Last time: Elliptic curve Diffie-Hellman

Agree On:

- Prime p
- An elliptic curve $y^2 = x^3 + ax + b$
- A point on the curve, $G = (X, Y)$.

e.g. NIST P-192

Alice picks secret d_A and computes $Q_A = d_A \cdot G$ sends to Bob

Bob picks secret d_B and computes $Q_B = d_B \cdot G$ sends to Alice

repeated doubling

Now: Alice does $d_A \cdot Q_B$ $(= d_A(d_B G))$

Bob does $d_B \cdot Q_A$ $(= d_B(d_A G))$

Equal!

Use x-coordinate as key.

(lots of computer demo time today)

You've just agreed on a ~~300~~ ~60 digit key K.

This is a 180-digit binary string you both know. Now you can use this key to encrypt your messages.

Write your message in binary:

Bob's message: 1 0 1 1 1 0 0 1 1 0 1 0 0 0 0 1 0 1 0 1 1

XOR with key: 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1 0 1 1 0

Send to Alice: 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 ...

Key again: 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1 0 1 1 0

XOR: 1 0 1 1 1 0 0 1 1 0 1 0 0 0 0 1 0 1 0 1 1

If message is longer than key, you'll run out.

Use "block cipher" (like AES)

Elliptic curve digital signatures. (ECDSA)

If Gerald wants to sign a digital message M, here's how it works:

1) Gerald generates a ~~public~~ private key d_G public key Q_G and puts Q_G in a database, on his website.

2) Given Message M, Gerald does some algorithm to generate a signature S. (using d_G)

3) Anyone who sees S, M and Q_G can verify that whoever generated S must know d_G.

Del

Details:

1) Gerald picks private d_G (a number)

2) computes $Q_G = d_G \cdot G$, and posts Q_G publicly.

3) To sign a message M:

a) Gerald computes hash(M) which "digests" M into 256-bit string (in a nonreversible way)

Call that h.

n is number of pts on curve, ~p.

b) Pick a random $k \in [1, n-1]$

c) Compute $R = kG$ in ~~given~~ elliptic curve.

d) Let r be x-coordinate of R.

e) Compute $s = k^{-1} \cdot (h + r\, d_G) \bmod n$

random, hash, x-coordinate, private key

f) Signature is (r, s)

Suppose somebody knows:

- M message
- (r, s) signature
- Q_G public.

They can check: whoever computed s must have known d_G:

1) Compute $s^{-1} \bmod n$ and then

$$R' = (hs^{-1}) \cdot G + (rs^{-1}) \cdot Q_G$$

simplifies to R.

2) If x-coordinate of R' is r, signature is valid!

For different k, you get different signatures, but they'll all pass verification.

If you know (r, s) & (r', s'), two valid signatures generated using same k, you can solve for d_G and generate your own signature!