MATH 470: Communications and Cryptography

Homework 10

Due date: 29 November 2023 *Name: Huy Lai*

Problem 1. Check that the points $P = (-1, 4)$ and $Q = (2, 5)$ are points on the elliptic curve $E : Y^2 = X^3 + 17$

Subproblem 1. Compute the points $P \oplus Q$ and $P \ominus Q$

Solution:

$$L : y - 4 = \frac{5-4}{2+1}(x+1) \to y = \frac{1}{3}x + \frac{13}{3}$$

$$\left(\frac{1}{3}x + \frac{13}{3}\right)^2 = x^3 + 17$$

$$\frac{1}{9}x^2 + \frac{26}{9}x + \frac{169}{9} = x^3 + 17$$

$$x^3 - \frac{1}{9}x^2 - \frac{26}{9}x - \frac{16}{9} = 0$$

$$(x+1)(x-2)\left(x + \frac{8}{9}\right) = 0$$

$$R_x = -\frac{8}{9}$$

We plug this into the line equation to find R_y

$$R_y = \frac{1}{3}R_x + \frac{13}{3}$$
$$= \frac{109}{27}$$

$$P \oplus Q = \left(-\frac{8}{9}, -\frac{109}{27}\right)$$

$$L : y - 4 = \frac{-5-4}{2+1}(x+1) \to y = -3x + 1$$

$$(-3x+1)^2 \equiv x^3 + 17$$

$$9x^2 - 6x + 1 \equiv x^3 + 17$$

$$x^3 - 9x^2 + 6x + 16 \equiv 0$$

$$(x+1)(x-2)(x-8) \equiv 0$$

$$R_x = 8$$

We plug this into the line equation to find R_y

$$R_y = -3R_x + 1$$
$$= -23$$

$$P \ominus Q = (8, 23)$$

Subproblem 2. Compute the points $2P$ and $2Q$

Solution:
We first find the implicit derivative

$$\begin{aligned} y^2 &= x^3 + 17 \\ 2yy' &= 3x^2 \\ y' &= \frac{3x^2}{2y} \end{aligned}$$

$$L : y - 4 = \frac{3(-1)^2}{2(4)}(x+1) \to y = \frac{3}{8}x + \frac{35}{8}$$

$$\left(\frac{3}{8}x + \frac{35}{8}\right)^2 = x^3 + 17$$

$$\begin{aligned} \frac{9}{64}x^2 + \frac{105}{32}x + \frac{1225}{64} &= x^3 + 17 \\ x^3 - \frac{9}{64}x^2 - \frac{105}{32} - \frac{137}{64} &= 0 \\ (x+1)^2\left(x - \frac{137}{64}\right) &= 0 \end{aligned}$$

$$R_x = \frac{137}{64}$$

We plug this into the line equation to find R_y

$$\begin{aligned} R_y &= \frac{3}{8}R_x + \frac{35}{8} \\ &= \frac{2651}{512} \end{aligned}$$

$$2P = \left(\frac{137}{64}, -\frac{2651}{512}\right)$$

$$L : y - 5 = \frac{3(2)^2}{2(5)}(x-2) \to y = \frac{6}{5}x + \frac{13}{5}$$

$$\left(\frac{6}{5}x + \frac{13}{5}\right)^2 = x^3 + 17$$

$$\begin{aligned} \frac{36}{25}x^2 + \frac{156}{25}x + \frac{169}{25} &= x^3 + 17 \\ x^3 - \frac{36}{25}x^2 - \frac{156}{25}x - \frac{256}{25} &= 0 \\ (x-2)^2\left(x + \frac{64}{25}\right) &= 0 \end{aligned}$$

$$R_x = -\frac{64}{25}$$

We plug this into the line equation to find R_y

$$\begin{aligned} R_y &= \frac{6}{5}R_x + \frac{13}{5} \\ &= -\frac{59}{125} \end{aligned}$$

$$2Q = \left(-\frac{64}{25}, \frac{59}{125}\right)$$

Problem 2. Make an addition table for E over $\mathbb{F}_p$, as we did in Table 6.1

