In [1]:

```python
import random
import sys
from typing import Callable, Tuple

sys.setrecursionlimit(100000)

from random import randrange

def gcd(a: int, b: int) -> int:
    """Calculates greatest common divisor of 2 numbers"""

    if (a == 0):
        return b
    return gcd(b % a, a)

def generate_relative_coprime(prime_to: int, get_candidate: Callable[[], int]) ->
 int:
    """
    Generates a relatevely prime number
    """

    commond_divisor = 0
    candidate = 0

    while commond_divisor != 1:
        candidate = get_candidate()
        commond_divisor = gcd(candidate, prime_to)

    return candidate

def get_relative_by_module(d, z):
    candidate = 0
    while (candidate * d) % z != 1:
        candidate = random.randrange(3, z, 2)

    return candidate

def generate_keypair(p: int, q: int) -> tuple:
    """
    Generates private and public keys using RSA algorithm.

    :param p: A prime number
    :param q: A prime number
    """

    n = p * q
    z = (p - 1) * (q - 1)

    d = generate_relative_coprime(z, lambda: random.randrange(3, z, 2))
    e = get_relative_by_module(d, z)

    return ((e, n), (d, n))

def encrypt(public_key: Tuple[int, int], plaintext: str) -> list:
    """
    Encrypts the message using public key
    """

    e, n = public_key
    cipher = [(ord(char) ** e) % n for char in plaintext]

    return cipher

def decrypt(private_key: Tuple[int, int], ciphertext: list) -> str:
    """
    Decrypts message using private key
    """

    d, n = private_key
```

```
    plain = [chr((char ** d) % n) for char in ciphertext]

    return ''.join(plain)

# A list of prime numbers
# [3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97
# ,101,103,107,109,113,127,131,137,139,149,151,157,163,167,173,179
# ,181,191,193,197,199,211,223,227,229,233,239,241,251,257,263,269
# ,271,277,281,283,293,307,311,313,317,331,337,347,349,353,359,367
# ,373,379,383,389,397,401,409,419,421,431,433,439,443,449,457,461
# ,463,467,479,487,491,499,503,509,521,523,541,547,557,563,569,571
# ,577,587,593,599,601,607,613,617,619,631,641,643,647,653,659,661
# ,673,677,683,691,701,709,719,727,733,739,743,751,757,761,769,773
# ,787,797,809,811,821,823,827,829,839,853,857,859,863,877,881,883
# ,887,907,911,919,929,937,941,947,953,967,971,977,983,991,997]

public, private = generate_keypair(859, 733)
print("Your public key is: ", public)
print("Your private key is: ", private)

message = 'KINO'

cipher = encrypt(public, message)

print("Your encrypted message is: ")
print(''.join(str(x) for x in cipher))
print("Your message is:")
print(decrypt(private, cipher))
```

```
Your public key is:  (367691, 629647)
Your private key is:  (470963, 629647)
Your encrypted message is:
14842664323462858349885
Your message is:
KINO
```