CISS451/MATH451: Cryptography and Computer Security
Final Exam

The goal is to derive the formulas for the addition of points in a general elliptic curve:

$$E : y^2 = x^3 + ax^2 + bx + c$$

and implement an Python function to add points in an elliptic curve.

Write $\mathcal{O}$ for the point at infinity.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points on E. (Therefore P, Q are finite points.) We want to derive the addition formulas for R where $R = P + Q$. Let $R = (x_3, y_3)$.

Q1. CASE: $x_1 \neq x_2$ (therefore $P \neq Q$)

(a) Let L be the line through P and Q be

$$L : y = \lambda x + \nu$$

Derive λ and ν in terms of x_1, y_1, x_2, y_2.

SOLUTION

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\nu = y_1 - \left(\frac{y_2 - y_1}{x_2 - x_1} \cdot x_1 \right)$$

(Continuing Q1.)

(b) Let $R' = (x_3', y_3')$ be the third point of intersection of E and L. Therefore P, Q, R' must satisfy the equations of both E and L:

$$
\begin{aligned}
y^2 &= x^3 + ax^2 + bx + c \\
y &= \lambda x + \nu
\end{aligned}
$$

Through substitution, remove the variable y. You will obtain (of course) a cubic equation in x. Derive and write down this equation in the form

$$\text{cubic polynomial} = 0$$

This cubic polynomial of x will have a, b, c, λ, ν in its coefficients. (You need not substitute λ and ν with their expressions from (a).)

Of course the three solutions of x gives the three x–coordinates of P, Q, and R'.

SOLUTION.

$$0 = x^3 + ax^2 + bx + c - \lambda^2 x^2 - 2\lambda\nu - \nu^2$$

(Continuing Q1.)

(c) We know that the equation (1) from (b) is satisfied by the x–coordinates of P, Q and R'. Therefore we must have the following factorization:

$$\text{cubic polynomial} = c(x - x_1)(x - x_2)(x - x_3')$$

where the left hand side is the cubic polynomial from equation (b). Note that the cubic has leading coefficient 1, i.e. the coefficient of x^3 is 1. Therefore

$$\text{cubic polynomial} = (x - x_1)(x - x_2)(x - x_3')$$

(i.e. $c = 1$). Compute the coefficent of x^2 on the right of the above equation in terms of x_1, x_2, x_3'.

SOLUTION.

$$0 = a - \lambda^2$$

(Continuing Q1.)

(d) By equating the coefficients of x^2 of both sides of the equation in (c), derive x_3' in terms of the given data (i.e. the coefficient of E, the coefficients of L, and the coordinates of P and Q.)

(The expression will contain λ. You need not substitute λ with its expression from (a).)

SOLUTION.

$$x_3' = -x_2 - x_1 - a + \lambda^2$$

(e) In (d), you've derived x_3' which is the x–coordinate of R'. Note that R' is on L. By substituting x_3' in L, compute the y–coordinate of R', i.e. y_3'.

(The expression contains λ and ν. You need not substitute these with their expressions in (a).)

SOLUTION.

$$y_3' = \lambda \cdot (-x_2 - x_1 - a + \lambda^2) + \nu$$

(f) By the (geometric) definition of $P + Q$, the point R is the reflection of R' about the x–axis. Using (e), state the coordinates of R, i.e. x_3 and y_3.

(The expression contains λ and ν. You need not substitute these with their expressions in (a).)

SOLUTION.

$$R = ((-x_2 - x_1 - a + \lambda^2), -(\lambda \cdot (-x_2 - x_1 - a + \lambda^2) + \nu))$$

As a summary, you can now state one case of your theorem on the addition formulas for E:

Let E be the elliptic curve

$$E : y^2 = x^3 + ax^2 + bx + c$$

and

$$P = (x_1, y_1), \quad Q = (x_2, y_2), \quad x_1 \neq x_2$$

Then

$$P + Q = (x_3, y_3)$$

where

$$\begin{aligned}
\lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\
\nu &= y_1 - \left(\frac{y_2 - y_1}{x_2 - x_1} \cdot x_1 \right) \\
x_3 &= -x_2 - x_1 - a + \lambda^2 \\
y_3 &= \lambda \cdot (-x_2 - x_1 - a + \lambda^2) + \nu
\end{aligned}$$

ASIDE: Of course a good researcher *always* checks his/her work. First the following points

$$P = (3, 5)$$

and

$$Q = \left(\frac{129}{10^2}, \frac{383}{10^3}\right)$$

are on the elliptic curve

$$E : y^2 = x^3 - 2$$

Compute $P + Q$ using your formulas. Check that the point is on E.

Now for the next case.

Recall that

$$E : y^2 = x^3 + ax^2 + bx + c$$

and $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points on E. We want to derive the addition formulas for R where $R = P + Q$. Let $R = (x_3, y_3)$.

