Cryptanalysis Course

Part III – DLPs in intervals

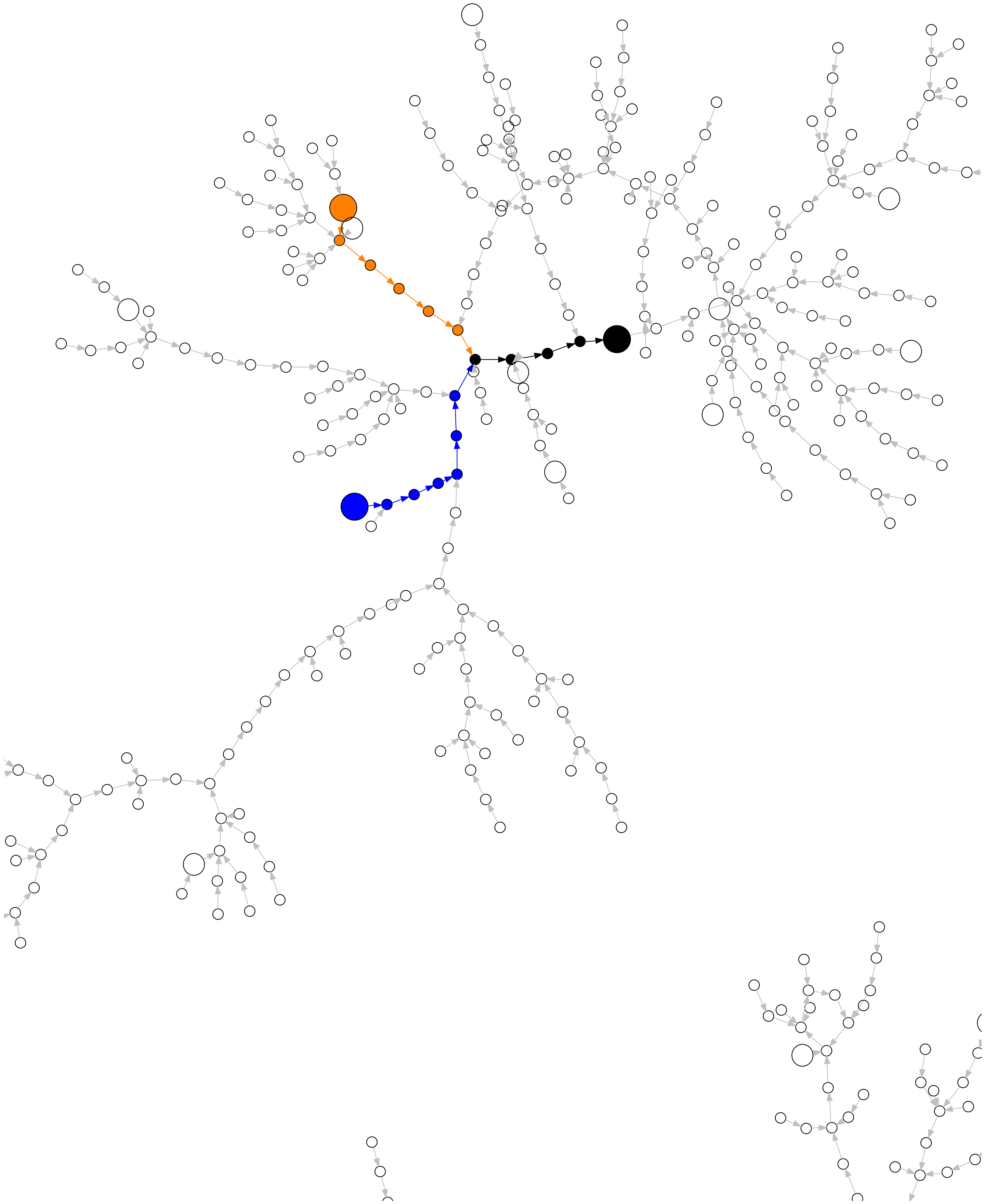
Tanja Lange

Technische Universiteit Eindhoven

30 November 2016

with some slides by

Daniel J. Bernstein



Additive walks

Generic rho method requires two scalar multiplications for each iteration.
Could replace by double-scalar multiplication; could further merge the 2-scalar multiplications across several parallel iterations.

