Lecture 6

Integer Factorisation, Elliptic Curves and Fermat Numbers*

*Six lectures on Algorithms, Trinity term 1999.
 lec06

Abstract

We outline the integer factorisation algorithms ECM, MPQS and NFS, and then compare their expected performance on "typical" or "random" large integers. Finally, we illustrate some of the conclusions by giving a brief historical summary of attempts to factor Fermat numbers.

Outline

Part 1: ECM, MPQS and NFS

- Notation and definitions
- Elliptic curves over finite fields
- The elliptic curve method (ECM)
- The quadratic sieve (QS and MPQS)
- The number field sieve (NFS)
  - Special (SNFS)
  - General (GNFS)

Part 2: Comparison Theorems

- Comparison of ECM and MPQS
- Comparison of ECM and GNFS

Part 3: History

- Attempts to factor Fermat numbers

Notation

n and N always denote positive integers.

p_n denotes a prime number with n decimal digits, e.g. $p_3 = 163$. Similarly, c_n denotes a composite number with n decimal digits, e.g. $c_4 = 1729$.

log or ln denotes the natural logarithm, lg or $\log_2$ denotes the logarithm to base 2.

Almost Always and Almost Never

If $P(n)$ is a predicate, we say that $P(n)$ holds *almost always* if

$$\lim_{N\to\infty} \frac{|\{n \le N : P(n)\}|}{N} = 1$$

and we say that $P(n)$ holds *almost never* if

$$\lim_{N\to\infty} \frac{|\{n \le N : P(n)\}|}{N} = 0 \,.$$

Example (Erdős–Kac): For any $\varepsilon > 0$, n almost always has between $(1-\varepsilon)\log\log n$ and $(1+\varepsilon)\log\log n$ prime factors.

Elliptic Curves Over Finite Fields

A curve of the form

$$y^2 = x^3 + ax + b \qquad (1)$$

over some field F is known as an *elliptic curve*. A more general cubic in x and y can be reduced to the form (1), which is known as the Weierstrass normal form, by rational transformations, provided $\mathrm{char}(F) \neq 2$ or 3.

There is a well-known way of defining an Abelian group $(G, +)$ on an elliptic curve over a field. If $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ are points on the curve, then the point $P_3 = (x_3, y_3) = P_1 + P_2$ is defined by –

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2,\ \lambda(x_1 - x_3) - y_1) \ ,$$

where

$$\lambda = \begin{cases} (3x_1^2 + a)/(2y_1) & \text{if } P_1 = P_2 \\ (y_1 - y_2)/(x_1 - x_2) & \text{otherwise.} \end{cases}$$

The zero element in G is the "point at infinity", (∞, ∞). We write it as 0.

Geometric Interpretation

The geometric interpretation of "+" is straightforward: the straight line P_1P_2 intersects the elliptic curve at a third point $P_3' = (x_3, -y_3)$, and P_3 is the reflection of P_3' in the x-axis.

More elegantly, if a straight line intersects the elliptic curve at three points Q_1, Q_2, Q_3 then

$$Q_1 + Q_2 + Q_3 = 0 \ .$$

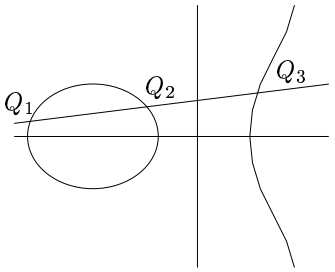


Figure 1: The Group Operation

Brief Description of ECM

The *elliptic curve method* (ECM) for integer factorisation was discovered by H. W. Lenstra, Jr. in 1985. Various practical refinements were suggested by Montgomery, Suyama, and others.

ECM uses groups defined by pseudo-random elliptic curves over $\mathrm{GF}(p)$, where $p > 3$ is the prime factor we hope to find. (Fortunately, we don't need to know p in advance.) By a theorem of Hasse (1934), the group order g for an elliptic curve over $\mathrm{GF}(p)$ satisfies

$$|g - p - 1| < 2\sqrt{p} \ .$$

By a result of Deuring, all g satisfying this inequality are possible.

ECM is similar to an earlier method, Pollard's "$p-1$" method, but the $p-1$ method has the disadvantage that the group is fixed and the method fails if $p-1$ has a large prime factor. We can think of ECM as a "randomised" version of the $p-1$ method.

Lenstra's Analysis of ECM

Consider applying ECM to a composite integer N with smallest prime factor p. Making an unproved but plausible assumption regarding the distribution of prime factors of random integers in "short" intervals, Lenstra showed that ECM will find p in an expected number

$$W(p) = \exp\left(\sqrt{(2 + o(1)) \log p \log\log p}\right)$$

of multiplications (mod N), where the "$o(1)$" term tends to zero as $p \to \infty$.

In Lenstra's algorithm the field F is the finite field $\mathrm{GF}(p)$ of p elements, where p is a prime factor of N. Since p is not known in advance, computation is performed in the ring Z/NZ of integers modulo N rather than in $\mathrm{GF}(p)$. We can regard this as using a redundant group representation.

