11-9-2024

Session 2: Group operations on elliptic curves

Nombre: Álvarez García Brandon Azarael

Nombre de la materia: Selected topics in cryptography

Grupo: 7CM1

Nombre de la profesora: Dra. Sandra Díaz Santiago

Programming exercises

Please represent a point in an elliptic curve using three coordinates (x, y, z). The point at infinity $\mathcal{O} = (0, 1, 0)$. Any other point in the elliptic curve will be represented as $(x, y, 1)$, where $x, y \in \mathbb{Z}_p$. Use this representation to do the following exercises.

1. Design and implement a function that as input receives a, b and p, i.e. the parameters given for an elliptic curve $y^2 = x^3 + ax + b$ mód p, and a point $P = (x, y, z)$. Your function must return *true* if $P \in \mathbb{E}(a, b)$ otherwise your function must return *false*.

Para las funciones hacemos uso de la clase que cree en la practica pasada, con el fin de ahorrar código e implementar una mejoría en la clase, para esta función, simplemente evaluamos y^2 y comprobamos si está en las raíces, de ser así, devolveremos verdadero, de lo contrario falso

```
3 usages   BrandonAlvarez
def isPoint(self,a,b,p,x,y):

    y2= (math.pow(x,3)+a*x+b)%p
    sr = (math.pow(y,2)) % p

    if(y2 == sr):
        return True
    else:
        return False
```

2. Design a function that as input receives a, b and p, i.e. the parameters given for an elliptic curve $y^2 = x^3+ax+b$ mód p, and a points $P \in \mathbb{E}(a, b)$. The output must be $-P = (x, -y) = (x, -y \text{ mód } p)$

De la practica anterior, se creo una función para generar los puntos, esta misma se reutiliza, con la característica de invertir la coordenada 'y' y aplicar el modulo como se muestra en la formula (x, -y mod p).

La función devolverá los puntos de la recta, pero de forma negada, al inicio se agrega la representación del punto al infinito (0,1,0)

```
1 usage   BrandonAlvarez *
def negativePoints(self,qr,evaluate,p,z=1):

    points = list()
    points.append([0,1,0])

    for i, value in enumerate(evaluate):
        for j,qrValue in enumerate(qr):

            if value == qrValue:
                y = (j*-1) % p
                points.append([i, y,z])

    return points

```

3. Design and implement a function that as input receives a, b and p, i.e. the parameters given for an elliptic curve $y^2 = x^3 + ax + b$ mód p, and two points $P, Q \in \mathbb{E}(a, b)$. Your function must calculate the result of point addition $P + Q$.

Para esta función, esta dividida en dos partes, la primera consiste en la creación de una función para el calculo del coeficiente de Bezout, necesario para el calculo de S, ya que al realizarlo de forma directa, arrojaba pseudovalores.

```
def euclides_extendido(self,alpha, n):
    a, b = alpha, n
    x, y = 1, 0
    # Actualización hasta que b es 0
    while b != 0:
        q, r = divmod(a, b)
        a, b = b, r
        x, y = y, x - q * y
        # Si x es negativo después del algoritmo, se agrega n a x para asegurarse de que el resultado esté en el rango [0, n).
    if x < 0:
        x += n
    return x
```

Una vez teniendo esta función, ahora si se trabaja con la función de suma de puntos P+Q.

Primeramente, se tratan los casos especiales vistos en clase, para poder minimizar los errores a la hora del calculo de la suma de los puntos

```python
def sumPoints(self,a,b,p,x1,y1,x2,y2,z1=1,z2=1):

    #Special cases
    if x1 == 0 and y1 == 1 and z1 == 0:
        print(f'El resultado es de la suma es:  ({x2},{y2})')
        return

    elif x2 == 0 and y2 == 1 and z2 == 0:
        print(f'El resultado es de la suma es:  ({x1},{y1})')
        return

    elif z1 == 0 and z2 == 0:
        print(f'El resultado es el punto al infinito')
        return

    if self.isPoint(a, b, p, x1, y1) and self.isPoint(a, b, p, x2, y2):
        if x1 == x2 and y1 == y2:
            self.doublingPoint(a,b,p,x1,y1)
        else:
            #Calculate s for P != Q
            s = ((y2 - y1) * (self.euclides_extendido(x2 - x1, p))) % p
      if s == 0:
          print('El punto es el infinito')
      else:
          x3 = (math.pow(s, 2) - x1 - x2) % p
          y3 = (s * (x1 - x3) - y1) % p
          print(f'\nLa suma de los puntos ({x1},{y1}) y ({x2},{y2}) es : ({int(x3)},{int(y3)})')
else:
    print('Punto invalido')
```

