> Math 418 : MAPLE Solution of Assignment #10 -- Your NAME

```
> restart; with(numtheory) :
```

Problem 2: The program multpt(k, pt, a, p).
This computes the multiple k*pt of a point pt on the elliptic curve
E: y^2 = x^3 + ax + b over F_p by the binary power method.
Note: It is assumed that pt lie on the curve E (and that \Delta_E <> 0).

(a) Program to add two points on E (from Assignment 9):
Recall: the point at infinity is denoted by [].

```
> addpts := proc(p1, p2, a, p) local lm, x3, y3;
    if p1 = [ ] then return(p2); fi;
    if p2 = [ ] then return(p1); fi;
    if modp(p1[1] - p2[1], p) = 0
     then if modp(p1[2] + p2[2], p) = 0 then return([ ]);
                 else lm := (3*p1[1]^2 + a)/(2*p1[2]); fi;
     else lm := (p2[2] - p1[2])/(p2[1] - p1[1]); fi;
    x3 := lm*lm - p1[1] - p2[1];
    y3 := lm*(p1[1] - x3) - p1[2];
    return([modp(x3, p), modp(y3, p)]); end:
> addpts([0, 1], [0, 1], 2, 5), addpts([ ], [0, 1], 2, 5);
                                  [1, 3], [0, 1]                    (1)
```

Multiplication of points by the binary power method:

```
> multpt := proc(k, pt, a, p) local n1, B, C;
    n1 := k; B := [ ]; C := pt;
    while n1 <> 0 do;
     if irem(n1, 2) = 1 then B := addpts(B, C, a, p); fi;
     n1 := floor(n1/2); C := addpts(C, C, a, p);
    od; return(B); end:
> multpt(5, [1, 1], 0, 11);
                                     [4, 3]                         (2)
```

(b) Test the program for k=211 and pt = [1,2] on E: y^2 = x^3 + 2x +1 mod 997:

```
> multpt(211, [1, 2], 2, 997);
                                   [355, 807]                       (3)
```

Similarly for k = 1299700 and k = 1298393 and pt = [1,1] on E: y^2 = x^3+223344x-223345 mod 1299709:

```
> multpt(1299700, [1, 1], 223344, 1299709);
                                 [576795, 359127]                   (4)
> multpt(1298393, [1, 1], 223344, 1299709);
                                      [ ]                           (5)
```

Remark: Thus, the order of [1,1] divides 1298393 = 67*19379:

```
> ifactor(1298393);
                                 (67) (19379)                       (6)
```

But since 67[1,1] <> [] and 19379<> [], we can conclude that ord([1,1]) = 1298393:

```
> multpt(67, [1, 1], 223344, 1299709); multpt(19379, [1, 1], 223344, 1299709);
```

$$[201313, 125513]$$

$$[715348, 316894] \quad \textbf{(7)}$$

(c)* Computing the order of a point on an elliptic curve

Program to compute the order of a point

```
> ordpt := proc(pt, a, p) local p1, k, n;
    p1 := pt; n := 1;
    while(p1 ≠ [ ]) do;
      p1 := addpts(p1, pt, a, p); n := n + 1; od;
    return(n); end;
ordpt := proc(pt, a, p)                                                    (8)
    local p1, k, n;
    p1 := pt; n := 1; while p1 <> [ ] do p1 := addpts(p1, pt, a, p); n := n + 1 end do; return
    n
end proc
```

Test the program for P_2 = [1,2] on E_2: y^2 = x^3 + 2x + 1/F_997

```
> P2 := [1, 2] : ordpt(P2, 2, 997);
```

$$961 \quad \textbf{(9)}$$

Thus, the point [1,2] on E_2 has order 961.

Problem 3: (Variant of Koblitz, VI.2, Exercises 2 and 11)

Let E: y^2 = x^3 - x + 188 be the given curve over F_p, p = 751.

(a) i. Convert the given message to a list of numbers

Recall: the numerals 0-9 have their usual value and A-Z the values 10-35

This can be done by hand or by using the following program:

```
> converttonumberlist := proc(str) local lis, i, r; lis := [ ];
    for i to length(str) do;
      lis := [op(lis), searchtext(substring(str, i..i, ),
        _0123456789abcdefghijklmnopqrstuvwxyz) − 2]; od;
    return(lis); end:
```

```
> converttonumberlist("This0is1top2secret");
```

$$[29, 17, 18, 28, 0, 18, 28, 1, 29, 24, 25, 2, 28, 14, 12, 27, 14, 29] \quad \textbf{(10)}$$

Apply this to the given message:

```
> ml := converttonumberlist("stop007");
```

$$ml := [28, 29, 24, 25, 0, 0, 7] \quad \textbf{(11)}$$

ii. Program to embed a plaintext m in an elliptic curve:

The elliptic curve is E: y^2 = x^3 + ax +b over F_p,
the parameter is k (such that 0 < k(m+1) < p.)
The program returns point [x,y] on E and an index j to indicate that x = mk+j.
-returns FAIL if no point with x-coordinate mk+ j can be found.
-uses Maple's msqrt program in the number theory package:

```
> embed := proc(m, k, a, b, p) local fx, j, x, y; x := k·m;
    for j to k do;
      x := x + 1; fx := x^3 + a·x + b mod p;
      y := msqrt(fx, p);
      if y ≠ FAIL then return([[x, y], j]); fi; od;
```

```
    return(FAIL); end:
> embed(2, 6, 1, 1, 31);
                          [[13, 14], 1]                          (12)
```

This is indeed a point on the given curve because

```
> evalb(modp(14^2, 31) = modp(13^3 + 13 + 1, 31));
                           true                                  (13)
```

iii. Apply this program to the given message and elliptic curve:

Recall: here p = 751 and E: y^2 = x^2 - x + 188.
We can use k = 20 because the largest message unit is m = 35 (= Z), and because (35+1)20 = 720 < p = 751.

Applying the program "embed" to the given message ml, we obtain

```
> seq(embed(ml[i], 20, -1, 188, 751), i = 1..nops(ml));
[[562, 201], 2], [[581, 20], 1], [[484, 590], 4], [[501, 596], 1], [[1, 376], 1], [[1, 376], 1],   (14)
    [[144, 190], 4]
```

Recall that the number after each point indicates the j that was used. To obtain only the sequence of points, we compute instead:

```
> e1 := [seq(embed(ml[i], 20, -1, 188, 751)[1], i = 1..nops(ml))];
    e1 := [[562, 201], [581, 20], [484, 590], [501, 596], [1, 376], [1, 376], [144, 190]]   (15)
```

(b) To translate a given point into a message, use:

```
> transl := (pt, k) → floor((pt[1] - 1)/k):
> transls := (pls, k) → map(transl, pls, k); #list version
                transls := (pls, k) → map(transl, pls, k)      (16)
```

Here we are given the following embedded message points:

```
> msg := [[361, 8], [241, 230], [201, 5], [461, 92], [581, 20]];
        msg := [[361, 8], [241, 230], [201, 5], [461, 92], [581, 20]]   (17)
> tl := transls(msg, 20);
                tl := [18, 12, 10, 23, 29]                     (18)
```

Thus, the message is: "icant" ("I can't").

(c) ElGamal (without a random number generator) on an elliptic curve:

Here: m is the embedded message, b the base point, public key pk; these are points on E: y^2 = x^3+ ax + b over F_p. Moreover, the secret ("random") key is k.
Note: this uses the programs "addpoints" and "multpt" of Problem 1.

```
> ElGamal := (m, b, pk, k, a, p) → [multpt(k, b, a, p), addpts(m, multpt(k, pk, a, p), a, p)];
    ElGamal := (m, b, pk, k, a, p) → [multpt(k, b, a, p), addpts(m, multpt(k, pk, a, p), a, p)]   (19)
```

Here p = 751, E: y^2 = x^3 -x + 188 as before. The base point and public key are:

```
> b := [0, 376]; pk := [201, 5];
                b := [0, 376]
               pk := [201, 5]                                  (20)
```

The list of secret keys to be used is:

```
> kls := [386, 209, 118, 589, 312, 483, 335];
        kls := [386, 209, 118, 589, 312, 483, 335]             (21)
```

Applying ELGamal to the list e1 of embedded message points above, we thus send the following encrypted (pairs of) message points:

```
> egl := [seq(ElGamal(e1[i], b, pk, kls[i], -1, 751), i = 1..nops(e1))];
                                                               (22)
```

$$egl := [[[676, 558], [385, 328]], [[595, 79], [212, 250]], [[261, 463], [77, 745]], [[126, 476], [66, 214]], [[551, 231], [501, 155]], [[97, 467], [733, 486]], [[63, 689], [380, 155]]] \quad (22)$$

Problem 4: Pollard's p-1 method:

Compute the gcd of a^k - 1 and n (where k = (n-1)/q).

> $pol := (a, k, n) \rightarrow igcd(Power(a, k) - 1 \bmod n, n);$

$$pol := (a, k, n) \rightarrow igcd((Power(a, k) - 1) \bmod n, n) \quad (23)$$

Apply this to n = 167767, k = 840 and a = 2, 3,5, ...

> $n := 167767; k := 840;$

$$n := 167767$$
$$k := 840 \quad (24)$$

> $pol(2, k, n), pol(3, k, n), pol(5, k, n);$

$$167767, 1, 127 \quad (25)$$

Thus, already for a = 5 we get the proper factor 127 and so 167727 = 127*1321:

> $\frac{n}{127}, ifactor(n);$

$$1321, (127)\,(1321) \quad (26)$$

>