ALGORITHM AND PROGRAM TO DETERMINE THE ORDER OF A POINT ON AN ELLIPTIC CURVE

DOCHIŢOIU, Ştefan
Hyperion University,
stefan.dochitoiu@yahoo.ca

Abstract

This work presents an iterative algorithm and a program for obtaining the order of a point on an elliptic curve over a finite field. Details are given based on the Mathematica integrated development environment.

Key-words: *cyclic group, order of points, elliptic curve*

AMS classification: 12E30

1. **Introduction**

Let K be a field. An elliptic curve over K is the set of points from K×K each of them verifying an equation of form:

$$y^2=x^3+a\,x+b, \qquad (1)$$

where the parameters $a, b\in$ K are such as the polynomial x^3+ax+b does not allow multiple roots. This field has an abelian group structure (in additive notation).

The aim of this work is to elaborate an iterative algorithm and program to determinate the order of an arbitrary element of the elliptic curve case of K=Z$_p$. The results of the program for this case apply in the domain of cryptography. First we will present certain aspects regarding elliptic curves in the plane R^2.

2. **Case of** K=R

In case K=R, the set of the points defined by the equation (1) has this graphic:

The composition operation (addition) of the curve points is done like this: given two points P_1 and P_2 on the curve, the straight determined by $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$ join the curve for the third time in point $P(x_3,y_3)$. The composition P_1+P_2 of those two points is by definition the point $P_3(x_3, -y_3)$, the symmetric of P against the horizontal axis.

The point of null effect of the curve is a conventional point, which is not expressible in coordinates and named the point from infinite, notated with O, to the vertical axis direction. By convention, the composition with this point is: $O+P=P$, whatever is the point P on the curve. The opposite $-P$ of the point $P(x, y)$ of the curve is the symmetric of P against the horizontal axis (having the coordinates x and $-y$), which is located on the curve as well.

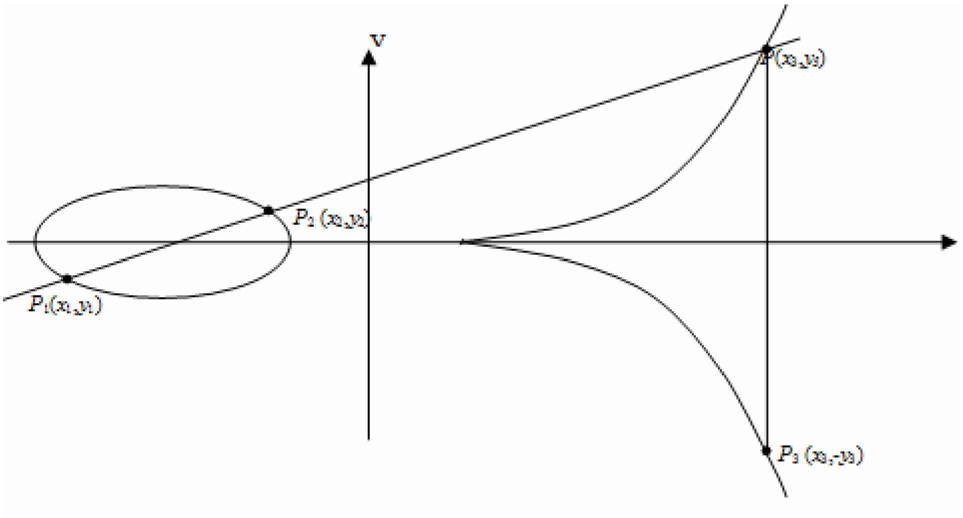

Fig. 1.

We will deduce the calculus formulas for the coordinates x_3 and $-y_3$ of the point P_1+P_2 function of the coordinates x_1, y_1, respectively x_2, y_2 of the points P_1 and P_2. For the beginning, we consider the case when x_1 is not equal with x_2. The straight line determinated by the points P_1 and P_2 has the equation:

$$y = x\frac{y_2 - y_1}{x_2 - x_1} - x_1\frac{y_2 - y_1}{x_2 - x_1} + y_1 .$$

We replace y in equation (1) and obtain the three degree equation in x:

$$\left(x\frac{y_2 - y_1}{x_2 - x_1} - x_1\frac{y_2 - y_1}{x_2 - x_1} + y_1 \right)^2 = x^3 + ax + b$$

The coefficient of x^2 is: $\left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2$ and represents the sum of the three roots (according to the Viète relations), two of those being known, namely as x_1 and x_2.