Aquí se observa primero se tratan los casos especiales en donde se trata el punto al infinito, en caso de que no se trate ningún caso especial , se comprueba que ambos puntos pertenezcan a la recta, ya que para poder realizar esta operación, ambos deben pertenecer, si pertenecen ambos puntos, se procede a realizar el cálculo respectivo usando aquí el algoritmo de Euclides extendido, para nuestro coeficiente S, si este resulta 0, se trata del punto al infinito, ya que seria el inverso del punto que estamos tratando, y si ambos puntos son iguales solo mandamos a llamar la función designada para el doblado de puntos.

4. Design a function that as input receives a, b and p, i.e. the parameters given for an elliptic curve $y^2 = x^3 + ax + b$ mód p, and a point $P \in \mathbb{E}(a, b)$. The output must be $2P = P + P$

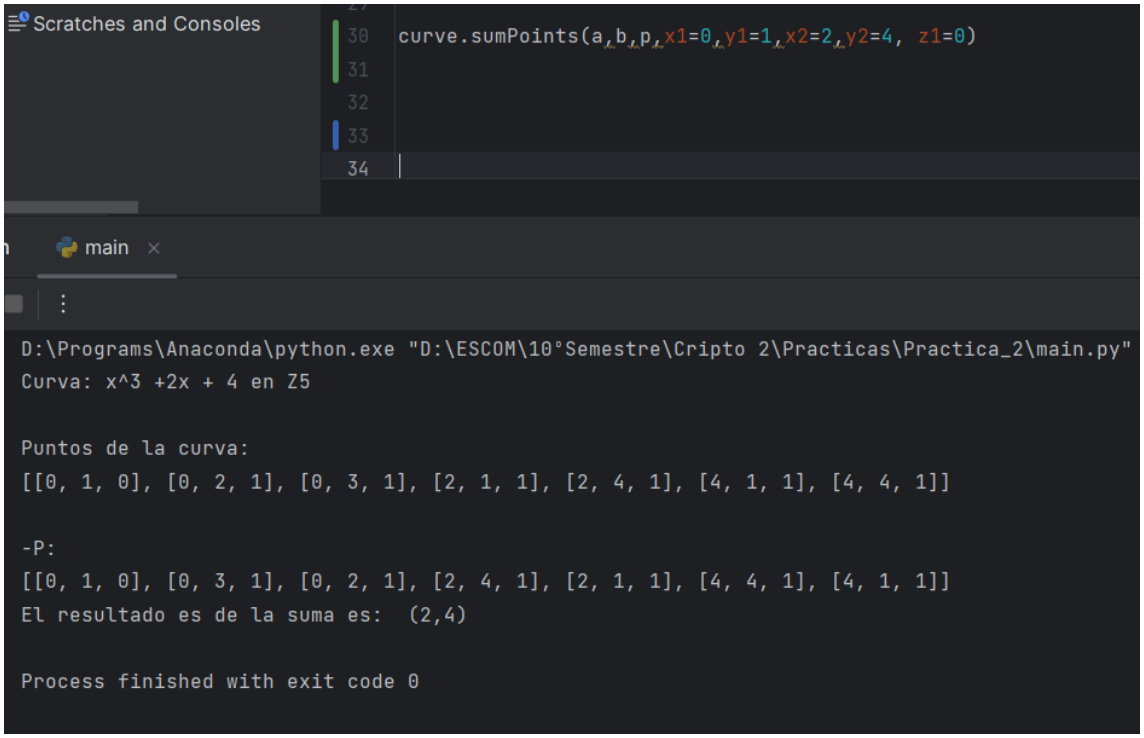
Para esta función resulta de forma mas sencilla ya que se trata de un solo punto, al igual que la función anterior, se comprueba el caso especial del punto al infinito y se compruebe que este punto pertenezca a la recta, de ser así simplemente volveríamos a realizar el cálculo, pero esta vez para P+P

