PARI/GP PROGRAMMING: POLLARD RHO ALGORITHM

BILL ALLOMBERT AND KARIM BELABAS

A text version of the list of commands to enter is available at
`http://ecc2015.math.u-bordeaux.fr/documents/ecc_rho.txt`

1. Pollard rho

The exercise is to implement a simple version of Pollard rho to solve the discrete logarithm problem on elliptic curves. Let E be an elliptic curve, P and Q to points, we want to find e such that $P = eQ$, assuming it exists. The idea is to find a point that can be written in two ways as a linear combination of P and Q:

$$X = a_1P + b_1Q \tag{1}$$
$$X = a_2P + b_2Q \tag{2}$$
(3)

and to solve the system modulo the order of Q. To find such collision, we use Floyd algorithm.

We need four ingredients:

1.1. A "random" function.

We need a function `rnd` that take a point on E and return either 0, 1 or 2, which hopefully behave like a random function. (Use `a.pol` to get a representative of the field element a in $\mathbb{Z}[X]$).

1.2. The ρ function.

We need a function `rho` that take $[X, a, b]$ such that $X = aP + bQ$, compute `h=rnd(X)` and return:

- if $h = 0$, return $[X + P, a + 1, b]$
- if $h = 1$, return $[X + Q, a, b + 1]$
- if $h = 2$, return $[2X, 2a, 2b]$

(The new triple still satisfies $X = aP + bQ$)

1.3. The Floyd algorithm.

The idea is to compute two sequences of points X_n and $Y_n = X_{2n}$ by recursion, such that $X_0 = P$ and $X_{n+1} = \rho(X_n)$, until we find n such that $X_n = X_{2n}$, while keeping track of a_n and b_n such that $X_n = a_nP + b_nQ$.

1.4. The discrete logarithm. Assuming $a_n \neq a_{2n}$, we can solve

$$a_n P + b_n Q = a_{2n} P + b_{2n} Q \tag{4}$$

to find e

1.5. Improvement. We can also stop if $X_n = -X_{2n}$ and solve a slightly different equation.

2. Solution

2.1. a random function.

```
rnd(P)=subst(P[1].pol,variable(P[1].pol),2)%3;
```

2.2. the rho function.

```
rho(E,P,Q,V)=
{
  my(X,a,b,h);
  [X,a,b] = V;
  h = rnd(X);
  if(h==0,[elladd(E,X,P),a+1,b],
     h==1,[elladd(E,X,Q),a,b+1],
          [ellmul(E,X,2),2*a,2*b]);
}
```

2.3. The Floyd algorithm.

```
floyd(E,P,Q)=
{
  my(X1,X2);
  X1=[P,1,0]; X2=[P,1,0];
  until(X1[1]==X2[1],
    X1=rho(E,P,Q,X1);
    X2=rho(E,P,Q,rho(E,P,Q,X2)));
  [X1,X2];
}
```

2.4. The discrete logarithm.

```
dlog(E,P,Q,o)=
{
  my(X1,X2);
  [X1,X2]=floyd(E,P,Q);
  -(X1[3]-X2[3])/(X1[2]-X2[2])%o;
}
```

2.5. Tests.

```
test(N)=
{
  p =randomprime(N); a =ffgen(p);
  until(isprime(ellcard(E)), E = ellinit([1,random(p)],a));
  Q = ellgenerators(E)[1];
  o = ellgroup(E)[1];
  e = random(ellcard(E));
  P = ellmul(E,Q,e);
}
test(2^35);
dlog(E,P,Q,o)
##
elllog(E,P,Q,o)
##
```