RSA

In [5]:

```python
import random
import math

def generate_keypair(p, q):
    """Generate public and private keys using the given prime numbers p and q"""
    n = p * q
    phi = (p - 1) * (q - 1)

    # Find a number e such that 1 < e < phi and gcd(e, phi) = 1
    e = random.randint(2, phi - 1)
    while math.gcd(e, phi) != 1:
        e = random.randint(2, phi - 1)

    # Find the modular multiplicative inverse of e mod phi
    d = pow(e, -1, phi)

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

def encrypt(plaintext, public_key):
    """Encrypt the plaintext using the given public key"""
    e, n = public_key
    ciphertext = [pow(ord(c), e, n) for c in plaintext]
    return ciphertext

def decrypt(ciphertext, private_key):
    """Decrypt the ciphertext using the given private key"""
    d, n = private_key
    plaintext = ''.join([chr(pow(c, d, n)) for c in ciphertext])
    return plaintext

# Example usage
p = 13
q = 17
public_key, private_key = generate_keypair(p, q)
print("Public key:", public_key)
print("Private key:", private_key)

plaintext = "Hello, world!"
print("Original plaintext:", plaintext)

ciphertext = encrypt(plaintext, public_key)
print("Encrypted ciphertext:", ciphertext)

decrypted_plaintext = decrypt(ciphertext, private_key)
print("Decrypted plaintext:", decrypted_plaintext)
```

```
Public key: (77, 221)
Private key: (5, 221)
Original plaintext: Hello, world!
Encrypted ciphertext: [89, 186, 10, 10, 76, 96, 2, 136, 76, 82, 10, 172, 50]
Decrypted plaintext: Hello, world!
```