One Trial of ECM

A *trial* (or *curve*) is the computation involving one random group G. The steps involved are –

1. Choose a parameter B.
2. Choose x_0, y_0 and a randomly in $[0, N)$. This defines $b = y_0^2 - (x_0^3 + ax_0) \bmod N$. Set $P \leftarrow P_0 = (x_0, y_0)$.
3. For each prime $\leq B$ take its maximal power $q \leq B$ and set $P \leftarrow qP$ in the group G defined by a and b.

If $P = 0$ then the trial succeeds as a factor of N will have been found during an attempt to compute an inverse mod N. Otherwise the trial fails.

The work involved in a trial is $O(B)$ group operations. There is a tradeoff involved in the choice of B, as a trial with large B is expensive, but a trial with small B is unlikely to succeed.

Optimal Choice of B

Making Lenstra's plausible assumption, one may show that the optimal choice of B is $B = p^{1/\alpha}$, where

$$\alpha \sim (2 \ln p / \ln \ln p)^{1/2} .$$

It follows that the expected run time is

$$T = p^{2/\alpha + o(1/\alpha)} .$$

The exponent $2/\alpha$ should be compared with 1 (for trial division) or 1/2 (for Pollard's "rho" method).

A Practical Problem

The optimal choice of B depends on the size of the factor p. Since p is unknown, we have to guess or use some sort of adaptive strategy.

Fortunately, the expected performance of ECM is not very sensitive to the choice of parameters, so the precise strategy does not matter much.

The Second Phase

Both the Pollard "$p - 1$" and Lenstra elliptic curve algorithms can be speeded up by the addition of a second phase. The idea of the second phase is to find a factor in the case that the first phase terminates with a group element $P \neq 0$, such that $|\langle P \rangle|$ is reasonably small (say $O(B^2)$). Here $\langle P \rangle$ is the cyclic group generated by P.

There are several possible implementations of the second phase. One of the simplest uses a pseudorandom walk in $\langle P \rangle$. By the birthday paradox argument, there is a good chance that two points in the random walk will coincide after $O(|\langle P \rangle|^{1/2})$ steps, and when this occurs a nontrivial factor of N can usually be found.

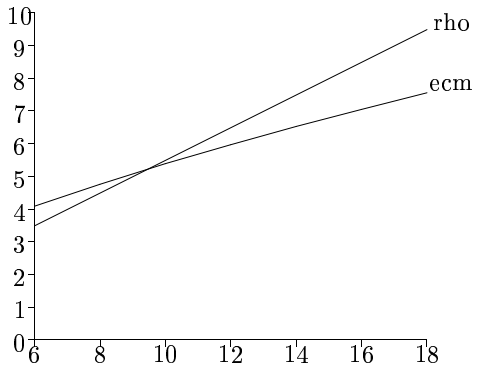
Expected Performance of ECM

In Table 1 we give a small table of $\log_{10} W$ for factors of D decimal digits. The precise figures depend on assumptions about the implementation.

Table 1: Expected work for ECM

digits D	$\log_{10} W$
20	7.35
30	9.57
40	11.49
50	13.22
60	14.80

Comparison with Pollard "rho"



$\log_{10} W$ versus decimal digits in factor

Because of the overheads involved with ECM, a simpler algorithm such as Pollard's "rho" is preferable for finding factors of up to about ten decimal digits, but for larger factors the advantage of ECM becomes apparent.

ECM Example

ECM can routinely find factors p of size about 30 decimal digits. The largest factor known to have been found by ECM is the 53-digit factor

$$p_{53} = 53625112691923843508117942\backslash 31151642817302190330034456 7$$

of $2^{677} - 1$, found by Conrad Curry in September 1998 using a program written by George Woltman and running on 16 Pentiums. The group order for the lucky trial was

$$g = 2^4 \cdot 3^9 \cdot 3079 \cdot 152077 \cdot 172259 \cdot 1067063 \cdot 3682177 \cdot 3815423 \cdot 8867563 \cdot 15880351$$

We expect only one in $2,400,000$ curves to have such a "smooth" group order.

Quadratic Sieve Algorithms

Quadratic sieve algorithms belong to a large class of algorithms which try to find two integers x and y such that $x \neq \pm y \pmod N$ but

$$x^2 = y^2 \pmod N . \qquad (2)$$

Once such x and y are found, then GCD $(x - y, N)$ is a nontrivial factor of N. One way to find x and y satisfying (2) is to find a set of *relations* of the form

$$u_i^2 = v_i^2 w_i \pmod N, \qquad (3)$$

where the w_i have all their prime factors in a moderately small set of primes (called the *factor base*). Each relation (3) gives a row in a matrix M whose columns correspond to the primes in the factor base.

Linear Algebra mod 2

Once enough rows have been generated, we can use sparse Gaussian elimination in GF(2) to find a linear dependency (mod 2) between a set of rows of M. Multiplying the corresponding relations now gives an expression of the form (2). With probability at least $1/2$, we have $x \neq \pm y \bmod N$ so a nontrivial factor of N will be found. If not, we need to obtain a different linear dependency and try again.