Additive walks

Generic rho method requires two scalar multiplications for each iteration.

Could replace by double-scalar multiplication; could further merge the 2-scalar multiplications across several parallel iterations.

More efficient: use *additive walk*:

Start with $W_0 = a_0 P$ and put

$f(W_i) = W_i + c_j P + d_j Q$

where $j = h(W_i)$.

Pollard's initial proposal:

Use $x(W_i)$ mod 3 as h

and update:

$$W_{i+1} = \begin{cases} W_i + P \text{ for } x(W_i) \bmod 3 = 0 \\ 2W_i \quad \text{ for } x(W_i) \bmod 3 = 1 \\ W_i + Q \text{ for } x(W_i) \bmod 3 = 2 \end{cases}$$

Easy to update a_i and b_i.

$$(a_{i+1}, b_{i+1}) = \begin{cases} (a_i + 1, b_i) \text{ for } x(W_i) \bmod 3 = 0 \\ (2a_i, 2b_i) \quad \text{ for } x(W_i) \bmod 3 = 1 \\ (a_i, b_i + 1) \text{ for } x(W_i) \bmod 3 = 2 \end{cases}$$

Additive walk requires only one addition per iteration.
h maps from $\langle P\rangle$ to $\{0, 1, \ldots, r-1\}$, and $R_j = c_j P + d_j Q$ are precomputed for each $j \in \{0, 1, \ldots, r-1\}$.

Easy coefficient update:
$W_i = a_i P + b_i Q$,
where a_i and b_i are defined recursively as follows:
$a_{i+1} = a_i + c_{h(W_i)}$ and
$b_{i+1} = b_i + d_{h(W_i)}$.

Additive walks have disadvantages:
The walks are noticeably nonrandom; this means they need more iterations than the generic rho method to find a collision.

This effect disappears as r grows, but but then the precomputed table $R_0, \ldots, R_{r-1}$ does not fit into fast memory. This depends on the platform, e.g. trouble for GPUs.
More trouble with adding walks later.

Randomness of adding walks

Let $h(W) = i$ with probability p_i.

Fix a point T, and let W and W' be two independent uniform random points.

Let $W \neq W'$ both map to T.

This event occurs if

Randomness of adding walks

Let $h(W) = i$ with probability p_i.

Fix a point T, and let W and W' be two independent uniform random points.

Let $W \neq W'$ both map to T.

This event occurs if simultaneously for $i \neq j$:
$T = W + R_i = W' + R_j$;
$h(W) = i$; $h(W') = j$.

These conditions have probability $1/\ell^2$, p_i, and p_j respectively.

Summing over all (i, j) gives the overall probability $\left(\sum_{i \neq j} p_i p_j\right) / \ell^2 = \left(\sum_{i,j} p_i p_j - \sum_i p_i^2\right) / \ell^2 = \left(1 - \sum_i p_i^2\right) / \ell^2$.

This means that the probability of an immediate collision from W and W' is $\left(1 - \sum_i p_i^2\right) / \ell$, where we added over the ℓ choices of T. In the simple case that all the p_i are $1/r$, the difference from the optimal $\sqrt{\pi \ell / 2}$ iterations is a factor of $1/\sqrt{1 - 1/r} \approx 1 + 1/(2r)$.

Various heuristics leading to standard $\sqrt{1-1/r}$ formula in different ways:

1981 Brent–Pollard;

2001 Teske;

2009 ECC2K-130 paper, eprint 2009/541.

Various heuristics leading to standard $\sqrt{1-1/r}$ formula in different ways:
1981 Brent–Pollard;
2001 Teske;
2009 ECC2K-130 paper, eprint 2009/541.

2010 Bernstein–Lange:
Standard formula is wrong!
There is a further slowdown from higher-order anti-collisions:
e.g. $W + R_i + R_k \neq W' + R_j + R_l$ if $R_i + R_k = R_j + R_l$.
$\approx 1\%$ slowdown for ECC2K-130.

Eliminating storage

Usual description: each walk keeps track of a_i and b_i with $W_i = a_i P + b_i Q$.

This requires each client to implement arithmetic modulo ℓ or at least keep track of how often each R_j is used.

For distinguished points these values are transmitted to server (bandwidth) which stores them as e.g. (W_i, a_i, b_i) (space).