Q2. CASE: $x_1 = x_2$, $y_1 \neq y_2$.

What is R? [Just state it. No need to give the reason because I already talked about it in class.]

SOLUTION.

$$R = \mathcal{O}$$

Now for the third case. Again, let

$$E : y^2 = x^3 + ax^2 + bx + c$$

and let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points on E. We want to derive the addition formulas for R where $R = P + Q$. Let $R = (x_3, y_3)$.

We now handle the case of $x_1 = x_2$ and $y_1 = y_2$, i.e. we want to compute $2P$. We need to be careful since for this case the tangent line can be vertical. We first handle the case where the tangent line is not vertical.

Q3. CASE: $x_1 = x_2$, $y_1 = y_2$, $y_1 \neq 0$. (Note that in this case $P = Q$ and $R = P + P = 2P$.)

(a) Let L be the line

$$L : y = \lambda x + \nu$$

tangent to the curve E at point P. Derive λ and ν terms of x_1, y_1.

SOLUTION.

$$\lambda = \frac{3{x_1}^2 + 2ax_1 + b}{2y_1}$$

$$\nu = y_1 - \left(\frac{3{x_1}^2 + 2ax_1 + b}{2y_1} \cdot x_1\right)$$

(Continuing Q3.)

(b) Let $R' = (x'_3, y'_3)$ be the third point of intersection of E and L. Therefore $P, Q(=P), R'$ must satisfy the equations of both E and L.

$$y^2 = x^3 + ax^2 + bx + c$$
$$y = \lambda x + \nu$$

Through substitution remove the variable y. You will obtain (of course) an equation in x of the form:

$$\text{cubic polynomial} = 0$$

Derive and write down this equation.

Of course the solutions to the above equation gives the three x–coordinates of P, $Q(=P)$, and R'.

SOLUTION.

$$0 = x^3 + ax^2 + bx + c - \lambda^2 x^2 - 2\lambda\nu - \nu^2$$

(Continuing Q3.)

(c) We know that the equation from (b) is satisfied by the x–coordinates of P, $Q(= P)$ and R'. Therefore we must have the following factorization:

$$\text{cubic polynomial} = c(x - x_1)(x - x_1)(x - x_3')$$

where the left hand side is the cubic polynomial from equation (b). Note that the cubic has leading coefficient 1, i.e. the coefficient of x^3 is 1. Therefore

$$\text{cubic polynomial} = (x - x_1)(x - x_1)(x - x_3')$$

(i.e. $c = 1$). Compute the coefficent of x^2 on the right of the above equation.

SOLUTION.

$$0 = a - \lambda^2$$

(d) By equating the coefficients of x^2 of both sides of the equation in (c), derive x_3' in terms of the given data.

(The answer will contain λ. You need not replace λ with its expression from (a).)

SOLUTION.

$$x_3' = -2x_1 - a + \lambda^2$$

(e) In (d), you've derived x_3' which is the x–coordinate of R'. Note that R' is on L. By substituting x_3' in L, compute the y–coordinate of R', i.e. y_3'

(The answer will contain λ and ν. You need not replace λ and ν with their expressions from (a).)

SOLUTION.

$$y_3' = \lambda \cdot (-2x_1 - a + \lambda^2) + \nu$$

(f) By the (geometric) definition of $P + Q$ (when $Q = P$), the point R is the reflection of R' about the x–axis. Using (e), state the coordinates of R, i.e. x_3 and y_3.

(The answer will contain λ and ν. You need not replace λ and ν with their expressions from (a).)

$$R = ((-2x_1 - a + \lambda^2), -(\lambda \cdot (-2x_1 - a + \lambda^2) + \nu))$$

As a summary, you can now state your theorem on addition formulas for finite points on our elliptic curve.

Let E be the elliptic curve
$$E : y^2 = x^3 + ax^2 + bx + c$$
and
$$P = (x_1, y_1), \quad y_1 \neq 0$$
be a point on E. Then
$$2P = P + P = (x_3, y_3)$$
where
$$\begin{aligned}
\lambda &= \frac{3{x_1}^2 + 2ax_1 + b}{2y_1} \\
\nu &= y_1 - \left(\frac{3{x_1}^2 + 2ax_1 + b}{2y_1} \cdot x_1\right) \\
x_3 &= -2x_1 - a + \lambda^2 \\
y_3 &= -(\lambda \cdot (-2x_1 - a + \lambda^2) + \nu)
\end{aligned}$$

ASIDE: Again, you should *always* checks your work. First the following point

$$P = (3, 5)$$

is on the elliptic curve

$$E : y^2 = x^3 - 2$$

Compute $2P = P + P$ using your formulas and check that the point is on E.