Sieving

In quadratic sieve algorithms the numbers w_i are the values of one (or more) polynomials with integer coefficients. This makes it easy to find relations by *sieving*. The inner loop of the sieving process has the form

while $j < bound$ **do**
begin
$s[j] \leftarrow s[j] + c$;
$j \leftarrow j + q$;
end

Here *bound* depends on the size of the (single-precision real) sieve array s, q is a small prime or prime power, and c is a single-precision real constant depending on q ($c = \Lambda(q) = \log p$ if $q = p^e$, p prime).

It is possible to use scaling to avoid floating point additions, which is desirable on a small processor without floating-point hardware.

MPQS

MPQS is a quadratic sieve method which uses several polynomials to improve the efficiency of sieving (an idea of Montgomery). MPQS can, under plausible assumptions, factor a number N in time

$$\Theta(\exp(c(\ln N \ln\ln N)^{1/2})) ,$$

where $c \sim 1$. The constants involved are such that MPQS is usually faster than ECM if N is the product of two primes which both exceed $N^{1/3}$. This is because the inner loop of MPQS involves only single-precision operations.

P-MPQS and PP-MPQS

In the "one large prime" (P-MPQS) variation w_i is allowed to have one prime factor exceeding B (but not too much larger than B). This is analogous to the second phase of ECM and gives a similar performance improvement.

In the "two large prime" (PP-MPQS) variation w_i can have two prime factors exceeding B – this gives a further performance improvement at the expense of higher storage requirements.

MPQS Examples

MPQS has been used to obtain many impressive factorisations. Arjen Lenstra and Mark Manasse (with many assistants scattered around the world) have factored several numbers larger than 10^{100}. For example, the 116-decimal digit number $(3^{329}+1)$/(known small factors) was split into a product of 50-digit and 67-digit primes. The final factorisation is

$$\begin{aligned} 3^{329}+1 \;=\; & 2^2 \cdot 547 \cdot 16921 \cdot 256057 \cdot 36913801 \cdot \\ & 177140839 \cdot 1534179947851 \cdot \\ & 24677078822840014266652 77\backslash \\ & 90367680629183726974352 41 \cdot p_{67} \end{aligned}$$

Such factorisations require many years of CPU time, but a real time of only a month or so because of the number of different processors which are working in parallel.

The Magic Words are ···

At the time of writing, the largest number factored by MPQS is the 129-digit "RSA Challenge" number RSA129. It was factored in 1994 by Atkins *et al.* RS&A had predicted in *Scientific American* that it would take millions of years to factor RSA129.

The factors of RSA129 allow decryption of a 'secret' message from RS&A. Using the decoding scheme $01 = A, 02 = B, \ldots, 26 = Z$, and 00 a space between words, the decoded message reads

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

It is certainly feasible to factor larger numbers by MPQS, but for numbers of more than about 110 decimal digits GNFS is faster. For example, to factor RSA129 by MPQS required 5000 Mips-years, but to factor the slightly larger number RSA130 by GNFS required only 1000 Mips-years.

The Special Number Field Sieve (SNFS)

Most of our numerical examples have involved numbers of the form

$$a^e \pm b\ , \tag{4}$$

for small a and b, although the ECM and MPQS factorisation algorithms do not take advantage of this special form.

The *special number field sieve* (SNFS) is a relatively new (*c.* 1990) algorithm which does take advantage of the special form (4). In concept it is similar to the quadratic sieve algorithm, but it works over an algebraic number field defined by a, e and b.

The details are rather technical and depend on concepts from algebraic number theory, so we simply give two examples to show the power of the algorithm.

SNFS Example 1

Consider the 155-decimal digit number

$$F_9 = N = 2^{2^9} + 1$$

as a candidate for factoring by SNFS. Note that $8N = m^5 + 8$, where $m = 2^{103}$. We may work in the number field $Q(\alpha)$, where α satisfies

$$\alpha^5 + 8 = 0,$$

and in the ring of integers of $Q(\alpha)$. Because

$$m^5 + 8 = 0 \pmod N,$$

the mapping $\phi : \alpha \mapsto m \bmod N$ is a ring homomorphism from $Z[\alpha]$ to Z/NZ.
The idea is to search for pairs of small coprime integers u and v such that both the algebraic integer $u + \alpha v$ and the (rational) integer $u + mv$ can be factored. The factor base now includes prime ideals and units as well as rational primes.

Example 1 continued

Because

$$\phi(u + \alpha v) = (u + mv) \pmod N,$$

each such pair gives a relation analogous to (3). The prime ideal factorisation of $u + \alpha v$ can be obtained from the factorisation of the *norm* $u^5 - 8v^5$ of $u + \alpha v$. Thus, we have to factor simultaneously two integers $u + mv$ and $|u^5 - 8v^5|$. Note that, for moderate u and v, both these integers are much smaller than N, in fact they are $O(N^{1/d})$, where $d = 5$ is the degree of the algebraic number field.

Using these and related ideas, Lenstra *et al* factored F_9 in June 1990, obtaining

$$\begin{aligned} F_9 \;=\; & 2424833 \cdot \\ & 74556028256478842083373 9\backslash \\ & 57362004549187833663426 57 \cdot p_{99}\ , \end{aligned}$$

where p_{99} is an 99-digit prime, and the 7-digit factor was already known (although SNFS was unable to take advantage of this).

