The parallelized kangaroo method

Edlyn Teske

University of Waterloo

ECC '99

The discrete logarithm problem (DLP):

G finite cyclic group ,
written multiplicatively.

$$g, h \in G \ ,$$

such that g generates the group:

$$\langle g \rangle = G \ .$$

Task: Find x such that

$$h = g^x \ .$$

Solve

$$h = g^x \ .$$

If we do not know anything else about x but

$$0 \leq x < n = \mathrm{ord(g)} \ :$$

- **Shanks' baby step-giant step method:**
  Time and space: $const \cdot \sqrt{n}$.

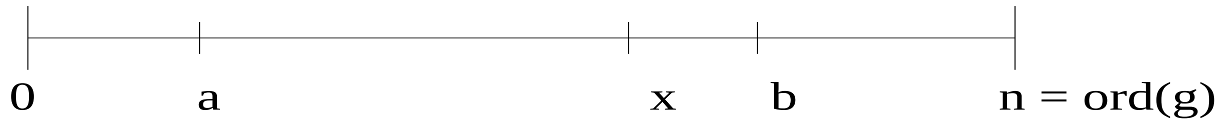
- **Pollard's rho method:**
  Expected time $\approx \sqrt{\pi n/2}$ group operations, can be parallelized with linear speed-up.
  See: Solution of the Certicom challenge problems, e.g.: ECC2-97 on Sept. 28, '99.

Solve

$$h = g^x \ ,$$

where

$$a \leq x \leq b \ :$$



- Baby step-giant step method
  Time and space: $const \cdot \sqrt{b-a}$.

- Pollard's rho method *cannot* exploit this knowledge.

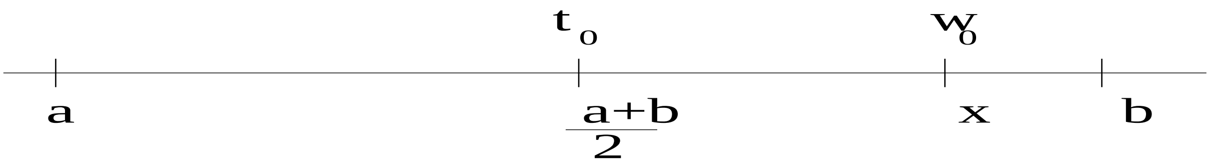
- **Pollard's kangaroo method:**
  expected running time: $2\sqrt{b-a}$ group operations.

Outline

- Serial version
- Parallel rho method $\neq$ kangaroo method (What is the *lambda* method?)
- Parallel version (van Oorschot & Wiener) with experimental results
- Pollard's parallel version with experimental results
- Comparison
- Implications for ECC

The kangaroo method (Pollard ’78)
in the version by v.Oorschot & Wiener (’94/’99)

Tame kangaroo T **:** $t_0 = g^{(a+b)/2}$.
Wild kangaroo W **:** $w_0 = h \quad (= g^x)$.



Set of jumps:

$$S = \{g^{s_0}, g^{s_1}, \ldots g^{s_r} \ : \quad s_i = O(\sqrt{b-a}),$$
$$r = O(\log(b-a))\} \ .$$

Hash function $v : G \to \{0, \ldots, r\}$.

$(G = \langle g \rangle)$.

$k = 0, 1, \ldots :$

Tame kangaroo: $t_{k+1} = t_k * g^{s_i}$,

where $i = v(t_k)$.

