# Batch binary Edwards

D. J. Bernstein
University of Illinois at Chicago

NSF ITR–0716498

Classic $\mathbf{F}_p^*$ index calculus needs to check smoothness of many positive integers $< p$.

Smooth integer: integer with no prime divisors $> y$. Typical: $(\log y)^2 \in (1/2 + o(1)) \log p \log \log p$.

Many: typically $y^{2+o(1)}$, of which $y^{1+o(1)}$ are smooth.

(Modern index calculus, NFS: smaller integers; smaller $y$.)

How to check smoothness?

Old answers: Trial division, time $y^{1+o(1)}$; rho, time $y^{1/2+o(1)}$, assuming standard conjectures.

Better answer: ECM etc. Time $y^{o(1)}$; specifically $\exp\sqrt{(2+o(1))\log y \log\log y}$, assuming standard conjectures.

Much better answer (using RAM): Known *batch* algorithms test smoothness of *many* integers simultaneously. Time per input: $(\log y)^{O(1)}$ $= \exp O(\log\log y)$.

General pattern:

Algorithm designer optimizes algorithm for *one* input.

But algorithm is then applied to *many* inputs! Oops.

Often much better speed from *batch* algorithms optimized for many inputs.

e.g. Batch ECDL: $\sqrt{\#}$ speedup.
Batch NFS: smaller exponent.
Can find many more examples.

Surprising recent example:
Batching can save time
in *multiplication*!

Largest speedups: $\mathbf{F}_2[x]$.

Consequence: New speed record
for public-key cryptography.
$\approx$ 30000 scalar mults/second
on a 2.4GHz Core 2 Quad for
a secure elliptic curve/$\mathbf{F}_{2^{251}}$.

Software release this month.

Surprising recent example:
Batching can save time
in *multiplication*!

Largest speedups: $\mathbf{F}_2[x]$.

Consequence: New speed record
for public-key cryptography.
$\approx$ 30000 scalar mults/second
on a 2.4GHz Core 2 Quad for
a secure elliptic curve/$\mathbf{F}_{2^{251}}$.

Software release this month.

Note: No subfields were exploited
in the creation of this record.

Batched conditional branches are slow and painful. Solution: *complete* curve operations.

2008 Bernstein–Lange–Rezaeian Farashahi: for $n \geq 3$, every ordinary elliptic curve over $\mathbf{F}_{2^n}$ can be written as a "complete binary Edwards curve."

Extremely fast formulas for complete differential addition. With good curve selection: $5\mathbf{M} + 4\mathbf{S}$ per bit.

Note 1: Need complete *curve*.
Need singularities at $\infty$
blowing up irrationally.

Symmetric, Edwards-like:
$x^2(y^2+y+d)$
$+\,x(y^2+\cdots)+(dy^2+\cdots)$,
with $y^2+y+d$ irreducible.

Note 2: Need complete *formulas*.
Warning: for odd characteristic,
$(x_1,y_1)+(x_2,y_2)=$
$$\left(\frac{x_1y_1+x_2y_2}{x_1x_2+y_1y_2},\frac{x_1y_1-x_2y_2}{x_1y_2-x_2y_1}\right)$$
is an *incomplete* addition law
on a complete Edwards curve!