Lecture 9

Tuesday, 4 October 2022 13:28

Some quirks/improvements for solving ECDLP:

(EC → elliptic curve)

given $P_A = (x_A, y_A) = aP$

on a Weierstrass curve. The we observe that

$-P_A = (x_A, -y_A) = (-a)P$ (minus)

meaning $-a \bmod N$, where $\text{ord}(P) = N$ ($\in [0, N-1]$)

Use this in the BSGS algorithm to match GS on x-coordinates only

$$jmP + P_A = \pm iP \longrightarrow \text{find match faster}$$

$$P_A = aP = \underbrace{(\pm i + jm)}_{\bmod N \text{ is } a} P$$

What about the Pollard rho-method?

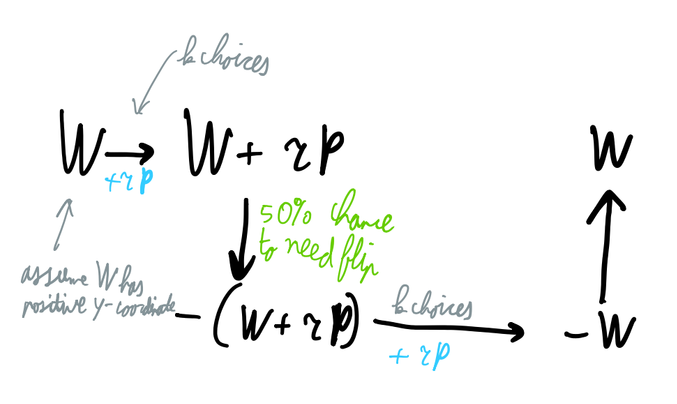
Identify S and $-S$ by looking only at $x(S)$

We can hope for $\sqrt{\frac{\pi N}{4}}$ by birthday paradox, so $\sqrt{2}$ speed-up.

Need to define f such that $f(S) = f(-S)$

Do this by having f take $|S|$ which we define as S or $-S$ so that the y-coordinate is positive. Minor slow-down compared to $\sqrt{2}$ speed up.

But:



Fruitless cycle of 2 steps; this will not give an answer

This happens with probability $\frac{1}{k} \cdot \frac{1}{2}$ (flip sign of $W + 2P$, pick same)

Solution: pick a large r (e.g. 1024)

and check for cycles after 1024 steps (just check whether $W_{i+2} = W_i$)

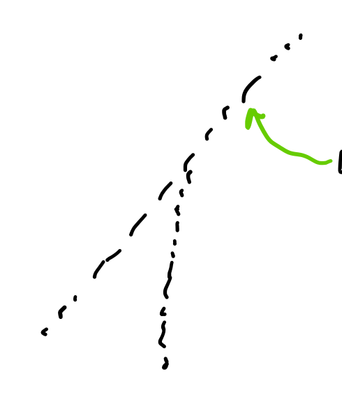
and if a cycle is found, escape it in a way that helps the computation

$W_i = b_i P + c_i P_A$

multiply by 2, to end up somewhere else in the field

$2 W_i = 2(b_i P + c_i P_A)$ preserves knowledge of b_i and c_i, but we need to take the same point to escape the cycle as next time. Of W_i and W_{i+1}, pick the one with the smaller x-coordinate.

This needs some bookkeeping, but it works.



important to escape the same way, else we lose the collision

Lesson from Pohlig-Hellman:

DLP in group of order N depends mostly on hardness of DLP in biggest prime-order subgroup.

$$N = \prod_{i=1}^{t} p_i^{e_i} \quad \text{with } p_i < p_{i+1},\ e_i \geq 1$$

DLP takes $O(\sqrt{p_t})$ time

Signatures: Public key can be used to verify signature.

Signature proves that the signer had the private key.

Signature links the signer to the message (non-repudiation) & proves authenticity & integrity (links to private key)

Alice is known by her public key, which is $P_A = aP$.

Identification protocol should show that A (Alice) knows a without leaking any info on it. We want to do a zero-knowledge (meaning that the other party learns nothing) proof of knowledge of a.

A: prover — has a, aP

B: verifier — has $P_A = aP$ but no a separately

1. pick random $r \in [0, N-1]$ "commitment" — sends $R = rP$ →
2. picks a random $h \in [0, N-1]$ "challenge" — ← sends h
3. computes $s \equiv r + ha$ — sends s → verifies $sP = R + hP_A$

This works: if A computed s correctly (meaning she knows a and r) the verification holds.

If A knew h before picking r: as an example, take $h = 23$

$sP = R + 23P_A$

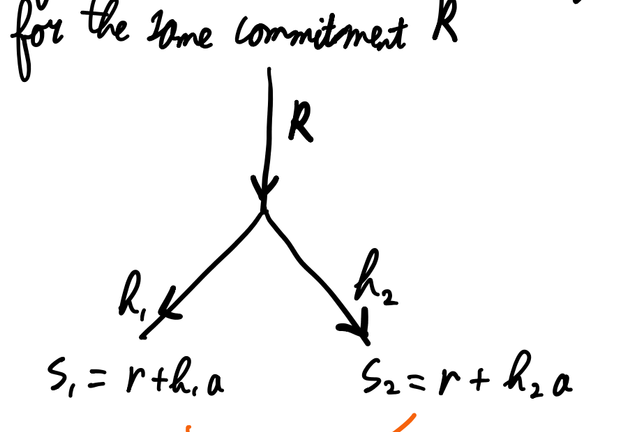
pick $R = -23P_A$, then $= 0P$

$R = -23P_A + P$ is valid for $s = 1$

If Alice knows h before committing to R, she can pick a random s, computes $R = -hP_A + sP$ as her commitment and answer the challenge with s.

Consequences:

1. "fake Alice" has $\frac{1}{N}$ chance of winning by guessing h.
2. Bob does not learn anything about a from (R, h, s) because he can compute valid triples himself.
   → zero knowledge
3. If A can answer 2 challenges for the same commitment R



$$a = \frac{s_1 - s_2}{h_1 - h_2} \bmod N \ldots$$

Same R, 2 different h: can compute a → this proves A has a but also shows her secret

But this means that a normal transcript (no rewinding) proves that she has a.

This is the Schnorr identification protocol.

Remove interaction to get signature scheme.

→ replace the challenge by the hash of the message, but we need to enforce committing to R before seeing h, so

$h = \text{hash}(R, m)$ ← this enforces ordering

Schnorr signature:

1. pick r, compute $R = rP$
2. compute $h = \text{hash}(R, m)$
3. compute $s \equiv r + ha \bmod N$
4. output (R, s) as signature

Now, security relies on DLP and on preimage & collision resistance of hash

To further protect, put R first so that knowing that $\text{hash}(m) = \text{hash}(m')$ doesn't make my signature on m valid for m'.

$\text{hash}(m) = \text{hash}(m')$ for SHA-2

implies

$\text{hash}(m, R) = \text{hash}(m', R)$, but

$\text{hash}(R, m) \neq \text{hash}(R, m')$