References

[1] Garey, M.R., and Johnson, D.S., *Computers and Intractability*, Freeman and Company, San Francisco, 1979.

[2] Horowitz, E., and Sahni, S., Computing partitions with applications to the knapsack problem, *J. Ass. Comput. Mach.*, 21, 2 (1974), 277–292.

[3] Karnin, E.D., A parallel algorithm for the knapsack problem, *IEEE Trans. on Comput.*, C-33, 5 (1984), 404–408.

[4] Schroeppel, R., and Shamir, A., A $TS^2 = O(2^n)$ time/space trade-off for certain NP-complete problems, in *Proc. 20th IEEE Symp. on Foundations of Comput. Sci.*, 1979, 328–336.

[5] Stanley, R.P., Theory and application of plane partitions, Parts 1 and 2, *Studies in Applied Math.*, 50 (1971), 167–188, 259–279.

[6] Tarjan, R.E., and Trojanowski, A.E., Finding a maximum independent set, *SIAM J. on Comput.*, 6, 3 (1977), 537–546.

Fast, Rigorous Factorization and Discrete Logarithm Algorithms

Carl Pomerance*
Department of Mathematics
The University of Georgia
Athens, Georgia 30602, USA

§1. Introduction.

The last decade has seen an exponential increase in activity concerning the related algorithmic problems of factoring integers and computing discrete logarithms in finite fields. Since an "answer" to an instance of one of these problems can be very easily verified, it is possible to profitably employ an efficient algorithm that has not been rigorously analyzed. In fact every practical factorization or discrete logarithm algorithm that purports to be sub-exponential in its worst case has only heuristic analyses.

In factoring, the fastest known algorithms all share a heuristic worst case running time of $L(N)^{1+o(1)}$ to factor N, where

$$L(N) = \exp(\sqrt{\log N \operatorname{loglog} N})$$

and log denotes the natural logarithm; see Coppersmith, Odlyzko, Schroeppel [7], Lenstra [12], Pomerance [16], and Schnorr, Lenstra [17]. The fastest rigorously proved factoring algorithm has expected running time $L(N)^{\sqrt{5/2} + o(1)}$, see Pomerance [16].

*Supported in part by an NSF grant.

For discrete logarithms in GF(q) (the finite field with q elements), the fastest known algorithms to compute a discrete logarithm from scratch have heuristic running time $L(q)^{1+o(1)}$ if q is prime and heuristic running time $\exp\{O((\log q)^{1/3}(\log\log q)^{2/3})\}$ if q is a power of 2; see Coppersmith [6], Coppersmith, Odlyzko, Schroeppel [7], and Odlyzko [15]. For the case of q prime, Adleman [1] has sketched a rigorous argument that discrete logarithms may be computed in time $L(q)^c$ for some constant c . Although this exponent c was not computed, using the tools cited in Adleman's paper, an upper bound of $2\sqrt{2} + o(1)$ may be inferred. Using the tools in Pomerance [16], a value of $c = \sqrt{5/2} + o(1)$ may be obtained. For GF(q) with q a power of 2, Hellman, Reyneri [21] have given a rigorous treatment similar to Adleman's obtaining an upper bound of $L(q)^{\sqrt{12} + o(1)}$. (Thanks are due to K. McCurley for this reference.)

The case when $q = p^n$ is a non-trivial power of an odd prime has been less well studied. For p relatively small, the techniques for $GF(2^n)$ carry over essentially intact. For $n = 2$ we have the recent work of ElGamal [9]. Odlyzko [15] points out that this method can be extended to the case n bounded. I do not consider these fields here.

In this paper, I shall present and rigorously analyze two similar random algorithms, one for factoring and one for computing discrete logarithms in GF(q) where q is prime or a power of 2. The factoring algorithm will have expected worst case running time $L(N)^{\sqrt{2} + o(1)}$ and the discrete logarithm algorithm will have expected worst case running time $L(q)^{\sqrt{2} + o(1)}$ for a preprocessing stage and expected worst case running time $L(q)^{\sqrt{1/2} + o(1)}$ for the actual discrete logarithm calculation.

Both methods are quite similar to previously considered algorithms. In particular, the factoring algorithm is a variant of Dixon's random squares method [8], which itself is based on ideas of Morrison, Brillhart

[14] and earlier writers. Here the random squares method is augmented with Lenstra's elliptic curve factoring method [12] and Wiedemann's coordinate recurrence method [20] for solving a sparse system of linear equations over a finite field. The discrete logarithm algorithm is based on the index calculus method of Western, Miller [19] (see [15] for furthe references). Again, the new ingredients are the elliptic curve method (i the case q prime) and the coordinate recurrence method.

It is perhaps paradoxical that the elliptic curve factoring method can be used as a subroutine in a rigorously analyzed algorithm while it itself has not been completely rigorously analyzed. The point is that th algorithms described here are random, so a subroutine need not work on al inputs. What will be shown is that a somewhat weakened form of the elliptic curve method works fairly rapidly for most inputs. The argument uses a new result of Friedlander, Lagarias [10] in analytic number theory

That it might be possible to rigorously employ his elliptic curve method as a subroutine in the random squares algorithm was suggested by Lenstra [12]. This method has already been used in the heuristic analyse of some discrete logarithm algorithms, see Coppersmith, Odlyzko, Schroeppel [7].

In [18], Seysen describes a random factoring algorithm that under th sole assumption of the Extended Riemann Hypothesis (ERH) can be proved to have expected running time $L(N)^{\sqrt{5/4} + o(1)}$. It is likely that by using some of the methods of this paper that the expected running time for Seysen's algorithm can be reduced to $L(N)^{1 + o(1)}$, still under the assumption of the ERH.

§2. A rigorous version of the elliptic curve factoring method.

If $v,w \ge 2$, let

$$k(v,w) = \prod_{j=2}^{[w]} j^{e_j}$$

where e_j is the largest integer with $j^{e_j} \le v + 2\sqrt{v} + 1$. One iteration of the elliptic curve method to factor N with parameters v,w is as follows (cf. Lenstra [12], paragraph (2.4)).

