THE AUSTRALIAN NATIONAL UNIVERSITY

TR-CS-96-02

Factorization of the Tenth and Eleventh Fermat Numbers

Richard P. Brent

February 1996

Joint Computer Science Technical Report Series

Department of Computer Science
Faculty of Engineering and Information Technology

Computer Sciences Laboratory
Research School of Information Sciences and Engineering

This technical report series is published jointly by the Department of Computer Science, Faculty of Engineering and Information Technology, and the Computer Sciences Laboratory, Research School of Information Sciences and Engineering, The Australian National University.

Please direct correspondence regarding this series to:

Technical Reports
Department of Computer Science
Faculty of Engineering and Information Technology
The Australian National University
Canberra ACT 0200
Australia

or send email to:

`Technical.Reports@cs.anu.edu.au`

A list of technical reports, including some abstracts and copies of some full reports may be found at:

`http://cs.anu.edu.au/techreports/`

Recent reports in this series:

TR-CS-96-01 Weifa Liang and Richard P. Brent. *Constructing the spanners of graphs in parallel*. January 1996.

TR-CS-95-08 David Hawking. *The design and implementation of a parallel document retrieval engine*. December 1995.

TR-CS-95-07 Raymond H. Chan and Michael K. Ng. *Conjugate gradient methods for Toeplitz systems*. September 1995.

TR-CS-95-06 Oscar Bosman and Heinz W. Schmidt. *Object test coverage using finite state machines*. September 1995.

TR-CS-95-05 Jeffrey X. Yu, Kian-Lee Tan, and Xun Qu. *On balancing workload in a highly mobile environment*. August 1995.

TR-CS-95-04 Department of Computer Science. *Annual report 1994*. August 1995.

FACTORIZATION OF THE TENTH AND ELEVENTH FERMAT NUMBERS

RICHARD P. BRENT

ABSTRACT. We describe the complete factorization of the tenth and eleventh Fermat numbers. The tenth Fermat number is a product of four prime factors with 8, 10, 40 and 252 decimal digits. The eleventh Fermat number is a product of five prime factors with 6, 6, 21, 22 and 564 decimal digits. We also note a new 27-decimal digit factor of the thirteenth Fermat number. This number has four known prime factors and a 2391-decimal digit composite factor. All the new factors reported here were found by the elliptic curve method (ECM). The 40-digit factor of the tenth Fermat number was found after about 140 Mflop-years of computation. We discuss aspects of the practical implementation of ECM, including the use of special-purpose hardware, and note several other large factors found recently by ECM.

1. INTRODUCTION

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

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

In 1732 Euler [36, p. 104] found that $641 = 5 \cdot 2^7 + 1$ is a factor of F_5, thus disproving Fermat's belief [38, pp. 206–208] that all F_n are prime. Euler apparently used trial division by primes of the form $64k+1$, see [37, p. 33]. 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.

The complete factorization 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}. In this section we give a summary of what has been achieved since 1732. Additional historical details and references can be found in [20, 45, 53], and some recent results are given in [26, 27, 40].

In the following, 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. $c_4 = 1729$. In cases where one factor can be found by division, we usually write it as p_n or c_n and leave its computation as an exercise. For example, Morain [63] showed that $2^{3539} + 1 = 3 \cdot p_{1065}$. It would be tedious to write out the 1065-digit prime in decimal (though easy in binary).

In 1880, Landry [50] factored $F_6 = 274177 \cdot p_{14}$. Landry's method was never published in full, but Williams [82] has attempted to reconstruct it.

1991 *Mathematics Subject Classification.* 11Y05, 11B83, 11Y55; Secondary 11-04, 11A51, 11Y11, 11Y16, 14H52, 65Y10, 68Q25.

Key words and phrases. computational number theory, Cunningham project, ECM, elliptic curve method, factorization, Fermat number, F_{10}, F_{11}, F_{12}, F_{13}, integer factorization.

Copyright © 1996, R. P. Brent Typeset using $\mathcal{AMS}$-LaTeX.

In the period 1877–1974, several small factors of F_n for various $n \geq 9$ were found by taking advantage of the special form (1) of these factors. For example, in 1903 Western [29] found the factor $p_7 = 2424833 = 37 \cdot 2^{16} + 1$ of F_9. Other examples may be found in [41, Table 2].

Significant further progress was only possible with the development of the digital computer and more efficient algorithms. In 1970, Morrison and Brillhart [64] factored

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

by the continued fraction method. Then, in 1980, Brent and Pollard [17] factored

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

by a modification of Pollard's "rho" method [5, 67]. The modification improved the efficiency of Pollard's original algorithm, for factors of the form (1), by a ratio conjectured to be of order $2^{n/2}/n$; for F_8 the ratio is about 6. The larger factor p_{62} of F_8 was first proved prime by Williams [17, §4] using the method of Williams and Judd [83].

The rho method could have factored F_7 (with a little more difficulty than F_8, see [17, Table 2]) if it had been invented earlier. Similarly, the multiple-polynomial quadratic sieve (MPQS) method [78], which is currently the best "general-purpose" method for composite numbers of up to about 100 decimal digits, could have factored both F_7 and F_8, but it was not available in 1980.

Logically, the next step after the factorization of F_8 was the factorization 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 its successor, MPQS. The difficulty was finally overcome by the invention of the (special) number field sieve (SNFS), based on a new idea of Pollard [52]. 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 [53, 54]. The factorization is

$$F_9 = 2424833 \cdot 7455602825647884208337395736200454918783366342657 \cdot p_{99} \,.$$

F_{10} has been a "most wanted" number in various lists of composite numbers ever since the factorization of F_9 in 1990, and in recent years it has usually been *the* most wanted number (see, for example, the list in Update 2.9 of [20]). F_{10} was proved composite in 1952 by Robinson [71], using Pépin's test on the SWAC. A small factor, 45592577, was found by Selfridge [72] in 1953 (also on the SWAC). Another small factor, 6487031809, was found by Brillhart [19] in 1962 on an IBM 704. Brillhart later found that the cofactor was a 291-digit composite. Thus, it was known that $F_{10} = 45592577 \cdot 6487031809 \cdot c_{291}$.

This paper describes the complete factorization of F_{10}. Using ECM we found a 40-digit factor $p_{40} = 4659775785220018543264560743076778192897$ on October 20, 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). Later, a more elementary proof was found, using Selfridge's "Cube Root Theorem" (see §9). Thus, the complete factorization is

$$F_{10} = 45592577 \cdot 6487031809 \cdot 4659775785220018543264560743076778192897 \cdot p_{252}$$

where $p_{252} = 13043 \cdots 24577$.

In §§2-4 we describe some variants of ECM and their performance, and in §5 we describe some implementations of ECM with which we attempted to factor F_{10} and other Fermat numbers. Details of the factorization of F_{10} are given in §6.

So far, this summary has been chronological, but now we backtrack, because F_{11} was completely factored in 1988, *before* the factorization of F_9 and F_{10}. In fact,

$$F_{11} = 319489 \cdot 974849 \cdot 167988556341760475137 \cdot 3560841906445833920513 \cdot p_{564}$$

where $p_{564} = 17346 \cdots 34177$. The two 6-digit factors were found by Cunningham [20, 30] in 1899, and remaining factors were found by the present author in May 1988. The 564-digit factor passed a probabilistic primality test, and a rigorous proof of primality was provided by Morain (see §9). As details of the factorization of F_{11} have not been published, apart from two brief announcements [8, 10], we describe the computation in §7.

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. Thus, the difficulty of completely factoring F_n by the fastest known method is not a monotonic function of n.

A brief summary of the history of factorization of $F_5, \ldots, F_{11}$ is given in Table 1. For a similar history of $F_{12}, \ldots, F_{22}$, see [25, p. 148].

In §8 we outline a project to find more factors of Fermat numbers by ECM, using inexpensive special-purpose hardware, and give the one success so far – a 27-digit factor of F_{13}, found in June 1995.

Rigorously proving primality of a number as large as the 564-digit factor of F_{11} is a nontrivial task. In §9 we discuss primality proofs and "certificates" of primality for the factors of F_n, $n \leq 11$.

The smallest Fermat number which is not yet completely factored is F_{12}. It is known [20] that

$$F_{12} = 114689 \cdot 26017793 \cdot 63766529 \cdot 190274191361 \cdot 1256132134125569 \cdot c_{1187} .$$

Thus, F_{12} has at least seven prime factors. If the distribution of second-largest prime factors of large random integers [47, 48] is any guide, it is unlikely that c_{1187} can be completely factored by ECM. For further discussion, see §10.

For one possible application of factorizations of Fermat numbers, observe that the factorization of $F_0, F_1, \ldots, F_{n-1}$ is necessary to determine the structure of a finite field with 2^{2^n} elements, because the multiplicative group of such a field has order $2^{2^n} - 1 = F_0 F_1 F_2 \cdots F_{n-1}$. We now know the structure of such fields for $0 \leq n \leq 12$.

1.1. **Acknowledgements.** Thanks are due to Hendrik Lenstra, Jr., for the ECM algorithm which made the factorization of F_{10} and F_{11} possible; and to Peter Montgomery and Hiromi Suyama for their practical improvements to ECM. John Pollard provided some of the key ideas with his "$p-1$" and "rho" methods. John Brillhart, Richard Crandall, Wilfrid Keller, Donald Knuth, John Selfridge and Daniel Shanks provided historical information, references, and/or corrections to drafts of this paper. Bruce Dodson, Arjen Lenstra, Peter Montgomery, Robert Silverman and Sam Wagstaff, Jr. provided information about other attempts to factor F_{10}. François Morain proved the primality of the 564-digit factor of F_{11}. Craig Eldershaw ported my ECM program to the AP1000. Harvey Dubner provided a Dubner Cruncher and encouraged me to implement ECM on it. Dennis Andriolo and Robert Dubner provided assistance with aspects of the Cruncher hardware and software. John Cannon and Herman te Riele provided other assistance and encouragement. Bob Gingold graciously volunteered spare computer cycles on a SparcCenter 2000. The ANU Supercomputer Facility provided computer time on a Fujitsu VP100 and VP2200/10, and the ANU-Fujitsu CAP Project provided access to a Fujitsu AP1000.

TABLE 1. Complete factorization of F_n, $n = 5, \ldots, 11$

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

2. Variants of ECM

The *elliptic curve method* (ECM) was discovered by H. W. Lenstra, Jr. [55] in 1985. Various practical refinements were suggested by Montgomery [58, 59], Suyama [80], and others [1, 7, 23]. We refer to [54, 60, 79] for a general description of ECM, and to [24, 44, 77] for relevant background.

Lenstra's key idea was to apply Pollard's "$p-1$" method [66] but to work over a different group G. If the method fails, another group can be tried. This is not possible for the $p-1$ method, because it uses a fixed group.

To be specific, suppose we attempt to find a factor of a composite number N, which we can assume not to be a prime power [53, §2.5]. Let p be the smallest prime factor of N. In practice it is desirable to remove small factors (up to say 10^4) by trial division before applying ECM, but we only need assume $p > 3$.

Although p is unknown, we describe the algorithms in terms of operations in the finite field $K = GF(p) = \mathbf{Z}/p\mathbf{Z}$. In practice we work modulo N (or sometimes modulo a multiple of N, if the multiple has a convenient binary representation), and occasionally perform GCD computations which will detect any nontrivial factor of N (probably p, though possibly a different factor of N). Working modulo N or modulo a multiple of N can be regarded as using a redundant representation for elements of K.