Subproblem 1. $E : Y^2 = X^3 + X + 2$ over $\mathbb{F}_5$

Solution:
$E(\mathbb{F}_5) = \{\mathcal{O}, (1,2), (1,3), (4,0)\}$

	$\mathcal{O}$	$(1,2)$	$(1,3)$	$(4,0)$
$\mathcal{O}$	$\mathcal{O}$	$(1,2)$	$(1,3)$	$(4,0)$
$(1,2)$	$(1,2)$	$(4,0)$	$\mathcal{O}$	$(1,3)$
$(1,3)$	$(1,3)$	$\mathcal{O}$	$(4,0)$	$(1,2)$
$(4,0)$	$(4,0)$	$(1,3)$	$(1,2)$	$\mathcal{O}$

Table 1: Addition Table

Subproblem 2. $E : Y^2 = X^3 + 2X + 3$ over $\mathbb{F}_7$

Solution:
$E(\mathbb{F}_7) = \{\mathcal{O}, (2,1), (2,6), (3,1), (3,6), (6,0)\}$

	$\mathcal{O}$	$(2,1)$	$(2,6)$	$(3,1)$	$(3,6)$	$(6,0)$
$\mathcal{O}$	$\mathcal{O}$	$(1,2)$	$(1,3)$	$(4,0)$	$(3,6)$	$(6,0)$
$(2,1)$	$(2,1)$	$(3,6)$	$\mathcal{O}$	$(2,6)$	$(6,0)$	$(3,1)$
$(2,6)$	$(2,6)$	$\mathcal{O}$	$(3,1)$	$(6,0)$	$(2,1)$	$(3,6)$
$(3,1)$	$(3,1)$	$(2,6)$	$(6,0)$	$(3,6)$	$\mathcal{O}$	$(2,1)$
$(3,6)$	$(3,6)$	$(6,0)$	$(2,1)$	$\mathcal{O}$	$(3,1)$	$(2,6)$
$(6,0)$	$(6,0)$	$(3,1)$	$(3,6)$	$(2,1)$	$(2,6)$	$\mathcal{O}$

Table 2: Addition Table

Subproblem 3. $E : Y^2 = X^3 + 2X + 5$ over $\mathbb{F}_{11}$

Solution:
$E(\mathbb{F}_{11}) = \{\mathcal{O}, (0,4), (0,7), (3,4), (3,7), (4,0), (8,4), (8,7), (9,2), (9,9)\}$

	$\mathcal{O}$	$(0,4)$	$(0,7)$	$(3,4)$	$(3,7)$	$(4,0)$	$(8,4)$	$(8,7)$	$(9,2)$	$(9,9)$
$\mathcal{O}$	$\mathcal{O}$	$(0,4)$	$(0,7)$	$(3,4)$	$(3,7)$	$(4,0)$	$(8,4)$	$(8,7)$	$(9,2)$	$(9,9)$
$(0,4)$	$(0,4)$	$(9,2)$	$\mathcal{O}$	$(8,7)$	$(9,9)$	$(8,4)$	$(3,7)$	$(4,0)$	$(3,4)$	$(0,7)$
$(0,7)$	$(0,7)$	$\mathcal{O}$	$(9,9)$	$(9,2)$	$(8,4)$	$(8,7)$	$(4,0)$	$(3,4)$	$(0,4)$	$(3,7)$
$(3,4)$	$(3,4)$	$(8,7)$	$(9,2)$	$(8,4)$	$\mathcal{O}$	$(9,9)$	$(0,7)$	$(3,7)$	$(4,0)$	$(0,4)$
$(3,7)$	$(3,7)$	$(9,9)$	$(8,4)$	$\mathcal{O}$	$(8,7)$	$(9,2)$	$(3,4)$	$(0,4)$	$(0,7)$	$(4,0)$
$(4,0)$	$(4,0)$	$(8,4)$	$(8,7)$	$(9,9)$	$(9,2)$	$\mathcal{O}$	$(0,4)$	$(0,7)$	$(3,7)$	$(3,4)$
$(8,4)$	$(8,4)$	$(3,7)$	$(4,0)$	$(0,7)$	$(3,4)$	$(0,4)$	$(9,2)$	$\mathcal{O}$	$(9,9)$	$(8,7)$
$(8,7)$	$(8,7)$	$(4,0)$	$(3,4)$	$(3,7)$	$(0,4)$	$(0,7)$	$\mathcal{O}$	$(9,9)$	$(8,4)$	$(9,2)$
$(9,2)$	$(9,2)$	$(3,4)$	$(0,4)$	$(4,0)$	$(0,7)$	$(3,7)$	$(9,9)$	$(8,4)$	$(8,7)$	$\mathcal{O}$
$(9,9)$	$(9,9)$	$(0,7)$	$(3,7)$	$(0,4)$	$(4,0)$	$(3,4)$	$(8,7)$	$(9,2)$	$\mathcal{O}$	$(8,4)$

