Factorization: state of the art

1. Batch NFS
2. Factoring into coprimes
3. ECM

D. J. Bernstein
University of Illinois at Chicago

Tanja Lange
Technische Universiteit Eindhoven
Finding small factors

Find smooth congruences by finding small factors of many congruences:

Never ending supply of congruences
↓ select
Smallest congruences
↓ find small factors
Partial factorizations using primes \( \leq y \)
↓ abort non-smooth
Smooth congruences
How to find small factors?

Could use trial division:
For each congruence,
remove factors of 2,
remove factors of 3,
remove factors of 5,
etc.; use all primes $p \leq y$.

$y^{3+o(1)}$ bit operations:
$y^{1+o(1)}$ per congruence.

Want something faster!
Early aborts

Never ending supply of congruences
↓ select
Smallest congruences
↓
Partial factorizations using primes \( \leq y^{1/2} \)
↓ early abort
Smallest unfactored parts
↓
Partial factorizations using primes \( \leq y \)
↓ final abort
Smooth congruences
Find small primes by trial division. Cost $y^{1/2+o(1)}$ for primes $\leq y^{1/2}$. Cost $y^{1+o(1)}$ for primes $\leq y$.

Say we choose “smallest” so that each congruence has chance $y^{1/2+o(1)}/y^{1+o(1)}$ of surviving early abort. Have reduced trial-division cost by factor $y^{1/2+o(1)}$.

Fact: A $y$-smooth congruence has chance $y^{-1/4+o(1)}$ of surviving early abort. Have reduced identify-a-smooth cost by factor $y^{1/4+o(1)}$. 
Example from Andrew Shallue:
A uniform random integer in
\[1, 2^{64} - 1\] has chance about
\(2^{-8.1}\) of being \(2^{15}\)-smooth,
chance about \(2^{-3.5}\) of having \(2^7\)-
unfactored part below \(2^{44}\), and
chance about \(2^{-9.8}\) of satisfying
both conditions.

Given congruence, find primes \(\leq 2^7\); abort if unfactored part
is above \(2^{44}\); then find primes \(\leq 2^{15}\). Compared to skipping the
abort: about \(2^{3.5}\) times faster,
about \(2^{1.7}\) times less productive;
gain \(2^{1.8}\).
More generally, can abort at $y^{1/k}, y^{2/k}$, etc. Balance stages to reduce cost per congruence from $y^{1+o(1)}$ to $y^{1/k+o(1)}$.

Fact: A $y$-smooth congruence has relatively good chance of surviving early abort. Have reduced identify-a-smooth cost by factor $y^{(1-1/k)/2+o(1)}$.

Increase $k$ slowly with $y$. Find enough smooth congruences using $y^{2.5+o(1)}$ bit operations.

Want something faster!
Sieving

Textbook answer: Sieving finds enough smooth congruences using only $y^{2+o(1)}$ bit operations.

To sieve: Generate in order of $p$, then sort in order of $i$, all pairs $(i, p)$ with $i$ in range and $i(n + i) \in p\mathbb{Z}$.

Pairs for one $p$ are $(p, p), (2p, p), (3p, p)$, etc. and $(p - (n \mod p), p)$ etc.

e.g. $y = 10, n = 611, i \in \{1, 2, \ldots, 100\}$:
For $p = 2$ generate pairs 
$(2, 2), (4, 2), (6, 2), \ldots, (100, 2)$
and
$(1, 2), (3, 2), (5, 2), \ldots, (99, 2)$.

For $p = 3$ generate pairs 
$(3, 3), (6, 3), \ldots, (99, 3)$ and
$(1, 3), (4, 3), \ldots, (100, 3)$.

For $p = 5$ generate pairs 
$(5, 5), (10, 5), \ldots, (100, 5)$ and
$(4, 5), (9, 5), \ldots, (99, 5)$.

For $p = 7$ generate pairs 
$(7, 7), (14, 7), \ldots, (98, 7)$ and
$(5, 7), (12, 7), \ldots, (96, 7)$. 
Sort pairs by first coordinate:
(1, 2), (1, 3), (2, 2), (3, 2), (3, 3), (4, 2), (4, 3), (4, 5), \ldots, (98, 2), (98, 7), (99, 2), (99, 3), (99, 5), (100, 2), (100, 3), (100, 5).

Sorted list shows that
the small primes in $i(n + i)$ are
2, 3 for $i = 1$;
2 for $i = 2$;
\ldots
2, 7 for $i = 98$;
2, 3, 5 for $i = 99$;
2, 3, 5 for $i = 100$. 
In general, for \( i \in \{1, \ldots, y^2\} \):

Prime \( p \) produces \( \approx y^2/p \) pairs \((p, p), (2p, p), (3p, p), \) etc.
and produces \( \approx y^2/p \) pairs \((p - (n \mod p), p)\) etc.