In Pollard's $p-1$ method the group G is the multiplicative group of the finite field K, i.e. the multiplicative group of integers modulo p. In ECM the group G is defined on an elliptic curve. There is no loss of generality in assuming that the elliptic curve is given in Weierstrass normal form

$$y^2 = x^3 + ax + b\,, \tag{2}$$

where a and b are constants such that

$$4a^3 + 27b^2 \neq 0 \tag{3}$$

in K. G consists of the set of points $(x, y) \in K \times K$ which lie on the curve and a "point at infinity" $\mathcal{O}$. A commutative and associative group operation is defined in a standard way [7, 23, 58, 77]. In

accordance with the usual convention we write the group operation additively[1]. The (additive) zero element of G is $\mathcal{O}$.

Let $g = |G|$ be the order of G. In the $p-1$ method $g = p-1$, but in ECM we have $g = p+1-t$, where, by a theorem of Hasse [42], $t = t(G)$ satisfies

$$t^2 < 4p\,. \tag{4}$$

By a result of Deuring [31], the result (4) is best possible, in the sense that all integer $t \in (-2\sqrt{p}, +2\sqrt{p})$ arise for some choice of a and b in the Weierstrass form (2). The number of curves with given t can be expressed in terms of the Kronecker class number of $t^2 - 4p$ (see [55]).

In practice, to avoid computation of square roots, we select a pseudo-random parameter a and initial point (x_1, y_1) on the curve, and then compute b from (2). The condition (3) can be checked by taking a GCD; in the unlikely event that it fails to hold then the GCD probably gives the factor p immediately. In practice the test is usually omitted.

2.1. **Other models.** We use the words "model" and "form" interchangeably. The Weierstrass form (2) is not necessarily the most efficient for computational purposes. With (2) we have to perform divisions modulo N. These are expensive because they involve an extended GCD computation. To avoid them, we can replace (x, y) by $(x/z, y/z)$ in (2) to get a homogeneous Weierstrass equation

$$y^2 z = x^3 + axz^2 + bz^3\,. \tag{5}$$

The points (x, y, z) satisfying (5) are thought of as representatives of elements of $P^2(K)$, the projective plane over K, i.e. the points (x, y, z) and (cx, cy, cz) are regarded as equivalent if $c \neq 0 \bmod p$. We write $(x : y : z)$ for the equivalence class containing (x, y, z). The additive zero element $\mathcal{O}$ is $(0 : 1 : 0)$ and we can test for it by computing $\mathrm{GCD}(N, z)$.

The equation (5) lacks symmetry. An elegant symmetrical alternative is the Cauchy form [23, §4.2]

$$x^3 + y^3 + z^3 = Dxyz\,, \tag{6}$$

where D is a constant. In practice we choose a pseudo-random point $(x_1 : y_1 : z_1)$ which defines D, but D does not have to be computed. The group operation has a symmetrical form given in [23, (4.21)], and $(x : y : z) + (y : x : z) = \mathcal{O}$.

Montgomery [58] suggested using the form

$$by^2 = x^3 + ax^2 + x \tag{7}$$

or, replacing (x, y) by $(x/z, y/z)$ as above,

$$by^2 z = x^3 + ax^2 z + xz^2\,. \tag{8}$$

Corresponding to the condition (3) we now have the condition $a^2 \neq 4$.

Not every elliptic curve can be expressed in the form (7) or (8) by rational transformations. However, by varying a in (7) or (8), we get a sufficiently large class of pseudo-random curves. The exact value of b in (7) or (8) is not important, but it is significant whether b is a quadratic residue (mod p) or not. In general we get two different groups, of order $p + 1 \pm t$, by varying b.

Assume that we start with a point $P_1 = (x_1 : y_1 : z_1)$ on (8). For positive integer n, we write $P_n = nP_1$ and suppose $P_n = (x_n : y_n : z_n)$. Montgomery [58, §10] shows in a direct way that, for positive integer m, n such that $P_m \neq \pm P_n$, we have an *addition formula*

$$x_{m+n} : z_{m+n} = z_{|m-n|}(x_m x_n - z_m z_n)^2 : x_{|m-n|}(x_m z_n - z_m x_n)^2 \tag{9}$$

[1] In [7, 10, 11] we wrote the group operation multiplicatively because of the analogy with the multiplicative group of $GF(p)$ in the $p-1$ method, and spoke of $\mathcal{O}$ as the identity element.

and a *duplication formula*

$$x_{2n} : z_{2n} = (x_n^2 - z_n^2)^2 : 4x_n z_n(x_n^2 + a x_n z_n + z_n^2) \ . \tag{10}$$

An alternative derivation of (9)–(10), using addition and duplication formulas for the Jacobian elliptic function $sn^2(u)$, is given in [23, p. 422]. This derivation makes the reason for associativity clear.

Note that (9)–(10) do not specify the y-coordinate. Fortunately, it turns out that the y-coordinate is not required for ECM, and we can save work by not computing it. In this case we write $P_n = (x_n : : z_n)$. Since $(x : y : z) + (x : -y : z) = \mathcal{O}$, ignoring the y component amounts to identifying P and $-P$.

Montgomery [58, §10] shows how (9) and (10) can be implemented to perform an addition and a duplication with 11 multiplications (mod N). The number 11 can be reduced under some circumstances.

2.2. **The first phase.** The first phase of ECM computes P_r for a large integer r. Usually r is the product of all prime powers less than some bound B_1. There is no need to compute r explicitly. By the prime number theorem, $\log r \sim B_1$ as $B_1 \to \infty$. (Here and below, "log" without a subscript denotes the natural logarithm.)

From (9)–(10), we can compute the x and z-components of (P_1, P_{2n}, P_{2n+1}) or $(P_1, P_{2n+1}, P_{2n+2})$ from the x and z-components of (P_1, P_n, P_{n+1}). Thus, from the binary representation of the prime factors of r, we can compute $(x_r : : z_r)$ in $O(\log r) = O(B_1)$ operations, where each operation is an addition or multiplication mod N. In fact, $(x_r : : z_r)$ can be computed with about $K_1 B_1$ multiplications mod N and a comparable number of additions mod N, where $K_1 = 11/\log 2$. If $z_1 = 1$ then K_1 can be reduced to $10/\log 2$. For details, see Montgomery [58, §10].

At the end of the first phase of ECM we check if $P_r = \mathcal{O}$ by computing $\mathrm{GCD}(z_r, N)$. If the GCD is nontrivial then the first phase of ECM has been successful in finding a factor of N. Otherwise we may continue with a second phase (see §3) before trying again with a different pseudo-random group G.

2.3. **The starting point.** An advantage of using (7) or (8) over (2) or (5) is that the group order is always a multiple of four (Suyama [80]; see [58, p. 262]). Also, it is possible to ensure that the group order is divisible by 8, 12 or 16. For example, if $\sigma > 5$ is a pseudo-random integer,

$$\begin{aligned} u : v &= \sigma^2 - 5 : 4\sigma, \\ x_1 : z_1 &= u^3 : v^3, \\ a' : a'' &= (v-u)^3(3u+v) : 4u^3 v, \end{aligned} \tag{11}$$

then the curve (8) with $a + 2 = a'/a''$ has group order divisible by 12. As starting point we can take $(x_1 : : z_1)$.

3. The second phase

Montgomery and others [7, 58, 59, 62] have described several ways to improve Lenstra's original ECM algorithm by the addition of a second phase, analogous to phase 2 of the Pollard $p-1$ method. We outline some variations which we have implemented. Phase 1 is the same in all cases, as described in §2.2.

We usually assume that Montgomery's form (8) is used with starting point $P_1 = (x_1 : : z_1)$ given by (11). Bounds $B_2 \geq B_1 > 0$ are chosen in advance. For example, we might choose $B_1 = 10^6$ and $B_2 = 100 B_1$ (see §4.4). In the following we assume that $B_2 \gg B_1$, so $B_2 - B_1 \simeq B_2$.

We define $B_3 = \pi(B_2) - \pi(B_1) \simeq B_2/\log B_2$. The time required for phase 2 is approximately proportional to B_3.

Suppose the group order g has prime factors $g_1 \geq g_2 \geq \cdots$. Phase 1 will usually be successful if $g_1 \leq B_1$. We say "usually" because it is possible that a prime factor of g occurs with higher multiplicity in g than in r, but this is unlikely and can be neglected when considering the average behaviour of ECM. Phase 2 is designed to be successful if $g_1 \leq B_2$ and $g_2 \leq B_1$. To a good approximation, phase 1 is successful if all prime factors of g are at most B_1, and phase 2 is successful if all but one prime factors of g are at most B_1, and that one factor is at most B_2.

In the following we describe several different versions of phase 2 (also called "continuations" because they continue after phase 1). Some versions are difficult to implement using only the formulae (9)–(10). For this reason, some programs use Montgomery's form (8) for phase 1 and convert back to Weierstrass form (2) or (5) for phase 2. For details of the transformation see [3, §4.2].

3.1. **The standard continuation.** Suppose phase 1 has computed $Q = P_r$ such that $Q \neq \mathcal{O}$. The "standard continuation" computes $sQ = (x_{rs} : : z_{rs})$ for each prime s, $B_1 < s \leq B_2$, and is successful if $\mathrm{GCD}(z_{rs}, N)$ is nontrivial for some such s. We can amortize the cost of a GCD by following a suggestion of Pollard [67]. We compute

$$\mathrm{GCD}\left(\prod_s z_{rs} \bmod N, N\right)$$

where the product mod N is taken over a sufficiently large set of s, and backtrack if the GCD is composite. This reduces the cost of a GCD essentially to that of a multiplication mod N.

There is no advantage in using phase 2 if sQ is computed using the standard binary method, which takes $O(\log s)$ group operations. It is much more efficient to precompute a small table of points $2dQ$, where $0 < d \leq D$ say. Then, given $s_1 Q$ for some odd s_1, we can compute $\min(s_1 + 2D, s_2)Q$, where s_2 is the next prime, using only one group operation. Thus, we can compute sQ for a sequence of values of s including all primes in $(B_1, B_2]$ and possibly including some composites (if $2D$ is smaller than the maximal gap between successive primes in the interval), with one group operation per point. Provided D is at least of order $\log B_2$, the work for phase 2 is reduced from $O(B_2)$ group operations to $O(B_3)$ group operations.

The standard continuation can be implemented efficiently in $O(\log N \log B_2)$ bits of storage. It is not necessary to store a table of primes up to B_2 as the odd primes can be generated by sieving in blocks as required. Even storing the primes to $\sqrt{B_2}$ is unnecessary, because we can sieve using odd integers $3, 5, 7, 9, \ldots$. The sieving does not need to be very efficient, because most of the time is spent on multiple-precision arithmetic to perform group operations. Sieving could be replaced by a fast pseudo-prime test, because it does not hurt ECM if a few composites are included in the numbers generated and treated as primes.

