

21BDS0340

Abhinav Dinesh Srivatsa

Cryptography and Network Security Lab

Digital Assignment – IV

Question 1

Code

```
def binary_list(n):
    binary = []
    while n > 0:
        binary.append(int(n % 2))
        n = int(n / 2)
    return binary

def exponentiate(m, n, mod):
    binary_power = binary_list(n)
    current_val = m % mod
    prod = 1
    for i in binary_power:
        if i == 1:
            prod = (prod * current_val) % mod
            current_val = (current_val * current_val) % mod
    return prod % mod

def factors(n):
    res = []
    i = 2
    while n != 1:
        if n % i == 0:
            n //= i
            res.append(i)
            continue
        i += 1
    return res

def totient(n):
    f = factors(n)
    t = 1
    count = 1
    for i in range(len(f) - 1):
        if f[i] == f[i + 1]:
            count += 1
        else:
            t *= f[i] ** count - f[i] ** (count - 1)
            count = 1
```

```

t *= f[len(f) - 1] ** count - f[len(f) - 1] ** (count - 1)
return t

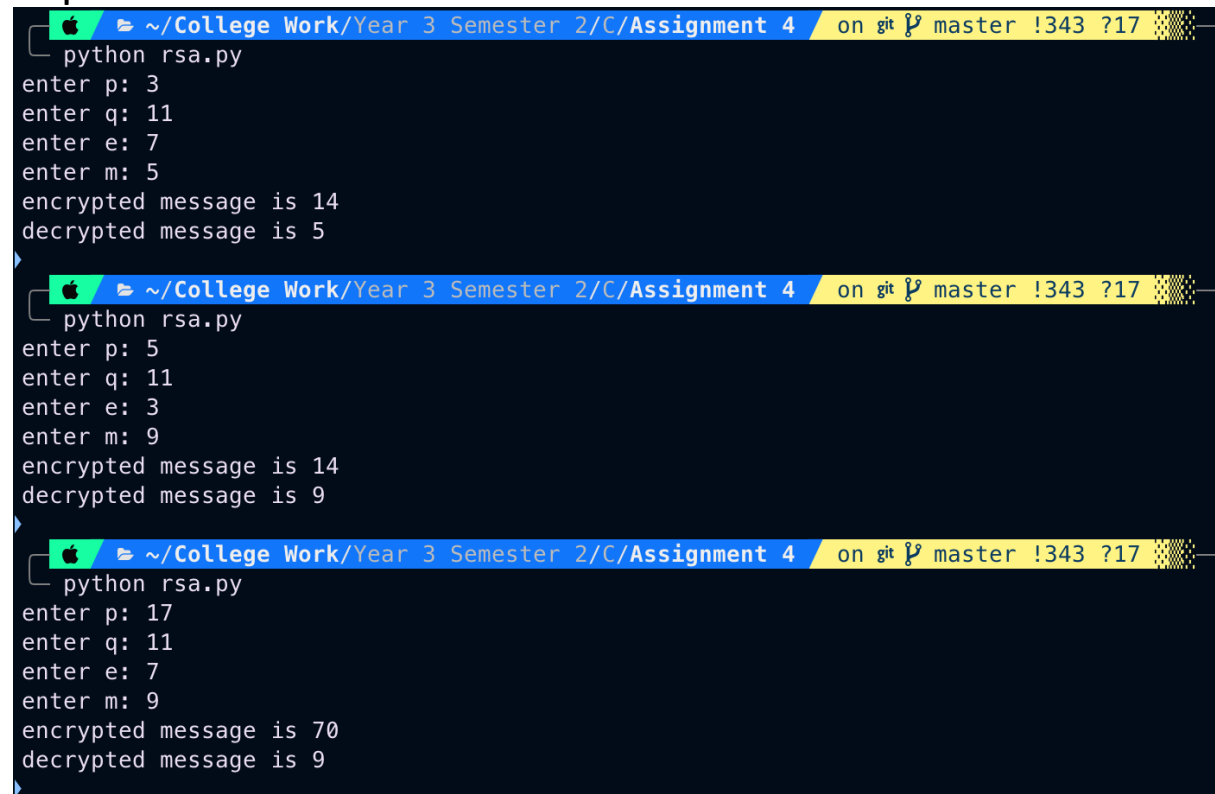
# key setup
p = int(input("enter p: "))
q = int(input("enter q: "))
e = int(input("enter e: "))
n = p * q
totient_n = (p - 1) * (q - 1)
d = exponentiate(e, totient(totient_n) - 1, totient_n)

# message encryption
m = int(input("enter m: "))
c = exponentiate(m, e, n)
print(f"encrypted message is {c}")

# message decryption
decrypted = exponentiate(c, d, n)
print(f"decrypted message is {decrypted}")