So,

$$\left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 = x_1 + x_2 + x_3 \Rightarrow x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2$$

and

$$y_3 = x_3 \frac{y_2 - y_1}{x_2 - x_1} - x_1 \frac{y_2 - y_1}{x_2 - x_1} + y_1 .$$

Thus the coordinates of the point P_1+P_2 are:

$$\begin{cases} x_3 = \left(\dfrac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2 \\ - y_3 = -y_1 - \dfrac{y_2 - y_1}{x_2 - x_1} (x_3 - x_1) \end{cases} \tag{2}$$

If the points P_1 and P_2 are identical (we denote P_0=P_1=P_2) then the tangent to the curve in the point P_0 should be in place of the straight P_1P_2. Taking into account that the curve equation is written on the form: F(x, y) = 0 (in which $F(x, y) = y^2 - x^3 - a\,x - b$), the tangent in point P_0 has the following equation:

$$F_{x_0}^{'} (x - x_0) + F_{y_0}^{'} (y - y_0) = 0 .$$

where, $F_{x_0}^{'} = \frac{\partial}{\partial x} F(x_0, y_0), F_{y_0}^{'} = \frac{\partial}{\partial y} F(x_0, y_0)$

Thus we obtain the tangent to the curve in point P_0 equation, $(-3x_0^2 - a)(x - x_0) + 2y_0(y - y_0) = 0$ with $y_0 \neq 0$ those intersection with the curve is obtained replacing $y = y_0 + \frac{3x_0^2 + a}{2y_0}(x - x_0)$ into the curve equation.

This way we obtain the three degree equation:

$$\left(y_0 + \frac{3x_0^2 + a}{2y_0}(x - x_0) \right)^2 - x^3 - a\,x - b = 0$$

with two known roots, namely $x_1 = x_2 = x_0$. Using again the first Viète formula one obtains: $2x_0 + x' = \left(\frac{3x_0^2 + a}{2y_0} \right)^2$, which gives $x' = \left(\frac{3x_0^2 + a}{2y_0} \right)^2 - 2x_0$, where we denoted x' the abscise of the point $2P_0$=P_0+P_0. The ordinate y' will be $y' = y_0 + \frac{3x_0^2 + a}{2y_0}(x' - x_0)$

So,

$$\begin{cases} x' = \dfrac{(3x_0^2+a)^2}{4y^2} - 2x_0 \\ y' = -y_0 - \dfrac{(3x_0^2+a)^2}{4y^2}(x_3 - x_0) \end{cases} . \qquad (3)$$

For every natural non-zero number k we denote P_k=k P_0. The point P_1 has the coordinates $x_1 = x_0$, $y_1 = y_0$, P_2 have the coordinates $x_2 = x'$, $y_2 = y'$. For $k \geq 3$ the following recurrent formulas are to be used:

$$\begin{cases} x_{n+1} = \left(\dfrac{y_n - y_0}{x_n - x_0}\right)^2 - x_n - x_0 \\ y_{n+1} = -y_0 - \dfrac{y_n - y_0}{x_n - x_0}(x_{n+1} - x_0) \end{cases} , \ \forall n = 2,...,k-1. \qquad (4)$$

At the moment when $x_n = x_0$ is obtained the recurrent formulas (4) could not be applied, but, yet, we obtained the order of P_0; namely it is n+1. Indeed, if x_n=x_0 then y_n=–y_0 and then P_n=n P_0=–P_0. That means (n+1)P_0 is the null element of the elliptic curve. Certainly n+1 is the smallest natural number with this property, so n+1 is the order of P_0.

3. **The supplied program for calculating the order of a curve point *p_0* (case of K=Z$_p$)**

For resolving the problem I used the programming IDE Mathematica – version 4.0. Now we continue by presenting the source code. Some comments at the source code referring to the functions to be used are included.

`/*COMMENT*/` is a markup tag that I used it here to demark the comments. The syntax of Mathematica 4.0 does not allow this markup tag but I used it here in the paper to give explanations at certain places into the source code.

/*

HERE IT IS A SUMARY FOR THE FUNCTIONS SUPPLIED BY THE MATHEMATICA IDE, NAMELY THOSE FUNCTIONS THAT OCCUR IN MY SOURCE CODE

(These functions haven't here a complete presentation but I tried to cover the needs of present work).

– Mod[*a,b*] Function taking two integer arguments (*a* and *b*). It returns the rest to the division of *a* by *b*.

– FullSimplify[*expression*] It searches in a set as large as possible of expressions that are equivalent with *expression*, the simplest expression and returns that expression.