Total number of pairs \( \approx \)
\[
\sum_{p \leq y} 2y^2/p \approx 2y^2 \log \log y.
\]

Easily generate pairs, sort, and finish checking smoothness, in \( y^2 (\log y)^{O(1)} \) bit operations.
Only \( (\log y)^{O(1)} \) bit operations per congruence.
Hidden costs

Is that what we do in record-setting factorizations? No!

Sieving has two big problems.

First problem:
Sieving needs large \( i \) range.
For speed, must use batch of \( \geq y^{1+o(1)} \) consecutive \( i \)'s.
Limits number of sublattices, so limits smoothness chance.

Can eliminate this problem using remainder trees.
Hidden costs, trees

Second problem with sieving, not fixed by remainder trees: Need $y^{1+o(1)}$ bits of storage.

Real machines don’t have much fast memory: it’s expensive.

Effect is not visible for small computations on single serial CPUs, but becomes critical in huge parallel computations.

How to quickly find primes above size of fast memory?
The rho method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 11$.

Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6)\cdots(\rho_{3575} - \rho_{7150})$.

Also many larger primes.

Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod $c$, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to $2^{20}$.
More generally: Choose $z$.
Compute $\gcd\{c, S\}$ where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z})$.

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y^{1/2+o(1)}$; so $y^{1/2+o(1)}$ mults mod $c$.
Early-abort rho: $y^{1/4+o(1)}$ mults.

Reason: Consider first collision in $\rho_1 \mod p$, $\rho_2 \mod p$, $\ldots$.
If $\rho_i \mod p = \rho_j \mod p$
then $\rho_k \mod p = \rho_{2k} \mod p$
for $k \in (j - i)\mathbb{Z} \cap [i, \infty] \cap [j, \infty]$. 
The $p - 1$ method

$S_1 = 2^{232792560} - 1$ has prime divisors


These divisors include

70 of the 168 primes $\leq 10^3$;
156 of the 1229 primes $\leq 10^4$;
296 of the 9592 primes $\leq 10^5$;
470 of the 78498 primes $\leq 10^6$;

etc.
An odd prime $p$ divides $2^{232792560} - 1$ iff order of 2 in the multiplicative group $\mathbb{F}_p^*$ divides $s = 232792560$.

Many ways for this to happen: $232792560$ has 960 divisors.

Why so many?

Answer: 

$$s = 232792560 = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19.$$
Can compute $2^{232792560} - 1$ using 41 ring operations.
(Side note: 41 is not minimal.)

Ring operation: $0, 1, +, −, \cdot$.

This computation: $1; 2 = 1 + 1$;
$2^2 = 2 \cdot 2$; $2^3 = 2^2 \cdot 2$; $2^6 = 2^3 \cdot 2^3$;
$2^{12} = 2^6 \cdot 2^6$; $2^{13} = 2^{12} \cdot 2$; $2^{26}$; $2^{27}$; $2^{54}$;
$2^{55}$; $2^{110}$; $2^{111}$; $2^{222}$; $2^{444}$; $2^{888}$; $2^{1776}$;
$2^{3552}$; $2^{7104}$; $2^{14208}$; $2^{28416}$; $2^{28417}$;
$2^{56834}$; $2^{113668}$; $2^{227336}$; $2^{454672}$; $2^{909344}$;
$2^{909345}$; $2^{1818690}$; $2^{1818691}$; $2^{3637382}$;
$2^{3637383}$; $2^{7274766}$; $2^{7274767}$; $2^{14549534}$;
$2^{14549535}$; $2^{29099070}$; $2^{58198140}$;
$2^{116396280}$; $2^{232792560}$. $2^{232792560} - 1$. 
Given positive integer $n$, can compute $2^{232792560} - 1 \mod n$ using 41 operations in $\mathbb{Z}/n$.

Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

e.g. $n = 8597231219$: \ldots

$2^{27} \mod n = 134217728$;
$2^{54} \mod n = 134217728^2 \mod n = 935663516$;
$2^{55} \mod n = 1871327032$;
$2^{110} \mod n = 1871327032^2 \mod n = 1458876811$; \ldots;
$2^{232792560} - 1 \mod n = 5626089344$. 
Given positive integer $n$, can compute $2^{2^{32792560}} - 1 \mod n$ using 41 operations in $\mathbb{Z}/n$.

Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

e.g. $n = 8597231219$: ... 