2009 ECC2K-130 paper:
Remember where you started.
If $W_i = W_j$ is the collision of distinguished points,
can recompute these walks with a_i, b_i, a_j, and b_j;
walk is deterministic!
Server stores 2^{45} distinguished points; only needs to know coefficients for 2 of them.

Our setup: Each walk remembers seed; server stores distinguished point and seed.
Saves time, bandwidth, space.

Negation and rho

$W = (x, y)$ and $-W = (x, -y)$
have same x-coordinate.
Search for x-coordinate collision.

Search space for collisions is
only $\lceil \ell/2 \rceil$; this gives factor $\sqrt{2}$
speedup . . . if $f(W_i) = f(-W_i)$.

To ensure $f(W_i) = f(-W_i)$:
Define $j = h(|W_i|)$ and
$f(W_i) = |W_i| + c_j P + d_j Q$.
Define $|W_i|$ as, e.g., lexicographic
minimum of $W_i, -W_i$.
This negation speedup
is textbook material.

Problem: this walk can
run into fruitless cycles!
Example: If $|W_{i+1}| = -W_{i+1}$
and $h(|W_{i+1}|) = j = h(|W_i|)$
then $W_{i+2} = f(W_{i+1}) =$
$-W_{i+1} + c_j P + d_j Q =$
$-(|W_i| + c_j P + d_j Q) + c_j P + d_j Q =$
$-|W_i|$ so $|W_{i+2}| = |W_i|$
so $W_{i+3} = W_{i+1}$
so $W_{i+4} = W_{i+2}$ etc.

If h maps to r different values
then expect this example to occur
with probability $1/(2r)$
at each step.
Known issue, not quite textbook.

Eliminating fruitless cycles

Issue of fruitless cycles is known and several fixes are proposed. See appendix of full version ePrint 2011/003 for even more details and historical comments.

Summary: most of them got it wrong.

Eliminating fruitless cycles

Issue of fruitless cycles is known and several fixes are proposed. See appendix of full version ePrint 2011/003 for even more details and historical comments.

Summary: most of them got it wrong.

So what to do?
Choose a big r, e.g. $r = 2048$.
$1/(2r) = 1/4096$ small;
cycles infrequent.

Define $|(x, y)|$ to mean
(x, y) for $y \in \{0, 2, 4, \ldots, p - 1\}$
or
$(x, -y)$ for $y \in \{1, 3, 5, \ldots, p - 2\}$.

Precompute points
$R_0, R_1, \ldots, R_{r-1}$ as known
random multiples of P.

Define $|(x,y)|$ to mean
(x,y) for $y \in \{0,2,4,\ldots,p-1\}$
or
$(x,-y)$ for $y \in \{1,3,5,\ldots,p-2\}$.

Precompute points
$R_0, R_1, \ldots, R_{r-1}$ as known
random multiples of P.
Can do full scalar multiplication in
inversion-free coordinates!
Start each walk at a point
$W_0 = |b_0 Q|$,
where b_0 is chosen randomly.
Compute $W_1, W_2, \ldots$ as
$W_{i+1} = |W_i + R_{h(W_i)}|$.

Occasionally, every w iterations, check for fruitless cycles of length 2.

For those cases change the definition of W_i as follows:

Compute W_{i-1} and check whether $W_{i-1} = W_{i-3}$.

If $W_{i-1} \neq W_{i-3}$, put $W_i = W_{i-1}$.

If $W_{i-1} = W_{i-3}$, put $W_i = |2\min\{W_{i-1}, W_{i-2}\}|$, where min means lexicographic minimum.

Doubling the point makes it escape the cycle.

Cycles of length 4, 6, or 12
occur far less frequently.
Cycles of length 4, or 6
are detected when checking
for cycles of length 12;
so skip individual ones.

Same way of escape:
define $W_i =$

$$|2\min\{W_{i-1}, W_{i-2}, W_{i-3}, W_{i-4}, W_{i-5}, W_{i-6}, W_{i-7}, W_{i-8}, W_{i-9}, W_{i-10}, W_{i-11}, W_{i-12}\}|$$

if trapped
and $W_i = W_{i-1}$ otherwise.

Do not store all these points!

When checking for cycle,
store only potential entry point
W_{i-13} (one coordinate, for
comparison) and the
smallest point encountered since
(to escape).