3.2. **The improved standard continuation.** The standard continuation can be improved – Montgomery's form (8) can be used throughout, and most group operations can be replaced by a small number of multiplications mod N. The key idea [58, §4] is that we can test if $\mathrm{GCD}(z_{rs}, N)$ is nontrivial without computing sQ. We precompute $2dQ$ for $0 < d \leq D$ as above, using $O(D)$ group operations. We can then compute mQ for $m = 1, 2D+1, 4D+1, \ldots$, using one application of (9) for each point after the first. The points mQ are updated as necessary, so only require $O(\log N)$ storage. Suppose $s = m + n$ is a prime, where n is even and $0 < n \leq D$. Now $sQ = \mathcal{O}$ implies $mQ = \pm nQ$. Since $mQ = (x_{mr} : : z_{mr})$ and $\pm nQ = (x_{nr} : : z_{nr})$ are known, it is sufficient to test if $\mathrm{GCD}(x_{mr} z_{nr} - x_{nr} z_{mr}, N)$ is nontrivial. We can avoid most of the GCDs as in §3.1, by computing $\prod(x_{mr} z_{nr} - x_{nr} z_{mr}) \bmod N$, where the product is taken over several m and n. Thus, the work is reduced to about three multiplications mod N per prime s. It can be reduced

to two multiplications mod N by ensuring that $z_{nr} = 1$, which involves a precomputation of order D. A reduction to one multiplication mod N is possible, at the cost of one extended GCD computation per point mQ. This is worthwhile if D is sufficiently large. Another reduction by a factor of almost two can be achieved by rational preconditioning [65]. For future reference, we assume K_2 multiplications mod N per comparison of points, where $1/2 \le K_2 \le 3$, the exact value of K_2 depending on the implementation.

There is an analogy with Shanks's baby and giant steps [74, p. 419]: giant steps involve a group operation (about 11 multiplications) and possibly an extended GCD computation, but baby steps avoid the group operation and involve only $K_2 \le 3$ multiplications.

To decide on the table size D, note that setting up the table and computation of the points mQ requires of order $D + B_2/(2D)$ applications of (9). If storage is not a consideration, the optimal D is approximately $\sqrt{B_2/2}$. However, provided $\sqrt{B_2} > D \gg \log B_2$, the setting up cost is $o(B_3)$, and the overall cost of phase 2 is about $K_2 B_3$ multiplications mod N. Thus, storage requirements for an efficient implementation of the improved standard continuation are not much greater than for the standard continuation.

3.3. **The birthday paradox continuation.** The "birthday paradox" continuation is an alternative to the (improved) standard continuation. It was suggested in [7] and has been implemented in several of our programs (see §5) and in the programs of A. Lenstra et al [3, 33, 54].

There are several variations on the birthday paradox idea. We describe a version which is easy to implement and whose efficiency is comparable to that of the improved standard continuation. Following a suggestion of Suyama, we choose a positive parameter e. The choice of e is considered below. For the moment the reader can suppose that $e = 1$.

Suppose, as in §3.1, that Q is the output of phase 1. Select a table size D. If storage permits, D should be about $\sqrt{B_3}$; otherwise choose D as large as storage constraints allow (for reasonable efficiency we only need $D \gg e \log B_3$). Generate D pseudo-random multiples of Q, say $Q_j = q_j^e Q$ for $j = 1, \ldots, D$. There is some advantage in choosing the q_j to be linear in j, i.e. $q_j = k_0 + k_1 j$ for some pseudo-random k_0, k_1 (not too small). In this case the Q_j can be computed by a "finite difference" scheme with $O(eD)$ group operations because the e-th differences of the multipliers q_j^e are constant. Another possibility is to choose q_j to be a product of small primes. For example, in our programs C–E (see §5) we use a set of $2\lceil \log_2 D \rceil$ odd primes and take q_j to be a product of $\lceil \log_2 D \rceil$ odd primes from this set, the choice depending on the bits in the binary representation of $j - 1$. This scheme requires $O(eD \log\log D)$ group operations and can be vectorized.

After generating the D points Q_j, we generate $\lfloor B_3/D \rfloor$ further pseudo-random points, say $\overline{Q}_k = \overline{q}_k^e Q$, where the $\overline{q}_k$ are distinct from the q_j. The choice $\overline{q}_k = 2^k$ is satisfactory. For each such point $\overline{Q}_k$, we check if $\overline{Q}_k = \pm Q_j$ for $j = 1, \ldots, D$. This can be done with $K_2 D$ multiplications mod N, as in the description of the improved standard continuation in §3.2. If D is sufficiently large, it is worthwhile to make the z-coordinates of Q_j and $\overline{Q}_k$ unity by extended GCD computations, which reduces K_2 to 1. (To reduce the number of extended GCDs, we can generate the points $\overline{Q}_k$ in batches and reduce their z-coordinates to a common value before performing one extended GCD.) Note that the points $\overline{Q}_k$ do not need to be stored, as only one (or one batch) is needed at a time. We only need store $O(D)$ points.

It is easy to see that most of the computation for the birthday paradox version of phase 2 amounts to the evaluation of a polynomial of degree D at $\lfloor B_3/D \rfloor$ points. Thus, fast polynomial evaluation schemes can be used [7, 47, 65, 84].

Both the improved standard continuation and the birthday paradox continuation make approximately B_3 comparisons of points which are multiples of Q, say nQ and $n'Q$, and usually succeed if $g_2 \le B_1$ and $n \pm n'$ is a nontrivial multiple of g_1. The improved standard continuation

ensures that $|n - n'|$ is prime, but the birthday paradox continuation merely takes pseudo-random n and n'. Since g_1 is prime, it would appear that the improved standard continuation is more efficient. However, taking the parameter $e > 1$ may compensate for this. The number of solutions of

$$x^{2e} = 1 \bmod g_1 \tag{12}$$

is $\mathrm{GCD}(2e, g_1 - 1)$. Thus, by choosing e as a product of small primes, we increase the expected number of solutions of (12). In fact, if $2e = 2^{e_1}3^{e_2}5^{e_3}\cdots$, it can be shown that the expected value of $\mathrm{GCD}(2e, q-1)$ for a random large prime q is $(e_1+1)(e_2+1)(e_3+1)\cdots$. Since g_1 is the largest prime factor of the group order g, it is reasonable to assume similar behaviour for $\mathrm{GCD}(2e, g_1 - 1)$. The number of solutions of equation (12) is relevant because we expect phase 2 to succeed if $g_2 \le B_1$ and $(q_j/\overline{q}_k)^{2e} = 1 \bmod g_1$.

The parameter e should not be chosen too large, because the cost of generating the points Q_j and $\overline{Q}_k$ is proportional to e. To ensure that this cost is negligible, we need $e \ll D$. In practice, assuming $D < 512$, a reasonable strategy is to choose the largest e from the set $\{1, 2, 3, 6, 12\}$ subject to the constraint $32e < D$.

3.4. **The FFT continuation.** Pollard suggested the use of the FFT to speed up phase 2 for his $p-1$ method, and Montgomery [59] has successfully implemented the analogous phase 2 for ECM. The FFT continuation may be regarded as an efficient generalisation of both the improved standard continuation and the birthday paradox continuation. We have not implemented it because of its complexity and large storage requirements.

3.5. **Comparison of continuations.** It is natural to ask which of the above versions of phase 2 is best. We initially implemented the birthday paradox continuation because of its simplicity. Also, the asymptotic analysis in [7, §7] indicated that it would be faster than the (improved) standard continuation. However, this was on the assumption that an asymptotically fast polynomial evaluation scheme would be used. In practice, D is rarely large enough for such a scheme to be significantly faster than standard polynomial evaluation. In our most recent implementation of ECM (program G of §5) we have used the improved standard continuation because of its slightly lower storage requirements and better (predicted) performance for factors of up to 40 decimal digits. If storage requirements and program complexity are not major considerations, then the FFT continuation is probably the best.

4. Performance of ECM

4.1. **Prime factors of random integers.** Let $n_1(N) \ge n_2(N) \ge \ldots$ be the prime factors of a positive integer N. The $n_j(N)$ are not necessarily distinct. For convenience we take $n_j(N) = 1$ if N has less than j prime factors.

For $k \ge 1$, suppose that $1 \ge \alpha_1 \ge \ldots \ge \alpha_k \ge 0$. Following Vershik [81], we define $\Phi_k = \Phi_k(\alpha_1, \ldots, \alpha_k)$ by

$$\Phi_k = \lim_{M\to\infty} \frac{\#\left\{N : 1 \le N \le M,\ n_j(N) \le N^{\alpha_j} \text{ for } j = 1, \ldots, k\right\}}{M}.$$

Informally, $\Phi_k(\alpha_1, \ldots, \alpha_k)$ is the probability that a large random integer N has its j-th largest prime factor at most N^{α_j}, for $j = 1, \ldots, k$. The cases $k = 1$ and $k = 2$ are relevant to ECM (see §§4.2-4.4). It is convenient to define $\Phi_1(\alpha_1) = 1$ if $\alpha_1 > 1$, and $\Phi_1(\alpha_1) = 0$ if $\alpha_1 < 0$. Vershik [81, Thm. 1] shows that

$$\Phi_k = \int_0^{\alpha_k} \int_{\theta_k}^{\alpha_{k-1}} \cdots \int_{\theta_2}^{\alpha_1} \Phi_1\left(\frac{\theta_k}{1 - \theta_1 - \cdots - \theta_k}\right) \frac{d\theta_1 \ldots d\theta_{k-1} d\theta_k}{\theta_1 \ldots \theta_{k-1}\theta_k}. \tag{13}$$

Knuth and Trabb Pardo [48] observe an interesting connection with the distribution of cycle lengths in a random permutation. In fact, the distribution of j-th largest prime factors of n-digit random integers is asymptotically the same as the distribution of j-th longest cycles in random permutations of n objects. Thus, the literature on random permutations [39, 75] is relevant.

We define

$$\begin{aligned} \rho(\alpha) &= \Phi_1(1/\alpha) \quad \text{for } \alpha > 0\ , \\ \rho_2(\alpha) &= \Phi_2(1, 1/\alpha) \quad \text{for } \alpha \ge 1\ , \\ \mu(\alpha,\beta) &= \Phi_2(\beta/\alpha, 1/\alpha) \quad \text{for } 1 \le \beta \le \alpha\ . \end{aligned} \tag{14}$$

Informally, $\rho(\alpha)$ is the probability that $n_1^\alpha \le N$, $\rho_2(\alpha)$ is the probability that $n_2^\alpha \le N$, and $\mu(\alpha,\beta)$ is the probability that both $n_2^\alpha \le N$ and $n_1^\alpha \le N^\beta$, for a large random integer N with largest prime factor n_1 and second-largest prime factor n_2. Note that $\rho(\alpha) = \mu(\alpha, 1)$ and $\rho_2(\alpha) = \mu(\alpha,\alpha)$.

The function ρ is usually called Dickman's function after Dickman [32], though some authors refer to Φ_1 as Dickman's function, and Vershik [81] calls $\varphi_1 = \Phi_1'$ the Dickman-Goncharov function.

It is known (see [7]) that ρ satisfies a differential-difference equation

$$\alpha\rho'(\alpha) + \rho(\alpha - 1) = 0 \tag{15}$$

for $\alpha \ge 1$. Thus, $\rho(\alpha)$ may be computed by numerical integration from

$$\rho(\alpha) = \frac{1}{\alpha}\int_{\alpha-1}^{\alpha} \rho(t)\,dt \tag{16}$$

for $\alpha > 1$. The function μ may be computed from [7, eqn. (3.3)]

$$\mu(\alpha,\beta) = \rho(\alpha) + \int_{\alpha-\beta}^{\alpha-1} \frac{\rho(t)}{\alpha - t}\,dt\ . \tag{17}$$

for $1 \le \beta \le \alpha$. The formula for μ given in [79, p. 447] is incorrect, as can be seen by considering the limit as $\beta \to 1+$.

The results (15)–(17) follow from Vershik's general result (13), although it is possible to derive them more directly, as in [7, 47, 48].

Sharp asymptotic results for ρ are given by de Bruijn [21, 57], and an asymptotic result for μ is stated in [7]. To predict the performance of phase 1 of ECM it is enough to know that

$$\frac{\rho(\alpha-1)}{\rho(\alpha)} \sim -\log\rho(\alpha) \sim \alpha\log\alpha \tag{18}$$

as $\alpha \to \infty$.

