MATHEMATICS OF COMPUTATION
VOLUME 61, NUMBER 203
JULY 1993, PAGES 445-462

A PRACTICAL ANALYSIS OF THE ELLIPTIC CURVE FACTORING ALGORITHM

ROBERT D. SILVERMAN AND SAMUEL S. WAGSTAFF, JR.

Dedicated to the memory of D. H. Lehmer

ABSTRACT. Much asymptotic analysis has been devoted to factoring algorithms. We present a practical analysis of the complexity of the elliptic curve algorithm, suggesting optimal parameter selection and run-time guidelines. The parameter selection is aided by a novel use of Bayesian statistical decision techniques as applied to random algorithms. We discuss how frequently the elliptic curve algorithm succeeds in practice and compare it with the quadratic sieve algorithm.

1. INTRODUCTION

The elliptic curve method (ECM) [8] and quadratic sieve (MPQS) [3] factoring algorithms are the most efficient, general-purpose, factoring algorithms known today. Furthermore, both ECM and MPQS have properties that make them ideal for parallel computation. The quadratic sieve is a deterministic algorithm whose run time is purely a function of the size of the number being factored. Its parallel implementation was discussed in [3]. The elliptic curve method is a random algorithm which finds small factors first. One implements it in parallel by performing independent random trials on multiple processors and quitting upon the first success.

In [9] Pomerance analyzed the asymptotic run time for MPQS, and Lenstra [7] used his results to analyze a single-step version of ECM. We summarize Lenstra's main result here. Let g be any positive real number, and let

$$L(p) = e^{\sqrt{\log p \log\log p}}, \qquad K(p) = e^{\sqrt{(2+o(1))\log p \log\log p}}.$$

Then, with probability at least $1-e^{-g}$, ECM will find a factor p of a larger integer N in time $gK(p)M(N)$, where $M(N)$ is the time to perform multiplication $\bmod N$ and $K(p)$ is the number of group operations per curve. Pomerance showed that the asymptotic run time for MPQS is $L(N)^{1+o(1)}$. Both of these asymptotic run times were heuristically derived, and depend on an unproved extension of a theorem of Pomerance, Canfield, and Erdös [9, 7].

Lenstra noted that one wants, in practice, to add a second step to the algorithm and that doing so can greatly shorten its run time. Montgomery [8] and

Received by the editor October 29, 1990; revised June 8, 1991.

1991 *Mathematics Subject Classification*. Primary 11A51, 11Y05; Secondary 68Q25, 14H52.

Key words and phrases. Elliptic curves, Dickman's function, smooth groups, quadratic sieve, factorization.

Brent [1] both give methods for implementing a second step of the algorithm. Brent further analyzed the run time and showed that a second step speeds the method by a factor of $\log p$, where p is the factor to be found, provided that one uses fast methods for polynomial evaluation.

ECM works by randomly selecting an elliptic curve E: $y^2 = Ax^3 + Bx + 1$, with nonzero discriminant and with coefficients in $\mathbb{Z}/N\mathbb{Z}$, and a random rational point P on the curve. If p is a prime dividing N, then the set of rational points on the curve over F_p, $p \neq 2, 3$, forms an Abelian group (known as the Mordell-Weil group) whose order lies in the interval $[p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$ [7]. One selects an integer B_1 and lets M be the product of all primes less than B_1 raised to some suitable power. One then computes MP by successive addition of points on the curve and hopes that MP will equal the identity of the group, $\bmod p$, but will not equal the identity of the group $\bmod N$. One then finds p by taking a GCD with N. The algorithm succeeds if the order of the group on the curve has all of its prime factors less than B_1.

The two-step version of the algorithm allows the order of the group to have all of its prime factors less than B_1, along with an additional, single prime factor between B_1 and some limit $B_2 > B_1$. For this reason, the study of the function which measures the probability that an integer x has all but one of its prime factors less than B_1 and a single factor between B_1 and B_2 is of primary importance to the analysis of the algorithm.

Lenstra gave, in his asymptotic analysis, the estimate $B(p) = L(p)^{1/\sqrt{2}+o(1)}$ for the optimal selection of B_1 for the one-step algorithm. Brent's analysis suggests that this should be lowered by $\log p$ for the two-step version. We define $\mathscr{K}(p)$ and $\mathscr{B}(p)$ as $K(p)/\log p$ and $B(p)/\log p$, respectively. Asymptotically, the factor of $\log p$ in the denominator is subsumed by the $o(1)$ in the exponents, but these definitions are useful for computational purposes. In this paper we discuss how the elliptic curve method and quadratic sieve perform for current real-world sized problems. ECM will find 10- to 15-digit factors quite quickly, 20- to 25-digit factors with an effort of perhaps ten hours, and 30- to 35-digit factors with considerable difficulty. Over the last several years the combined efforts of many researchers have resulted in hundreds of thousands of trials with ECM, and we have found exactly two 38-digit, one 37-digit, and one 36-digit factor. In this paper we discuss several related aspects of ECM. Among them are:

(1) How does one select its parameters?
(2) If ECM fails during one trial, how should its parameters be changed for the next?
(3) How does its run time vary with the size of factors found?
(4) How long should one use ECM before switching to the quadratic sieve?

In order to answer these questions, we require some background material regarding the distribution of the largest and smallest prime factors of an arbitrary positive integer.