Step 1. Choose a, x_0, y_0 at random in $\mathbb{Z}/N$ and let $b = y_0^2 - x_0^3 - ax_0$. Then $P := (x_0,y_0)$ is on the curve $E : y^2 = x^3 + ax + b$.

Step 2. Attempt to compute $k(v,w)P \bmod N$ on E by the addition procedure described in [12].

The addition procedure mimics addition of points on elliptic curves modulo a prime. Since N is presumably not prime, the procedure of Step 2 may break down. But this is good, for as shown in [12], if the addition procedure breaks down, a non-trivial divisor of N is necessarily revealed.

If Step 2 is completed, then the algorithm has been unsuccessful in factoring N. However, one then can go back to Step 1 and repeat the procedure gaining a new chance to factor N. If $n, w \ge 2$, let

$$\psi_0(n,w) = \#\{m \in (n-\sqrt{n}, n+\sqrt{n}) : \text{no prime factor of } m \text{ exceeds } w\}.$$

The following result is Corollary (2.8) in [12].

THEOREM A (Lenstra). *There is an effectively computable constant* $1 > c_1 > 0$ *with the following property. Let* N, v *be integers exceeding* 1 *such that* N *has at least two distinct prime factors and such that the smallest prime factor* p *of* N *satisfies* $3 < p \le v$. *If* w *is such that* $\psi_0(p,w) \ge 3$, *then the success probability of obtaining a non-trivial factorization of* N *with at most* h *iterations of the elliptic curve method with parameters* v,w *is at least*

$$1 - (1-c_1)^{h\psi_0(p,w)/(\sqrt{p}\,\log v)}.$$

Thus if h is a large constant times $\sqrt{p}\,(\log v)/\psi_0(p,w)$ we will expect success with at most h iterations. Since the running time for one iteration is of order $w \log v$ arithmetic operations mod N, to choose an optimal w we essentially wish to minimize the expression $\sqrt{p}\,w/\psi_0(p,w)$ (ignoring the factor $\log^2 v$).

Let

$$\psi(x,w) = \#\{m \le x : \text{no prime factor of } m \text{ exceeds } w\}.$$

If one conjectures that $\sqrt{p}/\psi_0(p,w)$ is about the same as $p/\psi(p,w)$, it is easy to choose an optimal w. Indeed, it is known that the expression $pw/\psi(p,w)$ for a fixed number $p > 3$ is minimized for $w = L(p)^{\sqrt{1/2} + o(1)}$ and the value of the fraction for this w is $L(p)^{\sqrt{2} + o(1)}$. That is, from the conjecture that $\sqrt{p}/\psi_0(p,w)$ is about the same as $p/\psi(p,w)$ one can deduce that the elliptic curve method is expected to factor N in time $L(p)^{\sqrt{2} + o(1)}$ if N is divisible by at least 2 distinct primes and its least prime p exceeds 3. (Of course, one does not know p in advance. The protocol for implementing this optimized version of the elliptic curve method involves a gradually increasing value of the parameter v in later iterations, always choosing $w = L(v)^{\sqrt{1/2}}$. When the prime p is finally trapped, we find that we probably have spent time $L(p)^{\sqrt{2} + o(1)}$ doing so.)

Although this optimized version of the elliptic curve method is based on an unproved hypothesis, Theorem A is just that, a theorem. Although the unproved hypothesis seems very difficult and is far from resolution, we do have theorems that point in this direction. The following result is a special case of Theorem 6 in [10].

THEOREM B (Friedlander-Lagarias). *For any fixed* $\varepsilon > 0$ *there is a positive constant* $c(\varepsilon)$ *such that uniformly for all* $x \ge 2$, y *with* $y \ge \exp\{(\log x)^{5/6 + \varepsilon}\}$, *the exceptional set of integers* $m \le x$ *for which*

$$\psi(m + \tfrac{1}{2}\sqrt{x}, y) - \psi(m - \tfrac{1}{2}\sqrt{x}, y) > \frac{1}{16}\,\rho\left(\frac{\log x}{\log y}\right)\sqrt{x} \qquad (2.1)$$

fails has cardinality at most $c(\varepsilon)x \cdot \exp\{-\frac{1}{2}(\log x)^{1/6}\}$.

Remarks. Here ρ denotes Dickman's function, the continuous solution on $[0,\infty)$ of the differential-delay equation

$$u\rho'(u) = -\rho(u-1), \ \rho(u) = 1 \ \text{ for } \ 0 \le u \le 1.$$

It is known (de Bruijn [4]) that

$$\rho(u) = \exp\{-(1 + o(1))u \log u\}. \qquad (2.2)$$

Theorem B does indeed point towards the unproved hypothesis mentioned above, since from de Bruijn [6], we have $\psi(x,y) \sim \rho\left(\frac{\log x}{\log y}\right) x$ for x,y satisfying the hypotheses of the theorem.

We shall apply Theorem B with $y = \exp\{(\log(x/4))^{6/7}\}$ and for m running through primes in $(x/4,x]$. Let $S(x)$ denote the set of primes $3 < p \le x$ for which

$$\frac{1}{\sqrt{p}}\,\psi_0(p, \exp\{(\log p)^{6/7}\}) > \exp\{-\tfrac{1}{6}(\log x)^{1/7} \log\log x\}. \qquad (2.3)$$

THEOREM B'. *If* $\pi(x)$ *denotes the number of primes* $p \le x$, *then*

$$\pi(x) - \#S(x) = O(x \cdot \exp\{-\tfrac{1}{2}(\log x)^{1/6}\}).$$

Proof. Let $N_1(x)$ denote the number of primes $p \in (x/4,x]$ for which (2.3) fails. For p in this range, we have

$$\sqrt{p} > \tfrac{1}{2}\sqrt{x}, \ \exp\{(\log p)^{6/7}\} > \exp\{(\log(x/4))^{6/7}\} .$$

In light of (2.2), if x is large enough, then the right side of (2.3) is smaller than the right side of (2.1) with $y = \exp\{(\log(x/4))^{6/7}\}$. Thus, for large x, if p is counted by $N_1(x)$, then p does not satisfy (2.1). Thus

$$N_1(x) \le c_2\, x \cdot \exp\{-\tfrac{1}{2}(\log x)^{1/6}\}$$

for some absolute constant c_2 and for all large x.