4.2. **Heuristic analysis of phase 1.** We first give a simple, heuristic explanation of why phase 1 of ECM works. Assume that, so far as its largest prime factor g_1 is concerned, the group order g behaves like a random integer near p. This is not quite correct, but is an accurate enough approximation to obtain asymptotic results. In §4.4 we take known divisors of g into account.

Let $\alpha = \log p/\log B_1$, so $B_1 = p^{1/\alpha}$. The probability that one curve succeeds in finding the factor p is close to the probability that $g_1 \le B_1$, and can be approximated by $\rho(\alpha)$. Thus, the expected number of curves for success is $C_1 = 1/\rho(\alpha)$. As each curve requires about $K_1B_1 = K_1p^{1/\alpha}$ multiplications (mod N), the expected number of multiplications (mod N) is

$$W(\alpha) = K_1p^{1/\alpha}/\rho(\alpha)\ . \tag{19}$$

Differentiating with respect to α, we see that the minimum $W(\alpha)$ occurs when

$$\log p = -\alpha^2 \rho'(\alpha)/\rho(\alpha) \ .$$

Using (15) and (18), we obtain the following asymptotic results for the optimal parameters:

$$\log p = \frac{\alpha\rho(\alpha-1)}{\rho(\alpha)} \sim \alpha^2 \log\alpha \ , \tag{20}$$

$$\log B_1 = \frac{\rho(\alpha-1)}{\rho(\alpha)} \sim \alpha\log\alpha \ , \tag{21}$$

$$\log C_1 = -\log\rho(\alpha) \sim \alpha\log\alpha \ , \tag{22}$$

$$\log W = \log K_1 - \frac{d}{d\alpha}\left(\alpha\log\rho(\alpha)\right) \sim 2\alpha\log\alpha \ , \tag{23}$$

$$\log(B_1/C_1) = -\alpha^2\frac{d}{d\alpha}\left(\frac{\log\rho(\alpha)}{\alpha}\right) \sim \alpha \ . \tag{24}$$

The result (24) is more delicate than the other asymptotic results because it involves cancellation of the terms of order $\alpha\log\alpha$ in (21) and (22).

From (20) and (23) we obtain

$$\log W \sim \sqrt{2\log p\log\log p} \tag{25}$$

as $p\to\infty$. Thus, $W = O(p^\epsilon)$ for any $\epsilon > 0$.

4.3. **Lenstra's analysis of phase 1.** Modulo an unproved but plausible assumption regarding the distribution of prime factors of random integers in "short" intervals, Lenstra [55] has made the argument leading to (25) rigorous. He shows that phase 1 of ECM, when applied to a composite integer N with smallest prime factor p, will find p in an expected number

$$W_1(p) = \exp\left(\sqrt{(2+o(1))\log p\log\log p}\right) \ , \tag{26}$$

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

$$T_1(p,N) = M(N)W_1(p) \ , \tag{27}$$

where $M(N)$ is the time required to multiply numbers modulo N.

The factor $M(N)$ is important because we are interested in Fermat numbers $N = F_n = 2^{2^n}+1$ which may be very large. By way of comparison, the obvious method of squaring (mod q) n times, for each $q \le p$ of the form (1), takes time

$$T_S(p,N) = O\left(2^{-n}np(\log p)^2\right) \ . \tag{28}$$

Comparing the bounds (27) and (28), we see that the second method is preferable if $(\log p)/n$ is sufficiently small (though it can not be too small, in view of (1)). In practice, ECM is only feasible on F_n for moderate n: the limit is about the same as the limit of feasibility of Pépin's test (currently $n \le 22$, see [26, 27]).

4.4. **Heuristic analysis of phase 2.** Lenstra's result (26) applies only to phase 1 and assumes that the Weierstrass form is used. To predict the improvement derived from phase 2 and the use of Montgomery's form, we have to use heuristic arguments. We assume that the group order g behaves (so far as the distribution of its largest two prime factors are concerned) like a random integer near p/d, where d takes account of known small divisors of g. If Montgomery's form is used with the curve chosen as in §2.3, then $d = 12$. A limited amount of experimental

data [7, 54, 59] confirms that this assumption is reasonable, though perhaps slightly conservative; ECM may perform slightly better than predicted.

If ECM is used with parameters B_1 and B_2, and the (improved) standard continuation is applied, then we expect a factor to be found if $g_2 \le B_1$ and $g_1 \le B_2$. If $\alpha = \log(p/d)/\log B_1$ and $\beta = \log B_2/\log B_1$, then (by our heuristic assumption) the probability that this occurs for any one curve is $\mu(\alpha,\beta)$. Thus, the expected number of curves required by ECM is $C(\alpha,\beta) = 1/\mu(\alpha,\beta)$, and (assuming that μ is small) the probability of success after at most t curves is

$$1 - (1-\mu(\alpha,\beta))^t \simeq 1 - \exp(-t/C)\ . \tag{29}$$

If the birthday paradox continuation is used, the performance depends on the exponent e, and it is possible for phase 2 to succeed with $g_1 > B_2$. From (13), the probability of success for one curve is approximately

$$\frac{1}{\phi(2e)} \sum \int_0^{1/\alpha} \int_\psi^{1-\psi} \left(1 - \exp\left(-\gamma B_3 (d/p)^\theta\right)\right) \Phi_1\left(\frac{\psi}{1-\theta-\psi}\right) \frac{d\theta d\psi}{\theta\psi}\ ,$$

where the sum is over the $\phi(2e)$ possible values of $g_1 \bmod 2e$, γ ranges over the corresponding values of $\mathrm{GCD}(2e, g_1 - 1)$, and $\alpha = \log(p/d)/\log B_1 > 2$. Assuming close to the optimal choice of parameters for factors of 30 to 40 decimal digits, numerical integration indicates that the expected cost of the birthday paradox continuation (without fast polynomial evaluation) is 15% to 20% more than for the (improved) standard continuation if $e = 6$, and 9% to 14% more if $e = 12$. Because the analysis is simpler, in the following we assume the improved standard continuation.

To choose optimal parameters, we note that the time to perform both phases on one curve is proportional to $K_1 B_1 + K_2 B_3$, provided overheads such as table initialization are negligible. The constants K_1 and K_2 depend on details of the implementation (see §3).

In principle, if we knew p in advance, we could choose B_1 and B_2 to minimize the expected run time, which is proportional to the expected number of multiplications mod N:

$$W = (K_1 B_1 + K_2 B_3)/\mu(\alpha,\beta)\ .$$

Recall that α and β are functions of B_1 and B_2, so this is a simple problem of minimization in two variables. Suppose that the minimum is W_{opt}. Tables of optimal parameters are given in [3, 7, 54, 59, 79], with each paper making slightly different assumptions. In Table 2 we give a small table of $\log_{10} W_{opt}$ for factors of D decimal digits. We assume that $K_1 = 11/\log 2$, $K_2 = 1$, and $\log_{10} p \simeq D - 0.5$. Some computed values of $\tau(p)$ are also shown in Table 2, where

$$\tau(p) = \frac{(\log W_{opt})^2}{\log p \log\log p}\ ,$$

so

$$W_{opt} = \exp\left(\sqrt{\tau(p) \log p \log\log p}\right)\ .$$

Since the expected run time is insensitive to changes in B_1 and B_2 near the optimal values, it is not important to choose them accurately. In practice, the significant point is that we do not know p in advance. Various strategies have been suggested to overcome this difficulty. Our strategy has been to increase B_1 as a function of the number of curves t which have been tried, using the fact that for the optimal choice we expect B_1/t to be about 330 for 30-digit factors and to be fairly insensitive to the size of the factor. Given B_1, we choose B_2 so the time for phase 2 is about half that for phase 1 (this choice is not far from optimal). If $B_1 \simeq 10^6$ this gives $B_2 \simeq 100 B_1$.

Once a factor p has been found, we can compute the efficiency E, defined as the ratio of the expected time to find p with optimal parameters to the expected time with the parameters

TABLE 2. Expected work for ECM

digits D	$\log_{10} W_{opt}$	τ
20	7.35	1.677
30	9.57	1.695
40	11.49	1.707
50	13.22	1.716
60	14.80	1.723

actually used. For an example in which we started with B_1 too small but gradually increased it to a value close to optimal, see Table 3.

From the asymptotic behaviour of the functions $\rho(\alpha)$ and $\mu(\alpha, \beta)$, it can be shown that the expected speedup S because of the use of phase 2 (standard continuation), compared to just using phase 1, is of order $\log\log p$. It is argued in [7, §7] that the birthday paradox continuation gives a speedup of order $\log p$ (though only if asymptotically fast polynomial evaluation is used; for our implementations the speedup is of order $\log\log p$). The speedup for the FFT continuation is probably of order $\log p$ at most. Although these speedups are important in practice, they are theoretically insignificant, because they can be absorbed into the $o(1)$ term in Lenstra's result (26). Thus, we expect $\tau(p) \to 2$ as $p \to \infty$, independent of whether phase 2 is used. Table 2 shows that the convergence is very slow and that $\tau(p) \simeq 1.7$ for p in the 25 to 45 digit range.

We note a lack of symmetry in (29) which may be of interest to cryptographers [70]. If t is much larger than C, say $t = 100C$, then the probability of failure is $\exp(-100)$, so we are almost certain to find a factor. On the other hand, if t is much smaller than C, say $t = C/100$, then $1 - \exp(-t/C) \simeq t/C \simeq 0.01$ is small, but not exponentially so. Thus, ECM has a non-negligible chance of finding factors which are much larger than expected. For example, if the work performed is sufficient to find 30-digit factors with probability 0.5, then with the same work there is about one chance in 100 of finding a factor of 40 digits and about one chance in 10000 of finding a factor of 50 digits (we do not attempt to be precise because the probabilities depend to some extent on how the parameters B_1 and B_2 are chosen).

5. SOME ECM IMPLEMENTATIONS

For future reference, we describe several of our implementations of ECM.

A. Our first implementation was written in 1985, mainly in Pascal, but with some assembler to speed up large-integer multiplications. It used Montgomery's forms (7) and (8) for phase 1, and converted back to Weierstrass normal form (2) for phase 2, which used the birthday paradox idea. Rational preconditioning [65] was used to speed up polynomial evaluation in phase 2. The implementation achieved $K_1 = 10/\log 2$ and $K_2 = 1/2$ (recall that, as in §2.2–§3.3, the number of multiplications mod N per curve is about $K_1 B_1 + K_2 B_3$). Program A ran on various machines, including Sun 3 and VAX, and found many factors of 25 decimal digits or less [18].

B. A simple Turbo Pascal implementation was written in 1986 for an IBM PC [9]. Program B uses the Cauchy form (6) and only phase 1 is implemented. The implementation of multiple-precision arithmetic uses strings of decimal digits, so is simple but inefficient. Program B is mainly used to generate tables [18], taking into account algebraic and Aurifeuillian factors [14], and accessing a database of over 190,000 known factors. As a byproduct, program B can produce lists of composites which are used as input to other programs, e.g. programs C–E below.