Details

The collection of relations took less than two months on a network of several hundred workstations. A sparse system of about 200,000 relations was reduced to a dense matrix with about 72,000 rows. Using Gaussian elimination, dependencies (mod 2) between the rows were found in three hours on a Connection Machine. These dependencies implied equations of the form $x^2 = y^2 \bmod F_9$. The second such equation was nontrivial and gave the desired factorisation of F_9.

SNFS Example 2

The current SNFS record is the 211-digit number $10^{211} - 1$, factored early in 1999 by a collaboration called "The Cabal". In fact, $10^{211} - 1 = 3^2 \cdot p_{93} \cdot p_{118}$, where

$$\begin{aligned} p_{93} \;=\; & 692624557324389620662 78\backslash \\ & 232267733671113810848 25\backslash \\ & 882817397343755705064 92\backslash \\ & 391931849524636731866879 \end{aligned}$$

and p_{118} may be found by division.

Details

The factorisation of $N = 10^{211} - 1$ used two polynomials

$$f(x) = x - 10^{35}$$

and

$$g(x) = 10x^6 - 1$$

with common root $m = 10^{35} \bmod N$. After sieving and reduction a sparse matrix over GF(2) was obtained with about 4.8×10^6 rows and weight (number of nonzero entries) about 2.3×10^8, an average of about 49 nonzeros per row. Montgomery's block Lanczos program took 121 hours on a Cray C90 to find 64 dependencies. Finally, the square root program needed 15.5 hours on one CPU of an SGI Origin 2000, and three dependencies to find the two prime factors.

The General Number Field Sieve (GNFS)

The *general number field sieve* (GNFS or just NFS) is a logical extension of the special number field sieve (SNFS).

When using SNFS to factor an integer N, we require two polynomials $f(x)$ and $g(x)$ with a common root m mod N but no common root over the field of complex numbers.

If N has the special form $a^e \pm b$ then it is usually easy to write down suitable polynomials with small coefficients, as illustrated by the two examples given above.

If N has no special form, but is just some given composite number, we can also find $f(x)$ and $g(x)$, but they no longer have small coefficients.

The "Base m" Method

Suppose that $g(x)$ has degree $d > 1$ and $f(x)$ is linear. d is chosen empirically, but it is known from theoretical considerations that the optimum value is

$$d \sim \left(\frac{3 \ln N}{\ln \ln N} \right)^{1/3} .$$

We choose $m = \lfloor N^{1/d} \rfloor$ and write

$$N = \sum_{j=0}^{d} a_j m^j$$

where the a_j are "base m digits" and $a_d = 1$. Then, defining

$$f(x) = x - m, \quad g(x) = \sum_{j=0}^{d} a_j x^j ,$$

it is clear that $f(x)$ and $g(x)$ have a common root m mod N. This method of polynomial selection is called the "base m" method.

Other Ingredients of GNFS

Having found two appropriate polynomials, we can proceed as in SNFS, but many difficulties arise because of the large coefficients of $g(x)$. The details are the subject of several theses.

Suffice it to say that the difficulties can be overcome and the method works!

Due to the constant factors involved, GNFS is slower than MPQS for numbers of less than about 110 decimal digits, but faster than MPQS for sufficiently large numbers, as anticipated from the theoretical run times.

Some Difficulties Overcome

Some of the difficulties which had to be overcome to turn GNFS into a practical algorithm are:

- Polynomial selection. The "base m" method is not very good. Brian Murphy has shown how a very considerable improvement (by a factor of more than ten for number of 140 digits) can be obtained.
- Linear algebra. After sieving a very large, sparse linear system over GF(2) is obtained, and we want to find dependencies amongst the rows. It is not practical to do this by Gaussian elimination because the "fill in" is too large. Montgomery showed that the Lanczos method could be adapted for this purpose. (This is nontrivial because a nonzero vector x over GF(2) can be orthogonal to itself, i.e. $x^T x = 0$.) His program works with blocks width 64.

Difficulties continued

- Square roots. The final stage of GNFS involves finding the square root of a (very large) product of algebraic numbers. Once again, Montgomery found a way to do this.
- An idea of Adleman, using quadratic characters, is essential to ensure that the desired square root exists with high probability.

Scalability of GNFS

At present, the main obstacle to a fully parallel and scalable implementation of GNFS is the linear algebra. Montgomery's block Lanczos program runs on a single processor and requires enough memory to store the sparse matrix. In principle it should be possible to distribute the block Lanczos solution over several processors of a parallel machine, but the communication to computation ratio will be high. There is a tradeoff here – by increasing the time spent on sieving we can reduce the size and weight of the resulting matrix.

If special hardware is built for sieving, as pioneered by Lehmer and recently proposed (in more modern form) by Shamir, the linear algebra will become relatively more important. The argument is similar to Amdahl's law: no matter how fast sieving is done, we can not avoid the linear algebra.

RSA140

At the time of writing, the largest number factored by GNFS is the 140-digit RSA Challenge number RSA140. It was split into the product of two 70-digit primes in February, 1999, by a team coordinated from CWI, Amsterdam. The amount of computer time required to find the factors was about 2000 Mips-years.