For large DLP
look for larger cycles;
in general, look for
fruitless cycles of even lengths
up to $\approx (\log \ell)/(\log r)$.

How to choose w?

Fruitless cycles of length 2 appear
with probability $\approx 1/(2r)$.
These cycles persist
until detected.
After w iterations,
probability of cycle $\approx w/(2r)$,
wastes $\approx w/2$ iterations
(on average) if it does appear.

Do not choose w
as small as possible!
If a cycle has *not* appeared then
the check wastes an iteration.

The overall loss is approximately $1 + w^2/(4r)$ iterations out of w. To minimize the quotient $1/w + w/(4r)$ we take $w \approx 2\sqrt{r}$.

Cycles of length $2c$ appear with probability $\approx 1/r^c$,
optimal checking frequency is $\approx 1/r^{c/2}$.
Loss rapidly disappears as c increases.
Can use lcm of cycle lengths to check.

Concrete example: 112-bit DLP

Use $r = 2048$. Check for 2-cycles every 48 iterations.
Check for larger cycles much less frequently.
Unify the checks for 4-cycles and 6-cycles into a check for 12-cycles every 49152 iterations.

Choice of r has big impact!
$r = 512$ calls for checking for 2-cycles every 24 iterations.
In general, negation overhead $\approx$ doubles when table size is reduced by factor of 4.

Bernstein, Lange, Schwabe
(PKC 2011):

Our software solves
random ECDL on the same curve
(with no precomputation)
in 35.6 PS3 years on average.

For comparison:
Bos–Kaihara–Kleinjung–Lenstra–
Montgomery software
uses 65 PS3 years on average.

Bernstein, Lange, Schwabe (PKC 2011):

Our software solves random ECDL on the same curve (with no precomputation) in 35.6 PS3 years on average.

For comparison: Bos–Kaihara–Kleinjung–Lenstra–Montgomery software uses 65 PS3 years on average.

First big speedup: We use the negation map.

Second speedup: Fast arithmetic.

Why are we confident this works?

We only have 1 PlayStation-3,
not 200 used in their record.
Don't want to wait for 36 years
to show that we actually compute
the right thing.

Why are we confident this works?

We only have 1 PlayStation-3,
not 200 used in their record.
Don't want to wait for 36 years
to show that we actually compute
the right thing.

Can produced scaled versions:
Use *same* prime field
(so that we can compare the field
arithmetic) and same curve shape
$y^2 = x^3 - 3x + b$
but vary b to get curves with
small subgroups.

This produces other curves, and many of those have smaller order subgroups.

Specify DLP in subgroup of size 2^{50}, or 2^{55}, or 2^{60} and show that the actual running time matches the expectation.

And that DLP is correct.

We used same property for a point to be distinguished as in big attack; probability is 2^{-20}.

Need to watch out that walks do not run into rho-type cycles (artefact of small group order).

We aborted overlong walks.

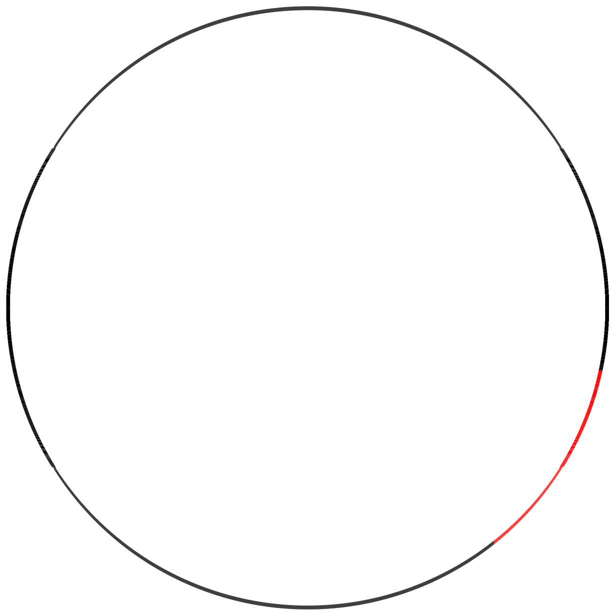
New record

Announced 29 Nov 2016,
most work by Ruben Niederhagen
(`@cryptocephaly` on twitter).