C. When a vector processor[2] became available early in 1988, a Fortran program MVFAC was written (based on program A, with some improvements and simplifications [11]). Vectorization is accomplished by working on a number of elliptic curves in parallel during phase 1. Phase 2 implements the birthday paradox idea as in §3.3. During phase 2 the program works on only one curve at a time, but takes advantage of the vector units during polynomial evaluation. Unlike program A, both phases use Montgomery's form (8), with $K_1 = 11/\log 2$ and $K_2 = 1$. The initial point is usually chosen to ensure that the group order is divisible by 12, as described in §2.3. Multiple-precision arithmetic (with base 2^{26}) in the inner loops is performed using double-precision floating-point operations. `INT` and `DFLOAT` operations are used to split a product into high and low-order parts. Operations which are not time-critical, such as input and output, are performed using the MP package [4]. Program C found the factorization of F_{11} (see §7) and many factors, of size up to 40 decimal digits, needed for [15, 16, 18]. Keller [46] used program C to find factors up to 39 digits of Cullen numbers.

D. MVFAC also runs with minor changes on other machines with Fortran compilers, e.g. Sun 4 workstations. For machines using IEEE floating-point arithmetic the base must be reduced to 2^{24}. Although the workstations do not have vector units, the vectorized code runs efficiently because of the effect of amortizing loop startup overheads over several curves and keeping most memory accesses in a small working set (and hence in the cache). Program D found the p_{40} factor of F_{10} (see §6).

E. Late in 1994, our Sun 4 code was ported to the Fujitsu AP1000, which is a parallel machine using 25 Mhz Sparc processors [35, 43]. The machine available to us has 128 processors. Each processor works on one or more curves independently, and communicates results (if any) after each phase. Program E found the p_{41} factor mentioned in §5.2, and other factors for an extension of the Cunningham project [18].

F. In December 1994, a new implementation was written in C for the Dubner Cruncher (see §8). Initially the C code was modelled on program B, so used the Cauchy form. Phase 2 was implemented using a storage-efficient version of the birthday paradox idea, much as described in §3.3. Program F found a p_{27} factor of F_{13} (see §8).

G. In June 1995 the Cruncher program was rewritten to use the Montgomery form (8) throughout, and to use a storage-efficient improved standard continuation for phase 2 (as described in §3.2). Program G is significantly faster than program F (about 35% faster on each curve, and the expected number of curves is reduced because the group order is divisible by 12). The Cruncher programs F and G do not use extended GCD computations. For this reason, the constants $K_1 = 12/\log 2$ (for program G) and $K_2 = 3$ are larger than for programs C–E. The effect on relative performance is less than might appear, because the overall performance is more sensitive to K_1 than to K_2.

5.1. **The multiplication algorithm.** Most of the cost of ECM is in performing multiplications mod N. Our programs A-G all use the classical $O(n^2)$ algorithm to multiply n-bit numbers. Karatsuba's algorithm [47, §4.3.3] or other "fast" algorithms [26, 28] would be preferable for large n. The crossover point depends on details of the implementation. Morain [62, Ch. 5] states that Karatsuba's method is worthwhile for $n \geq 800$ on a 32-bit workstation. The crossover point on a 64-bit vector processor is probably slightly larger. On a Cruncher the crossover is much larger because the multiplication time is essentially linear in n for $n < 10000$ (see Table 4).

Programs B–E do not take advantage of the special form of Fermat numbers when doing arithmetic mod N. However, the mod operation is implemented efficiently. For programs C–E

[2]Initially a Fujitsu VP100 with 7.5 nsec clock cycle; this machine was upgraded in mid-1991 to a Fujitsu VP2200/10 with 4.0 nsec (later 3.2 nsec) clock cycle and theoretical peak speed of 1000 Mflop (later 1250 Mflop). Times quoted for the VP2200 are for the faster version.

the operation $X \leftarrow Y \times Z \bmod N$ is coded as a single routine. As Y is multiplied by each digit d_1 of Z (starting with the most significant digit) and the sum accumulated in X, we also predict a quotient digit d_2, multiply N by d_2, and subtract. The predicted quotient digit can differ from the correct value by one, because a slightly redundant representation of the intermediate results allows a small error to be corrected at the next step (when working in base B, digits in the range $[0, B]$ are permitted). Also, the result of the operation is only guaranteed to lie in the interval $[0, 2N)$ and to be correct mod N. With these refinements, $X \leftarrow Y \times Z \bmod N$ can be performed almost as fast as $X \leftarrow Y \times Z$. For t-bit numbers, program C performs $9(t/26)^2 + O(t)$ flops per multiplication mod N; this takes $4(t/26)^2 + O(t)$ clock cycles on the VP2200/10.

For the Cruncher programs F-G, we avoid the mod operation as much as possible. If the number N to be factored is a composite divisor of $2^n \pm 1$, then the elliptic curve operations are performed mod $2^n \pm 1$ rather than mod N. At the end of each phase we compute a GCD with N (not with $2^n \pm 1$ because this would give known factors of $(2^n \pm 1)/N$). A minor penalty is that we work with n or $(n+1)$-bit numbers rather than with $\lceil \log_2 N \rceil$-bit numbers. The advantage is that we can perform the reductions mod $2^n \pm 1$ using binary shift and add/subtract operations, which are much faster (for large n) than multiply or divide operations. Thus, the Cruncher programs F-G run about twice as fast on Fermat or Mersenne numbers as on "general" numbers.

5.2. **Some examples.** By the end of 1995, at least 17 factors of 40 or more decimal digits had been found by ECM. To illustrate the performance of ECM, we mention three of these factors here. Details are available in [12]. Because of the selection criterion, these examples do not illustrate typical behaviour of ECM; rather, they illustrate the asymmetry noted at the end of §4.4.

The largest factor found by the author is

$$p_{41} = 25233450176615986500234063824208915571213 \ ,$$

a factor of $55^{126} + 1$. It was found using program E with $B_1 = 240000$, $D = 15$, $e = 1$. The curve is defined by $\sigma = 805816989$, and the group order is

$$g = 2^2 \cdot 3^2 \cdot 47 \cdot 61 \cdot 89 \cdot 233 \cdot 829 \cdot 3607 \cdot 18451 \cdot 54869 \cdot 75223 \cdot 149827 \cdot 345551 \ .$$

The group order g is exceptionally smooth. From §4.1, the probability that a random integer near $g/12$ has largest and second-largest prime factors as small as those of g is about

$$\mu\left(\frac{\log(g/12)}{\log 149827}, \frac{\log 345551}{\log 149827}\right) \simeq 3.6 \times 10^{-7} \ .$$

Another large factor found by the author, using program C with $B_1 = 370000$, $D = 255$, $e = 6$, is the 40-digit factor

$$p'_{40} = 9409853205696664168149671432955079744397$$

of $p_{252} - 1$, where p_{252} is the largest prime factor of F_{10}. See §9 for the application to proving primality of p_{252}. The curve is defined by $\sigma = 48998398$, and the group order is

$$g = 2^2 \cdot 3 \cdot 5 \cdot 17 \cdot 31^2 \cdot 53 \cdot 67 \cdot 233 \cdot 739 \cdot 5563 \cdot 7901 \cdot 20201 \cdot 57163 \cdot 309335137 \ .$$

The largest prime factor g_1 of g is about $836B_1$ which shows the virtue of the birthday paradox continuation. Note that $\mathrm{GCD}(2e, g_1 - 1) = 12$.

The largest factor known to have been found by ECM up to the end of 1995 is the 47-digit factor

$$p_{47} = 12025702000065183805751513732616276516181800961$$

of $5^{256}+1$, found in November 1995 by Peter Montgomery [61], using $B_1 = 3000000$ with his FFT continuation. For more details, see [12]. The group order is

$$g = 2^6 \cdot 3 \cdot 5 \cdot 7 \cdot 23 \cdot 997 \cdot 43237 \cdot 554573 \cdot 659723 \cdot 2220479 \cdot 2459497 \cdot 903335969 \ .$$

6. Factorization of F_{10}

When ECM was implemented on the Fujitsu VP100 in March 1988, some of the first numbers which we attempted to factor were the Fermat numbers F_9, F_{10}, F_{11} and F_{12}, using variants of program C. We were soon successful with F_{11} (see §7), but not with the other Fermat numbers, apart from rediscovering known factors. We continued attempts to factor F_9 by ECM (until it was done by SNFS, see §1), F_{10} and F_{12}, devoting of the order of 25% of available computer time to these Fermat numbers and the remainder to other projects. The phase 1 limit B_1 was increased as it became apparent that the unknown factors were difficult to find. However, there were some constraints on B_1. Batch jobs were limited to at most two hours and, to make efficient use of the vector units, we had to complete several curves in that time. The time for t curves done simultaneously was proportional to $t + t_{1/2}$, where $t_{1/2}$ depended on the startup cost of vector operations.

We ran about 2000 curves with $B_1 = 60000$ on the VP100 in the period March 1988 to late 1990. Each run on the VP100 took slightly less than two hours for 63 curves, with $t_{1/2} \simeq 10$. The VP100 was upgraded to a VP2200 in 1991, and we ran about 17360 curves with $B_1 = 200000$ on the VP2200 in the period August 1991 to August 1995. Each run on the VP2200 took slightly less than two hours for 62 curves, with $t_{1/2} \simeq 14$. The improvement in speed over the VP100 was partly due to rewriting the inner loop of program C to reduce memory references and improve the use of vector registers.

In September 1994 we started running program D on one or two 60 Mhz SuperSparc processors. Usually we used one processor for F_{10} and one for F_{12}. In July 1995 six more 60 Mhz SuperSparc processors became available for a limited period. We attempted to factor F_{10} on all eight SuperSparcs using computer time which was not required by other users. Details are given in Table 3. At the same time we were attempting to factor F_{12} to F_{15} using ECM on Dubner Crunchers (see §8).

In Table 3, F is an estimate of the expected number of times that the factor p_{40} should be found with the given B_1 and number of curves (see §4.4). E is an estimate of the efficiency compared to the optimal choice of $B_1 \simeq 3400000$. The programs used the birthday paradox continuation, but the estimates of E and F assume the improved standard continuation with $B_2 = 100B_1$, so they are only approximate (see §4.4). The last row of the table gives totals (for number of curves and F) and weighted means (for B_1 and E).

The p_{40} factor of F_{10} was found by a run which started on Oct 14 and finished on Oct 20, 1995. The run tried 10 curves with $B_1 = 2000000$ in about 114 hours of CPU time, 148 hours elapsed wall-clock time.

From the factorizations of $p_{40} \pm 1$ given in §9, it is easy to prove that p_{40} is prime, and also to see why the Pollard $p \pm 1$ methods did not find p_{40}.

The 252-digit cofactor c_{291}/p_{40} was rigorously proved to be prime by two independent methods (see §9).

6.1. **In retrospect.** From column F of Table 3, it appears that we were fortunate to find p_{40} as soon as we did. The probability is about $1 - \exp(-0.2434) \simeq 0.22$. We know of some other attempts. Bruce Dodson tried about 100 curves with $B_1 = 2 \times 10^6$, Peter Montgomery tried about 1000 curves with $B_1 \le 10^7$ (mean value unknown), Robert Silverman tried about 500 curves with $B_1 = 10^6$, and Samuel Wagstaff tried "a few dozen" curves with $150000 \le B_1 \le 400000$. There were probably other attempts of which we are unaware, but the size of the known

TABLE 3. ECM runs on F_{10}

B_1	curves	F	E	machine(s) and dates
6×10^4	2000	0.0010	0.14	VP100, Mar 1988 – Nov 1990
2×10^5	17360	0.0910	0.42	VP2200, Aug 1991 – Aug 1995
5×10^5	700	0.0152	0.69	Sparc × 2, Sep 1994 – Jul 1995
10^6	480	0.0262	0.87	Sparc × 8, Jul 1995 – Aug 1995
2×10^6	900	0.1100	0.98	Sparc × 8, Aug 1995 – Oct 1995
2.9×10^5	21440	0.2434	0.63	

composite factor of F_{10} (291 decimal digits) reduced the number of attempts. For example, Montgomery's Cray program was restricted to inputs of at most 255 digits, and Wagstaff did not use the Maspar [33] because it would have required a special "size 33" program.

