Homework 5 CS 290G Cryptographic Engineering

Magnus Settemsli Mogstad

May 2015

Let the elliptic curve equation $y^2 = x^3$ 3x + 4
defined over the finite field GF(29)

1 Hasse's theorem

GF(29) gives us p=29
Hasses theorem gives us that p+1-2$\sqrt{p}$ $\leq$ order(ϵ) $\leq$ p+1+2$\sqrt{p}$
29+1-2$\sqrt{29}$ $\leq$ order(ϵ(3,4,29)) $\leq$ 29+1+2$\sqrt{29}$
Take the floor value of $2\sqrt{29} = 10$
20 $\leq$ order(ϵ(3,4,29)) $\leq$ 40

2 Elements of the elliptic curve by enumeration

x	u = x^3-3x+4	y^2 = u mod 29
0	4	(0,2),(0,-2)=(0,27)
1	2	solution does not exist
2	6	(2,8),(2,-8)=(2,21)
3	22	(3,14),(3,-14)=(3,15)
4	56	solution does not exist
5	114	solution does not exist
6	202	(6,12),(6,-12)=(6,17)
7	326	(7,6),(7,-6)=(7,23)
8	492	(8,12),(8,-12)=(8,17)
9	706	solution does not exist
10	974	solution does not exist
11	1302	solution does not exist
12	1696	solution does not exist
13	2162	(13,4),(13,-4)=(13,25)
14	2706	(14,3),(14,-3)=(14,26)
15	3334	(15,12),(15,-12)=(15,17)
16	4052	solution does not exist
17	4866	(17,9),(17,-9)=(17,20)
18	5782	solution does not exist
19	6806	(19,7),(19,-7)=(19,22)
20	7944	solution does not exist
21	9202	(21,3),(21,-3)=(21,26)
22	10 586	(22,1),(22-1)=(22,28)
23	12 102	(23,3),(23,-3)=(23,26)
24	13 756	solution does not exist
25	15 554	solution does not exist
26	17 502	solution does not exist
27	19 606	solution does not exist
28	21 872	(28,8),(28,-8)=(28,21)

Table 1: Elements by enumeration

3 Find the exact order of the group

(0,2),(0,27),(2,8),(2,21),(3,14),(3,15),(6,12),(6,17),(7,6),(7,23),(8,12),(8,17),(13,4),(13,25),(14,3), (14,26),(15,12),(15,17),(17,9),(17,20),(19,7),(19,22),(21,3),(21,26),(22,1),(22,28),(23,3),(23,26), (28,8),(28,21) and (∞,∞)
The number of elements here are 31 therefore the order of order(ϵ(3,4,29))= 31.
We can also see that hasses theorem is correct from task 1

4 Find a primitive element of the group

Since the group order is prime, all elements of the group is primitive elements.
P=(2,21)

5 Compute [15]P using the binary method

P $\rightarrow$ [2]P $\rightarrow$ [3]P $\rightarrow$ [6]P $\rightarrow$ [7]P $\rightarrow$ [14]P $\rightarrow$ [15]P P=(2,21)
since x_1=x_2 and y_1=y_2 we get that [2]P = P$\bigoplus$P
m=15*9 mod 29 = 19
x_3 = 361-2-2 mod 29 = 9
y_3 = 20
Thus we have (x_3,y_3) = (2,21)$\bigoplus$(2,21) = (9,20)

Now we perform an addition and we have $x_1 \neq x_2$ and $y_1 \neq y_2$
[3]P = P$\bigoplus$[2]P
m=(12-21)*$(9-2)^{-1}$ mod 29 = -9*25 mod 29 = 4
x_3 = 49-2-9 mod 29 = 5
y_3 = 7*(2-9)-21 mod 29 = 7*(-7)-21 mod 29 = 25
[3]P=(5,25)

Next step is to take the double again to find [6]P
m = 12
x = 18
y = 22
[6]P=(18,22)

Then another addition to find [7]P
m = 20
x = 3
y = 17
[7]P=(3,17)
Then another double to find [14]P
m = 6
x = 1
y = 24
[14]P = (1,24)
Then the last step to find [15]P is to do another addition
m = 26
x = 6
y = 20
which means that the point [15]P = (6,20)

6 Compute [15]P using the canonical recoding binary method

$\frac{P}{P}$ $\rightarrow$ [2]P $\rightarrow$ [4]P $\rightarrow$ [8]P $\rightarrow$ [16]P $\rightarrow$ [15]P
P=(2,21) and use doubling to get [2]P the results we get are
m = 19
x = 9
y = 20
which gives us [2]P=(9,20)
The use double again [4]P the result is
m = 25
x = 27
y = 23
which gives is [4]P=(27,23)
Then double again to find [8]P and results is
m = 6
x = 11
y = 15
which gives [8]P=(11,15)
Then double again to find [16]P the double gives us
m = 18
x = 12
y = 25
The result is [16]P=(12,25)
Then we do a negative addition to get [15]P and the result is
m = 22
x = 6
y = 20
The result [15]P=(6,20) which is equal to the result we got in task 6.

7 Python code to calculate the double and addition

To calculate the addition and double in the two tasks over I used this python code because I tried to do it manually but I made many mistakes so I made the script

```
def addition(x1,y1,x2,y2,n):
        m=((y2-y1)*modinv(x2-x1,n))%n
        x3 = ((m**2)-x1-x2)%n
        y3 = (m*(x1-x3)-y1)%n
        return m,x3,y3
```

```
def double(x1,y1,x2,y2,n,a):
        m=((3*(x1**2)+a)*modinv(2*y1,n))%n
        x3=((m**2)-x1-x2)%n
        y3=((m*(x1-x3))-y1)%n
        return m,x3,y3

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
        if a < 0:
                a=a%m

        g, x, y = egcd(a, m)
        if g != 1:
                raise Exception('modular inverse does not exist')
        else:
                return x % m

def runDouble():
        m,x3,y3 = double(11,15,11,15,29,3)
        print "M", m
        print "X3", x3
        print "Y3", y3

def runAddition():
        m,x3,y3 = addition(2,-21,12,25,29)
        print "M", m
        print "X3", x3
        print "Y3", y3

def runEx():
        m,x3,y3 = double(3,10,3,10,23,1)
        print "M", m
        print "X3", x3
        print "Y3", y3
```