Elliptic curve over $\mathbf{F}_{2^{127}}$,
DLP in subgroup of order $2^{117.35}$.
Used parallel Pollard rho,
DP criterion: 30 top bits equal 0.

Expected
$\sqrt{\pi 2^{117.35}/4}/2^{30} \sim 379\,821\,956$
DPs, but ended up needing
968 531 433.
Computations ran on 64 to 576
FPGAs in parallel.

DLs in intervals



Want to use knowledge that DL is in a small interval $[a, b]$, much smaller than ℓ.

We can use this in baby-step giant-step algorithm.

How to use this in a memory-less algorithm?

Standard interval method:
Pollard's kangaroo method.

Pollard's kangaroos do small
jumps around the interval.

Standard interval method:
Pollard's kangaroo method.

Pollard's kangaroos do small
jumps around the interval.

Real kangaroos sleep

Standard interval method:
Pollard's kangaroo method.

Pollard's kangaroos do small
jumps around the interval.

Real kangaroos sleep

(at least outside Australia).

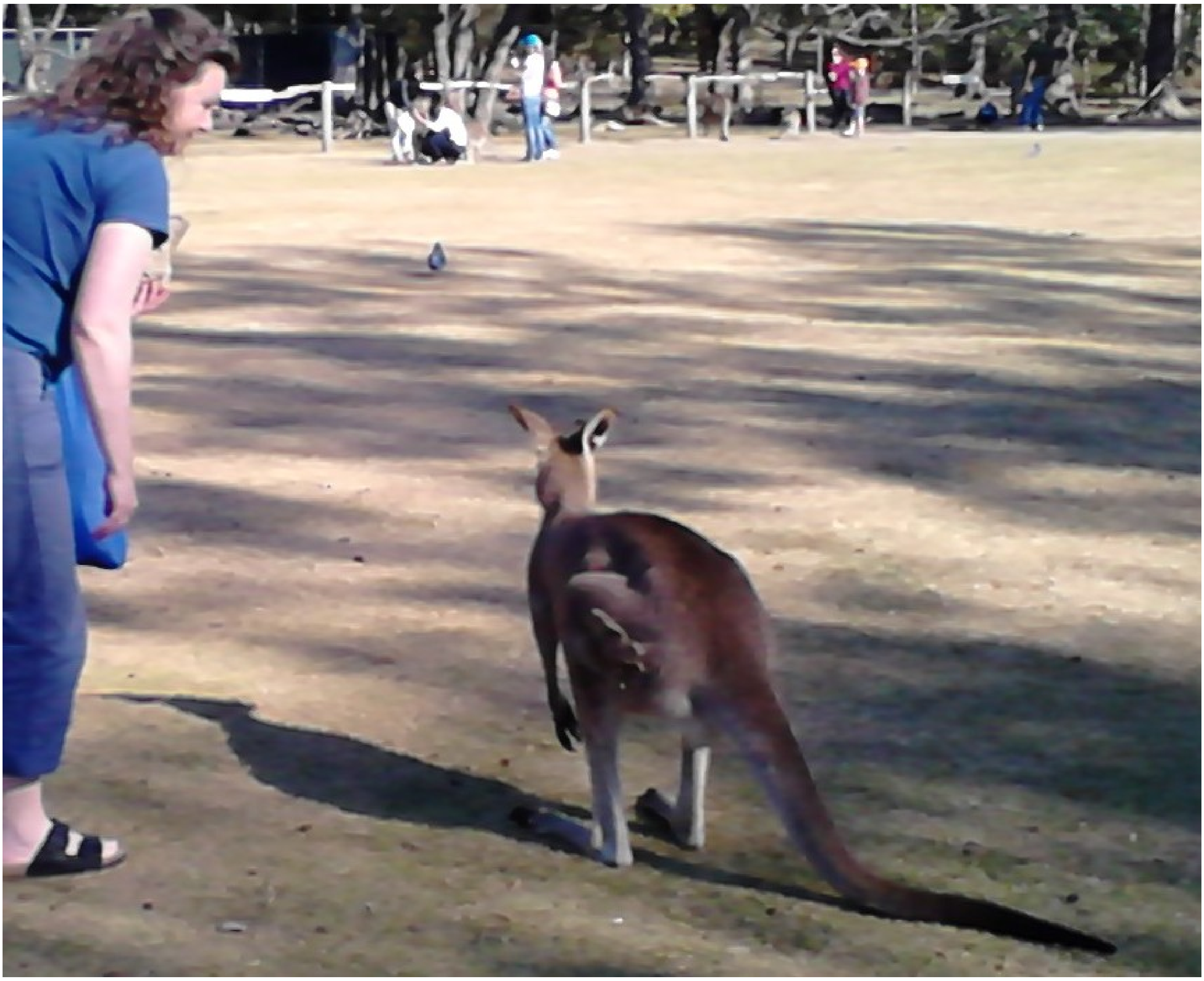
Kangaroo method

in Australia
Main actor:

The tame kangaroo

starts at a known
multiple of P, e.g. bP.

The tame kangaroo jumps.



Jumps are determined
by current position.

The tame kangaroo jumps.