From column E of the table, it is clear that the runs on the VP100 and VP2200 were inefficient. We should have implemented a form of checkpointing so that B_1 could be increased to at least 10^6, allowing us to take more than two hours per set of curves. At the time we did not know that the unknown factor had 40 decimal digits, though by mid-1995 we were reasonably confident that it had at least 35 digits. Our general strategy was to increase B_1 gradually, guided by estimates of the optimal B_1 and the expected number of curves for factors of 30–40 digits. We did not expect to find any factors much larger than 40 digits.

6.2. **The computational work.** Each curve on a 60 Mhz SuperSparc takes about $5.7 \times 10^{-6} B_1$ hours of CPU time. If a 60 Mhz SuperSparc is counted as a 60-Mips machine, then our computation took about 240 Mips-years. This is comparable to the 340 Mips-years estimated for sieving to factor F_9 by SNFS [53]. (SNFS has since been improved by Montgomery and others, so the 340 Mips-years could probably be reduced by an order of magnitude.)

Since the inner loops of programs C–E use floating-point arithmetic, Mflops are a more appropriate measure than Mips. The VP2200/10 is rated at 1250 Mflop (peak performance). If our factorization of F_{10} had been performed entirely on the VP2200, it would have taken about 6 weeks of machine time, or 140 Mflop-years. Cryptographers should note that this amounts to about 75 minutes on a 1 Teraflop machine.

The number of multiplications (mod N) is a machine-independent measure of the work to factor N. Each curve takes about $22.9\,B_1$ such multiplications. Overall, our factorization of F_{10} took 1.4×10^{11} multiplications (mod N), where $N = c_{291}$. (Table 2 predicts 3.3×10^{11} with the optimal choice of parameters.) Numbers mod c_{291} were represented with 38 digits and base 2^{26} (on the VP100/VP2200) or with 41 digits and base 2^{24} (on the Sparc), so each multiplication (mod N) took more than 10^4 floating-point operations.

6.3. **The group order.** The successful elliptic curve and starting point are defined by (8) and (11), with $\sigma = 14152267$ (derived from the starting date and time October 14, 15:22:54). Explicitly, we can take the elliptic curve in the form (7) as

$$by^2 = x^3 + 159744730829031835228495734317285840361 8x^2 + x \mod p_{40} .$$

This may also be written as $by^2 = x(x-k)(x-1/k) \bmod p_{40}$, where

$$k = 10368222255137077461535231735177787850 47 .$$

b is any quadratic non-residue (mod p_{40}), e.g. $b = 5$. The group order is

$$\begin{aligned} g &= p_{40} + 1 - 3674872259129499038 \\ &= 2^2 \cdot 3^2 \cdot 5 \cdot 149 \cdot 163 \cdot 197 \cdot 7187 \cdot 18311 \cdot 123677 \cdot 226133 \cdot 314263 \cdot 4677853 . \end{aligned}$$

The probability that a random integer near $g/12$ has largest prime factor at most 4677853 and second-largest prime factor at most 314263 is about 5.8×10^{-6}. The phase 1 limit for the successful run was $B_1 = 2 \times 10^6$, but programs C–E and G find p_{40} with B_1 as small as 314263 if the same curve and starting point are used.

7. Factorization of F_{11}

After the factorization of F_8 in 1980, no one predicted that F_{11} would be the next Fermat number to be completely factored. Program C, described in §5, was implemented on a Fujitsu VP 100 in March 1988. After failing to find any new factors of F_9 and F_{10}, we compiled "large" versions of program C suitable for F_{11} and F_{12}. We knew four prime factors of F_{12} from [41]. On May 6, 1988, a run found a fifth factor

$$p_{16} = 1256132134125569 = 2^{14} \cdot 7^2 \cdot 53 \cdot 29521841 + 1$$

of F_{12}. Later we learned that R. Baillie had already found this factor in July 1986, using the Pollard $p - 1$ method[3].

A large version of program C took slightly less than one hour for 20 curves with $B_1 = 16000$ on the number $c_{606} = F_{11}/(319489 \cdot 974849)$. On May 13, 1988, a 22-decimal digit factor

$$p_{22} = 3560841906445833920513 = 2^{13} \cdot 7 \cdot 677 \cdot p_{14} + 1$$

was found by phase 2. We had previously tried 68 curves unsuccessfully with $B_1 \simeq 15000$. Overall it took about 2.8×10^7 multiplications (mod c_{606}) and slightly less than four hours of machine time to find p_{22}. Dividing c_{606} by p_{22} gave a 584-digit composite number c_{584}.

The next run of program C, on May 17, 1988, found a 21-digit factor

$$p_{21} = 167988556341760475137 = 2^{14} \cdot 3 \cdot 373 \cdot 67003 \cdot 136752547 + 1$$

of c_{584}, again using 20 curves with $B_1 = 16000$. It was surprising that a larger factor (p_{22}) had been found first, but because of the probabilistic nature of ECM there is no guarantee that smaller factors will be found before larger ones. Overall, it took about 3.6×10^7 multiplications (mod c_{606} or c_{584}) and less than five hours of machine time to find both factors by ECM. It would have been feasible to find p_{21} (but not p_{22}) by the Pollard $p - 1$ method.

The quotient had 564 digits and, to our surprise (see §10), it passed a probabilistic primality test [47, p. 379]. If this test is applied to a composite number, the chance that it incorrectly claims the number is prime is less than 1/4. We ran many independent trials, so we were confident that the quotient was indeed prime and that the factorization of F_{11} was complete. This was verified by Morain, as described in §9. The complete factorization of F_{11} was announced in [8]:

$$F_{11} = 319489 \cdot 974849 \cdot 167988556341760475137 \cdot 3560841906445833920513 \cdot p_{564}$$

8. A new factor of F_{13}

The Dubner Cruncher [22, 34] is a board which plugs into an IBM-compatible PC. The board has a digital signal processing chip (LSI Logic L64240 MFIR) which, when used for multiple-precision integer arithmetic, can multiply two 512-bit numbers in 6.4 μsec. A software library has been written by Harvey and Robert Dubner [34]. This library allows a C programmer to use the Cruncher for multiple-precision integer arithmetic. Some limitations are:

[3]The resulting confusion explains why the fifth factor of F_{12} is not correctly attributed to Baillie in [25, p. 148].

1. Communication between the Cruncher and the PC (via the PC's ISA bus) is relatively slow, so performance is much less than the theoretical peak for numbers of less than say 1000 bits. For example, when multiplying 512-bit numbers on a Cruncher running in an 80386/40 PC, only about 10% of the peak performance is achieved.
2. Because of the slow communication it is desirable to keep operands in the on-board memory, of which only 256 KByte is accessible to the C programmer.

Program F (see §5) was implemented and debugged early in December 1994. Because of memory limitations, it could not be used for Fermat numbers larger than F_{15} (a number the size of F_{15} requires 4 KByte of storage). In the period January – June 1995 we used a Cruncher in an 80386/40 PC to attempt to factor F_{13} and F_{14} (and some non-Fermat numbers). We mainly used phase 1 limit $B_1 = 100000$. On F_{13} each curve took 137 minutes (91 minutes for phase 1 and 46 minutes for phase 2); on F_{14} each curve took 391 minutes. For comparison, program D running on a 60 Mhz SuperSparc on the much smaller number F_{10} takes 34 minutes per curve, and the Cruncher takes 17 minutes per curve. The Cruncher's advantage over the Sparc ranges from a factor of 2 near F_{10} to a factor of 22 for F_{14}. For numbers as large as F_{14} the Sparc program could be improved by using a faster multiplication algorithm such as Karatsuba's algorithm (see §5.1). However, since $391/137 < 3$, Karatsuba's algorithm would not be helpful on the Cruncher.

Three prime factors of F_{13} were known:

$$F_{13} = 2710954639361 \cdot 2663848877152141313 \cdot 3603109844542291969 \cdot c_{2417} .$$

The first factor was found by Hallyburton and Brillhart [41]. The second and third factors were found by Crandall [25] on Zilla net (a network of about 100 workstations) in January and May 1991, using ECM.

On June 16, 1995 our Cruncher program F found a fourth factor

$$p_{27} = 319546020820551643220672513 = 2^{19} \cdot 51309697 \cdot 11878566851267 + 1$$

after a total of 493 curves with $B_1 = 100000$, $B_3 = 485301$, and table size $D = 83$ (see §3.3). The overall machine time was about 47 days. We note that $p_{27} + 1 = 2 \cdot 3 \cdot 7^3 \cdot 59 \cdot p_{22}$. The factorizations of $p_{27} \pm 1$ explain why Pollard's $p \pm 1$ methods could not find the factor p_{27} in a reasonable time.

The successful curve was of the form (6) with initial point

$$(x_1 : y_1 : z_1) = (150400588188733400929847531 : 277194908510676462587880207 : 1) \bmod p_{27}$$

and group order (not divisible by 4 in this case)

$$g = 3^2 \cdot 7^2 \cdot 13 \cdot 31 \cdot 3803 \cdot 6037 \cdot 9887 \cdot 28859 \cdot 274471 .$$

Using Fermat's little theorem [20, p. lviii], we found c_{2417}/p_{27} to be composite. Thus, we now know that

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

At about the time that the p_{27} factor of F_{13} was found, program F was modified to give program G (see §5). Testing program G, we "rediscovered" the p_{27} factor after 871 curves with $B_1 = 100000$, $B_2 = 3542000$. After increasing B_1 to 500000, we found p_{27} again (this time by phase 1), after 216 curves. The expected numbers of curves, predicted as in §4.4, are 537 and 137 respectively. The successful curves are defined by (11) with $\sigma = 4009189$ and $\sigma = 8020345$

TABLE 4. ECM runs on F_n, $n = 12, \ldots, 15$

n	B_1	curves	hours/curve
12	10^6	550	5.9
13	5×10^5	430	6.6
14	5×10^5	300	17.2
15	2×10^5	360	21.9

respectively, and the group orders are

$$\begin{aligned} g &= 2^7 \cdot 3 \cdot 127 \cdot 3083 \cdot 3539 \cdot 9649 \cdot 18329 \cdot 3395653 \quad \text{and} \\ g &= 2^3 \cdot 3 \cdot 17 \cdot 23 \cdot 41 \cdot 113 \cdot 271 \cdot 3037 \cdot 10687 \cdot 12251 \cdot 68209 \ . \end{aligned}$$

The fact that programs F and G found the same 27-digit factor three times suggests (but does not prove) that the unknown factors of F_{13} are larger than p_{27}.

8.1. **Factorization using Crunchers.** The combination of a cheap PC and a Cruncher board ($US2,500) is very cost-effective for factoring large integers by ECM. Given the initial value $\sigma = 14152267$, the Cruncher program G running in an 80386/40 PC with $B_1 = 320000$ can find the p_{40} factor of F_{10} in only 56 minutes (this is reduced to 38 minutes in an 80486/50 PC). One Cruncher in an 80386/40 PC could have done our 240 Mips-year computation to factor F_{10} in approximately two years. For numbers approximately as large as F_{14}, the Cruncher runs about half as fast as the VP2200/10.