Now for the fourth case where we double a point with vertical tangent line. Again, let

$$E : y^2 = x^3 + ax^2 + bx + c$$

and let $P = (x_1, y_1)$ be a point on E with $y_1 = 0$.

Q4. State $2P$ in this case.

[There's no need to explain since I have already mentioned this in class.]

SOLUTION.

$$R = \mathcal{O}$$

We have now handled all cases of adding finite points, including cases where the points are distinct and the resulting tangle line is vertical and the case of doubling a finite point with vertical tangent line.

The only cases left are additions where at least one point is the point at infinity $\mathcal{O}$.

All these cases are easy since by definition of the behavior of $\mathcal{O}$,

$$P + \mathcal{O} = P = \mathcal{O} + P$$

This includes the case of

$$\mathcal{O} + \mathcal{O} = \mathcal{O}$$

In terms of notation, instead of writing (x, y) for finite points and $\mathcal{O}$ for the point at infinity, we will also write finite points as

$$(x : y : 1)$$

and the point at infinity as

$$(0 : 1 : 0)$$

I don't want to go into details here, but we're basically viewing the elliptic curve in a *projective* space. A 2-d space when placed in a corresponding 2-d projective space will have 3 coordinates.

But even for computational reasons, the projective notation is helpful. Why? Because for Python we can use a list $[x, y, 1]$ for finite points and $[0, 1, 0]$ to represent the point at infinity.

With the + now defined on the *projective* curve E, i.e. E with the point at infinity 0, one can prove that the resulting points form a group with neutral element 0. In other words for P, Q, R in E (including the case where P or Q or R is $\mathcal{O}$),

- $P + Q$ is also a point of E (closure)
- $(P + Q) + R = P + (Q + R)$ (associativity)
- There is some P' such that $P + P' = \mathcal{O} = P' + P$. We usually write the inverse of P as $-P$ (inverse)
- $P + \mathcal{O} = P = \mathcal{O} + P$ (neutral)

Note that by definition, if the line through P and Q is vertical, then

$$P + Q = \mathcal{O} = Q + P$$

This implies that the inverse of P is the point that is vertically above or below P. Let $P = (x, y)$. Since we write the inverse of P as $-P$, we have justs shown that

$$-P = (x, -y)$$

i.e.

$$-(x, y) = (x, -y)$$

Q5. You are already given the Python code for $\mathbb{Z}/N\mathbb{Z}$ (the ring of $\mathbb{Z}$ mod N.)

Using a Python list to represent points, i.e.

```
[x, y, 1]
```

for finite points and

```
[0, 1, 0]
```

to present the point at infinity, implement a function to add points on any elliptic curve

$$E : y^2 = x^3 + ax^2 + bx + c$$

This is how your function should look like:

```
def add(E, N, P, Q):
    ...
```

The second parameter is a positive integer for the mod. For instance if we're interested in $\mathbb{Z}/23\mathbb{Z}$ points, then N is 23. The first parameter E is a list of a, b, c where the equation for E is

$$E : y^2 = x^3 + ax^2 + bx + c$$

The values a, b, c are $\mathbb{Z}/N\mathbb{Z}$ integers. For instance when we want to study $\mathbb{Z}/23\mathbb{Z}$ points on

$$E : y^2 = x(x-1)(x+1) = x^3 - x = x^3 + 0x^2 + (-1)x + 0$$

we have $a = 0, b = -1, c = 0$. In this case the first parameter E is

```
[ZN(0, 23), ZN(-1, 23), ZN(0, 23)]
```

For instance note that $P = (1, 0)$ is a point on E. Therefore to compute $2P$ I would call this:

```
N = 23
E = [ZN(0, N), ZN(-1, N), ZN(0, N)]
P = [ZN(1, N), ZN(0, N), ZN(1, N)]
twoP = add(E, N, P, P)
```

Note that the function must of course work with the point at infinity. For instance this should work:

```
N = 23
E = [ZN(0, N), ZN(-1, N), ZN(0, N)]
P = [ZN(1, N), ZN(0, N), ZN(1, N)]
O = [ZN(0, N), ZN(1, N), ZN(0, N)]
P_add_O = add(E, N, P, O)
```

