ICS 5 ECC

41323 Gayatri

In [1]:

```
P = 101

def modmul(a, b, m = P):

    return ((a % m) * (b % m)) % m


def mod_pow(a, b, m = P):
    if b==0:

        return 1

    r = mod_pow(a, b//2, m)
    r = (r * r) % m
    if b % 2 == 1:

        r = (r * a) % m
    return r

def get_positive(a, m = P):
    a = a % m
    a += m
    a = a % m
    return a

def moddiv(a, b, m = P):


    return modmul(a, mod_pow(b, m-2, m), m)


class Point:


    def __init__(self, x, y):

        self.x = x
        self.y = y
    def __eq__(self, p2):

        return self.x == p2.x and self.y == p2.y
    def __str__(self) -> str:

        return f"({self.x}, {self.y})"


class EllipticCurve:


    def __init__(self, a, b):

        self.a = a
        self.b = b

    def add(self, p1, p2, m = P):
        l = 0
        if p1 == p2:
            num = 3 * p1.x * p1.x + self.a
            den = 2 * p1.y
        else:
            num = p2.y - p1.y
            den = p2.x - p1.x
        l = moddiv(num, den, m)
        x3 = l*l - p1.x - p2.x
        y3 = l*(p1.x - x3) - p1.y
        x3 = get_positive(x3, m)
        y3 = get_positive(y3, m)
        return Point(x3, y3)

    def mul(self, k, p):
        while k != 1:

            p = self.add(p, p)
            k -= 1
        return p

    def sub(self, p1, p2):



        np = Point(p2.x, -p2.y)
        return self.add(p1, np)


curve = EllipticCurve(2, 4) # Points lieing on this:{0, 2}, {0, 5}, {1, 0}, {2, 3}, {2, 4},
G = Point(0, 2)



def encrypt(P, U):

    k = 5
    c = [
       curve.mul(k, G),
       curve.add(P, curve.mul(k, U))
    ]
    return c


def decrypt(C, R):

    p = curve.sub(C[1], curve.mul(R, C[0]))
    return p


R = 5 # Private key
U = curve.mul(R, G) # Public key


plaintext = Point(6, 1)

ciphertext = encrypt(plaintext, U)
p = decrypt(ciphertext, R)

print(p)

assert(p == plaintext)
if(p==plaintext):
    print("p is same as plain text.")
```

```
(6, 1)
p is same as plain text.
```

In [ ]:

```

```