$2^{27} \mod n = 134217728$;  
$2^{54} \mod n = 134217728^2 \mod n = 935663516$;  
$2^{55} \mod n = 1871327032$;  
$2^{110} \mod n = 1871327032^2 \mod n = 1458876811$; ...;  
$2^{2^{32792560}} - 1 \mod n = 5626089344$.

Easy extra computation (Euclid):
$\gcd\{5626089344, n\} = 991$. 
This $p - 1$ method (1974 Pollard) quickly factored $n = 8597231219$. Main work: 27 squarings mod $n$.

Could instead have checked $n$’s divisibility by 2, 3, 5, . . . .  
The 167th trial division would have found divisor 991.

Not clear which method is better. Dividing by small $p$ is faster than squaring mod $n$.  
The $p - 1$ method finds only 70 of the primes $\leq 1000$; trial division finds all 168 primes.
Scale up to larger exponent
\[ s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\} : \]
using 136 squarings mod \( n \)
find 2317 of the primes \( \leq 10^5 \).

Is a squaring mod \( n \)
faster than 17 trial divisions?

Or
\[ s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\} : \]
using 1438 squarings mod \( n \)
find 180121 of the primes \( \leq 10^7 \).

Is a squaring mod \( n \)
faster than 125 trial divisions?

Extra benefit:
no need to store the primes.
Plausible conjecture: if \( K \) is 
\[
\exp \sqrt{\left( \frac{1}{2} + o(1) \right) \log H \log \log H}
\]
then \( p - 1 \) divides \( \text{lcm}\{1, 2, \ldots, K\} \) for \( H/K^{1+o(1)} \) primes \( p \leq H \).

Same if \( p - 1 \) is replaced by order of 2 in \( \mathbb{F}_p^* \).

So uniform random prime \( p \leq H \) divides \( 2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \) with probability \( 1/K^{1+o(1)} \).

\((1.4 \ldots + o(1))K\) squarings mod \( n \) produce \( 2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \mod n \).

Similar time spent on trial division finds far fewer primes for large \( H \).
Safe primes

This means numbers are easy to factor if their factors $p_i$ have smooth $p_i - 1$.

To construct hard instances avoid such factors – that’s it?

ANSI does recommend using “safe primes”, i.e., primes of the form $2p' + 1$ when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.
Interlude: Addition on a clock

$x^2 + y^2 = 1$, parametrized by $x = \sin \alpha$, $y = \cos \alpha$.

Sum of $(x_1, y_1)$ and $(x_2, y_2)$ is $(x_1y_2 + y_1x_2, y_1y_2 - x_1x_2)$. 
Examples of clock addition:

\[
2 \begin{pmatrix} 3 & 4 \\ 5 & 5 \end{pmatrix} = \begin{pmatrix} 24 & 7 \\ 25 & 25 \end{pmatrix}.
\]

\[
3 \begin{pmatrix} 3 & 4 \\ 5 & 5 \end{pmatrix} = \begin{pmatrix} 117 & -44 \\ 125 & 125 \end{pmatrix}.
\]

\[
4 \begin{pmatrix} 3 & 4 \\ 5 & 5 \end{pmatrix} = \begin{pmatrix} 336 & -527 \\ 625 & 625 \end{pmatrix}.
\]

Many equivalent formulations. e.g. Clock addition represents multiplication of norm-1 elements of \( C = \mathbb{R}[i]/(i^2 + 1) \).

\[
(x, y) \mapsto y + ix;
\]

\[
(4/5 + 3i/5)^3 = -44/125 + 117i/125.
\]
The $p + 1$ factorization method
(1982 Williams)

Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the
$232792560$th multiple of
$(3/5, 4/5)$ in the group $\text{Clock}(\mathbb{Q})$.

The integer $S_2 = 5^{232792560}X$
is divisible by
82 of the primes $\leq 10^3$;
223 of the primes $\leq 10^4$;
455 of the primes $\leq 10^5$;
720 of the primes $\leq 10^6$;
etc.
Given an integer \( n \), compute \( 5^{232792560} \mod n \) and compute \( \gcd \) with \( n \), hoping to factor \( n \).

Many \( p \)'s not found by \( \mathbb{F}_p^* \) are found by \( \text{Clock}(\mathbb{F}_p) \).

If \(-1\) is not a square mod \( p \) and \( p + 1 \) divides \( 232792560 \) then \( 5^{232792560} \mod p = 0 \).