Next, we let $N_2(x)$ denote the number of primes $p \in (x/16, x/4]$ for which (2.3) fails. Then $N_2(x) \le N_1(x/4)$, so that

$$N_2(x) \le c_2\,\frac{x}{4}\exp\{-\tfrac{1}{2}(\log \tfrac{x}{4})^{1/6}\},$$

if x is large. In general let $N_i(x)$ denote the number of primes $p \in (x/4^i, x/4^{i-1}]$ for which (2.3) fails where i is any integer with $4^i < \sqrt{x}$. For large x we have

$$N_i(x) \le c_2\,\frac{x}{4^i}\exp\{-\tfrac{1}{2}\left(\log \frac{x}{4^i}\right)^{1/6}\}$$

for all such i . Now

$$\pi(x) - \#S(x) = \sum_i N_i(x) + O(\sqrt{x}) ,$$

and so a simple calculation gives our result.

We are now in a position to prove the major result of this section.

THEOREM 2.1. *There is a random algorithm which on input of an integer* $N \ge 2$ *and a parameter* $v \ge 2$ *will produce integers* F, R *with* $N = FR$ *and the complete prime factorization of* F . *Moreover, with probability at least* $1 - (\log N)/N$ *no prime in* $S(v)$ *divides* R . *The running time is* $O((\log N)^4 \exp\{2(\log v)^{6/7}\})$.

Proof. By dividing out any factors 2, 3 from N, we may assume $(6,N) = 1$. This can be accomplished in at most $O(\log N)$ arithmetic operations, that is, in time $O((\log N)^3)$.

Let

$$w = \exp\{(\log v)^{6/7}\}, \ h = 1 + [\tfrac{1}{c_1}(\log v)(\log N)\exp\{\tfrac{1}{6}(\log v)^{1/7}\log\log v\}]$$

where c_1 is as in Theorem A. We apply the elliptic curve method with parameters v,w to N repeatedly until either we obtain a non-trivial factorization of N or we have done h iterations, whichever comes first. This can be accomplished in at most $O(hw \log v)$ arithmetic operations mod N, that is, in time

$$O((\log N)^3\exp\{2(\log v)^{6/7}\}).$$

If N is not a prime power and has a prime factor p in $S(v)$, then the probability that the above procedure produces a non-trivial factorization of N is at least

$$1-(1-c_1)^{h\psi_0(p,w)/(\sqrt{p}\log v)} > 1-\frac{1}{N} .$$

Indeed, by Theorem A, the probability is at least the first expression and since $p \in S(v)$, we have

$$\frac{h\psi_0(p,w)}{\sqrt{p}\log v} \geq \frac{1}{c_1}\log N ,$$

so that

$$1-(1-c_1)^{h\psi_0(p,w)/(\sqrt{p}\log v)} \geq 1-(1-c_1)^{\frac{1}{c_1}\log N} > 1-\frac{1}{N} .$$

If N has been factored, we apply the same procedure to the factors. If in turn, one of them factors, we again repeat the procedure on its factors, and so on. Since N is the product of at most $\log N$ primes, the time for all of these applications of the elliptic curve method is at most

$$O((\log N)^4 \exp\{2(\log v)^{6/7}\}).$$

With probability at least $1-(\log N)/N$, each factor m of N which the above procedure does not factor is either a power of a prime in $S(v)$ or is not divisible by any prime in $S(v)$. For each such m write m in the form k^a where k,a are integers and a is maximal. This can be accomplished in time $O((\log N)^4)$. If $k > v$, then stop working on the factor m of N. If $k \leq v$, apply the primality test in [2] to k, taking time at most

$$O((\log v)^{c_3\log\log\log(10v)})$$

for some constant c_3. Let F denote the product of all such k^a for which k has been proved prime and let $R = N/F$. Then with probability at least $1-(\log N)/N$ no prime in $S(v)$ divides R.

§3. Factoring with random squares.

It has long been known that if N is divisible by at least 2 distinct odd primes, then factoring N is random polynomial time equivalent to producing random solutions to the congruence $x^2 \equiv y^2 \bmod N$. Indeed, if $x^2 \equiv y^2 \bmod N$ is a random solution, then $(x-y,N)$ is a non-trivial factor of N with probability at least 1/2. Conversely, if the complete prime factorization of N is known and y is a random residue $\bmod N$, then it is a simple matter using the elementary theory of congruences to make a random choice in the set of $x \bmod N$ with $x^2 \equiv y^2 \bmod N$.

This concept is exploited successfully in several of the practical factoring algorithms, principally the continued fraction method [14] and the quadratic sieve [16]. However, these practical algorithms suffer two theoretical deficiencies. One, we cannot prove the congruences $x^2 \equiv y^2 \bmod N$ that are produced are truly random. Two, we cannot prove the methods are likely even to produce any congruences $x^2 \equiv y^2 \bmod N$, random or not. It should be stressed, however, that these deficiencies probably lie in our current methods of proof or in our ingenuity, not in the algorithms themselves which almost certainly work as claimed.

The random squares factoring method of Dixon [8] is a particularly simple and randomized version of the continued fraction method (from which the continued fraction has been removed!) that is amenable to rigorous analysis. By trading the practical advantages of the continued fraction method or the quadratic sieve for rigor, the true value of the random squares method lies in the theoretical analysis of the complexity of factoring, not in actually doing so.