2. Largest prime factor: Dickman's function

Dickman's function, $\rho(\alpha)$, is the probability that an integer $x \to \infty$ has its largest prime factor less than $x^{1/\alpha}$. It was extensively studied in [4]. We designate by $\mu(\alpha, \beta)$ the probability that x has its second largest prime factor less than $x^{1/\alpha}$ and its largest prime factor less than $x^{\beta/\alpha}$, $\alpha > \beta > 1$. Then the functional equations for ρ and μ are [6]

$$\rho(\alpha) = 1/\alpha \int_{\alpha-1}^{\alpha} \rho(t)\,dt \quad \text{and} \quad \mu(\alpha, \beta) = \frac{1}{\alpha-\beta}\int_{\alpha-\beta}^{\alpha-1} \rho(t)\,dt.$$

For the elliptic curve algorithm to succeed, the order of the Mordell-Weil group must be smooth up to B_1, with the exception of a single additional prime factor between B_1 and B_2. Designate $\mathscr{P}(B_1, B_2)$ as the probability of success with B_1 and B_2 as limits, where $B_2 > B_1$. Then

$$\mathscr{P}(B_1, B_2) \simeq \mu(\alpha, \beta) \quad \text{with } \alpha = \frac{\log p}{\log B_1}, \quad \beta = \frac{\log B_2}{\log B_1}.$$

One can numerically integrate $\rho(\alpha)$ by starting with the initial values $\rho(x) = 1$ for $0 \le x \le 1$. One can then set up an array of values for $\rho(\alpha)$ as α varies from (say) 3 to 1000, and compute their values by using Simpson's method for the integration. Tables for $\rho(\alpha)$ are also given in [6]. One can then use the tabulated values for ρ to numerically integrate $\mu(\alpha, \beta)$. The functions are well behaved and monotonic, and Simpson's method gives good accuracy. Our program agrees with published tables to the accuracy limits of those tables.

3. Smallest prime factor: Mertens's Theorem

We shall need estimates for the probability that a prime factor p, of a larger integer N, lies in a given interval. The probability that there is at least one divisor p, with $y < p < y^{1+\varepsilon}$ is 1 minus the probability that there is no divisor in the interval. On the assumption of independence, this is approximately

$$1 - \prod_{y<p<y^{1+\varepsilon}} \left(1 - \frac{1}{p}\right). \tag{3.1}$$

Formula (3.1) is not exact, owing to edge conditions at the interval boundaries. By Mertens's Theorem [5, p. 351] we have $\prod_{p<x}(1 - 1/p) \sim e^{-\gamma}/\log x$, so that (3.1) simplifies to

$$\Xi(N, y, \varepsilon) = \text{Prob}(\exists p \text{ with } y < p < y^{1+\varepsilon} \text{ and } p|N) \approx \frac{\varepsilon}{1+\varepsilon}. \tag{3.2}$$

This argument assumes that the probabilities for each prime in the interval are independent of one another. This assumption will be correct provided that N is big enough to have every prime p in the interval as a factor. Thus,

$$N \ge \prod_{y<p<y^{1+\varepsilon}} p,$$

and this is approximately $e^{y^{1+\varepsilon}}/e^{y}$ by the Prime Number Theorem.

In the case where N is not large enough to guarantee independence of probabilities, we shall need to use a standard inclusion/exclusion argument to correct for the lack of independence. Assume that $y = N^{\delta}$, and let $\mathscr{S}$ be the set of primes in the interval $(y, y^{1+\varepsilon})$. Then there can be up to $d = \lfloor 1/\delta \rfloor$ factors in $\mathscr{S}$. The probability is then

$$\Xi(N, y, \varepsilon) = \sum_{p\in\mathscr{S}}^{*} \frac{1}{p} - \frac{1}{2!}\sum_{p,q\in\mathscr{S}}^{*} \frac{1}{pq} + \frac{1}{3!}\sum_{p,q,r\in\mathscr{S}}^{*} \frac{1}{pqr} - \cdots. \tag{3.3}$$

The asterisks indicate that the reciprocal of the product under each summand is constrained to be less than or equal to N. The probability given in (3.3) can be written more generally as

$$\Xi(N, y, \varepsilon) = \sum_{j=1}^{d} \left(\frac{(-1)^{j+1}}{j!} \sum_{\substack{p_\iota \in \mathcal{S} \\ \prod_{\iota=1}^{j} p_\iota \le N}} \prod_{i=1}^{j} \frac{1}{p_i} \right), \tag{3.4}$$

where the constraint given on the inner sum guarantees that the product of primes in the inner product does not exceed N. This constraint will not be binding for $j \le \log N/((1+\varepsilon)\log y)$ because the product of j factors in this case cannot exceed N. If we truncate the sum at this point, by the Inclusion/Exclusion Inequality, the magnitude of the error is at most

$$\frac{1}{h!} \sum_{\substack{p_\iota \in \mathcal{S} \\ \prod_{\iota=1}^{h} p_\iota \le N}} \prod_{i=1}^{h} \frac{1}{p_\iota},$$

where h is the smallest integer exceeding $\log N/((1+\varepsilon)\log y)$. For the unconstrained cases we have

$$\sum_{p_\iota \in \mathcal{S}} \prod_{i=1}^{j} \frac{1}{p_i} = \left(\sum_{p_\iota \in \mathcal{S}} \frac{1}{p_i} \right)^j .$$

For the constrained cases we have

$$\sum_{\substack{p_\iota \in \mathcal{S} \\ \prod_{\iota=1}^{j} p_\iota \le N}} \prod_{i=1}^{j} \frac{1}{p_\iota} < \left(\sum_{p_\iota \in \mathcal{S}} \frac{1}{p_\iota} \right)^j ,$$

since clearly the constrained summand contains fewer cases than the unconstrained. A classical estimate also gives $\sum_{p<y} 1/p = \log\log y + \gamma_1 + o(1)$ [5, p. 351], where γ_1 is a computable constant. By using this estimate, and replacing the constrained summands by their upper bounds, we can obtain an approximation for the probability that there is a prime factor in $\mathcal{S}$ when N is not sufficiently large to guarantee independence.

We give an example of the calculations here. Suppose N is a 100-digit integer. The probability that N has a prime factor between 10^5 and 10^{10} can then be computed as follows. We clearly have $d = 19$, and that the product of any ten primes in the interval must be less than N. Thus, the first ten terms in our sum are unconstrained. The last nine are constrained, but their contribution is small, so replacing them with their upper bound will not incur much error. We have

$$\sum_{p=10^5}^{10^{10}} \frac{1}{p} = \log\log 10^{10} - \log\log 10^5 + o(1) \approx \log 2 .$$

Our estimate then becomes

$$\log 2 - \frac{1}{2!}\log^2 2 + \frac{1}{3!}\log^3 2 - \cdots + \frac{1}{19!}\log^{19} 2 \approx .500000000 .$$

Designate $S(N)$ as the smallest prime p that divides N. To estimate $\mathrm{Prob}(S(N) \in \mathscr{S})$, one can now simply compute $\Xi(N, y, \varepsilon)$ times 1 minus the probability that there is a factor less than y.

4. Optimal parameter selection

Parameters under our control are the step-1 and step-2 limits, and the number of curves. There are several ways to approach the problem of optimal parameter selection. One way is to maximize the probability of success divided by the work performed, allowing the latter to vary. This is perhaps the most natural approach because it maximizes the probability of success per unit work. Another is to fix the amount of work and simply maximize the probability of success. We look at the latter problem first.

Maximizing $\mathscr{P}(\alpha, \beta)$. On the assumption that step 2 runs K times as fast as step 1 (K will be implementation-dependent), the cost of running one curve is $B_1 + (B_2 - B_1)/K$. That is to say, if we can compute step 1 to B_1 in some fixed time, we can compute step 2 to KB_1 in the same amount of time. If we use L curves, the total run time is $T = L(B_1 + (B_2 - B_1)/K)$. If B_1 is small with respect by B_2, or K is large (as is usually the case), we can approximate this to $T \approx L(B_1 + B_2/K)$. We would then like to maximize the probability of success with L curves. This probability is

$$1 - [1 - \mathscr{P}(B_1, B_2)]^L. \tag{4.1}$$

Maximizing this with T fixed is equivalent to minimizing

$$[1 - \mathscr{P}(B_1, B_2)]^{(T/(B_1+B_2/K))}.$$

This in turn is equivalent to minimizing

$$Q(B_1, B_2) = \frac{T}{(B_1 + B_2/K)} \log(1 - \mathscr{P}(B_1, B_2)). \tag{4.2}$$

The Kuhn-Tucker conditions yield

$$\begin{aligned} \frac{T(\log(1 - \mathscr{P}(B_1, B_2))}{(B_1 + B_2/K)^2} &= \frac{-T\frac{\partial\mathscr{P}}{\partial B_1}}{(1 - \mathscr{P}(B_1, B_2))(B_1 + B_2/K)}, \\ \frac{T\log(1 - \mathscr{P}(B_1, B_2))}{K(B_1 + B_2/K)^2} &= \frac{-T\frac{\partial\mathscr{P}}{\partial B_2}}{(1 - \mathscr{P}(B_1, B_2))(B_1 + B_2/K)}. \end{aligned} \tag{4.3}$$

Solving for $\frac{\partial\mathscr{P}}{\partial B_1}$ and $\frac{\partial\mathscr{P}}{\partial B_2}$ gives, at the stationary points,

$$\begin{aligned} \frac{\partial\mathscr{P}}{\partial B_1} &= \frac{(\mathscr{P}(B_1, B_2) - 1)\log(1 - \mathscr{P}(B_1, B_2))}{(B_1 + B_2/K)}, \\ \frac{\partial\mathscr{P}}{\partial B_2} &= \frac{(\mathscr{P}(B_1, B_2) - 1)\log(1 - \mathscr{P}(B_1, B_2))}{K(B_1 + B_2/K)}. \end{aligned} \tag{4.4}$$

Whence

$$\frac{\partial\mathscr{P}}{\partial B_1} = K\frac{\partial\mathscr{P}}{\partial B_2}. \tag{4.5}$$

We remark that without the simplification $B_2 \approx B_2 - B_1$, K would become $K^2/(K-1)$ in (4.5). That (4.5) represents a minimum is easy to see because the function is convex. The result implies that if we have selected B_1 and B_2 optimally, then if we change B_1 by ΔB_1, then B_2 should be changed by $K\Delta B_1$. The partial derivatives in (4.4) can be directly determined from $\mathscr{P}(B_1, B_2)$, yielding

$$(4.6)\quad \begin{aligned} \frac{\partial \mathscr{P}}{\partial B_1} &= \frac{\mathscr{P}(B_1, B_2)}{B_1 \log(p/B_2)} + \frac{\rho(\alpha-\beta)}{B_1 \log B_1} - \frac{\rho(\alpha-1)\log p}{B_1 \log B_1 \log(p/B_2)}, \\ \frac{\partial \mathscr{P}}{\partial B_2} &= \frac{\log B_1 \mathscr{P}(B_1, B_2)}{B_2 \log^2(p/B_2)} + \frac{\rho(\alpha-\beta)}{B_2 \log(p/B_2)}. \end{aligned}$$

For simplicity of notation we set $\mathscr{V} = 1 - \mathscr{P}(B_1, B_2)$ and $\mathscr{W} = \log(p/B_2)$. Then, combining (4.6) with the optimality conditions given by (4.4) yields

$$(4.7)\quad \begin{aligned} \frac{(-\mathscr{V})\log(\mathscr{V})}{(B_1 + B_2/K)} &= \frac{\mathscr{P}(B_1, B_2)}{B_1 \mathscr{W}} + \frac{\rho(\alpha-\beta)}{B_1 \log B_1} - \frac{\rho(\alpha-1)\log p}{B_1 \log B_1 \mathscr{W}}, \\ \frac{(-\mathscr{V})\log(\mathscr{V})}{K(B_1 + B_2/K)} &= \frac{\log B_1 \mathscr{P}(B_1, B_2)}{B_2 \mathscr{W}^2} + \frac{\rho(\alpha-\beta)}{B_2 \mathscr{W}}. \end{aligned}$$

Solving (4.7) would yield expressions for the best values for B_1 and B_2, but (4.7) seems analytically intractable. To determine the optimal B_1 and B_2, we are therefore forced to resort to numerical evaluation of $\mathscr{P}(B_1, B_2)$.

To determine the optimal relationship between B_1 and B_2, the following procedure seems reasonable. Fix the size of the prime factor for which we are looking. Fix T, and allow B_1 and L to vary, adjusting B_2 accordingly. Compute $\mathscr{P}(B_1, B_2)$ by direct integration of Dickman's function and select the set of values for L, B_1, and B_2 that maximizes the probability of success. Table 1 shows the results of this computation for $p = 10^{15}$ and $K = 100$. The highest probability of success occurs where $L = 5$ and $B_2 \approx 41B_1$. Table 1 was recomputed for values of p ranging from 10^5 to 10^{40} and values of K varying from 10 to 500. Each time, the optimal value of B_2 was approximately $0.4KB_1$, provided that T was big enough to yield a nonnegligible chance of success. Space prohibits displaying all of the relevant data here. However, the estimate $B_2 = 0.4KB_1$ is a good general rule to use because the objective function is very flat in the neighborhood of the optimum. It should give results that are within a few percent of optimum.

The optimal value for B_1 could, in theory, be derived from $B(p)$, however we would need accurate estimates for its $o(1)$ term as a function of p, and we do not have such an expression.

Table 2 shows what happens if T is not big enough to give a nonnegligible chance of success. As p increases, $\mathscr{B}(p)$, and hence T, must increase also, and this cannot occur when T is constrained. This table displays, for $T = 20,000$, the optimal computed values for B_1 and B_2 as p is varied. The probability of success is more dependent on B_1 than B_2, and since T is constrained, we cannot maintain $B_2 = 0.4KB_1$ while B_1 is increasing, without reducing L. Once L reaches 1, however, it cannot be reduced further.

An intuitive explanation for this result is as follows. The success of the algorithm requires that a random number near p will be smooth up to B_1 and have a single prime factor between B_1 and B_2. The probability that this number is smooth up to B_1 clearly depends on the size of p, but the probability that it has a single prime factor between B_1 and B_2 will be independent of p if p is sufficiently large with respect to B_2.

TABLE 1. Optimal B_2/B_1 relationship; $p = 10^{15}$; $T = 20{,}000$; $K = 100$

Curves	B_1	B_2	$\mathcal{P}(B_1, B_2)$	B_2/B_1
1	12539	758620	.0635548	60.5
2	6535	352941	.0768417	54.0
3	4474	223713	.0816556	50.0
4	3546	148936	.0829785	42.0
5	2857	117142	.0829913	41.0
6	2491	100071	.0813971	40.2
7	2063	81485	.0796019	39.5
8	1837	71942	.0776121	39.1
9	1628	61050	.0752463	37.5
10	1498	54945	.0729526	36.7
11	1356	47489	.0706545	35.0
12	1277	43990	.0682703	34.4
13	1189	40183	.0660136	33.8
14	1094	35931	.0637429	32.8
15	1010	32333	.0615120	32.0
16	963	30203	.0593397	31.3
17	904	28054	.0573140	31.0
18	854	26395	.0552602	30.9
19	813	24979	.0533285	30.7
20	777	23646	.0515252	30.4

TABLE 2. Optimal B_2/B_1 relationship for fixed T $(K = 100)$

Digits	L	B_1	B_2	$\mathcal{P}(B_1, B_2)$	B_2/B_1
10	32	429	19974	.915860	46.6
11	22	631	28443	.698543	45.1
12	16	880	37871	.449772	43.0
13	10	1413	60082	.265791	42.5
14	8	1778	74006	.147927	41.6
15	4	3568	146688	.082823	41.1
16	4	3578	145745	.045130	40.7
17	3	4809	190510	.024959	39.6
18	2	7271	280162	.014252	38.5
19	1	14565	558007	.008191	38.3
20	1	14601	554427	.004646	38.0
21	1	14632	551360	.002598	37.7
22	1	14792	535502	.001428	36.2
23	1	15259	489356	.000768	32.1
24	1	15469	468559	.000411	30.3
25	1	15601	455514	.000215	29.2

Remark. We note, as a practical matter, that if one wants to perform ECM with just one curve, then one should use the $P - 1$ algorithm instead, since it is significantly faster.

Maximizing $\mathcal{P}(\alpha, \beta)/T$. We now examine the more interesting case of trying to maximize the probability of success per unit work. Table 3 shows the results. For each entry in the table a mesh was created by varying B_1 and B_2. We then looked for the point that maximized the probability of success divided by the cost per curve. The probability of success was again computed using direct integration of Dickman's function $\mu(\alpha, \beta)$. This table assumes that step 2 is 100 times as fast as step 1. Montgomery's program, the most efficient ECM program known to us, is even better. It achieves $K = 170$ on some machines. With $K = 100$ the cost per curve is $B_1 + (B_2 - B_1)/100$. The following columns are included in the table:

(1) D is the number of decimal digits in the factor for which we are looking.
(2) B_1 and B_2 are the optimal step-1 and step-2 limits.
(3) L is the expected number of curves to find a factor of this size. It is clearly $1/\mathcal{P}(B_1, B_2)$.
(4) *Inc* is the incremental cost ratio, that is, the cost for this row divided by the cost for finding a factor that is 1 digit smaller.

The minor fluctuations in the B_2/B_1 column (and others) are due to the fact that the response surface is extremely flat in the neighborhood of the global optimum. Furthermore, Dickman's ρ-function, needed to compute $\mu(\alpha, \beta)$, was precalculated only at fixed points and not interpolated for the second integration. Changes of ±10% in B_1 can result in less than a 1% change in the global response surface, even though the probability of success for a single curve can change significantly. Furthermore, the actual B_1- and B_2-parameters are prime numbers, and the entries in Table 3 are not taken as primes. For example, for $p = 10^4$, B_1 is given as 16 and B_2 as 800. These should actually be rounded to the nearest primes, yielding $B_1 = 17$, and $B_2 = 797$, changing the B_2/B_1 ratio substantially. Table 3 is very approximate for small p.

TABLE 3. Optimal ECM parameter selection ($K = 100$)

D	B_1	B_2	$\mathcal{P}(B_1, B_2)$	Average L	B_2/B_1	T	*Inc*
5	16	800	.456	2.19	50	52.6	
7	53	2650	.226	4.41	50	350.5	2.48
9	156	7176	.122	8.16	46	1858	2.25
11	405	19440	7.10e-2	14.1	48	8441	2.09
13	962	42328	4.07e-2	24.6	44	34034	1.98
15	2240	103017	2.60e-2	38.4	46	125599	1.90
17	4778	215010	1.61e-2	62.2	45	430918	1.84
19	9004	405180	9.39e-3	106	45	1.389e+6	1.78
21	18437	792791	6.20e-3	161	43	4.250e+6	1.74
23	34155	1.40e+6	3.86e-3	259	41	1.245e+7	1.70
25	66596	2.66e+6	2.66e-3	376	40	3.506e+7	1.66
27	133297	5.33e+6	1.95e-3	512	40	9.546e+7	1.64
29	247988	9.99e+6	1.38e-3	725	40	2.519e+8	1.62
31	374990	1.54e+7	8.20e-4	1219	41	6.447e+8	1.59
33	649996	2.66e+7	5.59e-4	1757	41	1.611e+9	1.57
35	1170924	4.80e+7	4.19e-4	2382	41	3.933e+9	1.56
37	1967442	8.13e+7	2.97e-4	3366	41	9.404e+9	1.54
39	3276490	1.31e+8	2.05e-4	4878	40	2.237e+10	1.52
41	5249667	2.09e+8	1.44e-4	6897	40	5.068e+10	1.51

Remark. It is possible to select the coefficients of an elliptic curve so that the group order is a priori divisible by 12, or alternatively 8, one-half the time and 16 one-half the time. This selection is assumed in Table 3. Clearly, if one does not make this selection, then one loses approximately one extra digit and should select parameters from Table 3 accordingly.

Lenstra's results stated in the introduction imply that the probability of success divided by the cost is

$$\frac{1-e^{-g}}{gK(p)M(N)}.$$

This is clearly maximized at $g = 1$ for any fixed N and p, so the optimal probability of success is $1 - 1/e$. Indeed, from Table 3, the global probability of success with multiple curves is

$$1-(1-\mathscr{P}(B_1, B_2))^{(1/\mathscr{P}(B_1, B_2))},$$

and this clearly approaches $1-1/e$ as $D \to \infty$ because $\mathscr{P}(B_1, B_2) \to 0$ at the appropriate rate. Furthermore, if one computes $1-(1-\mathscr{P}(B_1, B_2))^L$ from Table 3, one obtains very good approximations to $1-1/e$.

The $o(1)$ terms in $\mathscr{K}(p)$ and $\mathscr{B}(p)$ represent a class of functions of p that go to zero as p goes to infinity. From Table 3, we can estimate the $o(1)$ functions for $\mathscr{K}(p)$ and $\mathscr{B}(p)$. For each value of p we may find the corresponding value for $o(1)$ by setting $\mathscr{K}(p)$ and $\mathscr{B}(p)$ equal to the values in the T and B_1 columns, respectively, and then solving for $o(1)$. Thus, we obtain for $\mathscr{K}(p)$:

$$o(1) = \frac{[\log(T\log p)]^2}{\log p \log\log p} - 2,$$

and for $\mathscr{B}(p)$ we obtain:

$$o(1) = \frac{\log(B_1 \log p)}{\sqrt{\log p \log\log p}} - \frac{1}{\sqrt{2}}.$$

Table 4 (next page) gives these computed values, while Figure 1 plots them. Since these are $o(1)$ estimates, they must drop to zero as $p \to \infty$, and the data suggests that they might.

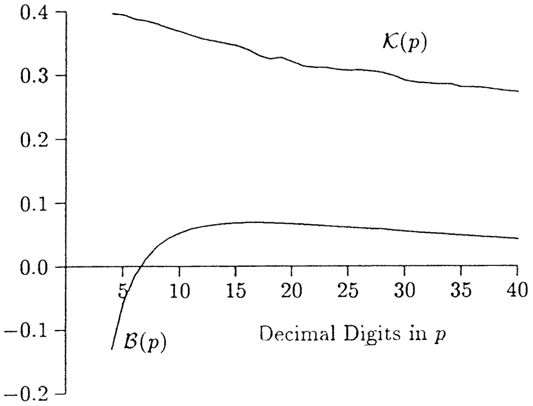

FIGURE 1. $o(1)$ estimates for $\mathscr{K}(p)$ and $\mathscr{B}(p)$

TABLE 4. $o(1)$ estimates for $\mathscr{K}(p)$ and $\mathscr{B}(p)$

$\log_{10} p$	$\mathcal{B}(p)$	$\mathcal{K}(p)$	$\log_{10} p$	$\mathcal{B}(p)$	$\mathcal{K}(p)$
4	-.130386	.396991	23	.063736	.311264
5	-.057283	.394847	24	.062588	.308229
6	-.015251	.388040	25	.061436	.306850
7	.011771	.385323	26	.060284	.307257
8	.031230	.379910	27	.059054	.305791
9	.044596	.373503	28	.058359	.303252
10	.053258	.368445	29	.056557	.298345
11	.059085	.362141	30	.055008	.290877
12	.062846	.356763	31	.053742	.287734
13	.065517	.353562	32	.052380	.286437
14	.067397	.350188	33	.051164	.284786
15	.068545	.347020	34	.049909	.285186
16	.068941	.340724	35	.048716	.280016
17	.069027	.330794	36	.047530	.280075
18	.068465	.325531	37	.046313	.278940
19	.067677	.327952	38	.045135	.276070
20	.066807	.320870	39	.043981	.274095
21	.066059	.313776	40	.042839	.272629
22	.064925	.311541			

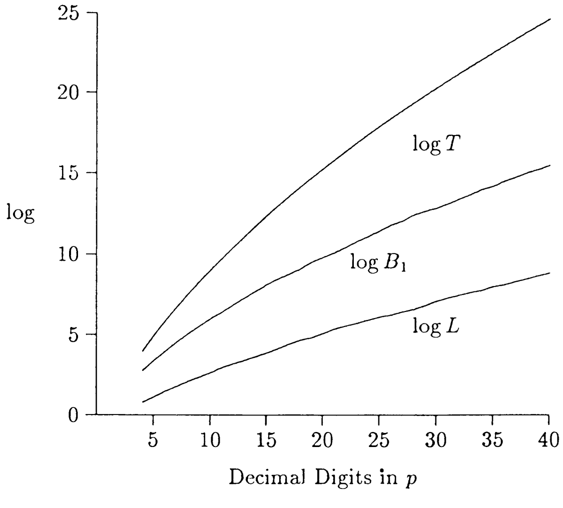

FIGURE 2. Logarithms of optimal L, B_1, and T

Figure 2 plots the logarithms of L, B_1, and T from Table 3.

Figure 3 plots the incremental cost ratio to find a factor of one more digit. It is easy to see that the incremental cost ratio is dropping slowly to 1 as p increases. Indeed,

$$\lim_{p\to\infty} \frac{\mathscr{K}(10p)}{\mathscr{K}(p)} = 1 .$$

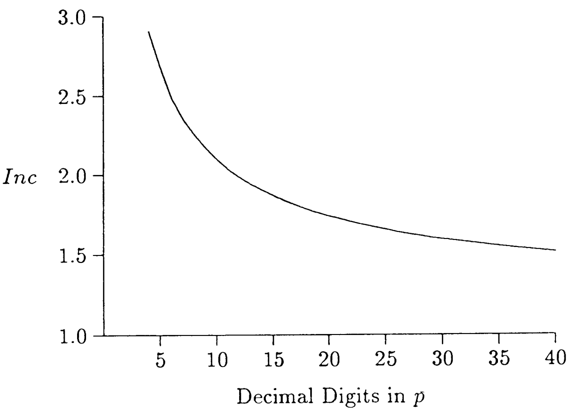

FIGURE 3. Incremental cost of adding one digit

The function behaves approximately like $O(p^{1/5})$, hence ECM behaves like an $O(p^{1/5})$ method for p between 5 and 40 digits.

5. CHANGING ECM PARAMETERS AFTER FAILURE

We now discuss the problem of how to select B_1 and B_2 when we do not know p. We assume that N is an odd, random composite integer. In practice, one always starts factoring N by trial division up to some limit $y_0 = \log^2 N$ or perhaps $y_0 = N^\varepsilon$ for small ε. To find the initial starting point for ECM, we need to compute the expected size of the smallest factor of N, given that it is greater than y_0, and then use that value. This can be readily computed, using the methods given in §3. The probability that the smallest prime factor is in an interval $(y, y^{1+\varepsilon})$ is the probability that there is a factor in that interval times the probability that there is no factor in any smaller interval. Table 5 (next page) presents the expected size of the smallest prime factor of N, for various N, given that trial division to $\log^2 N$ failed to find a factor. The third column shows the expected value of the size of $S(N)$, simply given that $S(N) > 10$. This table was computed by partitioning the interval $[\log y_0, \log\sqrt{N}]$ into 1000 pieces of equal length $\delta = (\log\sqrt{N} - \log y_0)/1000$. We also estimate the average size of a factor p in the interval $(10^d, 10^{d+\delta})$ to be $d + 1 + \delta/2$. Our estimate for the size of $E(S(N))$ now becomes

$$E(S(N)) \approx \sum_{d=\log y_0}^{\log\sqrt{N}} (d + 1 + \delta/2)D(d) \prod_{j=\log\log^2 N}^{d-\delta} (1 - D(j)),$$

where the product and sum are taken over increments of δ, and $D(d)$ is the probability that there is a factor in the interval.

Now, supposing that ECM fails, we can use the information obtained from the failure, along with Bayes's theorem, to re-estimate the expected size of the still unknown factor. Assuming that we run ECM and it fails, the failure will give us a sample distribution. A prior can be derived from (3.2). The density function for the prior is obtained by differentiating the distribution function $\text{Prob}(\exists p$ with $y_0 < p < 10^k$ and $p|N)$. This probability is approximately

$$\Xi(N, 3, k/\log(y_0) - 1) \approx (k - \log(y_0))/k.$$

TABLE 5. Expected size of $S(N)$ (in digits)

$\log_{10} N$	$\gamma_0 = \log^2 N$	$\gamma_0 = 10$
20	3.65	2.28
30	5.15	2.67
40	6.36	2.93
50	7.39	3.14
60	8.28	3.30
70	9.07	3.43
80	9.79	3.54
90	10.44	3.64
100	11.04	3.72
125	12.35	3.88
150	13.48	3.99
175	14.46	4.08
200	15.34	4.15

Differentiating this with respect to k yields $\log(\gamma_0)/k^2$ as our initial prior. Once we have run ECM and had it fail, we will use the sample distribution derived from the failure as the prior for the next trial. Alternatively, one can always use a uniform prior. In the parlance of Bayesian statistics, "the sample swamps the prior." Whether one uses a uniform prior, or the posterior from the last failure, makes little difference in practice.

Bayes's theorem states that the posterior density function is the product of the prior and sample density functions, times an appropriate integration constant. Thus, if $g(p)$ is the prior, and $h(p)$ is the sample, then $T(p)$ is the posterior, where

$$T(p) = kg(p)h(p) \quad \text{and} \quad k = \frac{1}{\int g(x)h(x)\,dx}.$$

An example will make this clear. Suppose (from Table 3) we chose $B_1 = 3350$, $B_2 = 151000$ and that we were looking for a 15-digit factor. We shall also assume that there are no factors less then (say) nine digits based upon trial division. With the given parameters, the probability of failing to find a factor of D digits is given in column 2 of Table 6. This gives us a sample distribution function. To get its density function, we compute its derivative numerically. Those values are given in the third column of Table 6. We assume a uniform prior for this calculation, so the expected value of the posterior will just be the expected value of the sample. The expected value of the posterior is 15.84. We then use this value to select the new ECM parameters. The actual calculations were carried out to more precision than is displayed in the table.

The same calculation was repeated for factors of 4 to 40 digits and the results are displayed in Table 7. This table gives, based upon a uniform prior, the new estimated size of the factor, given that the current search has failed and that the current search used the parameters from Table 3.

Remark. We note that the technique just described, for parameter reselection, can be applied in other instances as well. It is not unique to ECM. One can use the method in any random algorithm where the probability of success varies in a predictable way upon the value of the input parameters.

TABLE 6. Bayes theorem calculation

D	Prob	density
9	3.100e-10	1.315e-7
10	2.629e-7	4.039e-5
11	8.078e-5	1.891e-3
12	3.782e-3	1.953e-2
13	3.914e-2	7.789e-1
14	.1595	.1624
15	.3639	.2132
16	.5860	.1969
17	.7578	.1417
18	.8694	8.767e-2
19	.9331	4.906e-2
20	.9675	2.572e-2
21	.9846	1.266e-2
22	.9929	6.070e-3
23	.9968	2.837e-3
24	.9985	1.299e-3
25	.9994	5.731e-4
26	.9997	2.502e-4
27	.9998	1.075e-4

TABLE 7. Factor size re-estimation after failure

D	New D	D	New D
4	4.575	23	24.012
5	5.604	24	25.031
6	6.606	25	26.075
7	7.632	26	27.085
8	8.654	27	28.108
9	9.685	28	29.135
10	10.712	29	30.138
11	11.725	30	31.145
12	12.760	31	32.152
13	13.789	32	33.173
14	14.818	33	34.157
15	15.847	34	35.213
16	16.869	35	36.223
17	17.884	36	37.242
18	18.896	37	38.262
19	19.932	38	39.315
20	20.950	39	40.287
21	21.964	40	41.302
22	22.992		

6. Actual ECM run times

This section presents the actual running times of Montgomery's program on a SUN-3/60. On such a machine, step 2 of his program runs approximately 175 times as fast as step 1. That is to say, in an amount of time it takes to execute step 1 to B_1, step 2 can be executed to $175B_1$. Table 8 presents the amount of time it takes to execute Montgomery's program to various step-1 limits for three different-sized N, 50 digits, 100 digits, and 200 digits. Theoretically, it should take time proportional to B_1 to execute step 1 up to B_1, but Montgomery's program has some internal tricks that make it faster than linear for small B_1.

Table 8. Actual ECM run times in seconds (step 1)

B_1	50D	100D	200D
100	3	11	44
500	7	29	115
1000	12	46	184
5000	43	172	690
10000	81	323	1293
50000	247	989	3955
100000	439	1754	7027
200000	810	3241	12960
500000	1835	7338	29353

From the times given in Table 8, we can use the parameters given by Table 3 to estimate how long it will take to find a factor of a given size with probability $1 - 1/e$, for several different-sized N. The data is given in Table 9. Times for other values of N can be interpolated by noting that the multiplication algorithm used is an $O(\log^2 N)$ algorithm, and that multiplying the size of N by k changes the run time by a factor of k^2. Times for other p can be extrapolated by using the *Inc* column from Table 3.

Table 9. Probable time to find p (in seconds)

$\log_{10} p$	100-Digit N	200-Digit N
4	4.8	19.1
6	31.8	127
8	168	672
10	503	2012
12	1521	6084
14	4650	18600
16	14500	58000
18	45000	180000
20	110000	440000
22	270000	1.08e+6
24	645000	2.58e+6
26	1.60e+6	6.40e+6
28	4.10e+6	1.64e+7
30	1.04e+7	4.16e+7

7. Comparison with the quadratic sieve

ECM is a random algorithm, while MPQS [11] works like a deterministic algorithm in practice. ECM succeeds more quickly when N has a small prime factor, but since the factors are unknown, one would like to place an upper bound on the amount of time to spend using ECM, before switching to MPQS. We would also like to determine how long to run ECM in order to minimize the expected time to find a factor of N, using a combination of both methods. Throughout this section, we shall assume that trial division to $\log^2 N$ has already been performed.

Let $\nu(T, p)$ be the probability that we succeed in factoring N with ECM in time T, where p is the smallest prime factor of N. We note that T will depend on p. Given that we spend time T with ECM and that MPQS will take time x, the expected total time is

$$R(T, p, x) = \nu(T, p)T + (1 - \nu(T, p))(T + x). \tag{7.1}$$

Since we can always factor N in time x with MPQS by itself, we want $R(T, p, x) < x$ for all T. This yields an upper bound on the amount of time to spend with ECM:

$$T < \nu(T, p)x. \tag{7.2}$$

If we take $x = L(N)$, $\nu(T, p) = 1 - e^{-g}$, and $T = g\mathscr{K}(p)M(N)$, inequality (7.2) then becomes

$$\frac{g}{1 - e^{-g}} < \frac{L(N)}{\mathscr{K}(p)M(N)}. \tag{7.3}$$

We have good estimates for the values of the $o(1)$ term for $L(N)$ based upon the CPU-times presented in [3] and can therefore compute $L(N)$ and $\mathscr{K}(p)$ accurately. Alternatively, one can use values for $L(N)$ and $\mathscr{K}(p)$ taken from actual known run times. One can therefore solve (7.3) for g numerically if one assumes some value for p. The difficulty is that p is unknown. One should therefore solve (7.3) for many different values of p and then average over the different values of g, weighted by the probability that N has a factor near p. This analysis is, of course, predicated on N having only two prime factors. When it has more, the analysis becomes a great deal more complicated. The MPQS algorithm will work in the same amount of time regardless of how many factors are present. However, whenever ECM finds a factor, one presumably divides out that factor before continuing. This in turn affects $M(N)$ for the next trial and makes the analysis a great deal more difficult. One must also estimate the probability that the cofactor is prime.

We would also like to minimize the expected run time. A straightforward minimization of (7.1) with respect to g yields

$$g = \begin{cases} \log\left(\dfrac{L(N)}{\mathscr{K}(p)M(N)}\right) & \text{if } L(N) > \mathscr{K}(p)M(N), \\ 0 & \text{otherwise.} \end{cases} \tag{7.4}$$

In computing g, the values for $\mathscr{K}(p)M(N)$ can be taken from Table 9. Table 9 displays the actual timing estimates in the case where $g = 1$, that is, the probability of success is $1 - 1/e$.

Given an arbitrary N, ECM works best by selecting B_1 and B_2 small, running trials, then increasing B_1 and B_2 and repeating if there is no success. We estimate here the time to run ECM before switching to MPQS, to minimize (7.1), assuming that N will be small enough to factor with MPQS:

(1) Trial divide to $p \approx \log^2 N$.
(2) Select $\log p = E(S(N|p > \log^2 N))$ from Table 5.
(3) For that value of p compute g from (7.4). If $g = 0$, then quit. Select B_1 and B_2 from Table 3 and run ECM for time $g\mathscr{K}(p)M(N)$.
(4) If ECM fails, increase p and go to step (3).
(5) If ECM succeeds and the cofactor is composite, re-estimate $S(N)$ and go to step (3).
(6) The total time spent is the sum of the times spent in (3).

Remark. This procedure is different from one we would use if we were employing only ECM. In the case described above, g is varied according to (7.4). If we were using ECM only, we would always take $g = 1$, since that is optimal.

We only execute step (3) whenever earlier attempts have failed. The total expected time spent in step (3) is therefore

$$\sum_p g\mathscr{K}(p)M(N)[\text{probability that earlier trials failed}].$$

This sum runs over increasing p until g is negative. At that point it becomes faster to use MPQS. We note that even though B_1 and B_2 are chosen for a particular p, there is also a smaller chance that it will find a larger p, and a larger chance that it will find a smaller p. The probability that ECM succeeds on a given trial is approximately

$$\sum_d \frac{1}{d}\eta(d),$$

where $1/d$ is the probability that there is a $(d+1)$-digit factor, and $\eta(d)$ is the probability that ECM finds a factor of d digits with the value of g given by (7.4). This latter probability can be found by direct integration of Dickman's function as discussed in §§2 and 4. The probability of failure of a given trial is trivially just one minus the above sum.

Remark. This analysis has assumed that our goal is simply to find a factor of N, not to factor N completely. The latter case is extremely difficult to analyze because when ECM does find a factor one must estimate the probability that the cofactor is prime. This is difficult to estimate because the cofactor being considered is, in some sense, no longer a "random" integer. It does not have the same expected smoothness properties of some other arbitrary integer of about the same size.

Table 10 presents the amount of time one should run ECM before switching to MPQS. The values give the fraction of time one should spend with ECM, relative to the time it will actually take with MPQS. This table assumes that one is factoring a "random" integer whose factors up to $\log^2 N$ have been removed

TABLE 10. Time to run ECM before changing to MPQS

$\log_{10} N$	ECM Time
40	.72
50	.43
60	.16
70	.05
80	.0080
90	.0029
100	.0016
110	.00071

by trial division and that one switches to QS upon finding the first factor. The values given in Table 10 are approximate, and are rather sensitive to the starting value for $E(S(N))$. If one assumes that the smallest factor of N is larger than (say) $N^{0.2}$, then one clearly would spend more time with ECM before switching. The exact, optimal length of time can be computed by the procedure given by (1)–(6) above. We also remark that the values given are sensitive to the quality of implementation of the two methods.

ACKNOWLEDGMENTS

The authors would like to thank Claudia Spiro for helpful remarks regarding the estimates in §3 and Peter Montgomery for allowing us to use his ECM program.

BIBLIOGRAPHY

1. Richard P. Brent, *Some integer factorization algorithms using elliptic curves*, Research Report CMA-R32-85, The Centre for Mathematical Analysis, The Australian National University, 1985.

2. 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., Contemp. Math., vol. 22, Amer. Math. Soc., Providence, RI, 1983.

3. Tom R. Caron and Robert D. Silverman, *Parallel implementation of the quadratic sieve*, J. Supercomputing **1** (1988), 273–290.

4. N. G. DeBruijn, *On the number of uncancelled elements in the sieve of Eratosthenes*, Indag. Math. **12** (1950), 247–256.

5. G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*, 5th ed., Oxford Univ. Press, Oxford, 1979.

6. Donald E. Knuth and Luis Trabb Pardo, *Analysis of a simple factorization algorithm*, Theor. Comp. Sci. **3** (1976), 321–348.

→ H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. (2) **126** (1987), 649–673.

8. Peter L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Math. Comp. **48** (1987), 243–264.

9. Carl Pomerance, *Analysis and comparison of some integer factoring algorithms*, Math. Centrum Tracts (H. W. Lenstra, Jr. and R. Tijdeman, eds.), 1984, pp. 89–140.

10. Howard Raiffa and Robert Schlaifer, *Applied statistical decision theory*, MIT Press, Cambridge, MA, 1961.

11. Robert D. Silverman, *The multiple polynomial quadratic sieve*, Math. Comp. **48** (1987), 329–340.

THE MITRE CORPORATION, BURLINGTON ROAD, BEDFORD, MASSACHUSETTS 01730
E-mail address: bs@linus.mitre.org

DEPARTMENT OF COMPUTER SCIENCES, PURDUE UNIVERSITY, WEST LAFAYETTE, INDIANA 47907
E-mail address: ssw@cs.purdue.edu