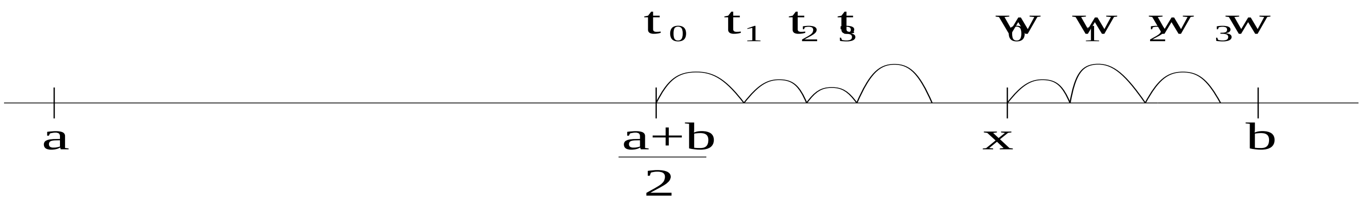
Wild kangaroo: $w_{k+1} = w_k * g^{s_j}$,

where $j = v(w_k)$.

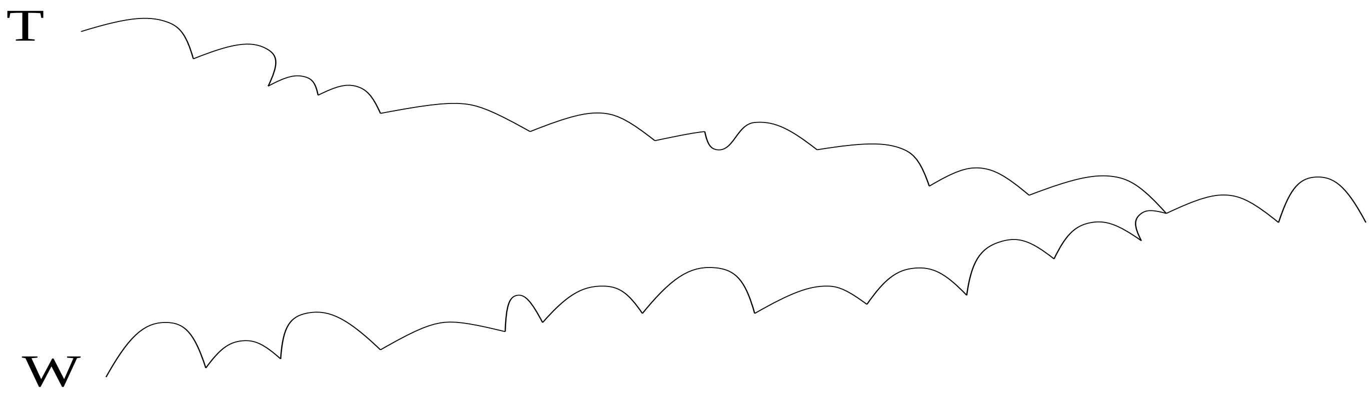
Keep track of the **distances** travelled:

$\delta_0(T) = \delta_0(W) = 0$,

$$
\begin{aligned}
\delta_{k+1}(T) &= \delta_k(T) + s_i \qquad (i = v(t_k)) \\
\delta_{k+1}(W) &= \delta_k(W) + s_j \qquad (j = v(w_k))
\end{aligned}
$$



Eventually, T and W meet:



$$\begin{aligned} t_k &= w_{k'} \\ &\Longleftrightarrow \\ g^{\frac{a+b}{2}} * g^{\delta_k(T)} &= g^x * g^{\delta_{k'}(W)} \end{aligned}$$

I.e.,

$$x \equiv \frac{a+b}{2} + \delta_k(T) - \delta_{k'}(W) \pmod{n} \ .$$

Even:

$$x = \frac{a+b}{2} + \delta_k(T) - \delta_{k'}(W) \ .$$

Expected running time:

$$\sqrt{b-a} + const \qquad \text{jumps/kangaroo} \quad ,$$

if set of jumps is chosen properly.

Proper choice of jumps g^{s_i}:

$$s_i = 2^i, \qquad i = 0, 1, \ldots, r \ ,$$

and such that

$$\text{mean value of the } s_i \approx \frac{\sqrt{b-a}}{2} \ .$$

(This has been verified in practice.)