```

Output



```

~/College Work/Year 3 Semester 2/C/Assignment 4 on git master !343 ?17
python rsa.py
enter p: 3
enter q: 11
enter e: 7
enter m: 5
encrypted message is 14
decrypted message is 5

~/College Work/Year 3 Semester 2/C/Assignment 4 on git master !343 ?17
python rsa.py
enter p: 5
enter q: 11
enter e: 3
enter m: 9
encrypted message is 14
decrypted message is 9

~/College Work/Year 3 Semester 2/C/Assignment 4 on git master !343 ?17
python rsa.py
enter p: 17
enter q: 11
enter e: 7
enter m: 9
encrypted message is 70
decrypted message is 9

```

Question 2

Code

```
def binary_list(n):
    binary = []
    while n > 0:
        binary.append(int(n % 2))
        n = int(n / 2)
    return binary

def exponentiate(m, n, mod):
    binary_power = binary_list(n)
    current_val = m % mod
    prod = 1
    for i in binary_power:
        if i == 1:
            prod = (prod * current_val) % mod
            current_val = (current_val * current_val) % mod
    return prod % mod

def factors(n):
    res = []
    i = 2
    while n != 1:
        if n % i == 0:
            n //= i
            res.append(i)
            continue
        i += 1
    return res

def totient(n):
    f = factors(n)
    t = 1
    count = 1
    for i in range(len(f) - 1):
        if f[i] == f[i + 1]:
            count += 1
        else:
            t *= f[i] ** count - f[i] ** (count - 1)
            count = 1
    t *= f[len(f) - 1] ** count - f[len(f) - 1] ** (count - 1)
    return t

def inverse(a, n):
    return exponentiate(a, totient(n) - 1, n)

def add(x1, y1, x2, y2, a, mod):
```

```

    if x1 == x2 and y2 == y1:
        return double(x1, y1, a, mod)
    l = ((y2 - y1) * inverse(x2 - x1, mod)) % mod
    x3 = (l * l - x1 - x2) % mod
    y3 = (l * (x1 - x3) - y1) % mod
    return x3, y3

def double(x1, y1, a, mod):
    l = ((3 * x1 * x1 + a) * inverse(2 * y1, mod)) % mod
    x3 = (l * l - x1 - x1) % mod
    y3 = (l * (x1 - x3) - y1) % mod
    return x3, y3

def negate(x1, y1, mod):
    return x1, mod - y1

def multiply(x1, y1, c, a, mod):
    xcurr, ycurr = x1, y1
    for i in range(c):
        xcurr, ycurr = add(x1, y1, xcurr, ycurr, a, mod)
    return xcurr, ycurr

# key generation
q = int(input("enter q: "))
a = int(input("enter a: "))
b = int(input("enter b: "))
na = int(input("enter na: "))
gx = int(input("enter gx: "))
gy = int(input("enter gy: "))
pax, pay = multiply(gx, gy, na, a, q)

# encryption
nb = int(input("enter nb: "))
mx = int(input("enter mx: "))
my = int(input("enter my: "))
c1x, c1y = multiply(gx, gy, nb, a, q)
c2x, c2y = multiply(pax, pay, nb, a, q)
c2x, c2y = add(c2x, c2y, mx, my, a, q)

print(f"encrypted message is ({c1x}, {c1y}), ({c2x}, {c2y})")

# decryption
decx, decy = multiply(c1x, c1y, na, a, q)
decx, decy = negate(decx, decy, q)
decx, decy = add(c2x, c2y, decx, decy, a, q)

print(f"decrypted message is ({decx}, {decy})")

```

Output

```
python ecc.py
enter q: 67
enter a: 2
enter b: 3
enter na: 2
enter gx: 2
enter gy: 22
enter nb: 4
enter mx: 24
enter my: 26
encrypted message is ((1, 41), (51, 30))
decrypted message is (24, 26)

python ecc.py
enter q: 211
enter a: 0
enter b: -4
enter na: 5
enter gx: 2
enter gy: 2
enter nb: 9
enter mx: 45
enter my: 67
encrypted message is ((75, 90), (9, 150))
decrypted message is (45, 67)
```