Elliptic Curve Cryptography.

We first look at the elliptic curve discrete logarithm problem (ECDLP).

ECDLP: let E be an elliptic curve over $\mathbb{F}_p$ with defining equation $F(x,y) \neq y^2 = f(x) = x^3 + ax + b$, $D_f \neq 0$. Take a point P of order N ($\underbrace{P \oplus P \oplus \dots \oplus P}_{N} = \mathcal{O}$) and choose a secret value $0 < s < N$. ECDLP: given E, P, N and $Q = sP$, find s.

Rmk. N is the smallest number, s.t. $NP = \mathcal{O}$.

The Double-and-Add Algorithm.

Notice that we need to compute multiples of P in $G(E)$ and ther a way to compute mP much faster than finding $P \oplus P$, followed by $P \oplus P \oplus P, \dots$

The algorith. , exactly 'in spirit' to the Fast Power algorithm that we used for finding $a^m \pmod p$.

Step 1. Write the binary expression of m:

$$m = m_0 \cdot \underbrace{1}_{=2^0} + m_1 \cdot 2^1 + m_2 \cdot 2^2 + \dots + m_r \cdot 2^r \text{ with } m_i \in \mathbb{Z}_2 = \{0,1\}.$$

Step 2. Compute $1 \cdot P = P = P_0$

$$2 \cdot P = P \oplus P = P_1$$

$$4 \cdot P = 2P \oplus 2P = P_2$$

$$\vdots$$

$$2^r P = 2^{r-1} P \oplus 2^{r-1} P = P_r$$

Step 3. Find $mP = m_0 \cdot P \oplus m_1 \cdot 2P \oplus \dots \oplus m_r \cdot 2^r P = m_0 \cdot P_0 \oplus m_1 P_1 \oplus \dots \oplus m_r P_r$

$= \bigoplus_{m_i = 1} P_i$

Example. Consider $E: y^2 = x^3 + 58x + 5$ over $\mathbb{F}_{71}$,
$P = (1,8) \in E$ ($\mathrm{ord}(P) = 77$). Let's find $Q = 61P$.

1. Check that $D_f = 4 \cdot (-13)^3 + 27 \cdot 5^2 \equiv 52 \neq 0 \pmod{71}$.

Rmk: the number of points on E is 77. In particular, P is a generator. This satisfies the bounds from Hasse's Theorem: $p+1-2\sqrt{p} \approx 55.15 \leq 77 \leq 88.85 \approx p+1+2\sqrt{p}$

Step 1. $61 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 1$

Step 2. $P_1 = P \oplus P = (35,31)$

$P_2 = P_1 \oplus P_1 = (20,19)$

$P_3 = P_2 \oplus P_2 = (24,13)$ (use the program)

$P_4 = P_3 \oplus P_3 = (9,7)$

$P_5 = P_4 \oplus P_4 = (36,16)$

Step 3. $Q = 61P = P_5 \oplus P_4 \oplus P_3 \oplus P_2 \oplus \overset{=P}{P_0} = (36,16) \oplus (9,7) \oplus (24,13) \oplus (20,19) \oplus (1,8) = (34,32) \oplus (24,13) \oplus (20,19) \oplus (1,8) = (23,67) \oplus (20,19) \oplus (1,8) = (0,17) \oplus (1,8) = (9,64)$

Rmk: we used only 9 operations instead of 60!

Old Friends, New context.

Elliptic Diffie-Hellman Key Exchange.

