totient function, I computed $\varphi(N) = \varphi(p)\varphi(q) = (p - 1)(q - 1) = n - (p + q - 1)$,

Because d is the modular multiplicative inverse of e, which means that

$d*e \equiv 1 \pmod{\varphi(N)}$,

and e is prime number of 65537, we know that there should be one and only one positive integer k so that

$d*e = k*\varphi(N)+1$

we just need to find the k so that $k*\varphi(N)+1 \% e=0$. We can calculate the d as:

$$d = \frac{k*\boldsymbol{\varphi}(N)+1}{e}$$

Another thing is d has to be an odd number because φ(N) is an even number and e is an odd number. This could be used to check if d is correct.

Finally, I got k=61123 and d= 862680125071763873 = 0xbf8db00c9b129a1

Task3

According to the paper, because of the public key N is generated by the product of two distinct prime number, a random number generator(RNG) is used to “randomly” choose this two prime numbers. Of course, if the prime number is big enough and the number of public key pairs in the system is limited, RSA cryptosystem is still reliable, due to the practical difficulty of factoring the product of two large prime numbers.

However, even for two big numbers, the Euclidean algorithm of finding their greatest common divisor (GCD) is very fast (e.g. GCD of two 1024-bit integers can be computed in microseconds). As long as two public keys n1 and n2 have a common divisor greater than 1, the two prime factors of both n1 and n2 can be easily get by dividing the common divisor. The rest of this problem to find out private key d is easy, as I described in the Task2.

If the RNG happens to choose the same prime number as one of factor for two public key n1 and n2 in a system, both of them would be vulnerable, because attackers just need to judge if any two of public keys have GCD greater than 1. The author of the paper observed 0.50% of TLS hosts have this vulnerability in the experiment.

What I did in this task to get my private key is explained above: First, I checked each public key in the list to see if it has common divisor greater than 1 with my public key. Then, if another public key matches this criteria, IT IS WALDO!

```
ubuntu@ubuntu-VirtualBox:~$ python find_waldo.py nzou3
your public key: ( 0x829bfe9ddab5ad033b058094a4c092e0f9213c32ccacf5461895abb53ea0a6acf0bdc619fb1fd820965ad065048e5c66bee932c5
d65b872a42bcdb71d65ddd9d2e63ebe882d167a6b9d6cbac64996c625e12b3433247f313b55d112a6162efa58d259a048281ee239fc0d3597d6eedd716a47
cb1ea1955afd675b5e85ea9d01d , 0x10001 )
your private key:  0x13523e5d5dc06882c4d5708b0cc04c3a29c0a7e61fef2a2d9f6f6cc2582c6ddbc195615bf0916a688fb589849822f33a63dd9d9c
34b4bf8f19e6545447d104fa888b75335de86c066818cf0a56a9615451b9b6d9c2e9db2f5ed92525bea62377fffb3e8eb525b6fedeb8527befee890b57e86
8aa3e7097b888d2769b23b58b21
your waldo:  e983d6cc21a4c594466920f3ab9ed7e6a81de3dbc1df92d3b6f53490
```