Aniruddha Jafa April 23, 2019

CS-302 Problem Set 4

Collaborators: *Paul Kurian, Vidur Singh*

Problem 1

:: Differences:
nature of elements:
In EC group we have points (x,y) that satisfy equation $y^2 \equiv_p x^3 + ax + b$ where p is some odd prime (in EC).

In $Z*_p$ we have integers less than p that are coprime to it. Further, if p is prime, one immediately knows all the values of elements in $Z*_p$ (2 to p-1), wheras to find points in an EC group we have to plug in x values and see if we can find a square root for the y values (see code to 4-4 h)

addition: in $Z*_p$, addition is similar to integer addition (except you do it mod p). In EC, point addition is very differnt from normal addition (see answer to Problem 2 here, where point addition is described).

multiplication: in $Z*_p$, multiplication is similar to integer multiplication (except you do it mod p). In EC, point multiplication is based on point addition (see answer to Problem 2 here, where point addition is described), and thus is very different from integer multiplication.

exponentiation: exponentiation as an operator exists in $Z*_p$; this is not the case for the EC group.

:: Similarities:
Both are groups: closed under some binary operation, identity, inverse, associative.

Both are commutative under their respective '+' operators.

Both operate in some modulo world, and have finite number of elements.

Problem 2

Let P = (x1, y1), Q = (x2, y2)

Note: addition has been described with respect to continuous EC. For the modular case, operations are similar expect they are done mod p, and instead of dividing we find the inverse mod p.

:: Basic addition: P + Q is obtained by drawing a line through P and Q, seeing the point where it intersects the elliptic curve, and reflecting it. This reflected point is P + Q.
The actual values in the above computation are obtained from the equation of the line through P and Q, and the equation of the elliptic curve itself.

:: Cases for point addition

i) point at infinity added to itself gives point at infinity

ii) point at infinity added to a non-infinity point gives us the non-infinity point itself

iii) If P and Q have the same x-coordinates, but different (or equal to 0) y coordinates, their addition yields the point at infinity

iv) P and Q have different x-coordinates

λ is the slope.

$\lambda = (y_2 - y_1)(x_2 - x_1)^{-1}$

$x_R = \lambda^2 - x_1 - x_2$

$y_R = \lambda(x_1 - x_R) - y_1$

v) point doubling (see below - "Doubling a point P")

:: Doubling a point P
Assume P is not the point at infinity. Then doubling is similar to addition. The slope (λ) is the derivative of the elliptic curve.

Consider P+P=R, let R have coordinates (x_R, y_R)

a) Find $\lambda^2 = (3x_1^2 + a)(2y_1)^{-1}$

b) $x_R = \lambda^2 - 2x_1$
c) $y_R = \lambda(x_1 - x_R) - y_1$

:: Cases where P + Q gives us the point at infinity
- both P and Q are infinity
- If P and Q have the same x-coordinates, but different (or equal to 0) y coordinates

Problem 3

Consider EC: $y^2 = x^3 + 2x + 2$ (mod 17)

point to double: (3,1)

$\lambda^2 = (3x_1^2 + a)(2y_1)^{-1}$ (mod p) = $(3.3^2 + 2)(2.1)^{-1}$ (mod 17) = 29.9 (mod 17) = 6

$x_R = \lambda^2 - 2x_1 = 6^2 - 2.3$ (mod 17) = 30 (mod 17) = 13

$y_R = \lambda(x_1 - x_R) - y1 =$ 6(3 - 13) - 1 (mod 17) = -61 (mod 17) = 7

The result of doubling is (13,7)

Problem 4

See code.

Problem 5

See code for part 1, part 2.

Part 3) Is the program susceptible to existential forgery?

No. In earlier case, we sent pair (m, S(m)). Mallory could work backwards to generate some random m' that would be accepted (even thought it's garbage). Now, we send pair (m, S(h(m)). Even if Mallory can follow the earlier procedure to make Bob accept some random string s' which pretends to be S(h(m) (exactly similar to what Mallory did in the non-hash case), Bob will still check if h(m') = s', and Mallory cannot fake this since hashes are one-way i.e. Mallory ecannot backwards-generate such an m'.