Towards 60 digits with ECM

Paul Zimmermann

LORIA/INRIA Lorraine, Nancy, France

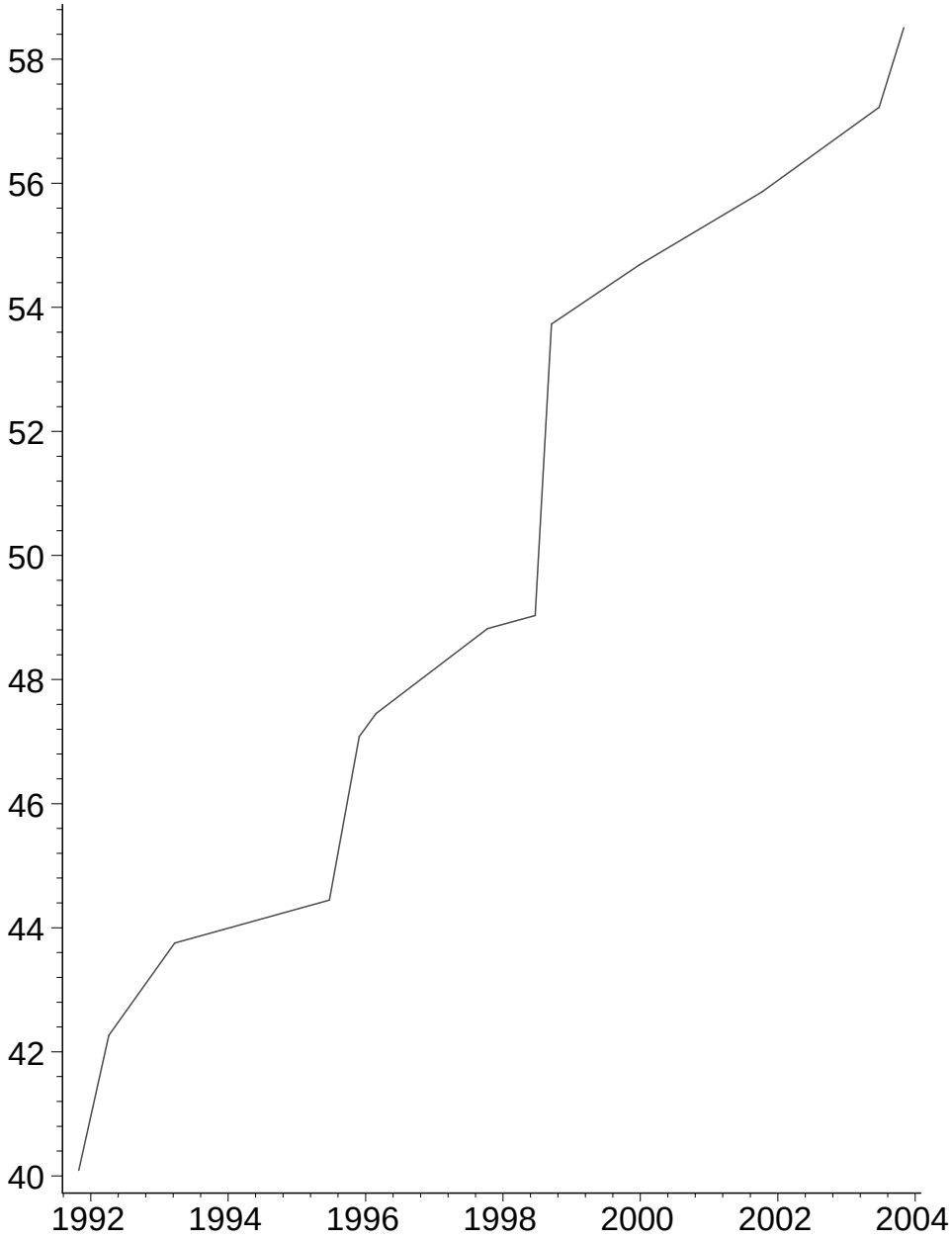
[presented by Herman te Riele]

(thanks to Richard Brent, Alexander Kruppa, Herman te Riele and Peter Montgomery for useful remarks)

ECM records

The following slide shows the number of digits of the ECM record versus year.