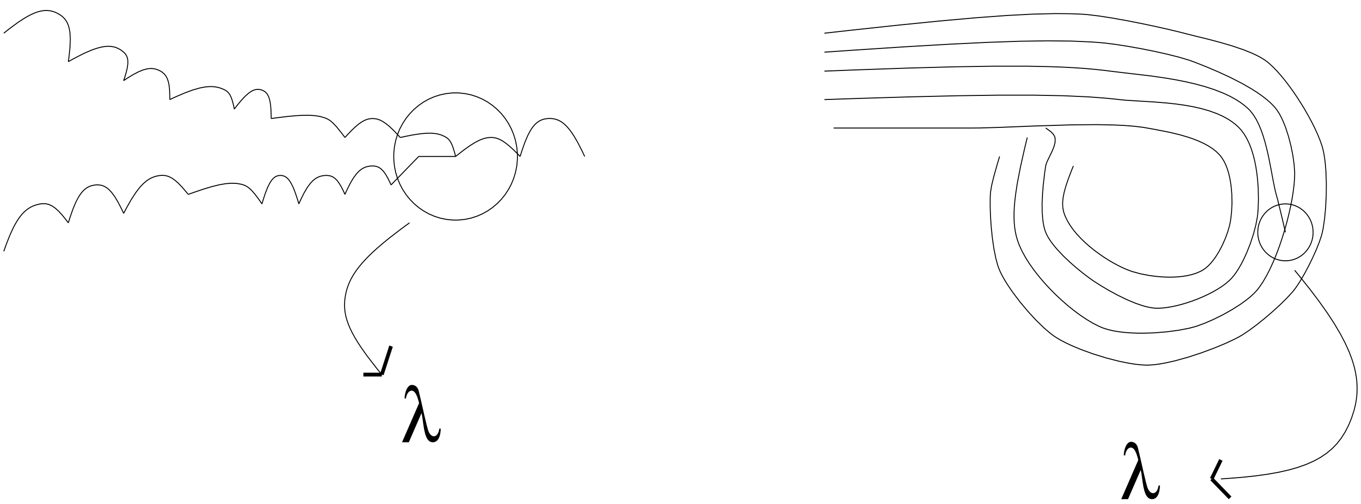
Q: How to detect collision?
A: Distinguished point method:

An element $g \in G$ is *distinguished* if, for example, the last f bits in its representation as binary string are zero.

- Can be done off-line

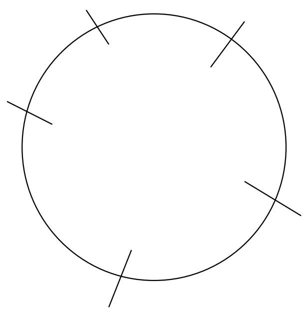
- Storage requirements:
  time-memory trade-off:
  $\approx 2^f$ jumps to find distinguished point.
  Expect $2\sqrt{b-a}/2^f$ elements to store.

Kangaroo method $\neq$ parallel rho method

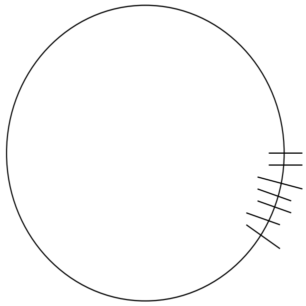


~~lambda method~~

Rho method:
Simulate a **random walk** in the group $\langle g \rangle$.