The random squares method with parameter v to factor N may be described briefly as follows. Let $Q(A)$ denote the least non-negative residue of $A \bmod N$. In stage 1, one continues to choose random integers

A until $\pi(v) + 1$ values of $Q(A)$ are found which completely factor with the primes up to v. In stage 2, a non-empty subset $Q(A_1),\ldots,Q(A_k)$ of the stage 1 successes is found whose product is a square, say x^2. If $y \equiv A_1 \ldots A_k \bmod N$, then clearly $x^2 \equiv y^2 \bmod N$.

It is not hard to show that if N is divisible by at least 2 distinct odd primes and if each $(A_i,N) = 1$, then $(x-y,N)$ is a non-trivial factor of N with probability at least 1/2 (see Lemma 3 in [8]). Indeed, among the various residues $A \bmod N$ for which $Q(A) = Q(A_1)$ at least half of these choices give a value of $y \bmod N$ (when substituted for A_1 in the definition of y) which lead to a non-trivial factorization of N. Since the algorithm is "indifferent" to which value of A gave the quadratic residue $Q(A_1)$ in stage 1, at least half of the instances of the algorithm which use $Q(A_1)$ in stage 2 lead to a non-trivial factorization of N.

To further specify the random squares method one must also describe the subroutines used to accomplish the main tasks of stage 1 and stage 2. That is, in stage 1 we must specify which method is used to recognize which values of $Q(A)$ produced factor completely with the primes up to v. Further, in stage 2 we must specify which method is to be used to find the non-empty subset. Once these subroutines are specified, the parameter v is then chosen so as to minimize the running time.

In [14] it was shown that the problem of finding the non-empty subset in stage 2 is really a problem in linear algebra. Indeed if

$$Q(A) = \prod_{i=1}^{\pi(v)} p_i^{a_i}$$

is the prime factorization of a stage 1 success, where p_i denotes the i-th prime and the exponents a_i are non-negative integers, then a linear dependency among the vectors $(a_1,\ldots,a_{\pi(v)}) \bmod 2$ corresponds to a non-empty subset of the numbers $Q(A)$ whose product is a square. Since we have $\pi(v) + 1$ such vectors, a linear dependency must exist.

In [8], Dixon used trial division by the primes up to v as the subroutine in stage 1 and Gaussian elimination as the subroutine in stage 2 and was able to show that with v chosen appropriately the running time is $\le L(N)^{3\sqrt{2} + o(1)}$. In [16] it was shown that with these subroutines the optimal choice of v is $L(N)^{1/2 + o(1)}$ and the running time is exactly $L(N)^{2 + o(1)}$. Furthermore, in [16] it was shown that if the Pollard-Strassen factorization method with the early abort strategy is used in stage 1 and the Coppersmith-Winograd elimination method is used in stage 2, then the optimal choice of v is $L(N)^{\sqrt{2/5} + o(1)}$ and the running time is exactly $L(N)^{\sqrt{5/2} + o(1)}$.

In this section we shall show that if the algorithm of Theorem 2.1 is used in stage 1 and a new elimination method of Wiedemann [20] is used in stage two, then the optimal choice of v is $L(N)^{\sqrt{1/2} + o(1)}$ and the running time is exactly $L(N)^{\sqrt{2} + o(1)}$.

Towards this end, we begin as follows. Let $\psi_1(x,y,z)$ denote the number of integers $n \le x$ divisible solely by primes p such that if $p > z$, then $p \in S(y)$.

LEMMA 3.1. *If* $y = L(x)^a$ *with* $a > 0$ *fixed and* $z = \exp\{64(\log\log x)^6\}$, *we have*

$$\psi_1(x,y,z) \sim \psi(x,y) \sim \rho\left(\frac{1}{a}\sqrt{\frac{\log x}{\log\log x}}\right) x = x \cdot L(x)^{-\frac{1}{2a} + o(1)} .$$

Proof. The equality follows from (2.2) and the latter asymptotic relation follows either from the main result in Hildebrand [11] or from Maier [13]. It remains to show the first asymptotic relation. Let $u = (\log x)/\log y = \frac{1}{a}\sqrt{\frac{\log x}{\log\log x}}$. First note that

$$0 \le \psi(x,y) - \psi_1(x,y,z) \le \sum_{p\in(z,y],\, p\notin S(y)} \psi(\tfrac{x}{p}, y)$$

$$= x \sum_{p\in(z,y],\, p\notin S(y)} \left(\frac{1}{p}\,\rho\left(u - \frac{\log p}{\log y}\right)\right)\left(1 + O\left(\frac{u \log u}{\log x}\right)\right)$$

$$\le \rho(u-1) \times \left(1 + O\left(\frac{u \log u}{\log x}\right)\right) \sum_{p\in(z,y],p\notin S(y)} \frac{1}{p} \qquad (3.1)$$

where again we use the papers [11], [13].

Using partial summation and Theorem B′ we have

$$\sum_{p\in(z,y],p\notin S(y)} \frac{1}{p} = \frac{1}{y}\Big(\sum_{p\in(z,y],p\notin S(y)} 1\Big) + \int_z^y \frac{1}{t^2}\Big(\sum_{p\in(z,t],p\notin S(y)} 1\Big)dt$$

$$\le \frac{1}{y}\Big(\sum_{p\le y,p\notin(y)} 1\Big) + \int_z^y \frac{1}{t^2}\Big(\sum_{p\le t,p\notin S(y)} 1\Big)dt$$

$$= O(\exp\{-\tfrac{1}{2}(\log y)^{1/6}\}) + O(\int_z^y \frac{1}{t}\exp\{-\tfrac{1}{2}(\log t)^{1/6}\}dt)$$

$$= O((\log z)^{5/6}\exp\{-\tfrac{1}{2}(\log z)^{1/6}\})$$

$$= O\left(\frac{(\text{loglog}x)^5}{\log x}\right) . \qquad (3.2)$$

Further from Lemma 3 in Alladi [3] it follows that

$$\rho(u-1) \sim \rho(u)u \log u \quad \text{as} \quad u \to \infty .$$

Putting this estimate and (3.2) into (3.1) it follows that

$$\psi_1(x,y,z) = \psi(x,y) + O\left(\rho(u) \times \frac{u\ (\log u)(\text{loglog}x)^5}{\log x}\right)$$

$$= \psi(x,y)\left(1 + O\left(\frac{(\text{loglog}\ x)^{11/2}}{(\log\ x)^{1/2}}\right)\right) ,$$

which proves the first asymptotic relation of the lemma.