Proof: \( p \equiv 3 \pmod{4} \), so 
\[
(4/5 + 3i/5)^p = 4/5 - 3i/5 \quad \text{and so} \quad (p + 1)(3/5, 4/5) = (0, 1)
\]
in the group \( \text{Clock}(\mathbb{F}_p) \), so \( 232792560(3/5, 4/5) = (0, 1) \).
The elliptic-curve method

Fix $a \in \{6, 10, 14, 18, \ldots \}$.

Define $x_1 = 2$, $z_1 = 1$,

$x_{2i} = (x_i^2 - z_i^2)^2$,

$z_{2i} = 4x_iz_i(x_i^2 + ax_iz_i + z_i^2)$,

$x_{2i+1} = 4(x_ix_{i+1} - z_iz_{i+1})^2$,

$z_{2i+1} = 8(x_iz_{i+1} - z_ix_{i+1})^2$.

Define $S_a = \lfloor \text{lcm}\{1,2,3,\ldots,B_1\} \rfloor$.

Have now supplemented $S_1, S_2$ with $S_6, S_{10}, S_{14}, \text{etc.}$.

Variability of $a$ is important.

\ldots As many curves as you want!
Point of $x_i, z_i$ formulas:

If $z_i(a^2 - 4)(4a + 10) \notin p\mathbb{Z}$ then $i$th multiple of $(2, 1)$ on the elliptic curve $(4a + 10)y^2 = x^3 + ax^2 + x$ over $\mathbb{F}_p$ is $(x_i/z_i, \ldots)$.

If $(a^2 - 4)(4a + 10) \notin p\mathbb{Z}$ and $\text{lcm} \in (\text{order of } (2, 1))\mathbb{Z}$ then $S_a \in p\mathbb{Z}$.

Order of elliptic-curve group depends on $a$ but is always in $[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$. 
e.g. $B_1 = 20$, $a = 10$, $p = 105239$:

$p$ divides $S_{10}$.

Have $232792560(2, 1) = \infty$ on the elliptic curve
$50y^2 = x^3 + 10x^2 + x$ over $\mathbb{F}_p$.

In fact, $(2, 1)$ has order $13167 = 3^2 \cdot 7 \cdot 11 \cdot 19$ on this curve.

Number of $\mathbb{F}_p$-points of curve is $105336 = 2^3 \cdot 3^2 \cdot 7 \cdot 11 \cdot 19$. 
Good news (for the attacker):  
All primes $\leq H$  
seem to be found after a reasonable number of curves.

Plausible conjecture: if $B_1$ is  
$\exp \sqrt{\left(\frac{1}{2} + o(1)\right) \log H \log \log \log H}$  
then, for each prime $p \leq H$,  
a uniform random curve mod $p$  
has chance $\geq 1/B_1^{1+o(1)}$ to find $p$.

If a curve fails, try another.  
Find $p$ using, on average,  
$\leq B_1^{1+o(1)}$ curves;  
i.e., $\leq B_1^{2+o(1)}$ squarings.  
Time subexponential in $H$. 
Overview of ECM

Stage 1: Point $P$ on $E$ over $\mathbb{Z}/n$, compute $R = sP$ for $s = \text{lcm}\{2, 3, \ldots, B_1\}$.

Stage 2: Small primes $B_1 < q_1, \ldots, q_k \leq B_2$ compute $R_i = q_iR$.

If the order of $P$ on the curve modulo $p_i$ divides $sq_i$, $R_i$ is the neutral element.

Let $\phi(\text{neutral}) = 0$, $\phi(P) \neq 0$. (Example uses $\mathbb{Z}$-coordinate in Montgomery representation.)

Compute $\gcd\{n, \prod \phi(R_i)\}$. 
Edwards curves

\[ x^2 + y^2 = 1 + dx^2y^2 \]

field \( k \) with \( 2 \neq 0, \; d \notin \{0, 1\} \).

Edwards addition law:

\[
(x_1, y_1) + (x_2, y_2) = \left( \frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right).
\]

Neutral element: \((0, 1)\).

Negation: \(- (x_1, y_1) = (-x_1, y_1)\).

Projective point \((X_1 : Y_1 : Z_1)\)

represents \((X_1/Z_1, Y_1/Z_1)\).

Addition costs \(10M + 1S + 1M_d\).

Doubling costs \(3M + 4S\).
Example: \( x^2 + y^2 = 1 - 30x^2y^2 \)

\[
\begin{align*}
\text{neutral} &= (0, 1) \\
(-1, 0) \text{ of order 4} \\
(0, -1) \text{ of order 2}
\end{align*}
\]

Compare to standard Jacobian
\[
V^2 = U^3 - 3UW^4 + bW^6:
\]

Addition \( 11M + 5S \).

Edwards saves \( 4S + 1M - 1M_d \).