Table 3: Addition Table

The code to developed to calculate this table is included at the end of this document.

Problem 3. Let E be an elliptic curve over $\mathbb{F}_p$ and let P and Q be points in $E(\mathbb{F}_p)$. Assume that Q is a multiple of P and let $n_0 > 0$ be the smallest solution to $Q = nP$. Also let $s > 0$ be the smallest solution to $sP = \mathcal{O}$. Prove that every solution to $Q = nP$ looks like $n_0 + is$ for some $i \in \mathbb{Z}$. (Hint. Write n as $n = is + r$ for some $0 \leq r < s$ and determine the value of r.)

Solution:
Following the hint, we write $n = is + r$ for some $0 \leq r < s$.

$$\begin{aligned} Q &= nP \\ &= (is + r)P \\ &= i(sP) + rP \\ &= i\mathcal{O} + rP \\ &= rP \end{aligned}$$

Since $n_0 P$ is the smallest multiple of P that is equal to Q, $r \geq n_0$.
If $r = n_0$ we are done, so suppose instead that $r > n_0$.

Then

$$\begin{aligned} \mathcal{O} &= Q - Q \\ &= rP - n_0 P \\ &= (r - n_0)P \end{aligned}$$

We know that sP is the smallest (nonzero) multiple of P that is equal to $\mathcal{O}$, so $r - n_0 \geq s$.
However, this contradicts $r < s$.
Therefore $r = n_0$, which proves that $n = is + n_0$

Problem 4. Use the double-and-add algorithm (Table 6.3) to compute nP in $E(\mathbb{F}_p)$ for the following curves and points, as we did in Fig. 6.4.

$$E : Y^2 = X^3 + 1828X + 1675, p = 1999, P = (1756, 348), n = 11$$

Solution:
$11P = (1068, 1540)$

Step i	n	$Q = 2^i P$	R
0	11	$(1756, 348)$	$\mathcal{O}$
1	5	$(1526, 1612)$	$(1756, 348)$
2	2	$(1657, 1579)$	$(1362, 998)$
3	1	$(1849, 225)$	$(1362, 998)$
4	0	$(586, 959)$	$(1068, 1540)$

Table 4: Compute $n \cdot P$ on $E \mod p$

Problem 5. Alice and Bob agree to use elliptic Diffie–Hellman key exchange with the prime, elliptic curve, and point

$$p = 2671, E : Y^2 = X^2 + 171X + 853, P = (1980, 431) \in E(\mathbb{F}_{2671})$$

Subproblem 1. Alice sends Bob the point $Q_A = (2110, 543)$. Bob decides to use the secret multiplier $n_B = 1943$. What point should Bob send to Alice?

Solution:
Bob sends the point $Q_B = n_B P = 1943P = (1432, 667) \in E(\mathbb{F}_{2671})$ to Alice.