You should have a folder containing this program which you should name `EC.py` and in the same folder you should have `ZN.py`. In your `EC.py` you should have on the few lines the following:

```
# Name: Brandy Poag
from ZN import *

def add(e, N, p, q):
    #print " e ", e

    a = e[0]
    b = e[1]
    c = e[2]
    #print "adding ", a, " ", b, " ", c
    x1= p[0]
    x2= q[0]
    y1= p[1]
    y2= q[1]
    #print "x1 ", x1, " y1 ", y1, " x2 ", x2, " y2 ", y2
    p_finite = p[2].data
    q_finite = q[2].data
    #print "p fin ", p_finite, "  q fin ", q_finite


    #check for infinite points
    if p_finite == 0:
        if q_finite == 0:
            return [ZN(0, N), ZN(1, N), ZN(0, N)]
        else:
            if (y2**2).data == (x2**3 + a*(x2**2) + b*x2 + c).data:
                return q
```

```
            else: return None
    elif q_finite == 0:
        if (y1**2).data == (x1**3 + a*(x1**2) + b*x1 + c).data:
            return p
        else: return None

    #make sure finite points are on the curve
    if not (y1**2 == x1**3 + a*(x1**2) + b*x1 + c) or \
       not (y2**2 == x2**3 + a*(x2**2) + b*x2 + c):
        print "print point not on curve"
        return None

    #handle finite points
    if not(x1 == x2):
        m = (y2 - y1) / (x2 - x1)
        #print "m  ", m
        c = y1 - (m * x1)
        #print "c  ", c
        x_3 = m**2 - x2 - x1 - a
        #print "x_3  ", x_3
        y_3 = m * x_3 + c
        #print "y_3  ", y_3
        return [ZN(x_3.data, N), ZN(-(y_3.data), N), ZN(1, N)]
    elif not(y1 == y2):
        return [ZN(0, N), ZN(1, N), ZN(0, N)]
    elif not(y1 == 0):
        m = (ZN(3, N) * x1**2 + ZN(2, N) * a*x1 + b)/(ZN(2, N) * y1)
        #print "m  ", m
        c = y1 - (m * x1)
        #print "c  ", c
        x_3 = m**2 - ZN(2, N)*x1 - a
        #print "x_3  ", x_3
        y_3 = m * x_3 + c
        #print "y_3  ", y_3
        return [ZN(x_3.data, N), ZN(-(y_3.data), N), ZN(1, N)]
    elif y1 == 0:
        return [ZN(0, N), ZN(1, N), ZN(0, N)]
    else:
        print "error with coordinates"
        return None

N = 7
```

```
E = [ZN(0, N), ZN(0, N), ZN(-2, N)]
#P = [ZN(1, N), ZN(0, N), ZN(1, N)]
#Q = [ZN(3, N), ZN(5, N), ZN(1, N)]
Q = [ZN(6, N), ZN(5, N), ZN(1, N)]
P = [ZN(3, N), ZN(2, N), ZN(1, N)]
O = [ZN(0, N), ZN(1, N), ZN(0, N)]

print "p: (", P[0].data,", ", P[1].data, ")"
print "q: (", Q[0].data,", ", Q[1].data, ")"
print "O: (", O[0].data,", ", O[1].data, ")"

print "P_add_P:"
P_add_P = add(E, N, P, P)
if P_add_P != None: print "[", P_add_P[0].data, " , ",
        P_add_P[1].data, " , ", P_add_P[2].data, "]"
else: print "no point returned"

print

print "Q_add_Q:"
Q_add_Q = add(E, N, Q, Q)
if Q_add_Q != None: print "[", Q_add_Q[0].data, " , ",
        Q_add_Q[1].data, " , ", Q_add_Q[2].data, "]"
else: print "no point returned"

print

print "P_add_Q:"
P_add_Q = add(E, N, P, Q)
if P_add_Q != None: print "[", P_add_Q[0].data, " , ",
        P_add_Q[1].data, " , ", P_add_Q[2].data, "]"
else: print "no point returned"

print

print "Q_add_P:"
Q_add_P = add(E, N, Q, P)
if Q_add_P != None: print "[", Q_add_P[0].data, " , ",
        Q_add_P[1].data, " , ", Q_add_P[2].data, "]"
else: print "no point returned"

print
```

```
print "P_add_O:"
P_add_O = add(E, N, P, O)
if P_add_O != None: print "[", P_add_O[0].data, " , ",
        P_add_O[1].data, " , ", P_add_O[2].data, "]"
else: print "no point returned"

print

print "O_add_P:"
O_add_P = add(E, N, O, P)
if O_add_P != None: print "[", O_add_P[0].data, " , ",
         O_add_P[1].data, " , ", O_add_P[2].data, "]"
else: print "no point returned"

print

print "O_add_O:"
O_add_O = add(E, N, O, O)
if O_add_O != None: print "[", O_add_O[0].data, " , ",
        O_add_O[1].data, " , ", O_add_O[2].data, "]"
else: print "no point returned"

print

print "Q_add_O:"
Q_add_O = add(E, N, Q, O)
if Q_add_O != None: print "[", Q_add_O[0].data, " , ",
        Q_add_O[1].data, " , ", Q_add_O[2].data, "]"
else: print "no point returned"

print

print "O_add_Q:"
O_add_Q = add(E, N, O, Q)
if O_add_Q != None: print "[", O_add_Q[0].data, " , ",
        O_add_Q[1].data, " , ", O_add_Q[2].data, "]"
else: print "no point returned"
```