Let $\omega(N)$ denote the number of distinct prime factors of N and let $\tau(N)$ denote the number of natural divisors of N.

Lemma 3.2. *Let* $x \ge 1$, *let* S *be an arbitrary set of primes, let* $N > 1$ *be an integer not divisible by any prime in* S, *let* $T(x)$ *denote the set of integers* $m \le x$ *not divisible by any prime outside of* S , *and let*

$$T_N(x) = \{m \in [1,N]: \ Q(m) = m^2 \bmod N \in T(x)\} .$$

Then

$$2^{\omega(N)} \cdot \#T(x) \ge \#T_N(x) \ge (\#T(\sqrt{x}))^4\Big(\sum_{t\in T(\sqrt{x})} \tau(t)\Big)^{-2} .$$

Proof. This result is a mild generalization of Lemma 3.1 in [16] where the set S consists of all the primes in an interval. That lemma is itself a generalization of Lemma 2 in Dixon [8]. The proof of the lemma at hand follows from the same argument.

Theorem 3.3. *Let* $a > 0$ *be fixed. If* $N > 1$ *is an integer not divisible by any prime up to* $L(N)^a$ *and* S *is the set of primes* $p \le L(N)^a$ *in* $S(L(N)^a)$ *together with all of the primes up to* $\exp\{64(\text{loglog}\ N)^6\}$, *then*

$$\#\, T_N(N) = N \cdot L(N)^{-\frac{1}{2a}+o(1)} ,$$

using the notation of Lemma 3.2.

Proof. From Lemma 3.1 we have

$$\#\, T(N) = N \cdot L(N)^{-\frac{1}{2a}+o(1)} , \quad \#\, T(\sqrt{N}) = \sqrt{N} \cdot L(N)^{-\frac{1}{4a}+o(1)} .$$

Thus our result will follow from Lemma 3.2 if we show

$$2^{\omega(N)} = L(N)^{o(1)} , \quad \sum_{t\in T(\sqrt{N})} \tau(t) = \sqrt{N} \cdot L(N)^{-\frac{1}{4a}+o(1)} . \qquad (3.3)$$

From the hypothesis that N has no prime factor up to $L(N)^a$, it follows that

$$\omega(N) \le \frac{1}{a}\sqrt{\frac{\log N}{\text{loglog}\ N}} ,$$

so that the first equality in (3.3) is immediate.

Since

$$\sum_{t\in T(\sqrt{N})} \tau(t) \ge \#\, T(\sqrt{N}) = \sqrt{N} \cdot L(N)^{-\frac{1}{4a}+o(1)}$$

from Lemma 3.1, we have half of the second equality in (3.3). To complete the proof it is sufficient to cite Lemma 3.2 in [16], where a quantity greater than or equal to $\Sigma_{t\in T(\sqrt{N})}\ \tau(t)$ is majorized by the expression $\sqrt{N} \cdot L(N)^{-\frac{1}{4a}+o(1)}$.

The following algorithm is the main goal of this section. The letter "R" stands for the random squares method, "E" stands for the elliptic curve method, and "C" the coordinate recurrence method.

Algorithm REC

Let $a > 0$ be fixed. On input of an integer $N > 1$, first use trial division to test N for prime factors up to $v = L(N)^a$. If this procedure produces a non-trivial factorization of N, then stop. Otherwise, let $z = \max\{3, \exp\{64(\log\log N)^6\}\}$. We iterate the following procedure until we have $\pi(v) + 1$ successes. The procedure is to choose a random integer $A \in [1, N-1]$, remove any prime factors up to z from $Q(A) = A^2 \bmod N$ by trial division, and if the unfactored portion exceeds 1, apply the algorithm of Theorem 2.1 with parameter $v = L(N)^a$ to this unfactored portion of $Q(A)$. A "success" is defined as a pair A, $Q(A)$ for which this procedure outputs the complete prime factorization of $Q(A)$ and none of these primes exceeds v.

For each of the $\pi(v) + 1$ successes A, $Q(A)$, let $\vec{v}(A)$ denote the vector $(a_1,\ldots,a_{\pi(v)}) \bmod 2$ where $Q(A) = \prod_{i=1}^{\pi(v)} p_i^{a_i}$ and p_i denotes the i-th prime. Use the coordinate recurrence method of Wiedemann [20] (Algorithm 1) to find a subset $\vec{v}(A_1),\ldots,\vec{v}(A_k)$ of the $\pi(v) + 1$ vectors with $\vec{v}(A_1) + \ldots + \vec{v}(A_k) = \vec{0}$. Let x be an integer with $x^2 = Q(A_1) \ldots Q(A_k)$ and let $y = A_1 \ldots A_k \bmod N$. Compute $(x-y, N)$. If this is a non-trivial factor of N, the algorithm has been successful.

From Theorems 2.1 and 3.3 we see that we shall expect to iterate the procedure with the random A's precisely $L(N)^{a + \frac{1}{2a} + o(1)}$ times to achieve the requisite number of successes. Since each iteration of this procedure has running time $L(N)^{o(1)}$, the expected time for the collection of all of the factored $Q(A)$'s is $L(N)^{a + \frac{1}{2a} + o(1)}$. Thus a choice of $a = \sqrt{1/2}$ will minimize the expected running time of this stage of the algorithm - it is $L(N)^{\sqrt{2} + o(1)}$.