In collaboration with Harvey Dubner, we are using several Crunchers in an attempt to find more factors of $F_{12}, \ldots, F_{15}$ by ECM. F_{14} is particularly interesting because it is known to be composite [73], but no prime factor is known. In Table 4 we give the current phase 1 limit B_1, the total number of curves (weighted in proportion to the phase 1 limit used) up to February 1996, and the time required per curve. The times are for program G running under DOS on a Cruncher installed in an 80386/40 PC, with B_2 chosen so phase 2 takes half as long as phase 1 (see §4.4).

9. PRIMALITY PROOFS

In [6] we give primality certificates for the prime factors of $F_5, \ldots, F_8$, using essentially the method pioneered by Lucas [56, p. 302], Kraitchik [49, p. 135] and Lehmer [51, p. 330]. To prove p prime, we completely factor $p-1$ and find a primitive root (mod p). The method is applied recursively to large prime factors of $p-1$.

Similar certificates can be given for the factors p_{49} and p_{99} of F_9, using Crandall's factorizations [53] of $p_{49} - 1$ and $p_{99} - 1$:

$$\begin{aligned} p_{49} - 1 &= 2^{11} \cdot 19 \cdot 47 \cdot 82488781 \cdot 1143290228161321 \cdot 43226490359557706629 \ , \\ p_{99} - 1 &= 2^{11} \cdot 1129 \cdot 26813 \cdot 40644377 \cdot 17338437577121 \cdot p_{68} \ , \quad \text{and} \end{aligned}$$

$$\begin{aligned} p_{68} - 1 &= 2 \cdot 3^3 \cdot 13 \cdot 1531 \cdot 173897 \cdot 1746751 \cdot 12088361983 \cdot \\ &\quad 1392542208042011209 \cdot 3088888502468305782559 \ . \end{aligned}$$

In these three cases the least primitive roots are 3.

For the penultimate factor p_{40} of F_{10}, the least primitive root is 5, and we have

$$\begin{aligned} p_{40} - 1 &= 2^{12} \cdot 3 \cdot 5639 \cdot 8231 \cdot 433639 \cdot 18840862799165386003967 \ , \\ p_{40} + 1 &= 2 \cdot 2887 \cdot 52471477 \cdot 31186157593 \cdot 493177304177011507 \ . \end{aligned}$$

The same method can not be applied to prove primality of the largest prime factors of F_{10} and F_{11}, because we have only incomplete factorizations:

$$\begin{aligned}
p_{252} - 1 &= 2^{13} \cdot 3 \cdot 13 \cdot 23 \cdot 29 \cdot 6329 \cdot 760347109 \cdot \\
&\quad 211898520832851652018708913943317 \cdot \\
&\quad 940985320569666416814967143295507974439 7 \cdot c_{158} \ , \\
p_{252} + 1 &= 2 \cdot 24407 \cdot 507702159469 \cdot c_{235} \ , \\
p_{564} - 1 &= 2^{13} \cdot 139 \cdot 1847 \cdot 32488628503 \cdot 1847272285831883 \cdot \\
&\quad 92147345984208191 \cdot 23918760924164258488261 \cdot c_{489} \ , \\
p_{564} + 1 &= 2 \cdot 3^2 \cdot 65231833 \cdot c_{555} \ .
\end{aligned}$$

We can apply Selfridge's "Cube Root Theorem" [20, Theorem 11] to p_{252}, since $p_{252} - 1 = F \cdot c_{158}$, where $F > 2 \times 10^{93}$ is completely factored, $p_{252} < 2F^3 + 2F$, and the other conditions of the Cube Root Theorem are easily verified. Thus, p_{252} is prime, and the factorization of F_{10} is complete.

The large factor p_{564} of F_{11} was proved prime by Morain (in June 1988) using a distributed version of his *ecpp* program [62, p. 13]. We have used the publicly available version of *ecpp*, which implements the "elliptic curve" method of Atkin and Morain [1, 2], to confirm this result. Version V3.4.1 of ecpp, running on a 60 Mhz SuperSparc, established the primality of p_{564} in 28 hours. It took only one hour to prove p_{252} prime by the same method. Primality "certificates" are available [13]. They can be checked using a separate program *xcheckcertif*.

10. When to use ECM, and prospects for F_{12}

When factoring large integers by ECM we do not usually know the size of the factors in advance. Thus, it is impossible to estimate how long ECM will require. In contrast, the running times of the MPQS and general number field sieve (GNFS) methods can be predicted fairly well, because they depend mainly on the size of the number being factored, and not on the size of the (unknown) factors [68]. An important question is how long to spend on ECM before switching to a more predictable method such as MPQS/GNFS.

Theorem 3 of Vershik [81] may be helpful. Roughly, it says that the ratios $\log q / \log p$ of logarithms of neighboring large prime divisors q, p $(q < p)$ of large random integers are asymptotically independent and uniformly distributed in $(0, 1)$. Using this theorem (assuming the numbers to be factored behave like random integers) or past experience gained by factoring a class of numbers (such as Cunningham numbers), we can make a rough estimate of the probability P that ECM will factor a given number in one unit of time (say one day of computer time). This estimate should take into account the information that ECM has already spent some (say U_0) units of time unsuccessfully searching for a factor. Silverman and Wagstaff [79, §7] suggest a Bayesian approach. As a simple approximation, we could use the results of §4.4 to estimate q such that the expected time for ECM to find a factor close to q is U_0, and then assume a lower bound q on the unknown factor. (This amounts to approximating the function in (29) by a step function.)

For example, if we are factoring $N \simeq 10^{100}$ and U_0 is such that $q \simeq 10^{30}$, then we could assume that the unknown factor p lies in the interval $(10^{30}, 10^{50})$ and that $1/\log_{10} p$ is uniformly distributed in the corresponding interval $(1/50, 1/30)$. The probability P can now be estimated, using the results of §4.4, if we assume that the parameters B_1 and B_2 are chosen optimally to find factors of size close to q. The estimate might, for example, be $1/P \simeq cU_0$, where $c \simeq 9$.

If the predicted running time of MPQS/GNFS exceeds $1/P$ units, then it is worthwhile to continue with ECM for a little longer. If ECM is unsuccessful, we repeat the procedure of

TABLE 5. Second-largest prime factors of F_n

n	7	8	9	10	11
ρ_2	0.98	0.45	0.82	0.27	0.06

estimating P. Eventually, either a factor will be found by ECM or the estimate of P will become so small that a switch to MPQS/GNFS is indicated. If a factor p is found by ECM, then the quotient N/p is either prime (so the factorization is complete) or much smaller than the original composite number N, and hence much more easily factored by MPQS/GNFS. (This is not true for SNFS, because knowledge of a non-algebraic factor does not greatly speed up SNFS, see [52, 53].)

Note that our approach is reasonable in the limiting case $N \to \infty$, because the assumption that $1/\log p$ is uniformly distributed in $(0, 1/\log q)$ gives a positive estimate for P. For example, replacing $N \simeq 10^{100}$ by $N \simeq F_{15}$ in the example above multiplies the constant c by a factor of about $2.5 = (1/30)/(1/30 - 1/50)$.

For the Fermat numbers F_n, $12 \le n \le 15$, the predicted probability of success for ECM is low, but the predicted running time of other methods is so large that it is rational to continue trying ECM. There is no practical alternative except the old method of trial division (see the discussion at the end of §4.3).

Although the Fermat numbers are not random integers, it is interesting to compute $\rho_2(\alpha)$ for $\alpha = \log F_n / \log p_2$, where p_2 is the second-largest prime factor of F_n and $\rho_2(\alpha)$ is defined by (14). The values for $n = 7, \ldots, 11$ are given in Table 5. For large random integers, we expect $\rho_2(\alpha)$ to be uniformly distributed (see §4.1). We see that F_{11} has a surprisingly small second-largest factor, and F_7 has a surprisingly large second-largest factor. The second-largest factors of F_8, F_9 and F_{10} are not exceptionally small or large.

The probability that a random integer N close to F_{12} has second-largest prime factor $p_2 < 10^{40}$ is 0.059. F_{12} has five known prime factors (see §1), and G. B. Gostin has shown by exhaustive search that these are its five smallest prime factors. The fifth-smallest is Baillie's $p_{16} = 1256132134125569$ (see §7). An indication of the likely size of the sixth-smallest prime factor can be obtained from Vershik's result [81, Thm. 3], which is paraphrased above. On the other hand, Harvey Dubner and the author have tried more than 500 curves with $B_1 = 10^6$, in an attempt to factor F_{12}, without finding more than the five known prime factors. Thus, from Table 2 and (29), we can be reasonably confident that the sixth-smallest prime factor of F_{12} is at least 10^{30}; a smaller factor would have been found with probability greater than 0.9.

The complete factorization of F_{12} may have to wait for the physical construction of a quantum computer capable of running Shor's algorithm [76], or a surprising new development such as a classical (deterministic or random) polynomial-time integer factoring algorithm. Here, "polynomial-time" means that the expected run time should be bounded above by a polynomial in the *length* (not the index) of the number to be factored.

REFERENCES

[1] A. O. L. Atkin and F. Morain, *Finding suitable curves for the elliptic curve method of factorization,* Math. Comp. **60** (1993), 399–405.

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

[3] W. Bosma and A. K. Lenstra, *An implementation of the elliptic curve integer factorization method,* Computational Algebra and Number Theory (edited by W. Bosma and A. van der Poorten), Kluwer Academic Publishers, Dordrecht, 1995, 119–136.

[4] R. P. Brent, *Algorithm 524: MP, a Fortran multiple-precision arithmetic package,* ACM Trans. on Mathematical Software **4** (1978), 71–81.