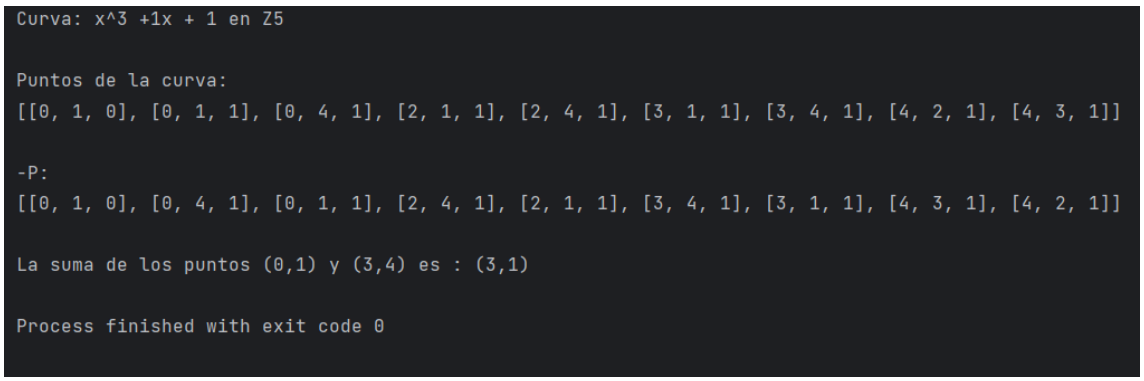
```python
    def doublingPoint(self,a,b,p,x1,y1,z1=1):

        if x1 == 0 and y1 == 1 and z1 == 0:
            print('Es el punto al infinito:  (inf,inf)')
            return

        if self.isPoint(a,b,p,x1,y1):

            #Calculate s when P = Q
            s = ((3 * math.pow(x1, 2) + a) * self.euclides_extendido(2 * y1, p)) % p
            if (s == 0):
                print("El punto resultante es el infinito")
            else:
                x3 = (math.pow(s, 2) - x1 - x1) % p
                y3 = (s * (x1 - x3) - y1) % p
                print(f"\nEl doblado del punto ({x1}, {y1}) es: ({int(x3)},{int(y3)})")

        else:
            print('El punto no pertenece a la curva')