The two polynomials used were

$$f(x) = x - 3443565780924253695177900 7$$

and

$$\begin{aligned} g(x) \;=\; & +439682082840x^5 \\ & +390315678538960x^4 \\ & -7387325293892994572x^3 \\ & -1902715324374298871 4824x^2 \\ & -6344102569446461791 3930613x \\ & +3185539170714743503 92223507494 . \end{aligned}$$

Polynomial Selection

The polynomial $g(x)$ was chosen (by the method of Murphy and Montgomery) to have a good combination of two properties: being unusually small over the sieving region, and having unusually many roots modulo small primes and small prime powers. The effect of the second property alone makes $g(x)$ as effective at generating relations as a polynomial chosen at random for an integer of 121 decimal digits. In effect judicious polynomial selection removed at least 19 digits from RSA140, making it much easier to factor.

The polynomial selection took 2000 CPU-hours on four 250 MHz SGI Origin 2000 processors. This is about 200 Mips-years, or 10% of the total factorisation time. It might have been better to spend a larger fraction of the time on polynomial selection – this is an interesting tradeoff.

Sieving

Sieving was done on about 125 SGI and Sun workstations running at 175 MHz on average, and on about 60 PCs running at 300 MHz on average. The total amount of CPU time spent on sieving was 8.9 CPU-years (about 1900 Mips-years).

The Linear Algebra

The resulting matrix had about 4.7×10^6 rows and weight about 1.5×10^8 (about 32 nonzeros per row). Using Montgomery's block Lanczos program, it took almost 100 CPU-hours and 810 MB of memory on a Cray C916 to find 64 dependencies among the rows of this matrix. Calendar time for this was five days.

RSA155

At the time of writing, an attempt to factor the 512-bit number RSA155 is well underway. I am willing to bet £100 that it will be factored before the year 2000.

Summary of Part 1

I have sketched some algorithms for integer factorisation. The most important are ECM, MPQS and GNFS. The algorithms draw on results in elementary number theory, algebraic number theory and probability theory. As well as their inherent interest and applicability to other areas of mathematics, advances in public key cryptography have lent them practical importance.

Despite much progress in the development of efficient algorithms, our knowledge of the complexity of factorisation is inadequate. We would like to find a polynomial time factorisation algorithm or else prove that one does not exist. Until a polynomial time algorithm is found or a quantum computer capable of running Shor's algorithm is built, large factorisations will remain an interesting challenge.

Predictions

From the predicted run time for GNFS, we would expect RSA155 to take 6.5 times as long as RSA140. On the other hand, Moore's law predicts that circuit densities will double every 18 months or so. Thus, as long as Moore's law continues to apply and results in correspondingly more powerful parallel computers, we expect to get almost 4 decimal digits per year improvement in the capabilities of GNFS, without any algorithmic improvements. A similar argument applies to ECM, for which we expect slightly more than 1 decimal digit per year in the size of factor found.

(When) Is RSA Doomed ?

512-bit RSA keys are clearly insecure. 1024-bit RSA keys should remain secure for at least thirty years, barring the unexpected (but unpredictable) discovery of a completely new algorithm which is better than GNFS, or the development of a practical quantum computer.

Part 2: Comparison of Algorithms

We now compare the expected behaviour of ECM, MPQS and GNFS on large, "random" or "typical" integers (*not* integers chosen by a cryptographer).

If N is a (large) integer with prime factors $p_1 \geq p_2 \geq \ldots$, we *assume* that the expected time to factor N by these three methods is $T_{ECM}(N), T_{MPQS}(N), T_{GNFS}(N)$ respectively, where

$$\log T_{ECM} = \sqrt{(2+o(1)) \log p_2 \log\log p_2}$$

$$\log T_{MPQS} = \sqrt{(1+o(1)) \log N \log\log N}$$

$$\log T_{GNFS} = \sqrt[3]{(c+o(1)) \log N (\log\log N)^2}$$

Here c is some positive constant, and the $o(1)$ terms are as $p_2 \to \infty$ or $N \to \infty$.

ECM and MPQS

Theorem

$T_{MPQS}(N) > e^{\sqrt{\log N}} T_{ECM}(N)$ almost always.

Idea of Proof

$$\left(\frac{\log T_{MPQS}}{\log T_{ECM}}\right)^2 \geq \frac{\log N}{(2+o(1)) \log p_2}$$

but from the known distribution of $\log p_2 / \log N$ this is at least $1+\varepsilon$ with probability at least $1 - O(\varepsilon^2)$. Thus, the Theorem holds if $e^{\sqrt{\log N}}$ is replaced by any $f(N)$ satisfying

$$\log f(N) = o\left(\sqrt{\log N \log\log N}\right) .$$

Corollary

For all $\varepsilon > 0$, $T_{ECM} < \varepsilon T_{MPQS}$ holds almost always.

ECM and GNFS

Theorem

For all $\varepsilon > 0$, $T_{GNFS} < \varepsilon T_{ECM}$ holds almost always.

However, *this is not the full story*, because ECM can find small factors quickly, and after dividing them out GNFS can finish the factorisation more quickly than if ECM had not been used.