[5] R. P. Brent, *An improved Monte Carlo factorization algorithm,* BIT **20** (1980), 176–184.
[6] R. P. Brent, *Succinct proofs of primality for the factors of some Fermat numbers,* Math. Comp. **38** (1982), 253–255.
[7] R. P. Brent, *Some integer factorization algorithms using elliptic curves,* Australian Computer Science Communications **8** (1986), 149–163. [Errata: (9.2) should read $by^2 = x^3 + ax^2 + x \pmod N$.] Also Report CMA-R32-85, Centre for Mathematical Analysis, Australian National University, Canberra, Sept. 1985, 20 pp.
[8] R. P. Brent, *Factorization of the eleventh Fermat number (preliminary report),* AMS Abstracts **10** (1989), 89T-11-73.
[9] R. P. Brent, *Factor: an integer factorization program for the IBM PC,* Report TR-CS-89-23, Computer Sciences Laboratory, Australian National Univ., Canberra, Oct. 1989, 7 pp.
[10] R. P. Brent, *Parallel algorithms for integer factorisation,* Number Theory and Cryptography (edited by J. H. Loxton), Cambridge University Press, 1990.
[11] R. P. Brent, *Vector and parallel algorithms for integer factorisation,* Proc. Third Australian Supercomputer Conference, Melbourne, 1990.
[12] R. P. Brent, *Large factors found by ECM,* available by ftp from `nimbus.anu.edu.au:/pub/Brent/champs.ecm` .
[13] R. P. Brent, *Primality certificates for factors of some Fermat numbers,* available by ftp from `nimbus.anu.edu.au:/pub/Brent/F10p252.cer, F11p564.cer` .
[14] R. P. Brent, *On computing factors of cyclotomic polynomials,* Math. Comp. **61** (1993), 131–149.
[15] R. P. Brent and G. L. Cohen, *A new lower bound for odd perfect numbers,* Math. Comp. **53** (1989), 431–437. Supplement, *ibid*, S7–S24.
[16] R. P. Brent, G. L. Cohen and H. J. J. te Riele, *Improved techniques for lower bounds for odd perfect numbers,* Math. Comp. **57** (1991), 857–868.
[17] R. P. Brent and J. M. Pollard, *Factorization of the eighth Fermat number,* Math. Comp. **36** (1981), 627-630. Preliminary announcement in AMS Abstracts **1** (1980), 565.
[18] R. P. Brent and H. J. J. te Riele, *Factorizations of $a^n \pm 1$, $13 \le a < 100$,* Report NM-R9212, Department of Numerical Mathematics, Centrum voor Wiskunde en Informatica, Amsterdam, June 1992. Also (with P. L. Montgomery), *Update 1 to Factorizations of $a^n \pm 1$, $13 \le a < 100$,* Report NM-R9419, Centrum voor Wiskunde en Informatica, Amsterdam, September 1994. Available by ftp from `nimbus.anu.edu.au:/pub/Brent/rpb134*.*.Z` .
[19] J. Brillhart, *Some miscellaneous factorizations,* Math. Comp. **17** (1963), 447–450.
[20] J. Brillhart, D. H. Lehmer, J. L. Selfridge, B. Tuckerman, and S. S. Wagstaff, Jr., *Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers*, 2nd ed., Amer. Math. Soc., Providence, RI, 1988. Also updates to Update 2.9, August 16, 1995.
[21] N. G. de Bruijn, *The asymptotic behaviour of a function occurring in the theory of primes,* J. Indian Math. Soc. **15** (1951), 25–32.
[22] C. Caldwell, *The Dubner PC Cruncher – a microcomputer coprocessor card for doing integer arithmetic*, review in J. Rec. Math. **25**(1), 1993.
[23] D. V. and G. V. Chudnovsky, *Sequences of numbers generated by addition in formal groups and new primality and factorization tests,* Adv. in Appl. Math. **7** (1986), 187–237.
[24] H. Cohen, *Elliptic curves,* From Number Theory to Physics (edited by M. Waldschmidt, P. Moussa, J.-M. Luck and C. Itzykson), Springer-Verlag, New York, 1992, 212–237.
[25] R. E. Crandall, *Projects in scientific computation,* Springer-Verlag, New York, 1994.
[26] R. E. Crandall, *Topics in advanced scientific computation,* Springer-Verlag, New York, 1996.
[27] R. Crandall, J. Doenias, C. Norrie, and J. Young, *The twenty-second Fermat number is composite,* Math. Comp. **64** (1995), 863–868.
[28] R. Crandall and B. Fagin, *Discrete weighted transforms and large-integer arithmetic,* Math. Comp. **62** (1994), 305–324.
[29] A. J. C. Cunningham and A. E. Western, *On Fermat's numbers,* Proc. London Math. Soc. (2) **1** (1904), 175.
[30] A. J. C. Cunningham and H. J. Woodall, *Factorisation of $y^n \mp 1$, $y = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers (n),* Hodgson, London, 1925.
[31] M. Deuring, *Die Typen der Multiplikatorenringe elliptischer Funktionenkörper,* Abh. Math. Sem. Hansischen Univ. **14** (1941), 197–272.
[32] K. Dickman, *On the frequency of numbers containing prime factors of a certain relative magnitude,* Ark. Mat., Astronomi och Fysik **22A**, 10 (1930), 1–14.
[33] B. Dixon and A. K. Lenstra, *Massively parallel elliptic curve factoring,* Proc. Eurocrypt '92, Lecture Notes in Computer Science **658**, Springer-Verlag, Berlin, 1993, 183–193.

[34] H. Dubner and R. Dubner, *The Dubner PC Cruncher: Programmers Guide and Function Reference,* February 15, 1993.
[35] C. Eldershaw and R. P. Brent, *Factorization of large integers on some vector and parallel computers,* Proceedings of Neural, Parallel and Scientific Computations **1** (1995), 143-148.
[36] L. Euler, *Observationes de theoremate quodam Fermatiano aliisque ad numeros primos spectantibus,* Comm. Acad. Sci. Petropol. **6**, ad annos 1732–33 (1738), 103–107; Leonhardi Euleri Opera Omnia, Ser. I, vol. II, Teubner, Leipzig, 1915, 1–5.
[37] L. Euler, *Theoremata circa divisores numerorum,* Novi. Comm. Acad. Sci. Petropol. **1**, ad annos 1747–48 (1750), 20–48.
[38] P. de Fermat, *Oeuvres de Fermat,* vol. II: *Correspondance,* P. Tannery and C. Henry (editors), Gauthier-Villars, Paris, 1894.
[39] V. Goncharov, *On the field of combinatory analysis,* Izv. Akad. Nauk SSSR Ser. Mat. **8** (1944), 3–48; English transl. in Amer. Math. Soc. Transl. (2) **19** (1962), 1–46.
[40] G. B. Gostin, *New factors of Fermat numbers,* Math. Comp. **64** (1995), 393–395.
[41] J. C. Hallyburton and H. Brillhart, *Two new factors of Fermat numbers,* Math. Comp. **29** (1975), 109–112. Corrigendum, *ibid* **30** (1976), 198.
[42] H. Hasse, *Beweis des Analogons der Riemannschen Vermutung für die Artinschen u. F. K. Schmidtschen Kongruenzzetafunktionen in gewissen elliptischen Fällen,* Nachr. Gesell. Wissen. Göttingen I **42** (1933), 253–262.
[43] H. Ishihata, T. Horie and T. Shimizu, *Architecture for the AP1000 highly parallel computer,* Fujitsu Sci. Tech. J. **29** (1993), 6–14.
[44] J.-R. Joly, *Equations et variétés algébriques sur un corps fini,* L'Enseignement Mathématique **19** (1973), 1–117.
[45] W. Keller, *Factors of Fermat numbers and large primes of the form* $k \cdot 2^n + 1$, Math. Comp. **41** (1983), 661–673. Also part II, preprint, Universität Hamburg, Sept. 27, 1992 (available from the author).
[46] W. Keller, *New Cullen primes,* Math. Comp. **64** (1995), 1733–1741.
[47] D. E. Knuth, *The art of computer programming, Volume 2: Seminumerical algorithms* (2nd ed.), Addison-Wesley, Menlo Park, CA, 1981.
[48] D. E. Knuth and L. Trabb Pardo, *Analysis of a simple factorization algorithm,* Theor. Comp. Sci. **3** (1976), 321–348.
[49] M. Kraitchik, *Théorie des nombres,* Tome 2, Gauthier-Villars, Paris, 1926.
[50] F. Landry, *Note sur la décomposition du nombre* $2^{64} + 1$ (Extrait), C. R. Acad. Sci. Paris **91** (1880), 138.
[51] D. H. Lehmer, *Tests for primality by the converse of Fermat's theorem,* Bull Amer. Math. Soc. **33** (1927), 327–340.
[52] 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.
[53] 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.
[54] A. K. Lenstra and M. S. Manasse, *Factoring by electronic mail,* Proc. Eurocrypt '89, Lecture Notes in Computer Science **434**, Springer-Verlag, Berlin, 1990, 355–371.
[55] H. W. Lenstra, Jr., *Factoring integers with elliptic curves,* Annals of Mathematics (2) **126** (1987), 649–673.
[56] E. Lucas, *Théorie des fonctions numériques simplement periodiques,* Amer. J. Math. **1** (1878), 184–239 & 289–321.
[57] J. van de Lune and E. Wattel, *On the numerical solution of a differential-difference equation arising in analytic number theory,* Math. Comp. **23** (1969), 417–421.
[58] P. L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization,* Math. Comp. **48** (1987), 243–264.
[59] P. L. Montgomery, *An FFT extension of the elliptic curve method of factorization,* Ph. D. dissertation, Mathematics, University of California at Los Angeles, 1992.
[60] P. L. Montgomery, *A survey of modern integer factorization algorithms,* CWI Quarterly **7** (1994), 337–366.
[61] P. L. Montgomery, personal communication by e-mail, November 29, 1995.
[62] F. Morain, *Courbes elliptiques et tests de primalité,* Ph. D. thesis, Université Claude-Bernard-Lyon I, France, 1990. Available by ftp from `ftp.inria.fr:/INRIA/publication/Theses/TU-0144.tar.Z` .
[63] F. Morain, *Distributed primality proving and the primality of* $(2^{3539} + 1)/3$, Advances in Cryptology – Eurocrypt '90, Springer-Verlag, 1990, 110–123.

[64] M. A. Morrison and J. Brillhart, *A method of factorization and the factorization of F_7,* Math. Comp. **29** (1975), 183–205.
[65] M. Paterson and L. Stockmeyer, *On the number of nonscalar multiplications necessary to evaluate polynomials,* SIAM J. on Computing **2** (1973), 60–66.
[66] J. M. Pollard, *Theorems in factorization and primality testing,* Proc. Cambridge Philos. Soc. **76** (1974), 521–528.
[67] J. M. Pollard, *A Monte Carlo method for factorization,* BIT **15** (1975), 331–334.
[68] C. Pomerance, *The number field sieve,* Proceedings of Symposia in Applied Mathematics **48**, Amer. Math. Soc., Providence, Rhode Island, 1994, 465–480.
[69] C. Pomerance, J. W. Smith and R. Tuler, *A pipeline architecture for factoring large integers with the quadratic sieve algorithm,* SIAM J. on Computing **17** (1988), 387–403.
[70] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems,* Comm. ACM **21** (1978), 120–126.
[71] R. M. Robinson, *Mersenne and Fermat numbers,* Proc. Amer. Math. Soc. **5** (1954), 842–846.
[72] J. L. Selfridge, *Factors of Fermat numbers,* MTAC **7** (1953), 274–275.
[73] J. L. Selfridge and A. Hurwitz, *Fermat numbers and Mersenne numbers,* Math. Comp. **18** (1964), 146–148.
[74] D. Shanks, *Class number, a theory of factorization, and genera,* Proc. Symp. Pure Math. **20**, American Math. Soc., Providence, R. I., 1971, 415–440.
[75] L. A. Shepp and S. P. Lloyd, *Ordered cycle lengths in a random permutation,* Trans. Amer. Math. Soc. **121** (1966), 340–357.
[76] P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring,* Proc. 35th Annual Symposium on Foundations of Computer Science, IEEE Press, Los Alamitos, California, 1994, 124.
[77] J. H. Silverman, *The arithmetic of elliptic curves,* Graduate Texts in Mathematics **106**, Springer-Verlag, New York, 1986.
[78] R. D. Silverman, *The multiple polynomial quadratic sieve,* Math. Comp. **48** (1987), 329–339.
[79] R. D. Silverman and S. S. Wagstaff, Jr., *A practical analysis of the elliptic curve factoring algorithm,* Math. Comp. **61** (1993), 445–462.
[80] H. Suyama, *Informal preliminary report* (8), personal communication, October 1985.
[81] 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. [Errata: on the right side of equation (5) "$-\alpha_k$" should be "$-\alpha_{k-1}$" and "φ" should be "φ_1".]
[82] H. C. Williams, *How was F_6 factored?,* Math. Comp. **61** (1993), 463–474.
[83] H. C. Williams and J. S. Judd, *Some algorithms for prime testing using generalized Lehmer functions,* Math. Comp. **30** (1976), 867–886.
[84] S. Winograd, *Evaluating polynomials using rational auxiliary functions,* IBM Technical Disclosure Bulletin **13** (1970), 1133–1135.

Computer Sciences Laboratory, RSISE, Australian National University, Canberra, ACT 0200, Australia

E-mail address: rpb@cslab.anu.edu.au