DAY – 3 : CRYPTOGRAPHY AND NETWORK SECURITY

12) Write a high level code for RSA system, the public key of a given user is e = 31, n = 3599. What is the private key of this user?

```
# define public key
e = 31
n=3599

# find p and q using factoring function
def factorize(n):
    for i in range(2, n):
        if n % i == 0:
            return i, n//i

p, q = factorize(n)

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

# find d such that e*d ≡ 1 mod phi
def extended_gcd(a, b):
    if b == 0:
        return a, 1, 0
```

```
    else:
        d, x, y = extended_gcd(b, a % b)
        return d, y, x - (a // b) * y

d = extended_gcd(e, phi)[1]

# make sure d is positive
if d < 0:
    d += phi

# print private key
print("The private key is:", d)
```

RESULT :

The private key is: 3031

13) Write a high level code for set of blocks encoded with the RSA algorithm and we don’t have the private key. Assume n = pq, e is the public key. Suppose also someone tells us they know one of the plaintext blocks has a common factor with n. Does this help us in any way?

```
import math
```

```
# Function to find the greatest common divisor (GCD) of two numbers
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)

# RSA public key
n = 13437231
e = 37

# Set of RSA-encoded blocks
ciphertext = [2314317, 3553782, 2855405, 5279019, 8534406, 11797456, 4142155]

# Find p and q (the prime factors of n)
for i in range(2, int(math.sqrt(n))+1):
    if n % i == 0:
        p = i
        q = n // i
        break

# Find the ciphertext block with a common factor with n
```

```
for c in ciphertext:
    if gcd(c, n) != 1:
        common_factor = gcd(c, n)
        break

# Find the corresponding plaintext block
for c in ciphertext:
    if gcd(c, n) == common_factor:
        common_plaintext = pow(c, e, n) % common_factor
        break

# Find the factorization of the remaining part of n
if common_factor == p:
    remaining_factor = q
else:
    remaining_factor = p

# Decode the remaining ciphertext blocks
plaintext = []
for c in ciphertext:
    if gcd(c, n) == 1:
        plaintext.append(pow(c, e, n) // remaining_factor)

# Print the plaintext blocks
```

```
print("Decoded plaintext blocks:")
for p in plaintext:
    print(p)
```

RESULT :

```
Decoded plaintext blocks:
0
0
2
```