Let $T_{ECM}^{(\lambda)}(N)$ be the expected time for ECM to find at least λk prime factors of N, where k is the total number of prime factors of N. (It does not matter how we count multiple factors.)

The "two thirds" Theorem

Let K be any positive constant, and $\lambda \in [0, 1]$.

If $\lambda < 2/3$ then $T_{ECM}^{(\lambda)} < KT_{GNFS}$ almost always;
if $\lambda > 2/3$ then $T_{ECM}^{(\lambda)} > KT_{GNFS}$ almost always.

Thus, it is better to use a combination of ECM and GNFS than either alone, and with a sensible strategy we expect to find about two thirds of the prime factors by ECM and the remaining one third by GNFS.

Part 3: Some History of Fermat Numbers

For a nonnegative integer n, the n-th *Fermat number* is $F_n = 2^{2^n} + 1$. It is known that F_n is prime for $0 \le n \le 4$, and composite for $5 \le n \le 23$. Also, for $n \ge 2$, the factors of F_n are of the form

$$k2^{n+2} + 1 \ .$$

In 1732 Euler found that $641 = 5 \cdot 2^7 + 1$ is a factor of F_5, thus disproving Fermat's belief that all F_n are prime[1]. Euler apparently used trial division by primes of the form $64k + 1$.

No Fermat primes larger than F_4 are known, and a probabilistic argument makes it plausible that only a finite number of F_n (perhaps only $F_0, \ldots, F_4$) are prime. It is known that F_n is composite for $5 \le n \le 23$.

[1] "Back of envelope" proof: working mod 641, $5.2^7 = -1 \Rightarrow 5^4.2^{28} = 1$, but $5^4 = -2^4$, so $2^{32} = -1$.

Factorisation of Fermat Numbers

The complete factorisation of the Fermat numbers $F_6, F_7, \ldots$ has been a challenge since Euler's time. Because the F_n grow rapidly in size, a method which factors F_n may be inadequate for F_{n+1}.

F_6

In 1880, Landry factored $F_6 = 274177 \cdot p_{14}$. Landry's method was never published in full, but Williams has attempted to reconstruct it.

Hand Computations

In the period 1877–1970, several small factors of F_n for various $n \ge 9$ were found by taking advantage of the special form of these factors. For example, in 1903 Western found the factor $p_7 = 2424833 = 37 \cdot 2^{16} + 1$ of F_9.

Significant further progress was only possible with the development of the digital computer and more efficient algorithms.

$\mathbf{F}_7$

In 1970, Morrison and Brillhart factored

$$F_7 = 59649589127497217 \cdot p_{22}$$

by the continued fraction method. This method has now been superseded by MPQS which, perhaps surprisingly, has never been the first to factor a Fermat number.

$\mathbf{F}_8$

In 1980, Brent and Pollard factored

$$F_8 = 1238926361552897 \cdot p_{62}$$

by a modification of Pollard's "rho" method. The "rho" method is now largely superseded by ECM.

Nowadays, F_7 and F_8 are "easy" to factor by ECM or MPQS.

$\mathbf{F}_9$

Logically, the next step after the factorisation of F_8 was the factorisation of F_9. It was known that

$$F_9 = 2424833 \cdot c_{148}$$

The 148-digit composite number resisted attack by methods such as Pollard rho, Pollard $p \pm 1$, and the elliptic curve method (ECM), which would have found "small" factors. It was too large to factor by the continued fraction method or even by MPQS.

The difficulty was finally overcome by the invention of the (special) number field sieve (SNFS), based on a new idea of Pollard.
In 1990, Lenstra, Lenstra, Manasse and Pollard, with the assistance of many collaborators and approximately 700 workstations scattered around the world completely factored F_9 by SNFS. As we already mentioned, the factorisation is

$$F_9 \quad = \quad 2424833 \cdot \ p_{49} \cdot p_{99} \ .$$

$\mathbf{F}_{10}$

After the factorisation of F_9 in 1990, F_{10} was the "most wanted" number in various lists of composite numbers.

F_{10} was proved composite in 1952 by Robinson, using Pépin's test on the SWAC. A small factor, 45592577, was found by Selfridge in 1953 (also on the SWAC). Another small factor, 6487031809, was found by Brillhart in 1962 on an IBM 704. Brillhart later found that the cofactor was a 291-digit composite.

Using ECM I found a 40-digit factor $p_{40} =$

$$4659775785220018543264560743076778192897$$

of F_{10} in October, 1995. The 252-digit cofactor c_{291}/p_{40} passed a probabilistic primality test and was soon proved to be prime using the method of Atkin and Morain (based, appropriately, on elliptic curves). Thus, the complete factorisation of F_{10} is

$$F_{10} \quad = \quad 45592577 \cdot 6487031809 \cdot p_{40} \cdot p_{252} \ .$$

$\mathbf{F}_{11}$

F_{11} was completely factored in 1988, *before* the factorisation of F_9 and F_{10}. In fact,

$$\begin{aligned} F_{11} \quad = \quad & 319489 \cdot 974849 \cdot \\ & 167988556341760475137 \cdot \\ & 3560841906445833920513 \cdot p_{564} \end{aligned}$$

The two 6-digit factors were found by Cunningham in 1899, and I found the remaining factors in May 1988, using ECM on a Fujitsu VP100. The 564-digit factor passed a probabilistic primality test, and a rigorous proof of primality was provided by Morain.