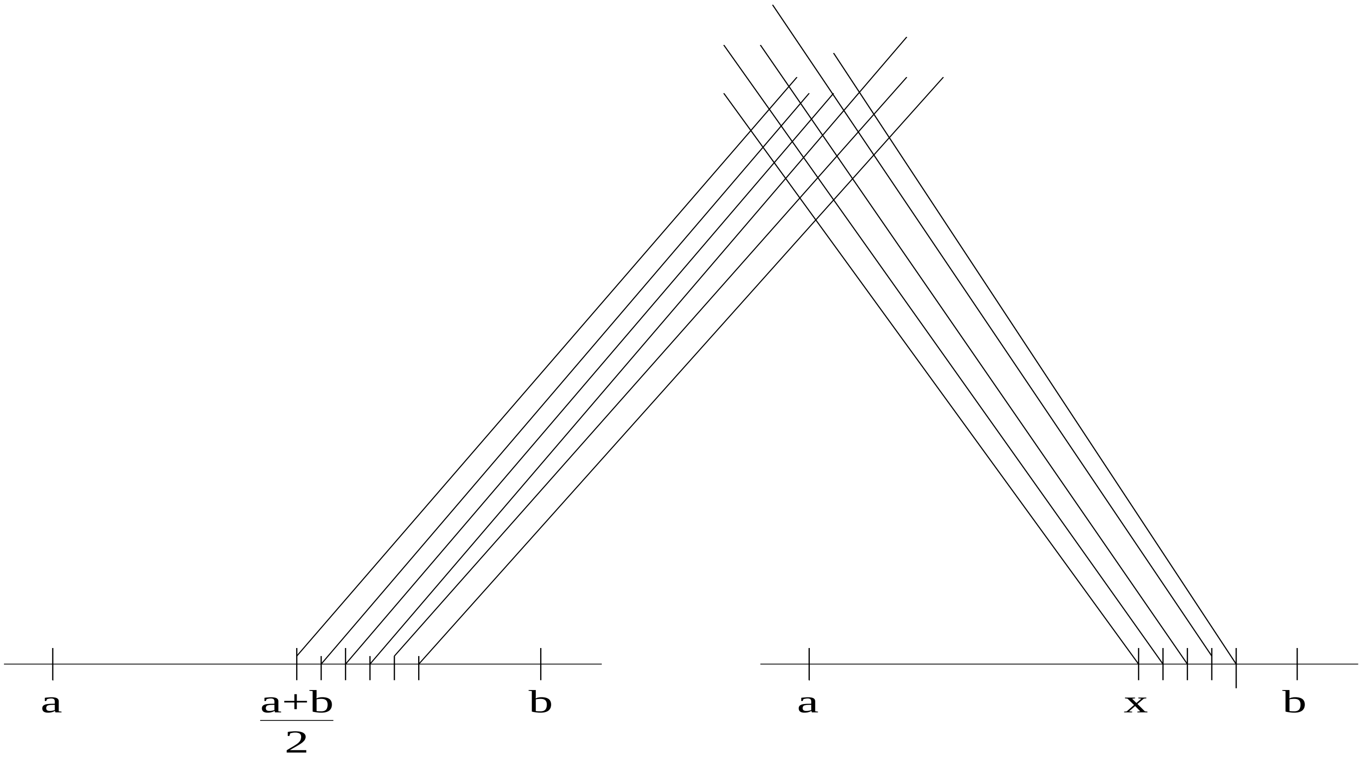
Kangaroo method:
Jumps g^{s_i}, where the s_i are viewed as **small** distances (in the order of $\sqrt{b-a}$).
"Kangaroos cannot leap between continents."



If $a = 0$ and $b =$ ord(g), the kangaroo method takes about 1.6 times longer than the rho method.
Point of turnover: $b - a <$ 0.4ord(g).

Parallelization:
Not one tame & one wild kangaroo,
but two *herds*

$$T_1, \ldots T_n$$
$$W_1, \ldots W_n$$



This can be parallelized:
Use $m = 2n$ machines, one kangaroo on each machine (and modify set of jumps).

Parallelization (I)

Version by van Oorschot & Wiener

Choose small distance ν, not a power of two.