Recall that for $G=(\mathbb{F}_p^\times, \times)$, Alice and Bob created their shared key as $K = g^{n_A n_B}$, where $g \in \mathbb{F}_p^\times$ was an element of high order N, n_A and n_B were their private keys. Alice sent Bob her public key g^{n_A} and Bob replied with his public key g^{n_B}, after which they both recovered the shared key via

$$K = (g^{n_A})^{n_B} = (g^{n_B})^{n_A}.$$

The generalization is completely straightforward. Namely, let E be a smooth elliptic curve ($D_f \neq 0$) and $P \in E$ a point of order N. As before $1 < n_A, n_B < N$ are the private keys of Alice and Bob. Their public keys are $Q_A = n_A P$ and $Q_B = n_B P$, while the shared key is

$$K = \underbrace{n_A(n_B P)}_{= n_A Q_B} = \underbrace{n_B(n_A P)}_{= n_B Q_A}.$$

computed by Alice

computed by Bob

Elliptic ElGamal PKC (public key cryptosystem).

Recall: $G = (\mathbb{F}_p^\times, *)$, Bob sends Alice a pair of numbers $(c_1, c_2) \equiv (g^{k_B}, m A^{k_B})$, where $m \in \mathbb{F}_p^\times$ is his plaintext message, k_b his private key, A Alice's public key.
Alice recovered m via

$$m \equiv c_2 \cdot c_1^{-k_A}.$$

Now: Bob's message (plaintext) is $M \in E/\mathbb{F}_p$;

$C_1 = k_B P =: Q_B$

$C_2 = M \oplus k_B Q_A$, where $Q_A = k_A P$ (k_A is Alice's private key)
k_B is Bob's private key.

$$\text{Bob} \longrightarrow \boxed{(C_1, C_2)} \longrightarrow \text{Alice}.$$

Alice recovers M as $M = C_2 \ominus k_A C_1 = M \oplus k_B Q_A \ominus k_A k_B P =$
$= M \oplus k_B k_A P \ominus k_A k_B P$

There are a few issues:

1. encoding the messages as points on E (we can not encode the message as the x-coord. of a point, since there might be no point with such x-coord. on E);
2. we need to transmit both coordinates of the points C_1 and C_2, this is a lot of data.

Rmk. $C_2 \oplus k_A C_1$ and $C_2 \ominus k_A C_1$ are very different.

Possible resolutions:

To bypass the second issue, one can send an extra bit of information for each of C_1, C_2:

$$\beta_{1,2} = \begin{cases} 0, & 0 \le y_{C_1, C_2} \le p/2 \\ 1, & p/2 < y_{C_1,C_2} \le p. \end{cases}$$

Rmk. For any $Z \in E$, there are two points on E with first
$\overset{\shortparallel}{(z_x, z_y)}$
coordinate z_x: E and $\ominus E = (z_x, -z_y)$. Notice that $-z_y \equiv p - z_y$, so either $0 \le z_y < p/2$ and $p/2 < p - z_y < p$ or the other way round.

Rmk. The receiver needs to be able to extract square roots modulo p reasonably fast, such algorithms exist.

A nice resolution to the first problem was proposed by Menezes and Vanstone. The corresponding PKC is known as MV-ElGamal cryptosystem.

Step 1. As usual, a trusted party chooses a prime p and a smooth elliptic curve $E/\mathbb{F}_p$ with a point P on it. Alice chooses her private key k_A and publishes the corresponding public key $Q_A = k_A P$

Step 2. Bob needs to send Alice a message $m \in \mathbb{F}_p^\times$. He separates (breaks) it into two messages (m_1 and m_2). Using his shared key with Alice $S = k_B Q_A$, he computes $C_1 \equiv m_1 x_S$ and $C_2 \equiv m_2 y_S$, where $S=(x_S, y_S)$. He also computes his public key $Q_B = k_B P$ and sends the data to Alice:

$$\text{Bob} \longrightarrow \boxed{(Q_B, C_1, C_2)} \longrightarrow \text{Alice}$$

<u>Step 3.</u> Alice recovers the original plaintext message via computing their shared key $S = k_A Q_B$, followed by

$$m_1 \equiv C_1 \cdot x_S^{-1} \pmod{p}$$

$$m_2 \equiv C_2 \cdot y_S^{-1} \pmod{p}.$$