The second stage of the algorithm involving the coordinate recurrence method is also probabilistic. Since each vector $\vec{v}(A)$ has at most $O((\log N)/\log\log N) = L(N)^{o(1)}$ non-zero entries, the running time for this stage will be $L(N)^{2a+o(1)} = L(N)^{\sqrt{2} + o(1)}$ with $a = \sqrt{1/2}$. Note that Algorithm 1 in [20] involves solving $A\vec{x} = \vec{b}$ where A is a non-singular square matrix. If this procedure is applied when A is possibly singular, then the algorithm will either solve the equation for $\vec{x}$ or find a non-trivial solution to $A\vec{x} = \vec{0}$. We use this algorithm as follows. Take the first $\pi(v)$ vectors and from the matrix A by writing these vectors as columns. Let $\vec{b}$ be the $\pi(v) + 1$- st vector written as a column. Thus we shall either find a linear dependency involving the $\pi(v) + 1$ - st vector or we shall find a linear dependency among just the first $\pi(v)$ vectors. In either case, we have found the requisite linear dependency.

The integer x may be as large as $L(N)^{aL(N)^a}$. However, it is only necessary to compute $x \bmod N$. This can be done in time $L(N)^{a + o(1)}$ by first finding the prime factorization of x and then computing $x \bmod N$. An alternative is to follow the algorithm described in [14].

The coordinate recurrence method (Algorithm 1 of [20]) is probabilistic. According to Proposition 3 in [20], the expected number of iterations of Algorithm 1 before a linear dependency is found is of order $\log \pi(v) = L(N)^{o(1)}$. Note that we might use Algorithm 2 in [20]. This has deterministic running time $L(N)^{2a + o(1)}$, but requires more space - $L(N)^{2a + o(1)}$ against $L(N)^{a + o(1)}$ for Algorithm 1.

Assuming we successfully find the linear dependency, the probability Algorithm REC will produce a non-trivial factor of N is at least 1/2, provided N is divisible by at least 2 distinct odd primes - see the discussion earlier in this section.

Summing up we have the following analysis of Algorithm REC.

Theorem 3.4. *With* $a = \sqrt{1/2}$, *the expected running time of Algorithm REC is* $L(N)^{\sqrt{2} + o(1)}$ *and the space required is* $L(N)^{\sqrt{1/2} + o(1)}$. *If* N *is divisible by at least* 2 *distinct odd primes, the probability that Algorithm REC will produce a non-trivial factor of* N *is at least* 1/2.

§4. Discrete logarithms in GF(p).

Let p denote a fixed prime exceeding 3. In this section we shall see how a certain natural analog of Algorithm REC from section 3 can be used to compute discrete logarithms in GF(p). The general idea is as follows (cf. Adleman [1], Western-Miller [19]). Suppose g is a primitive element in GF(p), $x \in GF(p)^*$, and we wish to compute $\log_g x$, that is, some number $y \bmod (p-1)$ with $g^y = x$. Let $v \geq 2$ be a parameter. Suppose the least positive residue of g^e in GF(p) factors completely over the primes up to v :

$$g^e \equiv \prod_{i=1}^{\pi(v)} p_i^{a_i} \bmod p.$$

Taking the $\log_g$ of both sides we obtain

$$e \equiv \sum_{i=1}^{\pi(v)} a_i \log_g p_i \bmod (p-1). \tag{4.1}$$

Stage one is to choose random exponents $e \in \{1,\dots,p-1\}$ until so many values are found with $g^e \bmod p$ factoring completely with the primes up to v that the corresponding system of equations of the form (4.1) for the unknown quantities $\log_g p_i \bmod (p-1)$ for $i = 1,\dots,\pi(v)$ has full rank. (What it means for a system of linear equations over $\mathbb{Z}/(p-1)$ to have full rank is that when considered over $\mathbb{Z}/q$, the system has full rank for each prime factor q of $p-1$.)

Stage two is to solve the system of equations of the form (4.1) for the quantities $\log_g p_i \bmod (p-1)$ for $i = 1,..,\pi(v)$.

If x is one of the primes up to v , or more generally, if x factors completely with the primes up to v , then it is an easy matter to write $\log_g x \bmod (p-1)$ as a linear combination of the now known quantities $\log_g p_i \bmod (p-1)$ and so compute it. Probably, though, x will not be in this form in which case we enter stage three of the algorithm. This involves choosing random exponents e until one is found with the least positive residue of $g^e x$ factoring completely with the primes up to v :

$$g^e x \equiv \prod_{i=1}^{\pi(v)} p_i^{b_i} \bmod p .$$

Thus

$$\log_g x \equiv -e + \sum_{i=1}^{\pi(v)} b_i \log_g p_i$$

and we are done.

Stages one and two comprise a precomputation part of the algorithm. If we next want to compute $\log_g x'$ for some other $x' \in GF(p)$, we need only stage 3 for x' since we already know the logarithms of the small primes.

This general algorithm is known as the index calculus algorithm (the word index is synonymous with discrete logarithm). As with the random squares method, the version of the index calculus algorithm presented here will use the elliptic curve method and the coordinate recurrence method as subroutines. The only difficulty in the analysis will concern showing when the system of equations of the form (4.1) is expected to have full rank. To solve this problem, we shall amend the index calculus algorithm slightly by also considering values of $g^e p_i \bmod p$ for random exponents e and for the various p_i. We begin with the following simple lemma, the main idea of which appears in [1].

LEMMA 4.1. *Let* V *be a vector space over a field* F *with* $\dim V = k$ *and* $0 < k < \infty$. *Let* S *be a finite set of vectors in* V *and let* $b_1,\dots,b_k$ *be a basis of* V. *Let* $\ell = [2\log_2 k] + 3$ *where* $\log_2$ *refers to the binary logarithm. With the uniform distribution over* S, *say we make* $2k\ell$ *independent choices of elements from* S *(with replacement), labeling the chosen vectors* $v_1,\dots,v_{k\ell}$, $w_1,\dots,w_{k\ell}$. *Let* V' *be the subspace of* V *spanned by* $v_1,\dots,v_{k\ell}$ *and the vectors* $b_j + w_{(j-1)\ell+i}$ *for* $j = 1,\dots,k$ *and* $i = 1,\dots,\ell$. *Then with probability at least* $1 - 1/(2k)$, $V' = V$.