The reason why F_{11} could be completely factored before F_9 and F_{10} is that the difficulty of completely factoring numbers by ECM is determined mainly by the size of the *second-largest* prime factor of the number.

The second-largest prime factor of F_{11} has 22 digits and is much easier to find by ECM than the 40-digit factor of F_{10} or the 49-digit factor of F_9.

Summary, $\mathbf{F}_5, \ldots, \mathbf{F}_{11}$

A brief summary of the history of factorisation of $F_5, \ldots, F_{11}$ is given in the Table.

Table 2: Complete factorisation of F_n, $n = 5, \ldots, 11$

n	Factorisation	Date	Comments
5	$p_3 \cdot p_7$	1732	Euler
6	$p_6 \cdot p_{14}$	1880	Landry
7	$p_{17} \cdot p_{22}$	1970	Morrison and Brillhart
8	$p_{16} \cdot p_{62}$	1980	Brent and Pollard (p_{16}, p_{62})
		1980	Williams (primality of p_{62})
9	$p_7 \cdot p_{49} \cdot p_{99}$	1903	Western (p_7)
		1990	Lenstra *et al* (p_{49}, p_{99})
10	$p_8 \cdot p_{10} \cdot p_{40} \cdot p_{252}$	1953	Selfridge (p_8)
		1962	Brillhart (p_{10})
		1995	Brent (p_{40}, p_{252})
11	$p_6 \cdot p'_6 \cdot p_{21} \cdot p_{22} \cdot p_{564}$	1899	Cunningham (p_6, p'_6)
		1988	Brent (p_{21}, p_{22}, p_{564})
		1988	Morain (primality of p_{564})

$\mathbf{F}_{12}$

The smallest Fermat number which is not yet completely factored is F_{12}. It is known that

$$\begin{aligned} F_{12} &= 114689 \cdot 26017793 \cdot \\ &\quad 63766529 \cdot 190274191361 \cdot \\ &\quad 1256132134125569 \cdot c_{1187}\ , \end{aligned}$$

where the 16-digit factor was found by Baillie in 1986, using the Pollard $p-1$ method (and rediscovered in 1988 using ECM).

F_{12} has at least seven prime factors, spoiling a "conjecture" based on the observation that F_n has exactly $n-6$ prime factors for $8 \le n \le 11$.

$\mathbf{F}_{13}$

It is known that

$$\begin{aligned} F_{13} &= 2710954639361 \cdot \\ &\quad 2663848877152141313 \cdot \\ &\quad 3603109844542291969 \cdot \\ &\quad 319546020820551643220672513 \cdot c_{2391}\ , \end{aligned}$$

where the 13-digit factor was found by Hallyburton and Brillhart (1975), and the two 19-digit factors were found by Crandall (1991).

I found the 27-digit factor in June 1995, using ECM on an IBM PC equipped with a Dubner Cruncher board.

$\mathbf{F}_{14}$

$F_{14} = c_{4933}$ is composite, but no nontrivial factors are known. The smallest prime factor probably has at least 30 decimal digits.

$\mathbf{F}_{15}$

$$F_{15} = 1214251009 \cdot 2327042503868417 \cdot c_{9840},$$

where the 13- and 16-digit prime factors were found by Kraitchik (1925) and Gostin (1987). In July, 1997, Brent, Crandall, Dilcher & Van Halewyn found a 33-digit factor

$$p_{33} = 168768817029516972383024127016961$$

using ECM. The quotient is c_{9808}.

$\mathbf{F}_{16}$

$$\begin{aligned} F_{16} &= 825753601 \cdot \\ &\quad 188981757975021318420037633 \cdot c_{19694} \end{aligned}$$

where the 9-digit factor was found by Selfridge (1953), and the 27-digit factor was found in December 1996 by Brent, Crandall & Dilcher using ECM.

$\mathbf{F}_{17}$

$$F_{17} = 31065037602817 \cdot c\,.$$

$\mathbf{F}_{18}$

$$F_{18} = 18631489 \cdot 81274690703860512587777 \cdot c\,,$$

where the 23-digit factor was found by McIntosh and Tardif in April 1999, using ECM.

$\mathbf{F}_{19}, \ldots, \mathbf{F}_{24}$

$F_{19}, F_{20}, F_{21}, F_{23}$ are composite and some small factors are known.

F_{22} is composite but no factors are known.

The status of F_{24} is unknown.

References

[1] A. O. L. Atkin and F. Morain, Elliptic curves and primality proving, *Math. Comp.* **61** (1993), 29–68. Programs available from `ftp://ftp.inria.fr/INRIA/ecpp.V3.4.1.tar.Z` .

[2] D. Atkins, M. Graff, A. K. Lenstra and P. C. Leyland, The magic words are squeamish ossifrage, *Advances in Cryptology: Proc. Asiacrypt'94*, *LNCS* **917**, Springer-Verlag, Berlin, 1995, 263–277.