Subproblem 2. What is their secret shared value?

Solution:
Their secret shared value is the x-coordinate of the point

$$\begin{aligned} n_B Q_A &= 1943(2110, 543) \\ &= (2424, 911) \in E(\mathbb{F}_{2671}) \end{aligned}$$

Therefore, their shared secret is $x = 2424$

Problem 6. Use the elliptic curve factorization algorithm to factor each of the numbers N using the given elliptic curve E and point P.

Subproblem 1. $N = 589, E = Y^2 = X^3 + 4X + 9, P = (2, 5)$

Solution:

n	$n! \cdot P \mod N$
1	$P = (2, 5)$
2	$2! \cdot P = (564, 156)$
3	$3! \cdot P = (33, 460)$
4	$4! \cdot P = (489, 327)$

Table 5: Factor Algorithm

Subproblem 2. $N = 26167, E = Y^2 = X^3 + 4X + 128, P = (2, 12)$

Solution:

n	$n! \cdot P \mod N$
1	$P = (2, 12)$
2	$2! \cdot P = (23256, 1930)$
3	$3! \cdot P = (21778, 1960)$
4	$4! \cdot P = (22648, 14363)$
5	$5! \cdot P = (5589, 11497)$
6	$6! \cdot P = (7881, 16198)$

Table 6: Factor Algorithm

Subproblem 3. $N = 1386493, E = Y^2 = X^3 + 3X - 3, P = (1, 1)$

Solution:

n	$n! \cdot P \mod N$
1	$P = (1, 1)$
2	$2! \cdot P = (7, 1386474)$
3	$3! \cdot P = (1059434, 60521)$
4	$4! \cdot P = (81470, 109540)$
5	$5! \cdot P = (870956, 933849)$
6	$6! \cdot P = (703345, 474777)$
7	$7! \cdot P = (335675, 1342927)$
8	$8! \cdot P = (1075584, 337295)$
9	$9! \cdot P = (149824, 1003869)$
10	$10! \cdot P = (92756, 1156933)$

Table 7: Factor Algorithm

Subproblem 4. $N = 28102844557, E = Y^2 = X^3 + 18X - 453, P = (7, 4)$

Solution:

n	$n! \cdot P \mod N$
1	$1! \cdot P = (7, 4)$
2	$2! \cdot P = (1317321250, 11471660625)$
3	$3! \cdot P = (15776264786, 10303407105)$
4	$4! \cdot P = (27966589703, 26991329662)$
5	$5! \cdot P = (11450520276, 14900134804)$
6	$6! \cdot P = (4793277431, 9752445932)$
7	$7! \cdot P = (12906753800, 1428645019)$
8	$8! \cdot P = (1616742079, 27619685544)$
9	$9! \cdot P = (27320991957, 3646973433)$
10	$10! \cdot P = (2545790696, 7677839234)$
11	$11! \cdot P = (18170261705, 12466766800)$
12	$12! \cdot P = (4901920162, 11601259538)$
13	$13! \cdot P = (15079106517, 15513790103)$
14	$14! \cdot P = (5984263366, 23520062125)$
15	$15! \cdot P = (23472354261, 24075984511)$
16	$16! \cdot P = (24690955854, 627519762)$
17	$17! \cdot P = (19558800958, 2490325171)$
18	$18! \cdot P = (18404539321, 19844111656)$
19	$19! \cdot P = (21523159699, 7698818121)$
20	$20! \cdot P = (13876671025, 10706479792)$
21	$21! \cdot P = (21288501342, 22276398067)$
22	$22! \cdot P = (9646041624, 223733998)$
23	$23! \cdot P = (1704727047, 8275613638)$
24	$24! \cdot P = (25959867777, 9003083411)$
25	$25! \cdot P = (10400016599, 11715538594)$
26	$26! \cdot P = (22632202481, 6608272585)$
27	$27! \cdot P = (25446531195, 2223850203)$
28	$28! \cdot P = (12412875644, 7213676617)$