Jumps are determined by current position. Average jump distance is $\sqrt{b-a}$.

The tame kangaroo jumps.

Jumps are determined by current position. Average jump distance is $\sqrt{b-a}$.

The tame kangaroo jumps.

Jumps are determined by current position. Average jump distance is $\sqrt{b - a}$.

The tame kangaroo stops

after a fixed number of jumps (about $\sqrt{b-a}$ many).

The tame kangaroo installs a trap and waits.

The wild kangaroo

starts at point Q.

Follows the same instructions for jumps.

But we don't know where
the starting point Q is.
Know $Q = nP$ with $n \in [a, b]$.

Hope that the paths of the tame and wild kangaroo intersect.

Similar to the rho method the kangaroos will hop on the same path from that point onwards.

Eventually the wild kangaroo falls into the trap.
(Or disappears in the distance if paths have not intersected.
Start a fresh one
from $Q + P, Q + 2P, \ldots.$)

Same story in math

Kangaroo = sequence $X_i \in \langle P \rangle$.

Starting point $X_0 = s_0 P$.

Distance $d_0 = 0$.

Step set: $S = \{s_1 P, \ldots, s_L P\}$,

with s_i on average

$s = \beta\sqrt{b-a}$.

Hash function

$H : \langle P \rangle \to \{1, 2, \ldots, L\}$.

