Project 3 : Implement and attack plain RSA.

Key choices: e and d

Choosing key e :

```
# Totient
phi_n = (p - 1) * (q - 1)

# Choose e
# e has to be coprime with phi_n AND be 1 < e <= phi_n
while True:
    # Choosing a random int and checking if it is prime with totient. If it is, then stop loop.
    e = random.randrange(2 ** (1024 - 1), 2 ** 1024 - 1)
    if is_coprime(e, phi_n):
        break
```

```
def is_coprime(p, q):
    """
    Checks if 2 primes numbers are coprime
    :param p: first prime number
    :param q: second prime number
    :return: True or False
    """
    # Return True if math.gcd(p, q) == 1
    if math.gcd(p, q) == 1:
        return True
    else:
        return False
```

So, to find *e* the public key, I need to compute the totient phi_n which is the product of the subtraction of the primes minus 1. Then I will take a random integer between 2^{1024-1} to 2^{1024}-1 where 1024 is the key size.
I need to test if it is coprime with phi_n, which means that their GCD is 1. If it is, then I break the loop because I found *e*. Else, I pick another random value.

Choosing key d :

```
d = modinv(e, phi_n)
```

```
def get_gcd(nb1, nb2):
    """
    Using the Euclidian algorithm, it computes the GCD of 2 numbers
    :param nb1: first number
    :param nb2: second number
    :return: the GCD, x that will be used in the modinv() and y
    """
    s = 0
    x = 1
    t = 1
    y = 0
    r = nb2
    gcd = nb1

    while r != 0:
        quotient = gcd // r
        gcd, r = r, gcd - quotient * r
        x, s = s, x - quotient * s
        y, t = t, y - quotient * t

    return gcd, x, y


def modinv(nb1, nb2):
    """
    Reurns the modular invert
    :param nb1: public key
    :param nb2: phi(n)
    :return: private key
    """
    gcd, x, y = get_gcd(nb1, nb2)

    if x < 0:
        x += nb2

    return x
```

To find *d*, I need to compute the modular invert of e. To do so, as e is a big number, I apply the Euclidian algorithm to compute, with e and phi_n, the value I need.

Encryption and decryption time

Encryption time: 0.24615979194641113 second

Decryption time: 0.5050091743469238 second

Program execution time: 1.47080659866333 second

The decryption takes twice the time of the encryption. Both, they last 0.751168966293335.

With the chosen ciphertext attack, the program execution time takes the double of encryption + decryption time.