Elliptic Curves #2

Friday, November 6, 2020 11:29 AM

Last Time, we defined elliptic curves and defined the addition operation going over how to add P + Q and how to calculate 2P.

0) Show Added Python Code.
1) Review both how to do P+Q, 2P
2) Talk about how to multiply a point by a large integer k. How to calculate kP.
3) On paper, we'll talk about how to exchange keys using something very similar to Diffie-Hellman, but with Elliptic Curves.

1 More Example

$p = 37, a = 2, b = 5$
$y^2 = x^3 + 2x + 5 \pmod{37}$

Make sure $4a^3 + 27b^2 \neq 0 \pmod{37}$
$4(2)^3 + 27(5)^2 = 32 + 675 = 707 = 4 \pmod{37}$

We need points.

P (11, 10), Q(27, 13)

P + Q
---
delta = (13 - 10)/(27 - 11) mod 37
= 3/16 mod 37
= $3(16^{-1})$ mod 37
= 3(7) = 21 mod 37

37 = 2 x 16 + 5
16 = 3 x 5 + 1
16 - 3 x 5 = 1
16 - 3(37 - 2 x 16) = 1
16 - 3 x 37 + 6 x 16 = 1 1
7 x 16 - 3 x 37 = 1

7 x 16 = 1 mod 37

$x_R = \text{delta}^2 - x_p - x_q = 21^2 - 11 - 27 = 441 - 38 = 403 = 33 \pmod{37}$
$y_R = -y_P + \text{delta}(x_p - x_r) = -10 + 21(11 - 33) = -10 - 21x22 = -472 = 9 \pmod{37}$

P + Q = (33, 9)

2P (P = (11,10))

$$\Delta = \frac{3x_p^2 + a}{2y_p} \equiv \frac{3 \cdot 11^2 + 2}{20}$$

$$\equiv 365 \times 20^{-1} \bmod 37$$
$$\equiv (-5) \times 20^{-1} \bmod 37$$
$$\equiv \frac{-5}{20} = -1 \times 4^{-1} \bmod 37$$
$$= -1 \times (-9) \bmod 37$$
$$\equiv 9 \bmod 37$$

37 = 9 x 4 + 1
37 - 9 x 4 = 1
-9 x 4 = 1 (mod 37)

$x_R = \text{delta}^2 - 2x_p = 9*9 - 2(11) = 81 - 22 = 59 = 22 \pmod{37}$
$y_R = \text{delta}(x_p - x_r) - y_p = 9(11 - 22) - 10 = 9(-11) - 10 = -109 = 2 \pmod{37}$

2P = (22, 2)

How to calculate kP quickly

```
Point multiply(Point P, int k) {

   if (k == 1) return P

   if (k%2 == 0) {
      Point half = multiply(P, k/2)
      return add(half, half)
   }

   Point almost = multiply(P, k-1)
   return add(almost, P)
}
```

Almost identical to the fast modular exponentiation…more generally, this paradigm is called Divide and Conquer.

What to realize as the big picture

When we start with some point P, and add it to itself, over and over again, it creates a random permutation of the points in the group. (Note that sometimes the point P isn't a generator and it loops back to itself before it hits everything in the group, but typically the size of the cycle is usually pretty big, so we don't require in cryptosystems to use a "generator" that loops through ALL the points.)

Like RSA and Diffie-Hellman, the idea is that we start somewhere and randomly jump through stuff in the group. If I give you the end point it's hard to know how many times I jumped or where I jumped from to get there.

Essentially, multiplication of elliptic curves is hard to do backwards…

If I give you P, kP but not k, it's hard to calculate k.

Diffie-Hellman with Elliptic Curves

Pick al elliptic curve $E_p(a, b)$, we will pick a base point G(x, y) that is on the curve. Want to make sure it's order is large. (How long it takes to cycle…) These are the public elements, the curve and the point G. Let the order of the point be n.

Alice selects a secret integer, n_A, this is Alice's Private key. She sends to Bob $P_A = n_A \times G$. ($n_A < n$)

Bob selects a secret integer, n_B, this is Bob's Private key. He sends to Alice $P_B = n_B \times G$.

Alice takes P_B and multiplies it by n_A, so she calculates $n_A n_B \times G$.
Bob takes P_A and multiplies it by n_B, so she calculates $n_B n_A \times G$.
This point is their shared key…

Item #1: How do you convert an arbitrary point into bits? (I don't know the answer to this…)

This is definitely more complicated than RSA!!!
So why is there so much interest in it and why is it used?
To get the same security as RSA, you don't need to use as many bits for your prime number…this, in turn, makes ECC faster than RSA to achieve comparable security.