Update function

$d_{i+1} = d_i + s_{H(X_i)}, \quad i = 0, 1, 2, \ldots,$

$X_{i+1} = X_i + s_{H(X_i)} P, \quad i = 0, 1, 2, \ldots$

Tame kangaroo starts at

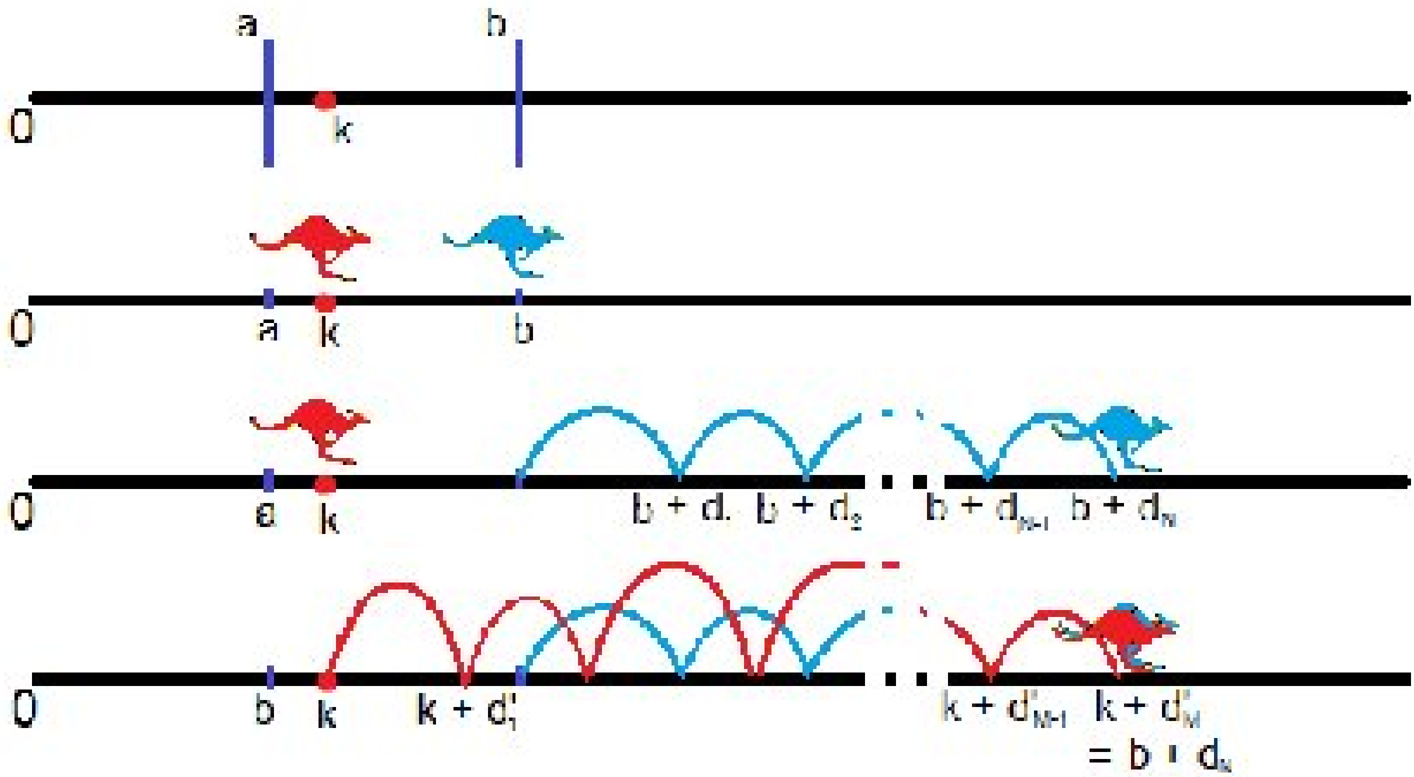
$X_0 = bP$,

wild kangaroo starts at

$X_0' = Q = nP$.

Trap: distance d_N,

endpoint $X_N = (b + d_N)P$.



Picture credit:

Christine van Vredendaal.

Parallel kangaroo method

Use an entire herd

of tame kangaroos,

all starting

around $((b-a)/2)P$ ...

. . . and define certain spots as distinguished points

Also start a herd of
wild kangaroos around Q.
Hope that one wild and
one tame kangaroo
meet at one distinguished point.

Pairings

Let $(G_1, +), (G_2, +)$ and $(G_T, \cdot)$ be groups of prime order ℓ and let

$e : G_1 \times G_2 \to G_T$

be a map satisfying

$e(P + Q, R') = e(P, R')e(Q, R'),$
$e(P, R' + S') = e(P, R')e(P, S').$

Request further that e is non-degenerate in the first argument, i.e., if for some P $e(P, R') = 1$ for all $R' \in G_2$, then P is the identity in G_1

Such an e is called a *bilinear map* or *pairing*.

Consequences of pairings

Assume that $G_1 = G_2$,
in particular $e(P, P) \neq 1$.

Then for all triples
$(P_1, P_2, P_3) \in \langle P \rangle^3$
one can decide in time polynomial
in $\log \ell$ whether
$\log_P(P_3) = \log_P(P_1) \log_P(P_2)$
by comparing
$e(P_1, P_2)$ and $e(P, P_3)$.
This means that the decisional
Diffie-Hellman problem is easy.

The DL system G_1 is at most as secure as the system G_T.

Even if $G_1 \neq G_2$ one can transfer the DLP in G_1 to a DLP in G_T, provided one can find an element $P' \in G_2$ such that the map $P \to e(P, P')$ is injective.

Pairings are interesting attack tool if DLP in G_T is easier to solve; e.g. if G_T has index calculus attacks.

We want to define pairings
$G_1 \times G_2 \to G_T$
preserving the group structure.

The pairings we will use
map to the multiplicative group of
a finite extension field $\mathbf{F}_{q^k}$.
More precisely, $G_T \subset \mathbf{F}_{q^k}$, order ℓ.

To embed the points of order ℓ
into $\mathbf{F}_{q^k}$ there need to be ℓ-th
roots of unity are in $\mathbf{F}^*_{q^k}$.

The *embedding degree* k satisfies
k is minimal with $\ell \mid q^k - 1$.