Proof. Let $V_0 = \{0\}$ and for $j = 1,\dots,k$, let V_j denote the subspace of V spanned by $v_1,\dots,v_{j\ell}$. Then the probability that

$$\#(S \cap V_k) \ge \frac{1}{2}\cdot \#S \tag{4.2}$$

is at least $1 - k\cdot 2^{-\ell}$. Indeed, if (4.2) fails, then for any fixed $j = 0,\dots,k-1$, the probability that each of $v_{j\ell+1},\dots,v_{(j+1)\ell}$ is in V_j is less than $2^{-\ell}$. But if one of these vectors is not in V_j, then $\dim V_{j+1} > \dim V_j$. Thus, if (4.2) fails, then with probability at least $1 - k\cdot 2^{-\ell}$ we have $0 < \dim V_1 < \dots < \dim V_k$, so that $V_k = V$. But if $V_k = V$, then (4.2) obviously holds. This proves our assertion about the probability that (4.2) holds.

Assume now that (4.2) holds. Let W_j be the subspace of V spanned by V_k and the vectors $b_j + w_{(j-1)\ell+i}$ for $i = 1,\dots,\ell$. Fix any $j = 1,\dots,k$. Then with probability at least $1 - 2^{-\ell}$ we have $b_j \in W_j$. Indeed, if $w_{(j-1)\ell+i} \in V_k$, then

$$b_j \in \operatorname{span}(V_k,\ b_j + w_{(j-1)\ell+i}) \subset W_j .$$

Further, from (4.2) the probability that at least one of $w_{(j-1)\ell+i} \in V_k$ for $i = 1,\dots,\ell$ is at least $1 - 2^{-\ell}$. Thus our assertion follows.

Thus with probability at least $1 - 2k\cdot 2^{-\ell}$ we have both (4.2) and $b_j \in W_j$ for each $j = 1,\dots,k$. In this case V' contains the basis $b_1,\dots,b_k$, so $V' = V$. It remains to note that $1 - 2k\cdot 2^{-\ell} > 1 - 1/(2k)$.

Remark. It should be clear from the proof that if we allow repeated elements in S so that S is now a multi-set the same result holds. More generally the same result holds if we replace the uniform distribution on S with an arbitrary distribution.

We are now in a position to consider the following algorithm. The letter "I" stands for the index calculus algorithm, "E" stands for the elliptic curve method, and "C" the coordinate recurrence method.

ALGORITHM IEC

On input of a prime $p > 3$, a primitive element $g \bmod p$, a non-zero residue $x \bmod p$, and a parameter $v = L(p)^a$ where $a > 0$ is fixed, do the following. Let $z = \max\{3, \exp\{64(\log\log p)^6\}\}$, let $k = \pi(v)$, and let $\ell = [2\log_2 k] + 3$. Iterate the following procedure until we have $k\ell$ successes. The procedure is to choose a random integer $e \in \{1,\dots,p-1\}$, form $g^e \bmod p$, remove any prime factors up to z by trial division, and if the unfactored portion exceeds 1, continue with the algorithm of Theorem 2.1 with parameter v. A "success" is defined as a pair e, $g^e \bmod p$ where this procedure outputs the complete prime factorization of $g^e \bmod p$ and no prime involved exceeds v. Next, for $j = 1,\dots,k$ continue with the same procedure applied to $(p_j g^e) \bmod p$ for random choices of e until we have ℓ successes for each j. Here, p_j denotes the j-th prime. This concludes stage one of the algorithm.

Next, let $y_j = \log p_j \bmod (p-1)$. Each success among the first $k\ell$ is of the form

$$g^e \equiv \prod_{i=1}^{k} p_i^{a_i} \bmod p$$

and each success among the latter $k\ell$ is of the form

$$p_j g^e \equiv \prod_{i=1}^{k} p_i^{b_i} \bmod p.$$

The former relations lead to equations of the form

$$e \equiv \sum_{i=1}^{k} a_i y_i \bmod (p-1)$$

while the latter relations lead to equations of the form

$$e \equiv -y_j + \sum_{i=1}^{k} b_i y_i \bmod (p-1) .$$

We use the coordinate recurrence method [20] to solve these equations for $y_1,\ldots,y_k$. This concludes stage two of the algorithm.

Finally, choose random exponents e until one is found (using the same procedure as in stage one) with $(xg^e) \bmod p$ factoring completely with the primes up to v. If

$$xg^e \equiv \prod_{i=1}^{k} p_i^{c_i} \bmod p,$$

then $\log_g x \equiv -e + \sum_{i=1}^{k} c_i y_i \bmod (p-1)$. This concludes the third and final stage of the algorithm.

Remarks. If e is a random variable with uniform distribution in $\{1,\ldots,p-1\}$ then so is $xg^e \bmod p$ for any fixed $x \not\equiv 0 \bmod p$. Thus from Therorem 2.1 and Lemma 3.1, the probability that the procedure just described will produce the complete prime factorization of $(xg^e) \bmod p$ with no prime involved exceeding v is $L(p)^{-\frac{1}{2a} + o(1)}$. Thus the expected running time of stage 1 is $L(p)^{a + \frac{1}{2a} + o(1)}$ and the expected running time of stage 3 is $L(p)^{\frac{1}{2a} + o(1)}$.