Source: `ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Richard.Brent/champs.txt`.


58
56
54
52
50
48
46
44
42
40
1992
1994
1996
1998
2000
2002
2004

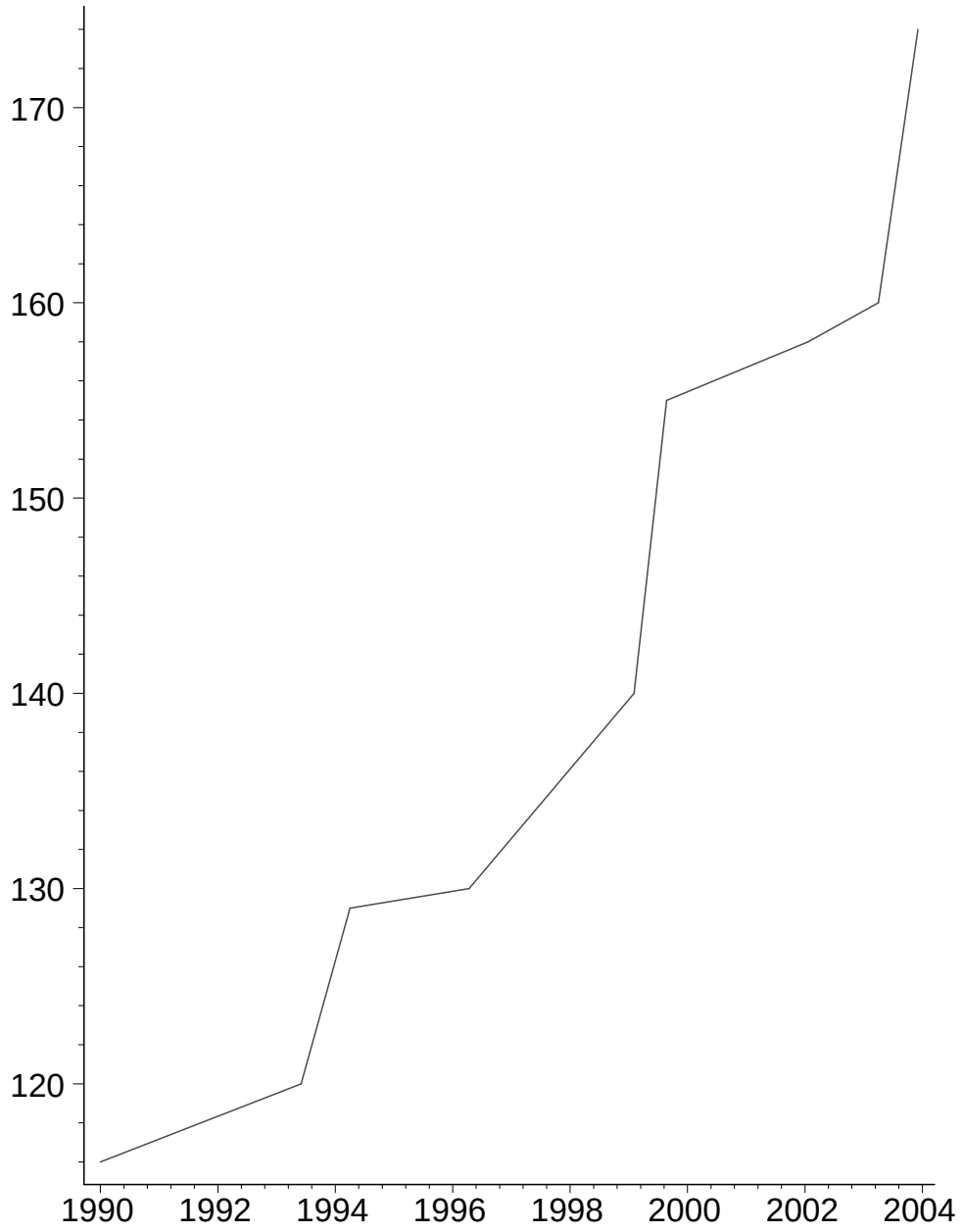
QS/GNFS records

The following slide shows the number of digits of the QS/GNFS record versus year.