First tame kangaroo T_1 starts at

$$g^{(a+b)/2} \ ,$$

kangaroos $T_2, T_3, \ldots, T_n$ start at

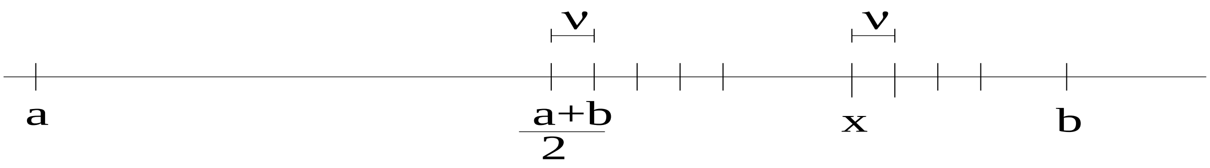
$$T_1 * g^{\nu}, T_1 * g^{2\nu}, \ldots, T_1 * g^{(n-1)\nu} \ .$$

First wild kangaroo W_1 starts at

$$h$$

and kangaroos $W_2, \ldots, W_n$ start at

$$W_1 * g^{\nu}, W_2 * g^{2\nu}, \ldots, W_1 * g^{(n-1)\nu} \ .$$



Set of jumps: distances s_i = powers of 2 as before, but with mean value

$$\frac{m\sqrt{b-a}}{4}$$

instead of $\sqrt{b-a}/2$

(m = number of machines, = total number of kangaroos) .
(I.e., serial case: $m = 2$.)

Theory predicts **linear speed-up**:
each kangaroo is expected to make

$$2\sqrt{b-a}/m$$

jumps.

Experimental results

$p = 10^{15} + 37$; $E : y^2 = x^3 + 5x + 19 \pmod p$

$E(\mathbb{F}_p)$ is cyclic.
Take generator P, and let

$$Q = xP ,$$

with $x \in [a, b]$.

$[a, b] = [0, 10^8] \quad , \quad [0, 10^{10}] \quad , \quad [0, 10^{12}]$.

Do this 1000 times,
and take average performances for
20, 100 and 1000 kangaroos.

Simulated parallelization.

Implementation: in C++, using the Computer Algebra System LiDIA.

	$[0, 10^8]$		$[0, 10^{10}]$		$[0, 10^{12}]$	
20 kangaroos	1109	2.22	10945	2.19	117405	2.35
100 kangaroos	209	2.09	2105	2.1	23856	2.38
1000 kangaroos	26	2.6	216	2.16	2087	2.09

Wiener's parallelization

left entry: # jumps/kangaroo

right entry : $\frac{\#\text{ jumps}}{\text{kangaroo}} \cdot \frac{\#\text{ kangaroos}}{\sqrt{b}}$

(Average values, taken over 1000 runs for $b = 10^8, 10^{10}$, and 100 runs for $b = 10^{12}$.)

	$[0,10^8]$	$[0,10^{10}]$	$[0,10^{12}]$
20 kangaroos	1.8	1.7	1.7
100 kangaroos	4.2	3.7	3.2
1000 kangaroos	52.6	34	25.8

Wiener's parallelization

useless collisions

(Average values, taken over 1000 runs for $b = 10^8, 10^{10}$, and 100 runs for $b = 10^{12}$.)

	$[0, 10^8]$	$[0, 10^{10}]$	$[0, 10^{12}]$
20 kangaroos	2.09	2.08	2.29
100 kangaroos	2.05	2.05	2.32
1000 kangaroos	2.5	2.12	2.08

Wiener's parallelization

$\frac{\#\ \text{jumps}}{\text{kangaroo}} \cdot \frac{\#\ \text{kangaroos}}{\sqrt{b}}$

Version: after tame-tame or wild-wild collision one of the two colliding kangaroos is immediately set off again.

Parallelization (II)
Pollard's version

Let u, v be coprime,
$u + v \le m = \#$machines, $u \approx v$.

For example: $m = 100$: $u = 49$, $v = 51$.