– (++ and --) Increments, respectively decrements, an integer value.

– Sort[*list*] Sorts the list *list* according to the canonical order.

– Table[*EPRESSIONoFi,{i,k}*] Creates a list formed with successive evaluations to the expression *EPRESSIONoFi* for *i*=1, 2, 3, …, k (in this order).

– Infixed form for arithmetic or relational operators are, each of them, nothing but forms equivalent with applying certain functions with operands passed to them as arguments.

For instance:

I. The infixed operator "/" (example a/b) is nothing but the function Rational[a,b] which returns the division of *a* by *b*.

II. The comparison operator "==" (example *a*==*b*). For the usage of this work it tests the equality of two numbers. It is nothing but the function Equal[*a*,*b*] which returns the logic value True for case of *a* is equal to *b* and the logic value False when *a* and *b* are not equal.

– Protect[*symbol*] confer to the symbol *symbol* the quality of being "Protected". This quality communicates to the Mathematica system to avoid any changes that would be intended to be done over the definition of the symbol *symbol*. There is a single exception for this rule of the "Protect" quality, namely the "Unprotect" function effect. Such quality settings, done either by the programmer or by the Mathematica system (for the symbols charged with features those the system supplies) are called "Attributes". For instance, the symbol "Rational" has the attribute "Protected".

– Unprotect[*symbol*] cancels the attribute "Protected" for the symbol *symbol*.

– MapAll[*function*,*expression*] With arguments of this kind, "MapAll" applies the function *function* to each sub expression of the expression *expression* and evaluates the new expression obtained this way. MapAll[*function*,*expression*] could be written in the equivalent manner: *function* //@ *expression*.

– Divide[*a*,*b*] has the same definition as Rational[*a*,*b*].

– For[] Function to realize iterative expressions evaluations.

– If[] Function for expressions conditional evaluation or choose between two evaluations alternatives.

– *list*[[*i*]] Returns the position *i* element in list *list*.

– Part[*expression*,i_1 , i_2 , i_3 , …] is the equivalent form to the expression *expression*[[i_1]] [[i_2]] [[i_3]]…

– length[*list*] If the argument is a list, the function returns the number of the list's elements (that is to say the length of the list).

– Append[*list*,*element*] Adds to the list *list* the element *element* as the last element of the list and returns the obtained list.

FORMATTING CONVENTIONS FOR IDENTIFIERS THAT OCCUR INTO THE SOURCE CODE

For formal parameters into the functions' definition and system-supplied functions should be used straight letters; for any other identifiers the italic formatting should be used.

Also we mention that every formatting that is done in this paper could be done inside a Mathematica 4.0 session as well.

Given x_0 and y_0 the Mathematica system finds, by evaluating the below expressions, the order of an element in the elliptic curve group, namely *n*+1 where *n* is the greatest number such as the recurrent relations (4) make sense. The program could be applies for K=Z$_p$ with no other logic restrictions over the chosen *p* but *p* should be a prime natural number. For this work exposition we put *p*=101. Also into the next code there are the following numerical settings: x_0=2 (understood as being 2 modulo *p*) and y_0=39 (understood as being 39 modulo *p*). These x_0 and y_0 verify equation (1)

```
 */
P=101;
Unprotect[Rational];
Rational[a_Integer, b_Integer]:= Mod[a,p]*PowerMod[b,-1,p];
Protect[Rational];
Mod[x_Integer]:=Mod[x,p]
a=-1; b= 0;
x' := mod//@$\left(\frac{(3x_0^2+a)^2}{2y_0^2}-2x_0\right)$
y' :=  mod//@ FullSimplify $\left(-y_0-\frac{(3x_0^2+a)^2}{2y_0^2}(x'-x_0\right)$
x₁ := x₀; y₁ := y₀; x₂ := x'; y₂ := y';
$x(u\_,v\_)$ := mod//@$\left(\left(\frac{v-y_0}{u-x_0}\right)^2-x_0-u\right)$
$x(u\_,v\_,xuv\_)$ := mod//@$\left(-y_0-\frac{v-y_0}{u-x_0}(xuv-x_0)\right)$
F:=(u=x₂; v=y₂; n=2;
    While[Mod[u-x+0+, p] ≠ 0,
      xx = x[u, v];
      yy =y[u, v, xx];
      u = xx; v = yy; n++]; n+1)
```

After the above evaluations have been done, the evaluation of the expression "*f*" gives the number 52 as the result and this is the order of the point whose coordinates are x_0=2 and y_0=39. We remark that the point order should divide the elliptic curve group order whatever be the point (according to Lagrange Theorem).

