Elliptic Curves Recitation Notes

1 Introduction

These are the notes for recitation 8 on elliptic curves. They are essentially the same as Prof. Rivest's notes from the following link:
http://courses.csail.mit.edu/6.857/2016/files/L13-groups-DH-key-exchange-elliptic-curves.pdf

2 Definition of Elliptic Curves

Let p be a prime number and let a and b be elements of Z_p such that $4a^3 + 27b^2 \neq 0$ (mod p) (*).
The equation (where x,y elements of Z_p) $y^2 = x^3 + ax + b$(mod p) (**) defines an algebraic curve.
If point (x,y) belongs on the curve, then point (x,-y) also belongs on the curve. Also, if r_1, r_2, r_3 are roots of the equation then it is true that:
$[(r_1 - r_2)(r_2 - r_3)(r_3 - r_1)]^2 = -(4a^3 + 27b^2)$ which from the condition (*) means that the roots are distinct.

Definition 1. *The points on the curve (**) are:*
E=$\{(x, y) : y^2 = x^3 + ax + b(mod\ p)\} \cup \{\infty\}$. *Here "$\infty$" denotes the "point at infinity".*

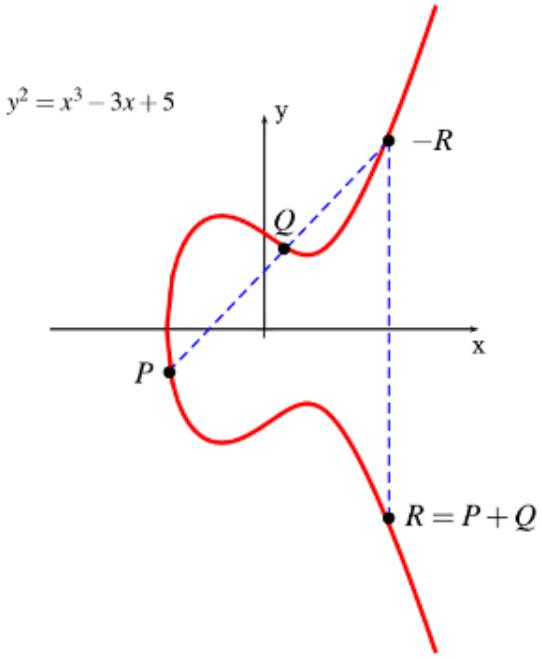
Fact 1. $|E| = p + 1 + t$ *where* $|t| \leq 2\sqrt{p}$

Fact 2. $|E|$ *can be computed efficiently.*

Fact 3. *A binary operation "+" can be defined on E such that (E,+) is a finite abelian group. In this group* ∞ *is the identity element (P+*∞*=P). The inverse of (x,y) is (x,-y) (which as we said also belongs in the curve). The inverse of* ∞ *is* ∞ *itself.*

3 Operations in Elliptic Curves

Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and $R = P + Q = (x_3, y_3)$. Intuitively P and Q define a line. Let -R be the third point in the curve on this line. Then the symmetric R is defined to be P+Q.



```
09:14:47 notes $ /Applications/sage/sage
Detected SAGE64 flag
Building Sage on OS X in 64-bit mode
----------------------------------------------------------------------
| Sage Version 4.6.2, Release Date: 2011-02-25                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
sage: # some experiments with elliptic curves with sage
sage: # first define a field mod 101
sage: F = Zmod(101)
sage: F
Ring of integers modulo 101
sage: # example of multiplication in F
sage: F(10)*F(11)
9
sage: # define elliptic curve over F
sage: E = EllipticCurve(F,[0,1])
sage: E
Elliptic Curve defined by y^2 = x^3 + 1 over Ring of integers modulo 101
sage: P = E.random_point()
sage: P
(96 : 49 : 1)
sage: # note coordinates are in projective form (X : Y : Z) representing
sage: # point x = X/Y, y = Y/Z, with Z = 0 for point at infinity.
sage: # get another point
sage: Q = E.random_point()
sage: Q
(29 : 94 : 1)
sage: P+Q
(21 : 77 : 1)
sage: # check commutativity
sage: Q+P
(21 : 77 : 1)
sage: # get third point
sage: R = E.random_point()
sage: R
(76 : 58 : 1)
sage: # check associativity
sage: P + (Q+R)
(53 : 2 : 1)
sage: (P+Q)+R
(53 : 2 : 1)
sage: # find size of this group
sage: E.order()
102
sage: # what are factors of 102?
sage: factor(102)
2 * 3 * 17
sage: # so possible orders of elements are 1,2,3,6,17,34,51,102
sage: P.order()
51
sage: Q.order()
51
sage: R.order()
51
sage: # none of P,Q, R are a generator (i.e. have order 102)
sage: # let's find one
sage: R = E.random_point()
```

```
sage: R.order()
102
sage: # bingo
sage: # what does identity look like?
sage: P-P
(0 : 1 : 0)
sage: I = P-P
sage: I
(0 : 1 : 0)
sage: I+P
(96 : 49 : 1)
sage: P+I
(96 : 49 : 1)
sage: -P
(96 : 52 : 1)
sage: # note that inverses just negate Y component, modulo 101
sage: # look at some small powers of generator R
sage: for i in range(15): print i, i*R
....:
0 (0 : 1 : 0)
1 (72 : 85 : 1)
2 (15 : 89 : 1)
3 (9 : 86 : 1)
4 (84 : 21 : 1)
5 (52 : 44 : 1)
6 (87 : 61 : 1)
7 (90 : 65 : 1)
8 (35 : 31 : 1)
9 (10 : 71 : 1)
10 (18 : 51 : 1)
11 (93 : 14 : 1)
12 (4 : 41 : 1)
13 (38 : 38 : 1)
14 (76 : 58 : 1)
sage: # find discrete log of P, base R
sage: R.discrete_log(P)
80
sage: 80*R
(96 : 49 : 1)
sage: 80*R==P
True
sage: # find discrete log of Q, base R
sage: R.discrete_log(Q)
58
sage: # find elements of each possible order
sage: R.order()
102
sage: S = 2*R
sage: S.order()
51
sage: S = 3*R
sage: S.order()
34
sage: S = 6*R
sage: S.order()
17
sage: S = 17*R
sage: S.order()
```

```
6
sage: S = 34*R
sage: S.order()
3
sage: S = 51*R
sage: S.order()
2
sage: S
(100 : 0 : 1)
sage:
```