Table 8: Factor Algorithm

Problem 7. This exercise asks you to compute some numerical instances of the elliptic curve digital signature algorithm described in Table 6.7 for the public parameters

$$E : y^2 = x^3 + 231x + 473, p = 17389, q = 1321, G = (11259, 11278) \in E(\mathbb{F}_p)$$

You should begin by verifying that G is a point of order q in $E(\mathbb{F}_p)$

Subproblem 1. Samantha's private signing key is $s = 542$. What is her public verification key? What is her digital signature on the document $d = 644$ using the random element $e = 847$?

Solution:
Samantha's public verification key is

$$\begin{aligned} V &= sG \\ &= 542(11259, 11278) \\ &= (8689, 1726) \in E(\mathbb{F}_p) \end{aligned}$$

Her signature on $d = 644$ using $e = 874$ is obtained by first computing

$$eG = (8417, 8276) \in E(\mathbb{F}_p)$$

Then she calculates

$$s_1 = x(eG) \mod q = 491 \quad \text{and} \quad s_2 \equiv (d + ss_1)e^{-1} \equiv 290 \mod q$$

Subproblem 2. Tabitha's public verification key is $V = (11017, 14637)$. Is $(s_1, s_2) = (907, 296)$ a valid signature on the document $d = 993$?

Solution:
Victor computes

$$v_1 \equiv ds_2^{-1} \equiv 106 \mod q \quad \text{and} \quad v_2 \equiv s_1 s_2^{-1} \equiv 311 \mod q$$

Then

$$v_1 G + v_2 V = (8833, 4526) \in E(\mathbb{F}_p)$$

and

$$x(v_1 G + v_2 V) \mod q = 8833 \mod 1321 = 907$$

this is equal to to s_1, so the signature is valid.

Subproblem 3. Umberto's public verification key is $V = (14594, 308)$. Use any method that you want to find Umberto's private signing key, and then use the private key to forge his signature on the document $d = 516$ using the random element $e = 365$.

Solution:
To find Umberto's private signing key, we need to find an s such that

$$sG = V$$

Solving the Elliptic Curve Discrete Log Problem gives us that $s = 1294$

We can then forge a signature on the document $d = 516$ using the ephemeral key $e = 365$ by first computing $eg = (3923, 12121) \in E(\mathbb{F}_p)$ and then

$$s_1 = x(eG) \mod q = 1281 \quad \text{and} \quad s_2 \equiv (d + ss_1)e^{-1} \equiv 236 \mod q$$

