```python
def GCD(a: int, b: int) -> bool:
    if b == 0:
        return a == 1
    else:
        return GCD(b, a % b)

def modular_inverse(a: int, m: int) -> int:
    m0, t1, t2 = m, 0, 1
    if m == 1:
        return 0
    while a > 1:
        q = a // m
        m, a = a % m, m
        t1, t2 = t2 - q * t1, t1
    if t2 < 0:
        t2 += m0
    return t2

def RSA(p: int, q: int, msg: int):
    n = p * q

    phi = (p-1)*(q-1)

    e = 2
    while e < phi:
        if GCD(e, phi):
            break
        else:
            e += 1

    d = modular_inverse(e, phi)

    print(f'Public key  (e, n): {e, n}')
    print(f'Private key (d, n): {d, n}')

    print(f'Original message : {msg}')

    C = pow(msg, e, n)
    print(f'Encrypted message: {C}')

    M = pow(C, d, n)
    print(f'Decrypted message: {M}')


RSA(3, 11, 31)
```

OUTPUT:

```
Public key  (e, n): (3, 33)
Private key (d, n): (7, 33)
Original message : 31
Encrypted message: 25
Decrypted message: 31
```