In [1]:

```
import random
from math import pow

a = random.randint(2, 10)

def gcd(a, b):
    if a < b:
        return gcd(b, a)
    elif a % b == 0:
        return b;
    else:
        return gcd(b, a % b)

 # Generating large random numbers
def gen_key(q):
    key = random.randint(pow(10, 20), q)
    while gcd(q, key) != 1:
        key = random.randint(pow(10, 20), q)
    return key

# Modular exponentiation
def power(a, b, c):
    x = 1
    y = a

    while b > 0:
        if b % 2 == 0:
            x = (x * y) % c;
        y = (y * y) % c
        b = int(b / 2)
    return x % c

# Asymmetric encryption
def encrypt(msg, q, h, g):
    en_msg = []

    k = gen_key(q)# Private key for sender
    s = power(h, k, q)
    p = power(g, k, q)

    for i in range(0, len(msg)):
        en_msg.append(msg[i])

    print("g^k used : ", p)
    print("g^ak used : ", s)
    for i in range(0, len(en_msg)):
        en_msg[i] = s * ord(en_msg[i])
    return en_msg, p

def decrypt(en_msg, p, key, q):
    dr_msg = []
    h = power(p, key, q)
    for i in range(0, len(en_msg)):
        dr_msg.append(chr(int(en_msg[i]/h)))
    return dr_msg

def main():
    msg = input("Enter message : ")  #encryption

    q = random.randint(pow(10, 20), pow(10, 50))
    g = random.randint(2, q)

    key = gen_key(q)# Private key for receiver
    h = power(g, key, q)
    print("g used : ", g)
    print("g^a used : ", h)
    en_msg, p = encrypt(msg, q, h, g)
    dr_msg = decrypt(en_msg, p, key, q)
    dmsg = ''.join(dr_msg)
    print("Decrypted Message :", dmsg);


if __name__ == '__main__':
    main()
```

```
Enter message : Hello I am Jessica
g used :  1608227613860619657641283723809633416913370623750
g^a used :  7619410708506500463246725465829658456557694395124
g^k used :  7201851107829805858139068237105810019127099370858
g^ak used :  4954892873456155794277013864868274523544693621502
Decrypted Message : Hello I am Jessica
```

In [ ]:

```
```