To check that the signature is valid, we compute

$$v_1G + v_2V = (3923, 12121) \in E(\mathbb{F}_p)$$

and

$$x(v1G + v2V) \mod q = 3923 \mod 1321 = 1281$$

which is equal to s_1

```python
from math import inf, isinf


class Curve:
    a: int
    b: int
    N: int

    def __init__(self, a: int, b: int, N: int):
        assert (4 * a ** 3 + 27 * b * b) % N, "Degenerate elliptic curve"
        self.a = a % N
        self.b = b % N
        self.N = N

    def __str__(self) -> str:
        return f"y^2 = x^3 + {self.a}x + {self.b}"

    def __contains__(self, point: "Point") -> bool:
        x, y = point.x, point.y
        return pow(y, 2, self.N) == (x * x * x + self.a * x + self.b) % self.N

    def __eq__(self, other: "Curve") -> bool:
        if not isinstance(other, Curve):
            return False
        if self.N != other.N:
            return False
        return self.a == other.a and self.b == other.b

    def __hash__(self) -> int:
        return hash((self.a, self.b, self.N))


class Point:
    x: int
    y: int
    curve: Curve

    I: "Point" = None
```

```
    def __init__(self, x: float, y: float, curve: Curve):
        self.x = x % curve.N if not isinf(x) else inf
        self.y = y % curve.N if not isinf(x) else inf
        self.curve = curve

    def __str__(self) -> str:
        if isinf(self.x) and isinf(self.y):
            return "$\mathcal{O}$"
        return f"$({self.x}, {self.y})$"

    def __eq__(self, other) -> bool:
        if not isinstance(other, Point):
            return False

        if isinf(self.x) ^ isinf(other.x):
            return False
        if isinf(self.x) and isinf(other.x):
            return True

        if self.curve.N != other.curve.N:
            return False
        return self.x == other.x and self.y == other.y

    def __hash__(self) -> int:
        return hash((self.x, self.y, self.curve))

    def __add__(self, other: "Point") -> "Point":
        if self == Point.I:
            return other

        if other == Point.I:
            return self

        if self.x == other.x and not ((self.y + other.y) % self.curve.N):
            return Point.I

        if self.x != other.x:
```

```
            x1, y1 = self.x, self.y
            x2, y2 = other.x, other.y

            m: int = ((y2 - y1) * pow(x2 - x1, -1,
                      self.curve.N)) % self.curve.N
            x3: int = (m * m - x1 - x2) % self.curve.N
            y3: int = (m * (x1 - x3) - y1) % self.curve.N

            return Point(x3, y3, self.curve)

        else:
            x1, y1, a = self.x, self.y, self.curve.a

            m: int = ((3 * x1 * x1 + a) * pow(y1 << 1, -
                      1, self.curve.N)) % self.curve.N
            x3: int = (m * m - (x1 << 1)) % self.curve.N
            y3: int = (m * (x1 - x3) - y1) % self.curve.N

            return Point(x3, y3, self.curve)

    def __rmul__(self, scalar: int) -> "Point":
        current = self
        result = Point.I
        while scalar:
            if scalar & 1:
                result = result + current
            current = current + current
            scalar >>= 1
        return result

    def __lt__(self, other: "Point") -> bool:
        if self.x != other.x:
            return self.x < other.x
        return self.y < other.y


def legendre(a: int, p: int) -> int:
    return pow(a, (p - 1) >> 1, p)
```

```python
def tonelli(n: int, p: int) -> int:
    if not n:
        return 0
    assert legendre(n, p) == 1, "not a square (mod p)"
    q = p - 1
    s = 0
    while not (q & 1):
        q >>= 1
        s += 1
    if s == 1:
        return pow(n, (p + 1) >> 2, p)
    for z in range(2, p):
        if p - 1 == legendre(z, p):
            break
    c = pow(z, q, p)
    r = pow(n, (q + 1) >> 1, p)
    t = pow(n, q, p)
    m = s
    t2 = 0
    while (t - 1) % p:
        t2 = (t * t) % p
        for i in range(1, m):
            if not ((t2 - 1) % p):
                break
            t2 = (t2 * t2) % p
        b = pow(c, 1 << (m - i - 1), p)
        r = (r * b) % p
        c = (b * b) % p
        t = (t * c) % p
        m = i
    return r


def main():
    Point.I = Point(inf, inf, None)
    a, b, N = 2, 5, 11
```

```
    curve = Curve(a, b, N)
    qr = set([0])

    # calculate all qr's mod N
    for i in range(curve.N):
        if legendre(i, curve.N) == 1:
            qr.add(i)
    qr = sorted(qr)

    # find the x for all of these roots
    points = set()
    for i in range(N):
        x = (i * i * i + curve.a * i + curve.b) % curve.N
        print(f"i={i}, x={x}")
        if x in qr:
            r = tonelli(x, curve.N)
            points.add(Point(i, r, curve))
            points.add(Point(i, -1 * r % curve.N, curve))
    points = sorted(points)
    points.insert(0, Point.I)

    # the table
    for p1 in points:
        print(f"{p1}", end=" & ")
        for p2 in points:
            print(f"{p1 + p2}", end=" & ")
        print("\\\\ \hline")


if __name__ == "__main__":
    main()
```