E is supersingular if
for $|E(\mathbf{F}_q)| = q + 1 - t$, $q = p^r$,
it holds that $t \equiv 0 \bmod p$.

Otherwise it is ordinary.

Example:
$y^2 + y = x^3 + a_4 x + a_6$ over $\mathbf{F}_{2^r}$
is supersingular:
Each (x, y) point also gives
$(x, y + 1) \neq (x, y)$.

All points come in pairs,
except for ∞,
so $|E(\mathbf{F}_{2^r})| = 1+$even,
so $t \equiv 0 \bmod 2$.

Embedding degrees

Let E be supersingular and $q = p \geq 5$, i.e $p > 2\sqrt{p}$.

Hasse's Theorem states $|t| \leq 2\sqrt{p}$.
E supersingular implies $t \equiv 0 \bmod p$, so $t = 0$ and $|E(\mathbf{F}_p)| = p + 1$.

Obviously
$(p+1) \mid p^2 - 1 = (p+1)(p-1)$
so $k \leq 2$ for supersingular curves over prime fields.

Distortion maps

For supersingular curves there exist maps
$\phi : E(\mathbf{F}_q) \to E(\mathbf{F}_{q^k})$
i.e. maps $G_1 \to G_2$, giving $\tilde{e}(P,P) \neq 1$ for $\tilde{e}(P,P) = e(P,\phi(P))$.
Such a map is called a *distortion map*.

These maps are important since the only pairings we know how to compute are variants of *Weil pairing* and *Tate pairing* which have $e(P,P) = 1$.

Examples:

$y^2 = x^3 + a_4x$,

for $p \equiv 3 \pmod 4$.

Distortion map

$(x, y) \mapsto (-x, \sqrt{-1}y)$.

$y^2 = x^3 + a_6$, for $p \equiv 2 \pmod 3$.

Distortion map $(x, y) \mapsto (jx, y)$

with $j^3 = 1, j \neq 1$.

In both cases, $\#E(\mathbf{F}_p) = p + 1$,

so $k = 2$.

Example from Tuesday:

$p = 1000003 \equiv 3 \bmod 4$ and $y^2 = x^3 - x$ over $\mathbf{F}_p$.

Has $1000004 = p + 1$ points.

$P = (101384, 614510)$ is a point of order 500002.

$nP = (670366, 740819)$.

Construct $\mathbf{F}_{p^2}$ as $\mathbf{F}_p(i)$.

$\phi(P) = (898619, 614510i)$.

Invoke magma and compute

$e(P, \phi(P)) = 387265 + 276048i$;

$e(Q, \phi(P)) = 609466 + 807033i$.

Solve with index calculus to get $n = 78654$.

(Btw. this is the clock).

Summary of pairings

Menezes, Okamoto, and Vanstone for E supersingular:
For $p = 2$ have $k \leq 4$.
For $p = 3$ we $k \leq 6$
Over $\mathbf{F}_p$, $p \geq 5$ have $k \leq 2$.
These bounds are attained.

Not only supersingular curves:
MNT curves are non-supersingular curves with small k.
Other examples constructed for pairing-based cryptography –
but small k unlikely to occur for random curve.

Summary of other attacks

Definition of embedding degree does not cover all attacks.
For $\mathbf{F}_{p^n}$ watch out that pairing can map to $\mathbf{F}_{p^{km}}$ with $m < n$.
Watch out for this when selecting curves over $\mathbf{F}_{p^n}$.

Anomalous curves:
If $E/\mathbf{F}_p$ has $\#E(\mathbf{F}_p) = p$ then transfer $E(\mathbf{F}_p)$ to $(\mathbf{F}_p, +)$.
Very easy DLP.
Not a problem for Koblitz curves, attack applies to order-p subgroup.

Weil descent:
Maps DLP in E over $\mathbf{F}_{p^{mn}}$
to DLP on variety J over $\mathbf{F}_{p^n}$.
J has larger dimension; elements represented as polynomials of low degree. $\Rightarrow$ index calculus.

This is efficient if dimension of J is not too big.

Particularly nice to compute with J if it is the Jacobian of a hyperelliptic curve C.
For genus g get complexity $\tilde{O}(p^{2-\frac{2}{g+1}})$ with the factor base described before, since polynomials have degree $<= g$.