First tame kangaroo T_1 starts at

$$g^{(a+b)/2} \quad ,$$

kangaroos $T_2, T_3, \ldots, T_n$ start at

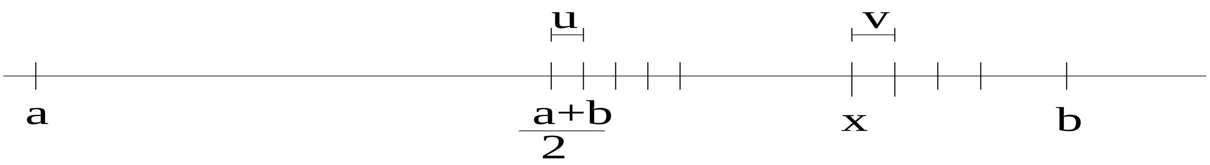
$$T_1 * g^u, T_1 * g^{2u}, \ldots, T_1 * g^{(n-1)u} \quad .$$

First wild kangaroo W_1 starts at

$$h$$

and kangaroos $W_2, \ldots, W_n$ start at

$$W_1 * g^v, W_2 * g^{2v}, \ldots, W_1 * g^{(n-1)v} \quad .$$



Set of jumps: g^{r_i}, with distances

$$r_i = s_i uv \ ,$$

with $s_i =$ powers of 2 as before.

Mean value of the r_i:

$$\frac{\sqrt{(b-a)uv}}{2}$$

instead of $\sqrt{b-a}\,/2$.

Observe:

Jumping does not change the residue classes modulo uv of the distances of the kangaroos.

The equation

$$\frac{a+b}{2} + iv = x + ku \pmod{uv} .$$

has a unique pair of solutions

$$[\ i \pmod{u}\ ,\ k \pmod{v}\ ] .$$

Hence, exactly one pair of tame and wild kangaroos travels in the same residue class modulo uv.

This pair is the only pair to meet.

Especially:
No collisions among tame kangaroos.
No collisions among wild kangaroos.

	$[0, 10^8]$	$[0, 10^{10}]$	$[0, 10^{12}]$
20 kangaroos	2.13	2.11	2.20
100 kangaroos	2.19	2.07	2.11
1000 kangaroos	2.0	2.13	2.14

Pollard's parallelization

$\frac{\text{\# jumps}}{\text{kangaroo}} \cdot \frac{\text{\# kangaroos}}{\sqrt{b}}$

Comparison of both parallel versions

- $\approx$ the same expected number of steps in both versions.

- van Oorschot & Wiener:
  may require occasional communication from the central server to the clients.

- Pollard:
  requires reliable system of machines.
  E.g.: large parallel processor.

- Running-time analysis and experiments have been carried out only under the assumption that all machines work with the same speed.
  In practice, this assumption may be more crucial for Pollard's version.

Implications for Elliptic Curve Cryptography

Let P be a point on an elliptic curve over a finite field, of order $|\langle P\rangle|$.

ECC with $|\langle P\rangle| =$ 160-bit prime is considered secure.

Public key: point Q.

Private key: x such that $Q = xP$.

ECC can be sped up if private key is limited to a restricted range, say $[0, b]$, $b < 2^{160}$.

But: This decreases the level of security!

Parallel kangaroo method $\Longrightarrow$
If b is n-bit number, ECC with restricted range $[0, b]$ for private key is only as secure as ECC with $|\langle P \rangle| = n$-bit prime.

The 97-bit Certicom challenge problems have been solved. $\Longrightarrow$ With approximately the same effort, we can break ECC with 97-bit number b.

Arithmetic progressions

Assume that we know numbers r and q such that

$$x \equiv r \pmod{q} \ .$$

That is,

$$x = r + qy \qquad \text{where} \qquad 0 \le y \le \frac{\text{ord(g)}}{q} \ .$$

Then, instead of using the rho method to solve

$$g^x = h$$

we use the kangaroo method to solve

$$(g^q)^y = h * g^{-r} \ ,$$

where the solution y comes from an interval of length $\lfloor \text{ord(g)/q} \rfloor$.

$\Longrightarrow$ speed-up of factor $\approx \dfrac{\sqrt{q}}{1.6}$.