[3] H. Boender and H. J. J. te Riele, Factoring integers with large prime variations of the quadratic sieve, *Experimental Mathematics* **5** (1996), 257–273. Also `ftp://ftp.cwi.nl/pub/CWIreports/NW/NM-R9513.ps.Z` .

[4] R. P. Brent, *Large factors found by ECM,* Oxford University Computing Laboratory, May 1999. `ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Richard.Brent/champs.txt` .

[5] R. P. Brent, Factorization of the tenth Fermat number, *Math. Comp.* **68** (1999), 429-451. Preliminary version: `ftp://ftp.comlab.ox.ac.uk:/pub/Documents/techpapers/Richard.Brent/rpb161tr.dvi.gz` .

[6] R. P. Brent, Some parallel algorithms for integer factorisation, *Proc. Fifth International Euro-Par Conference* (Toulouse, France, 1-3 Sept 1999), to appear. `ftp://ftp.comlab.ox.ac.uk:/pub/Documents/techpapers/Richard.Brent/rpb193.ps.gz` .

[7] R. P. Brent, R. E. Crandall, K. Dilcher and C. Van Halewyn, Three new factors of Fermat numbers, *Math. Comp.*, to appear. `ftp://ftp.comlab.ox.ac.uk:/pub/Documents/techpapers/Richard.Brent/rpb175.dvi.gz` .

[8] S. Cavallar, B. Dodson, A. K. Lenstra, P. Leyland, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele and P. Zimmermann, *Factorization of RSA-140 using the number field sieve*, announced 4 February 1999. `ftp://ftp.cwi.nl/pub/herman/NFSrecords/RSA-140` .

[9] M. Elkenbracht-Huizing, An implementation of the number field sieve, *Experimental Mathematics*, **5** (1996), 231–253.

[10] L. Euler, Observationes de theoremate quodam Fermatiano aliisque ad numeros primos spectantibus, *Comm. Acad. Sci. Petropol.* **6**, ad annos 1732–33 (1738), 103–107; also *Leonhardi Euleri Opera Omnia*, Ser. I, vol. II, Teubner, Leipzig, 1915, 1–5.

[11] L. Euler, Theoremata circa divisores numerorum, *Novi. Comm. Acad. Sci. Petropol.* **1**, ad annos 1747–48 (1750), 20–48.

[12] P. de Fermat, *Oeuvres de Fermat,* vol. II: *Correspondance,* P. Tannery and C. Henry (editors), Gauthier-Villars, Paris, 1894.

[13] A. K. Lenstra and H. W. Lenstra, Jr. (editors), The development of the number field sieve, *Lecture Notes in Mathematics* **1554**, Springer-Verlag, Berlin, 1993.

[14] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard, The factorization of the ninth Fermat number, *Math. Comp.* **61** (1993), 319–349.

[15] A. K. Lenstra and M. S. Manasse, Factoring by electronic mail, *Proc. Eurocrypt '89*, *LNCS* **434**, Springer-Verlag, Berlin, 1990, 355–371.

[16] A. K. Lenstra and M. S. Manasse, Factoring with two large primes, *Math. Comp.* **63** (1994), 785–798.

[17] H. W. Lenstra, Jr., Factoring integers with elliptic curves, *Annals of Mathematics* (2) **126** (1987), 649–673.

[18] P. L. Montgomery, Square roots of products of algebraic numbers, *Mathematics of Computation 1943 – 1993, Proc. Symp. Appl. Math.* **48** (1994), 567–571.

[19] P. L. Montgomery, A block Lanczos algorithm for finding dependencies over $GF(2)$, *Advances in Cryptology: Proc. Eurocrypt'95*, *LNCS* **921**, Springer-Verlag, Berlin, 1995, 106–120.

[20] A. M. Odlyzko, The future of integer factorization, *CryptoBytes* **1**, 2 (1995), 5–12. Available from `http://www.rsa.com/rsalabs/pubs/cryptobytes` .

[21] R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Comm. ACM* **21** (1978), 120–126.

[22] RSA Laboratories, Information on the RSA challenge, `http://www.rsa.com/rsalabs/html/challenges.html` .

[23] R. S. Schaller, Moore's law: past, present and future, *IEEE Spectrum* **34**, 6 (June 1997), 52–59.

[24] J. L. Selfridge, Factors of Fermat numbers, *MTAC* **7** (1953), 274–275.

[25] A. Shamir, *Factoring large numbers with the TWINKLE device* (extended abstract), preprint, 1999. Announced at Eurocrypt'99.

[26] P. W. Shor, Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Computing* **26** (1997), 1484–1509.

[27] I. N. Stewart and D. O. Tall, *Algebraic Number Theory*, second edition, Chapman and Hall, 1987.

[28] A. M. Vershik, The asymptotic distribution of factorizations of natural numbers into prime divisors, *Dokl. Akad. Nauk SSSR* **289** (1986), 269–272; English transl. in *Soviet Math. Dokl.* **34** (1987), 57–61.

[29] H. C. Williams, How was F_6 factored?, *Math. Comp.* **61** (1993), 463–474.