Doubling \( 3M + 5S \).

Edwards saves \( 1S \).
Example: $x^2 + y^2 = 1 - 30x^2y^2$

Compare to standard Jacobin

$V^2 = U^3 - 3UW^4 + bW^6$:

Addition $11M + 5S$.

Edwards saves $4S + 1M - 1M_d$.

Doubling $3M + 5S$.

Edwards saves $1S$. 
Twisted Edwards curves

\[ ax^2 + y^2 = 1 + dx^2y^2 \] with \( a \neq 0, d \neq 0, a \neq d. \]


Addition law: \((x_1, y_1) + (x_2, y_2) = \left( \frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \frac{y_1y_2-ax_1x_2}{1-dx_1x_2y_1y_2} \right).\)

Projective addition:

\[ 10M + 1S + 1M_d + 1M_a. \]

Projective doubling:

\[ 3M + 4S + 1M_a. \]
Advantages of twisted Edwards

- More flexible: not necessarily a point of order 4.
- Covers all Montgomery curves.
- Covers even more curves by applying a 2-isogeny.
- Saves time when $d$ is ratio of small integers.

Fewer mulmods per curve

Measurements of EECM-MPFQ for $B_1 = 1000000$:

$b = 1442099$ bits in

$s = \text{lcm}\{1, 2, 3, 4, \ldots, B_1\}$.

$P \mapsto sP$ is computed using

$1442085 \ (= 0.99999b) \ \text{DBL} + 98341 \ (0.06819b) \ \text{ADD}.$

These DBLs and ADDs use

$5112988\text{M} \ (3.54552b\text{M}) + 5768340\text{S} \ (3.99996b\text{S}) + 9635920\text{add} \ (6.68187b\text{add}).$
Compare to GMP-ECM 6.2.3:

\[ P \mapsto sP \] is computed using
\[ 2001915 \ (1.38820b) \ \text{DADD} + \\
194155 \ (0.13463b) \ \text{DBL}. \]

These DADDs and DBLs use
\[ 8590140M \ (5.95669bM) + \\
4392140S \ (3.04566bS) + \\
12788124\text{add} \ (8.86772b\text{add}). \]
Compare to GMP-ECM 6.2.3:

\[ P \mapsto sP \] is computed using
\[ 2001915 \, (1.38820b) \, \text{DADD} + 194155 \, (0.13463b) \, \text{DBL}. \]

These DADDs and DBLs use
\[ 8590140M \, (5.95669bM) + 4392140S \, (3.04566bS) + 12788124\text{add} \, (8.86772b\text{add}). \]

Could do better! 0.13463bM are actually 0.13463bM_d.

M_d: mult by curve constant.

Small curve, small \( P \), ladder
\[ \Rightarrow 4bM + 4bS + 2bM_d + 8b\text{add}. \]
EECM still wins.
HECM handles 2 curves using
\[2bM + 6bS + 8bM_d + \cdots\]
(1986 Chudnovsky–Chudnovsky, et al.); again EECM is better.
HECM handles 2 curves using
\[2bM + 6bS + 8bM_d + \cdots\]
(1986 Chudnovsky–Chudnovsky, et al.); again EECM is better.

What about NFS? \(B_1 = 587?\)

Measurements of EECM-MPFQ:

\[b = 839 \text{ bits in } s.\]

\(P \mapsto sP\) is computed using
\[833 \ (0.99285b) \ \text{DBL} + 131 \ (0.15614b) \ \text{ADD}.\]

These DBLs and ADDs use
\[3552M \ (4.23361bM) + 3332S \ (3.97139bS) + 6308\text{add} \ (7.51847b\text{add}).\]
Note: smaller window size in addition chain, so more ADDs per bit.

Compare to GMP-ECM 6.2.3:
Note: smaller window size in addition chain, so more ADDs per bit.

Compare to GMP-ECM 6.2.3:

$P \mapsto sP$ is computed using

$4785\text{M} \ (5.70322\text{bM}) +
2495\text{S} \ (2.97378\text{bS}) +
7053\text{add} \ (8.40644\text{badd})$.

Even for this small $B_1$, EECDM beats Montgomery ECM in operation count.
Notes on current stage 2:

1. EECM-MPFQ jumps through the \( j \)'s coprime to \( d_1 \).
   GMP-ECM: coprime to 6.

2. EECM-MPFQ computes Dickson polynomial values using Bos–Coster addition chains.
   GMP-ECM: ad-hoc, relying on arithmetic progression of \( j \).

3. EECM-MPFQ doesn’t bother converting to affine coordinates until the end of stage 2.

4. EECM-MPFQ uses NTL for poly arith in “big” stage 2.
Faster mulmods

ECM is bottlenecked by mulmods:
  • practically all of stage 1;
  • curve operations in stage 2 (pumped up by Dickson!);
  • final product in stage 2, except fast poly arith.

GMP-ECM does mulmods with the GMP library.

... but GMP has slow API, so GMP-ECM has $\geq 20000$ lines of new mulmod code.
Obviously EECM-MPFQ doesn’t include new mulmod code!
Obviously EECM-MPFQ doesn’t include new mulmod code!

MPFQ library (Gaudry–Thomé) does arithmetic in $\mathbb{Z}/n$ where number of $n$ words is known at compile time.

Better API than GMP: most importantly, $n$ in advance.

EECM-MPFQ uses MPFQ for essentially all mulmods.
GMP-ECM 6.2.3/GMP 4.3.2:
Tried 1000 curves, $B_1 = 2000$, typical 240-bit $n$, on 3.2GHz Phenom II x4.
Stage 1: $7.4 \cdot 10^6$ cycles/curve.
GMP-ECM 6.2.3/GMP 4.3.2:

Tried 1000 curves, $B_1 = 2000$, typical 240-bit $n$, on 3.2GHz Phenom II x4.

Stage 1: $7.4 \cdot 10^6$ cycles/curve.

EECM-MPFQ,
same 240-bit $n$, same CPU, 1000 curves, $B_1 = 2000$: $5.2 \cdot 10^6$ cycles/curve.

Some speedup from Edwards; some speedup from MPFQ.
What about stage 2?

GMP-ECM, 1000 curves, 
\( B_1 = 587, \ B_2 = 15366, \)
Dickson polynomial degree 1: 
\( 6.6 \cdot 10^6 \) cycles/curve. 
Degree 3: \( 9.5 \cdot 10^6. \)
What about stage 2?

GMP-ECM, 1000 curves, 
$B_1 = 587$, $B_2 = 15366$, 
Dickson polynomial degree 1: 
$6.6 \cdot 10^6$ cycles/curve. 
Degree 3: $9.5 \cdot 10^6$.

EECM-MPFQ, 1000 curves, 
$B_1 = 587$, $d_1 = 420$, range $20160$ 
for primes $420i \pm j$: 
$2.6 \cdot 10^6$ cycles/curve. 
Degree 3: $3.1 \cdot 10^6$. 
Summary: EECM-MPFQ uses fewer mulmods than GMP-ECM; takes less time than GMP-ECM; and finds more primes.
Summary: EECM-MPFQ uses fewer mulmods than GMP-ECM; takes less time than GMP-ECM; and finds more primes.

Are GMP-ECM and EECM-MPFQ fully exploiting the CPU? No!

Three recent efforts to speed up mulmods for ECM: Thorsten Kleinjung, for RSA-768; Alexander Kruppa, for CADO; and ours—see next slide.
Our latest mulmod speeds,
(working on update)
interleaving vector threads
with integer threads:

4×3GHz Phenom II 940:
202 \cdot 10^6 192-bit mulmods/sec.

4×2.83GHz Core 2 Quad Q9550:
114 \cdot 10^6 192-bit mulmods/sec.

6×3.2GHz Cell (Playstation 3):
102 \cdot 10^6 195-bit mulmods/sec.
$500 GTX 295
is one card with two GPUs;
60 cores; 480 32-bit ALUs.
Runs at 1.242GHz.

Our latest CUDA-EECM speed:
481 \cdot 10^6 210-bit mulmods/sec.

For \approx $2000 can build PC
with one CPU and four GPUs:
1300 \cdot 10^6 192-bit mulmods/sec.
Curve details – torsion points

Curve over $\mathbb{Q}$ has some torsion points: points of finite order.

All possible torsion groups (Mazur’s theorem):

$\mathbb{Z}/m$ for $m \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12\}$,

$\mathbb{Z}/2 \times \mathbb{Z}/2m$ for $m \in \{1, 2, 3, 4\}$.

If a point has finite order on the curve over $\mathbb{Q}$ then the point has the same finite order over $\mathbb{Z}/n$ and over $\mathbb{F}_p$. 
Don’t choose $P$ as a torsion point.

Minimize trouble by choosing curve with torsion $\mathbb{Z}/1$?
No: people try to use curves with many torsion points.

1987/1992 Montgomery, 1993 Atkin–Morain had suggested using torsion $\mathbb{Z}/12$ or $\mathbb{Z}/2 \times \mathbb{Z}/8$.

2008–2010 B.–Birkner–L.–Peters construct families of Edwards curves with torsion $\mathbb{Z}/12$ or $\mathbb{Z}/2 \times \mathbb{Z}/8$. 
Impact of large $\mathbb{Q}$-torsion

20 bit primes, stage 1 only.
Multiplications per prime found vs. $B_1$. 
Why people want big torsion

Standard series of heuristic approximations for “random” elliptic curve \( E \):

\[ \text{Pr[prime } p \in [1, R] \text{ is found by } E] \]
Why people want big torsion

Standard series of heuristic approximations for “random” elliptic curve $E$:

\[
\text{Pr[prime } p \in [1, R] \text{ is found by } E] \sim \text{Pr[prime } p \in [1, R] \text{ has smooth } \#\langle P \text{ in } E(\mathbb{F}_p) \rangle]
\]
Why people want big torsion

Standard series of heuristic approximations for “random” elliptic curve $E$:

$\Pr[\text{prime } p \in [1, R] \text{ is found by } E]$?
$\asymp \Pr[\text{prime } p \in [1, R] \text{ has smooth } \#\langle P \text{ in } E(F_p) \rangle]$?
$\asymp \Pr[p \in [1, R] \text{ has smooth } \#E(F_p)]$
Why people want big torsion

Standard series of heuristic approximations for “random” elliptic curve $E$:

Pr[prime $p \in [1, R]$ is found by $E$]

Pr[prime $p \in [1, R]$ has smooth $\#\langle P \text{ in } E(F_p) \rangle$]

Pr[$p \in [1, R]$ has smooth $\#E(F_p)$]

Pr[integer $\in [1, R]$ is smooth].
Standard series of heuristic approximations when ECM uses a curve \( E \) known to have \( t \) torsion points:

\[
\Pr[\text{prime } p \in [1, R] \text{ is found by } E] \]
Standard series of heuristic approximations when ECM uses a curve $E$ known to have $t$ torsion points:

\[ \Pr[\text{prime } p \in [1, R] \text{ is found by } E] \sim \Pr[\text{integer } \in t\mathbb{Z} \cap [1, R] \text{ is smooth}] \]
Standard series of heuristic approximations when ECM uses a curve $E$ known to have $t$ torsion points:

$\Pr[\text{prime } p \in [1, R] \text{ is found by } E]$?

$\Pr[\text{integer } \in t\mathbb{Z} \cap [1, R] \text{ is smooth}]$

$\Pr[\text{integer } \in \mathbb{Z} \cap [1, R/t] \text{ is smooth}].$
Standard series of heuristic approximations when ECM uses a curve $E$ known to have $t$ torsion points:

$Pr[\text{prime } p \in [1, R] \text{ is found by } E]$

$\Rightarrow Pr[\text{integer } \in t\mathbb{Z} \cap [1, R] \text{ is smooth}]$

$\Rightarrow Pr[\text{integer } \in \mathbb{Z} \cap [1, R/t] \text{ is smooth}].$

Larger $t \Rightarrow$ smaller $R/t$

$\Rightarrow$ larger $Pr.$
More primes per curve

Probability vs. $B_1$, 30-bit primes.
Influence of $d_1$

Multiplications per prime found vs. $B_1$; different $d_1$’s, same $E$. 
Faster twisted Edwards curves

Dual addition law by Hisil–Wong–Carter–Dawson.

\[
\begin{pmatrix}
\frac{x_1 y_1 + x_2 y_2}{ax_1 x_2 + y_1 y_2} & \frac{x_1 y_1 - x_2 y_2}{x_1 y_2 - y_1 x_2}
\end{pmatrix}
\]

Use extended coordinates \((X : Y : Z : T)\) with \(T = XY/Z\); bouncing between projective and extended coordinates.

Addition: \(9M + 1M_a\).

Only \(8M\) for \(a = -1\).

Doubling: \(3M + 4S + 1M_a\).

Note the addition speedup for \(a = -1\).
Faster ECM?

Let’s look closer at

\[-x^2 + y^2 = 1 - 30x^2y^2:\]
Singularity at infinity blows up to two points of order 2. EECM paper proved: arbitrary $d$ with $a = -1$ cannot achieve highest torsion such as $\mathbb{Z}/12$ and $\mathbb{Z}/2 \times \mathbb{Z}/8$. 
Singularity at infinity blows up to two points of order 2.
EECM paper proved:

arbitrary $d$ with $a = -1$
cannot achieve highest torsion such as $\mathbb{Z}/12$ and $\mathbb{Z}/2 \times \mathbb{Z}/8$. 
“Starfish on strike”

Is the sacrifice in torsion justified by the ADD speedup?
Modified EECM-MPFQ to support new curves.
“Starfish on strike”

Is the sacrifice in torsion justified by the ADD speedup?
Modified EECM-MPFPQ to support new curves.

Happy observation:
Gain in \# modular multiplications per curve outweighs loss in \# primes found per curve.
“Starfish on strike”

Is the sacrifice in torsion justified by the ADD speedup? Modified EECM-MPFQ to support new curves.

Happy observation:
Gain in # modular multiplications per curve outweighs loss in # primes found per curve.

Surprising phenomenon: \( \mathbb{Z}/6 \) \(-x^2 + y^2 = 1 + dx^2y^2\) family finds more primes than \( \mathbb{Z}/12 \).
“Starfish on strike”

Is the sacrifice in torsion justified by the ADD speedup?
Modified EECM-MPFQ to support new curves.

Happy observation:
Gain in \# modular multiplications per curve outweighs loss in \# primes found per curve.

Surprising phenomenon: \( \mathbb{Z}/6 -x^2 + y^2 = 1 + dx^2y^2 \) family finds more primes than \( \mathbb{Z}/12 \).

Even more benefit from precomputing best curves.
Number of $b$-bit primes found by 1000 different curves $-x^2 + \cdots$ with $\mathbb{Z}/2 \times \mathbb{Z}/4$ torsion:

<table>
<thead>
<tr>
<th>$b$</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>curve 1</td>
<td>15486</td>
<td>22681</td>
<td>46150</td>
</tr>
<tr>
<td>#1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>curve 2</td>
<td>14845</td>
<td>21745</td>
<td>43916</td>
</tr>
<tr>
<td>#2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>curve 3</td>
<td>14537</td>
<td>21428</td>
<td>43482</td>
</tr>
<tr>
<td>#3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#500</td>
<td>13706</td>
<td>19979</td>
<td>40993</td>
</tr>
<tr>
<td>#1000</td>
<td>13379</td>
<td>19475</td>
<td>40410</td>
</tr>
</tbody>
</table>
Number of $b$-bit primes found by 1000 different curves $x^2 + \cdots$ with $\mathbb{Z}/12$ torsion:

<table>
<thead>
<tr>
<th>$b$</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>curve</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>#1</td>
<td>16276</td>
<td>23991</td>
<td>48076</td>
</tr>
<tr>
<td>curve</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>#2</td>
<td>16275</td>
<td>23970</td>
<td>48028</td>
</tr>
<tr>
<td>curve</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>#3</td>
<td>16273</td>
<td>23965</td>
<td>48020</td>
</tr>
<tr>
<td>#500</td>
<td>15977</td>
<td>23590</td>
<td>47521</td>
</tr>
<tr>
<td>#1000</td>
<td>15313</td>
<td>22714</td>
<td>45987</td>
</tr>
</tbody>
</table>
Number of $b$-bit primes found by 1000 different curves $-x^2 + \cdots$ with $\mathbb{Z}/6$ torsion:

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>curve #1</td>
<td>[932]</td>
<td>$\frac{825}{2752}, \frac{1521}{1504}$</td>
<td>$336, 80$</td>
</tr>
<tr>
<td></td>
<td>16328</td>
<td>24160</td>
<td>$\frac{527}{67}$</td>
</tr>
<tr>
<td>curve #2</td>
<td>[94]</td>
<td>$\frac{825}{2752}, \frac{1521}{1504}$</td>
<td>$48378$</td>
</tr>
<tr>
<td></td>
<td>16289</td>
<td>24119</td>
<td></td>
</tr>
<tr>
<td>curve #3</td>
<td>[785]</td>
<td>[982]</td>
<td>$\frac{825}{2752}, \frac{1521}{1504}$</td>
</tr>
<tr>
<td></td>
<td>16287</td>
<td>24113</td>
<td>$48357$</td>
</tr>
<tr>
<td>#500</td>
<td>16037</td>
<td>23735</td>
<td>$47867$</td>
</tr>
<tr>
<td>#1000</td>
<td>15399</td>
<td>22790</td>
<td>$45828$</td>
</tr>
</tbody>
</table>
1. “ECM using Edwards curves.”
   Prototype software: GMP-EECM.
   New rewrite: EECM-MPFQ.

2. “ECM on graphics cards.”
   Prototype CUDA-EECM.

3. “The billion-mulmod-per-second PC.”
   Current CUDA-EECM,
   plus fast mulmods on
   Core 2, Phenom II, and Cell.

4. “Starfish on strike.”
   Integrated into EECM-MPFQ.

5. Not covered in this talk:
   early-abort ECM optimization.