Cf

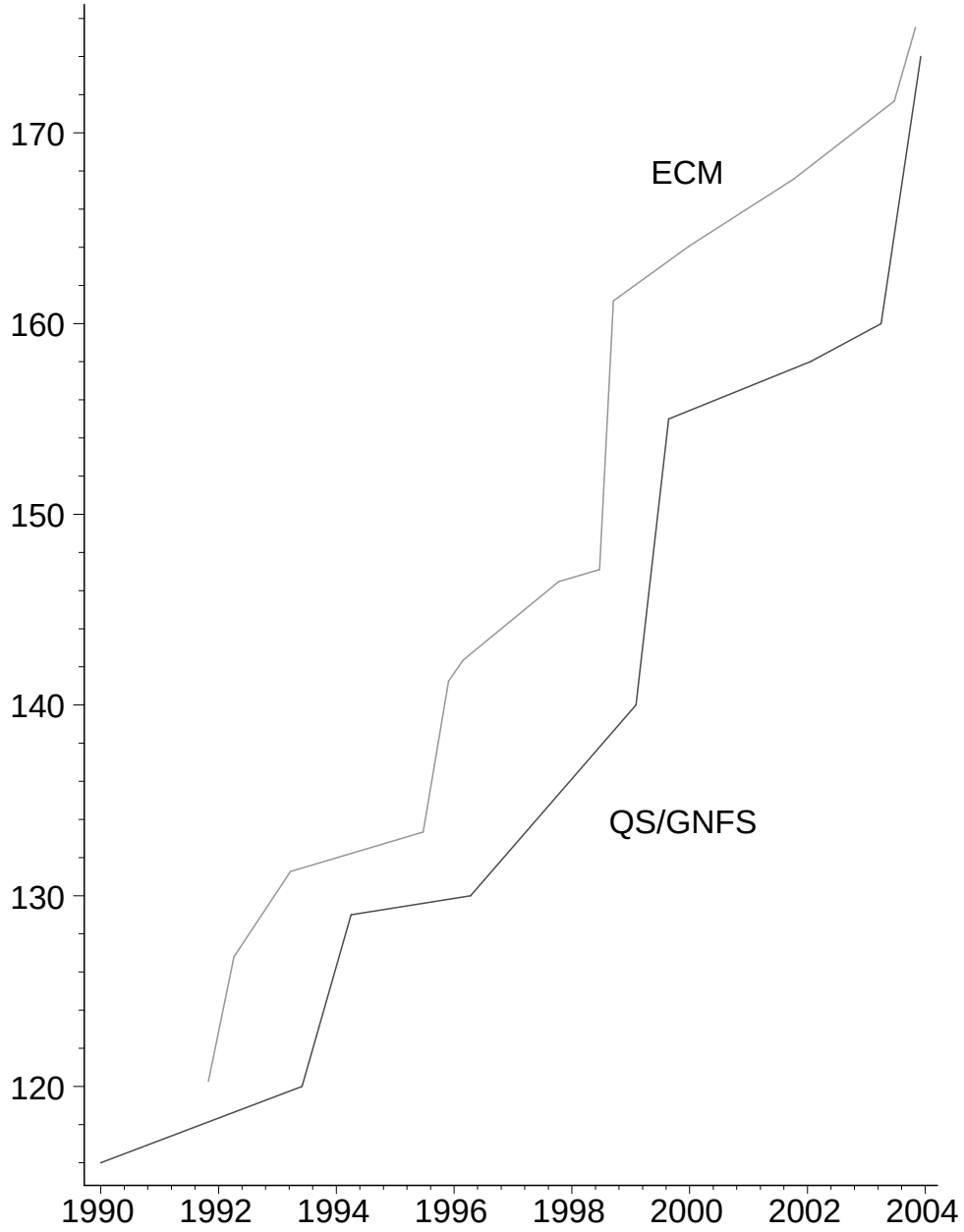
`http://www.loria.fr/~zimmerma/records/factor.html`

and `http://www.crypto-world.com/FactorRecords.html`.