```

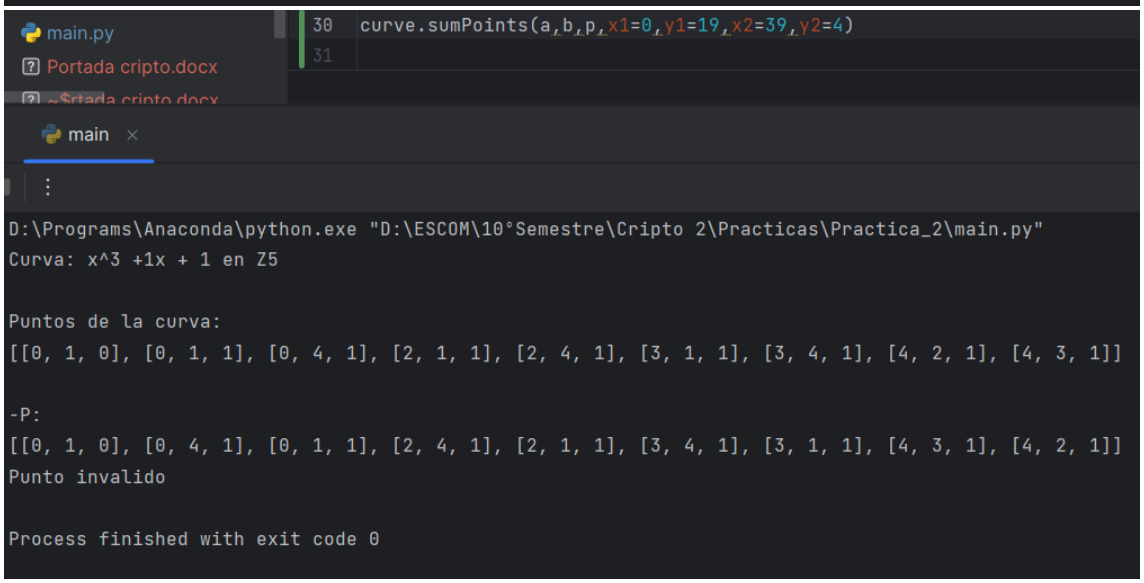
Pruebas

```
Curva: x^3 +2x + 4 en Z5

Puntos de la curva:
[[0, 1, 0], [0, 2, 1], [0, 3, 1], [2, 1, 1], [2, 4, 1], [4, 1, 1], [4, 4, 1]]

-P:
[[0, 1, 0], [0, 3, 1], [0, 2, 1], [2, 4, 1], [2, 1, 1], [4, 4, 1], [4, 1, 1]]

La suma de los puntos (0,2) y (2,4) es : (4,4)

Process finished with exit code 0
```

```
Curva: x^3 +2x + 4 en Z5

Puntos de la curva:
[[0, 1, 0], [0, 2, 1], [0, 3, 1], [2, 1, 1], [2, 4, 1], [4, 1, 1], [4, 4, 1]]

-P:
[[0, 1, 0], [0, 3, 1], [0, 2, 1], [2, 4, 1], [2, 1, 1], [4, 4, 1], [4, 1, 1]]

La suma de los puntos (0,2) y (4,4) es : (0,3)

Process finished with exit code 0
```


Scratches and Consoles
30 curve.sumPoints(a,b,p,x1=0,y1=1,x2=2,y2=4, z1=0)
31
32
33
34
main
D:\Programs\Anaconda\python.exe "D:\ESCOM\10°Semestre\Cripto 2\Practicas\Practica_2\main.py"
Curva: x^3 +2x + 4 en Z5
Puntos de la curva:
[[0, 1, 0], [0, 2, 1], [0, 3, 1], [2, 1, 1], [2, 4, 1], [4, 1, 1], [4, 4, 1]]
-P:
[[0, 1, 0], [0, 3, 1], [0, 2, 1], [2, 4, 1], [2, 1, 1], [4, 4, 1], [4, 1, 1]]
El resultado es de la suma es: (2,4)
Process finished with exit code 0


Curva: x^3 +1x + 1 en Z5
Puntos de la curva:
[[0, 1, 0], [0, 1, 1], [0, 4, 1], [2, 1, 1], [2, 4, 1], [3, 1, 1], [3, 4, 1], [4, 2, 1], [4, 3, 1]]
-P:
[[0, 1, 0], [0, 4, 1], [0, 1, 1], [2, 4, 1], [2, 1, 1], [3, 4, 1], [3, 1, 1], [4, 3, 1], [4, 2, 1]]
La suma de los puntos (0,1) y (3,4) es : (3,1)
Process finished with exit code 0


main.py
Portada cripto.docx
30 curve.sumPoints(a,b,p,x1=0,y1=19,x2=39,y2=4)
31
main
D:\Programs\Anaconda\python.exe "D:\ESCOM\10°Semestre\Cripto 2\Practicas\Practica_2\main.py"
Curva: x^3 +1x + 1 en Z5
Puntos de la curva:
[[0, 1, 0], [0, 1, 1], [0, 4, 1], [2, 1, 1], [2, 4, 1], [3, 1, 1], [3, 4, 1], [4, 2, 1], [4, 3, 1]]
-P:
[[0, 1, 0], [0, 4, 1], [0, 1, 1], [2, 4, 1], [2, 1, 1], [3, 4, 1], [3, 1, 1], [4, 3, 1], [4, 2, 1]]
Punto invalido
Process finished with exit code 0