RSA

In [11]:
```
import random
import ast
```

In [12]:
```
def isPrime(n):
    if n==0 or n==1:
        return False

    for i in range(2, int(n**0.5)+1):
        if(n%i)==0:
            return False

    return True

def generate_prime():
    primes = [i for i in range(1, 999) if isPrime(i)]
    return random.choices(primes, k=2)
```

In [13]:
```
class RSA:

    def __init__(self, p, q):
        self.p = p
        self.q = q
        self.N = p * q
        self.product = (p-1) * (q-1)
        self.generateKey()

    def generateKey(self):

        for i in range(1, 999999):
            if(self.product % i !=0):
                self.E = i
                break

        for i in range(1, self.product-1):
            if((i*self.E) % self.product) == 1:
                self.D = i
                break

        print("Encryption Key : ", self.E)
        print("Decryption Key : ", self.D)

    def encrypt(self, text):
        ct = (int(text) ** self.E) % self.N

        return ct

    def decrypt(self, cipher):
        dt = (int(cipher) ** self.D) % self.N

        return dt

#     def encrypt(self, text):
#         pt = []
#         ct = []

#         for i in text:
#             pt.append(ord(i))

#         for i in pt:
#             ct.append((i**self.E)%self.N)

#         return ct


#     def decrypt(self, cipher):
#         dt = []

#         for i in cipher:
#             dt.append(chr((i**self.D)%self.N))

#         return ''.join(dt)
```

In [14]:
```
if __name__ == "__main__":

    p = int(input("Enter p : "))
    q = int(input("Enter q : "))

    rsa = RSA(p, q)

    plaintext = input("Enter Plaintext : ")

    ct = rsa.encrypt(plaintext)

    print("Ciphertext : ", ct)

    dt = rsa.decrypt(ct)

    print("Decrypted Text : ", dt)
```

```
Enter p : 17
Enter q : 29
Encryption Key :  3
Decryption Key :  299
Enter Plaintext : 10
Ciphertext :  14
Decrypted Text :  10
```

In [ ]: