Elliptic Curve Cryptography

Analog of Diffie-Hellman Key Exchange

We can use elliptic curves to exchange keys, very similar to the Diffie-Hellman Key Exchange.

The first public key will be an elliptic curve $E_p(a, b)$, for a large prime number p.

Next, pick a base point G = (x_1, y_1) which has a very large order, *n*, on the curve. As you might expect, the order of a point, G, is the smallest positive integer *n* such that *n*G = 0, where 0 is the origin point.

Both the curve and the base point are the public keys for the system.

Alice and Bob can exchange keys as follows:

Alice picks a secret value n_A < n and sends Bob the point n_A x G.

Bob picks a secret value n_B < n, and sends Alice the point n_B x G.

Alice takes the point Bob sends her and multiplies it by n_A.
Bob takes the point Alice sends her and multiplies it by n_B.

After this, both Alice and Bob have n_A x n_B x G as their shared key.

Similar to the discrete log problem, when Eve sees either n_A x G or n_B x G, she can not determine either n_A or n_B. Similarly, she can't use the two values n_A x G and n_B x G together to combine in some way to create n_A x n_B x G.

In class, we looked at the Elliptic Curve $E_{23}(1, 1)$ using the point P = (3, 10) as a base point. We found out that this point had order 28. Below is a list of each number from 1 to 28 multiplied by point P. The number to the left of the point represents what we are multiplying by:

1. (3, 10)
2. (7, 12)
3. (19, 5)
4. (17, 3)
5. (9, 16)
6. (12, 4)
7. (11, 3)
8. (13, 16)
9. (0, 1)
10. (6, 4)
11. (18, 20)
12. (5, 4)
13. (1, 7)
14. (4, 0)
15. (1, 16)
16. (5, 19)
17. (18, 3)
18. (6, 19)
19. (0, 22)
20. (13, 7)
21. (11, 20)
22. (12, 19)
23. (9, 7)
24. (17, 20)
25. (19, 18)
26. (7, 11)
27. (3, 13)
28. (0, 0)

Thus, if we were using this curve and the point G = (3, 10) as our base point, if Alice chose n_A = 11 and Bob chose n_B = 16, then Alice would send Bob (18, 20) and Bob would send Alice (5, 19). Both, when multiplying would end up with 11 x 16 x G = 176 x G = 20 x G = (13, 7).

Here’s a better representation:

Alice: Picks n_A = 11, sends Bob (18, 20).

Bob: Picks n_B = 16, sends Alice (5, 19).

Alice: Receives (5, 19). Mutliplies it by n_A = 11 and retrieves the point (13, 7).

Bob: Receives (18, 20). Multiplies it by n_B = 16 and retrieves the point (13, 7).

Analog of El Gamal Cryptosystem

Just like the key exchange, our global public elements are an elliptic curve $E_p(a, b)$ and a point G on the curve with a large order, *n*.

Let Alice create her own set of keys so others can send messages to her. She first selects a private key $n_A < n$. She then calculates the corresponding public key, $P_A = n_A$ x G. (Still very similar to the previous key exchange.)

If Bob wants to send Alice a message, he can generate a random integer, $k < n$.

Then he calculates $C_1 = kG$ and $C_2 = P_m + kP_A$, where P_m is the plaintext message (encoded as a point) and P_A, as previously discussed, is Alice’s public key. He sends this pair (C_1, C_2) to Alice, very similar to El Gamal, where Bob generates a random secret value k and uses that to send a pair of cipher texts. Also notice that the same plaintext can be encrypted in *n* different ways, depending on the choice of *k*.

When Alice receives (C_1, C_2), she takes C_1 and multiplies it by n_A.

Note that n_A x C_1 = n_A x *k* x G = (n_A x *k*) x G.
Similarly, note that kP_A = *k* x (n_A x G) = (*k* x n_A) x G = (n_A x *k*) x G.

Thus, after Alice calculates temp = n_A x C_1. Then, to reveal the plaintext, she can just calculate

$P_m = C_2$ – temp. Indeed, notice that $P_m + kP_A$ – temp = P_m + (n_A x *k*) x G - (n_A x *k*) x G = P_m, since the last two terms cancel. (It’s like moving forward around the circle some number of slots and then moving backwards around the circle the same number of slots, when we think about adding each copy of G as moving one slot around a circle with *n* slots.)