Cryptography I
Homework sheet 11

Erwin van Duijnhoven
0719562

Rick Staals
0716184

December 11, 2013

Question 1

For computing discrete logarithms on elliptic curves it is important that collisions can be recognized. This means that point addition has to be computed in affine coordinates, costing at least one inversion per group operation. For this task Weierstrass curves are the most efficiënt, so even if a curve is given in Edwards form for efficient and secure implementation of the cryptosystem, the cryptanalyst will transform it to (short) Weierstrass form in order to attack it.

The curve $y^2 = x^3 - 3x + 910$ over $\mathbb{F}_{2347}$ is an elliptic curve with order $l = 2389$, l is prime. Implement Pollard's rho method (without negation is OK) to compute the discrete logarithm of $Q = (699, 835)$ to the base $P = (2232, 361)$, i.e. the integer $0 < k < l$ with $Q = kP$.

The answer to this exercise consists of a number (the result) and the program computing it. Please test that your program runs and that the result is correct.

Solution:

With $w_i = (w_i^x, w_i^y)$ we use as function $f(w_i) = w_i^x \mod 3$.

$$w_{i+1} = \begin{cases} w_i + P \\ w_i + Q \\ 2w_i \end{cases}, a_{i+1} = \begin{cases} a_i + 1 \\ a_i \\ 2a_i \end{cases}, b_{i+1} = \begin{cases} b_i \\ b_i + 1 \\ 2b_i \end{cases} \text{ for } f(w_i) \equiv \begin{cases} 0 \\ 1 \\ 2 \end{cases}$$

The code we used is on the second page. We found that $k = 1414$.

```
AddW[P_, Q_, prime_] := If[P[[1]] = Q[[1]] && P[[2]] = Q[[2]],
    Lambda = Mod[(3 P[[1]]^2 - 3) PowerMod[2 P[[2]], -1, prime], prime];
    xnew = Mod[Lambda Lambda - P[[1]] - Q[[1]], prime];
    ynew = Mod[-Lambda (xnew - P[[1]]) - P[[2]], prime];
    Return[{xnew, ynew}];,
    Lambda = Mod[(P[[2]] - Q[[2]]) PowerMod[P[[1]] - Q[[1]], -1, prime], prime];
    xnew = Mod[Lambda Lambda - Q[[1]] - P[[1]], prime];
    ynew = Mod[-Lambda (xnew - Q[[1]]) - Q[[2]], prime];
    Return[{xnew, ynew}];
   ];

prime = 2347;
order = 2389;

P = {2232, 361};
Q = {699, 835};

PRstep[{x_, a_, b_}] := Which[
    Mod[x[[1]], 3] = 0, {AddW[x, P, prime], Mod[a + 1, order], b},
    Mod[x[[1]], 3] = 1, {AddW[x, Q, prime], a, Mod[b + 1, order]},
    Mod[x[[1]], 3] = 2, {AddW[x, x, prime], Mod[2 a, order], Mod[2 b, order]}
   ];

W1 = PRstep[{P, 1, 0}];
W2 = PRstep[PRstep[{P, 1, 0}]];

While[W1[[1]] ≠ W2[[1]],
   W1 = PRstep[W1];
   W2 = PRstep[PRstep[W2]];
 ];
c = GCD[W1[[3]] - W2[[3]], order];
d = PowerMod[(W1[[3]] - W2[[3]])/c, -1, order/c];
k = Mod[((W2[[2]] - W1[[2]]) d)/c, order/c]
W = P;
For[i = 1, i < k, i++, W = AddW[W, P, prime];];
W
W = Q
```