Based on this remark the evaluation of the next expressions written below represents a more consistent test of the prior source code.

```
/*Let g=7 be the generator of the multiplicative group of Zp with p=101*/
g=7;
/*
Into the variable t1 is recorded the sorted list of all modulo p elements that could be written on the form x³−x, where x is a modulo p element too.
*/
```

```
t1 = Sort[Table[ { Mod[x^3 −x,p],x},{x, p −1} ] ]

/*
Into the variable t1 is recorded the sorted list of all modulo p elements that are
perfect squares in modulo p .
*/

t2 = Sort[Table[{Mod[g^(2*i ),p],2*i },{ i , Divide[p-1,2] } ] ]

\/*
      Next program finds and records into the variable txy the list of all coordinates
pairs {x,y} for which y^2 is in list t2 such as x and y verify equation (1).
 */

lst12 = {};
l1 =Length[ t1 ] ; l2 = Length[ t2 ];
t =Table[Equal[ t1[[ i ]] [[1]],t2[[ j ]] [[1]] ],{ i , l1 },{ j , l2 } ];
For[ i =1, i <= l1 , i++,
   For[ j =1 , j <= l2, j++,
     If[ t[[ i ]] [[ j ]] , lst12=Append[ lst12 , { i , j }]
       ]
     ]
   ];
lst12;
tper = Table[ {Mod[ g^Divide[ t2[[ lst12[[ i ]] [[2]] ]] [[2]] , 2] , p] ,
        t1[[ lst12[[ i ]] [[1]] ]] , t2[[lst12[[ i ]] [[2]] ]] } , { i ,Length[ lst12 ] } ];
txy = Table[ { Part[ tper, i , 2, 2],Part[tper, i , 1]} , { i , Length[tper ] } ]
```

After all of the above Mathematica expressions have been evaluated, this is the "*txy*" evaluation result:

```
{{20,100},{89,100},{93,100},{2,39},{63,3},{9,35},{42,35},{50,35},{35,97
},{5,76},{29,76},{67,76},{91,90},{22,83},{56,27},{58,86},{3,78},{23,78},
{75,78},{7,72},{49,6},{77,21},{21,7},{33,7},{47,7},{54,70},{68,70},{80,7
0},{24,93},{52,41},{94,88},{26,73},{78,73},{98,73},{43,49},{45,68},{79,
22},{10,92},{34,53},{72,53},{96,53},{66,40},{51,54},{59,54},{92,54},{38
,71},{99,87},{8,10},{12,10},{81,10}}
```

The cardinal of prior list of pairs could be easily found by the expression Length[*txy*]. Its evaluation result is 50. The elliptic curve group order finding requires to be added at the above list the following elements:

- For each recorded *y* (modulo *p*) in form of pairs $\{x, y\}$ (as coordinates of the elliptic curve points *P*), it should be added the pair of coordinates $\{x, -y\}$ representing the coordinates of the points $P(x, -y) = -P$. After that, we note, we obtained a number of 100 points.
- To these 100 points just obtained we add the null effect point, *O*, and obtain 101 points.

– Then we add at the 101 points, the elliptic curve points whose ordinate is equal with zero (y = 0). This means finding all abscises x which satisfy $x^3-x=0$, equation having three roots in Z_p (namely -1, 0 and 1). These totalize a number of three points which must be added more. So we obtain the result 104 for the order of the elliptic curve group.

The calculated orders of all elements of the elliptic curve group that are recorded into the variable *txy* are obtained by this evaluation:

```
Table[(x0=txy[[ i]] [[1]] ; y0=txy[[ i]] [[2]] ; f),{ i,Length[txy]}]
```

The result is:

```
{26,26,52,52,52,52,52,52,26,52,52,52,13,52,52,4,26,26,26,52,13,52,52,26,13,26,26
,26,26,26,26,13,26,52,52,13,52,26,26,26,4,52,52,13,52,52,52,26,52,52,52,52,26}
```

We observe that all the numbers in the prior list divide 104.

REFERENCES

1. Koblitz, N., *A Course in Number Theory and Cryptography*, Springer Verlag Berlin Heidelberg New York, Second Printing, 1988
2. Koblitz, N., *Algebraic Aspects of Cryptography*, Springer Verlag Berlin Heidelberg New York, Corrected Second Printing, 1999