170
160
150
140
130
120
1990
1992
1994
1996
1998
2000
2002
2004

QS/GNFS vs. ECM

The following slide compares the number of digits of the ECM record (multiplied by 3) with that of the QS/GNFS record.


170
160
150
140
130
120
1990
1992
1994
1996
1998
2000
2002
2004
ECM
QS/GNFS

Cheating

(Common work with Richard Brent)

With the current rules, it is rather easy to cheat with the ecm record.

0. Find a 60-digit factor p with gnfs or snfs.

1. Generate a random curve over p and compute its group order.

2. Loop to step 1 if the curve is not smooth enough.

For example, consider the 60-digit factor from 12,549M c180 found by CWI in September 2003 (here on a 3Ghz P4):

```
> GroupOrder := function (p, sigma)
     K:=GF(p); v:=K!(4*sigma); u:=K!(sigma^2-5);
     x:=u^3;   b:=4*x*v;       a:=(v-u)^3*(3*u+v);
     A:=a/b-2; x:=x/v^3;       b:=x^3 + A*x^2 + x;
     return FactoredOrder(EllipticCurve([0,b*A,0,b^2,0]));
  end function;
> p:=322410908070969630339041359359164154612901586904078700184707;
> time GroupOrder(p, 20031212);
[ <2, 2>, <3, 2>, <287167511077, 1>, <67038758223301, 1>,
  <190097716221949, 1>, <24471977313382286 29, 1> ]
Time: 19.620
```

According to
`http://www.loria.fr/~zimmerma/records/ecmnet.html`,
we need about 50,000 tries to find a curve with
$g_1 \leq 260000000$ and $g_2 \leq 2317234062121$ (default value for g_1 in GMP-ECM 5.0.3).

With Magma, it would take about one week to find such a smooth curve.
We have found in a few hours:

```
20041348 [ <2, 4>, <3, 2>, <391063, 1>, <1197631, 1>, <82011967, 1>,
  <126033329,1>, <1926338723, 1>, <4654300159, 1>, <51585518429, 1> ]
```

Without Cheating

- the composite should not be easy by gnfs: > 174 digits
- the composite should not be easy by snfs: > 244 digits

Among the 133 Cunningham composites of
$174 < d \leq 190$ digits, there are 97 of SNFS difficulty
> 244, of 183 digits on average.

Consider numbers of 180 digits. Assume all factors up to 45 digits have been found.

According to a theorem of Vershik (cf Brent's F10 paper), the proportion of factorizations with a smaller factor of d digits behaves like $1/d$. The probability of having $d = 60$ is thus:

```
> f := d -> 1/d:
> evalf(f(60)/add(f(d),d=45..90));
                          .02347931378
```

Thus with 97 composites, we expect about $97 \cdot 0.0234 \approx 2.3$ factors of 60 digits (and about 42 factors of 45 to 60 digits).

Optimal Parameters

Alexander Kruppa designed a program `countsmooth` to estimate the proportion of smooth curves:

```
% ./countsmooth -B1 260000000 -B2 2.4e12 -N 2.64e58 \
    -tests 10000000 -S 120 -D 2282280 -a -1 -theo
B1-smooth: 13, B2-smooth: 166,
   found by Brent-Suyama: 29.870082, Total: 208
```

This means that with a step 1 bound of $260M$, a step 2 bound of $2.4 \cdot 10^{12}$, and B.-S. polynomial Dickson(120), for a p60 with a group order divisible by 12 ($2.64e58 \approx 10^{59.5}/12$), about 208 curves out of 10^7 will be successful (13 in step 1, 166 in step 2, 29 with Brent-Suyama).

Estimated Time

For one curve with $B_1 = 260,000,000$, $B_2 = 2.3 \cdot 10^{12}$, and polynomial $D_{120}(x)$ for Brent-Suyama's extension, the estimated time on a 3Ghz P4 for a c180 is about 1.9 hour for step 1, and 1.7 hour for step 2.

With an expected 48000 curves, this gives 7200 cpu days, or about 20 cpu years, to find a 60-digit factor in a composite having one.

With the 97 Cunningham composites from 175 to 190 digits, this gives an estimate of about 840 cpu years to find a p60.

But in our hunt for a p60, we may also find a p61, p62, ... With the above ECM parameters, we have estimated the probability to find a p61, ..., p67, and the correction factor is:

```
> f := d -> 1/d:
> l := [[60,208],[61,154],[62,102],[63,82],[64,55],
        [65,26],[66,26],[67,15]]:
> evalf(add(l[i][2]*f(l[i][1]),i=1..nops(l))
       /(l[1][2]*f(l[1][1])));
                    3.119769350
```

This gives an estimate of 270 cpu years to find a (prime) factor of 60 digits or more.

Current Work on GMP-ECM

Alex Kruppa has implemented `countsmooth`, to determine optimal parameters (see above).

Laurent Fousse (new developer) is implementing the new algorithm for POLYEVAL based on Tellegen's principle. Works in $\frac{3}{2}M(n)\log n$ instead of $\frac{7}{2}M(n)\log n$ for Montgomery's algorithm. Requires to implement the "middle product" for Karatsuba, Toom-Cook 3-way and 4-way.

About Tellegen's principle

For any operation that can be written in matrix form $C = M \cdot B$, the *transposed* operation corresponds to the transposed matrix: $B = M^t \cdot C$.

Tellegen's principle says that if an algorithm exists to compute $C = M \cdot B$, then this algorithm can be transposed to an algorithm to compute $B = M^t \cdot C$ in the *same number of multiplications*.

Example: the multiplication by $a_0 + a_1 x + ... + a_{n-1} x^{n-1}$

corresponds to a Toeplitz matrix:

$$\begin{pmatrix} a_0 & 0 & \dots & 0 \\ a_1 & a_0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \dots & a_0 \\ 0 & a_{n-1} & \dots & a_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & a_{n-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{2n-4} \\ c_{2n-3} \\ c_{2n-2} \end{pmatrix}$$

where $c_k = \sum_{i+j=k} a_i b_j$.

The multiplication by the transposed matrix (with reverting a) corresponds to the *middle product* (Hanrot, Quercia, Zimmermann, *The Middle Product Algorithm, I. Speeding up the division and square root of power series.*, to appear in AAECC):

$$\begin{pmatrix} a_{n-1} & \dots & a_0 & 0 & \dots & 0 \\ 0 & \ddots & a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & a_{n-1} & a_{n-2} & \dots & a_0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{2n-3} \\ c_{2n-2} \end{pmatrix} = \begin{pmatrix} b_{n-1} \\ b_n \\ \vdots \\ b_{2n-2} \end{pmatrix}$$

where $b_k = \sum_{i+j=k} a_i c_j$.

Bostan, Lecerf and Schost (*Tellegen's Principle into Practice*, ISSAC'03, `http://www.stix.polytechnique.fr/~bostan/publications/BoLeSc03.pdf`) have extended that to other operations, in particular they have shown:

• the multipoint evaluation transposes into computing the first $m+1$ coefficients of the Taylor expansion of

$$\sum_{j=0}^{n-1} \frac{c_j}{1 - a_j x} \tag{1}$$

• (1) can be computed in $O(\frac{3}{2} M(n) \log n)$ using a new algorithm `UpTree`

• thus the transposed algorithm `TUpTree` gives the multipoint evaluation in $O(\frac{3}{2} M(n) \log n)$.