Some comment is needed for stage two. First, from Lemma 4.1, the system of equations has full rank with probability at least $1 - 1/(2k)$. Second, the coordinate recurrence method must be applied in a finite field and $\mathbb{Z}/(p-1)$ is not one. There are two exits from this dilemma. One is to apply algorithm REC from section 3 to p-1, completely factoring it

in expected time $L(p)^{\sqrt{2} + o(1)}$. Next, for each prime q that divides $p-1$ we apply the coordinate recurrence method to the system of equations considered over $\mathbb{Z}/q$. If $q^2 | p-1$ then we use a Hensel lifting argument to solve the system of equations over $\mathbb{Z}/q^2$ (again by the coordinate recurrence method over $\mathbb{Z}/q$), and so on if a higher power of q divides $p-1$. Solutions over the various $\mathbb{Z}/q^a$ are then glued together with the Chinese Remainder Theorem to form the solution over $\mathbb{Z}/(p-1)$.

The other way to solve the system via the coordinate recurrence method does not involve trying especially hard to factor p-1. If this method is applied to a system over a non-field it could well break down when it tried to invert a non-invertible element. In our case this would just produce a non-trivial factorization of the modulus. The coordinate recurrence method can then be begun again for the various factors of $p-1$. If the method does not break down with a particular modulus, it is expected to produce the required solution.

Since the system of equations is sparse, the expected running time to solve the system of equations is $L(p)^{2a+o(1)}$ and the space is $L(p)^{a+o(1)}$.

It is clear that to minimize the time for stage one we should choose $a = \sqrt{1/2}$. This leads to a running time of $L(p)^{\sqrt{2} + o(1)}$ for stage one and the same running time for stage two. We sum up our results in the following theorem.

THEOREM 4.2. *Suppose* $p > 3$ *is prime. Algorithm IEC with parameter* $L(p)^{\sqrt{1/2}}$ *is expected to complete preprocessing for the discrete logarithm problem over* $GF(p)$ *in time* $L(p)^{\sqrt{2} + o(1)}$ *and in space* $L(p)^{\sqrt{1/2} + o(1)}$. *After the preprocessing stage has been completed, any discrete logarithm in* $GF(p)$ *may be computed in expected time and space* $L(p)^{\sqrt{1/2} + o(1)}$.

§5. DISCRETE LOGARITHMS OVER $GF(2^n)$.

In the last section we represented elements of $GF(p) = \mathbb{Z}/p$ by their least positive residue. Since this is an integer it made sense to talk about an element of $GF(p)$ factoring into small primes. We would like to have a similar situation in $GF(2^n)$. Recall that if $f(x) \in (\mathbb{Z}/2)[x]$ is irreducible of degree n then $GF(2^n) = (\mathbb{Z}/2)[x]/(f(x))$. Since each coset in this quotient structure has a unique representative with degree $< n$, we may represent $GF(2^n)$ by the polynomials in $(\mathbb{Z}/2)[x]$ with degree $< n$. Since $(\mathbb{Z}/2)[x]$ is a unique factorization domain, it thus makes sense to talk about an element of $GF(2^n)$ factoring into small primes (low degree irreducibles).

In fact, the situation for $GF(2^n)$ is somewhat easier than with $GF(p)$. While we do not have an analog of the elliptic curve method, we do have random polynomial time algorithms to factor polynomials in $(\mathbb{Z}/2)[x]$ (see references on p.235 of [15]). Thus on presentation of a polynomial $h(x) \in (\mathbb{Z}/2)[x]$ of degree $< n$, we can determine a complete factorization of $h(x)$ into irreducibles in expected time less than $(\log n)^c$ for some absolute constant $c > 0$.

The only other difference with $GF(p)$ is that we need an analogy to Lemma 3.1 which gives the proportion of polynomials in $(\mathbb{Z}/2)[x]$ up to a certain degree all of whose irreducible factors have small degrees. Such a result may be found in Odlyzko [15].

LEMMA 5.1 (Odlyzko). *Suppose* $m^{1/100} \le d \le m^{99/100}$. *The proportion of polynomials in* $(\mathbb{Z}/2)[x]$ *of degree* $\le m$ *all of whose irreducible factors have degrees* $\le d$ *among all polynomials in* $(\mathbb{Z}/2)[x]$ *with degree* $\le m$ *is* $\exp\{-(1+o(1))u \log u\}$ *where* $u = m/d$.

If we choose $v = [\log_2(L(2^n)^a)]$ for a fixed $a > 0$, then the number of irreducible polynomials in $(\mathbb{Z}/2)[x]$ with degree $\le v$ is $2^{(1+o(1))v} = L(2^n)^{a+o(1)}$ (cf. Odlyzko [15]). Further, from Lemma

5.1, the number of members of $GF(2^n)$ whose principal representative is a product of irreducible factors of degree $\le v$ is $2^n \cdot L(2^n)^{-\frac{1}{2a} + o(1)}$. Thus the expected time to complete stage one is $L(2^n)^{a + \frac{1}{2a} + o(1)}$. The value $a = \sqrt{1/2}$ minimizes this expression, giving the running time of $L(2^n)^{\sqrt{2} + o(1)}$.

Finally, the coordinate recurrence method may still be used in stage two. The discrete logarithms are integers defined $\mathrm{mod}(2^n-1)$. If 2^n-1 is composite, the same devices as discussed in section 4 may be used to get around this problem. Summing up we have the following.

THEOREM 5.2. *Algorithm IEC of section 4 with the changes discussed above is expected to complete preprocessing for the discrete logarithm problem in* $GF(2^n)$ *in time* $L(2^n)^{\sqrt{2} + o(1)}$ *and in space* $L(2^n)^{\sqrt{1/2} + o(1)}$. *After preprocessing, any discrete logarithm may be computed in expected time and space* $L(2^n)^{\sqrt{1/2} + o(1)}$.

Remark. While the algorithms of sections 3 and 4 are in a sense near to the best we know of, even allowing heuristic - empirical algorithms, the algorithm of this section is far from the best discrete logarithm algorithm over $GF(2^n)$. As mentioned in the introduction, the algorithm of Coppersmith [6] has a heuristic running time of $\exp\{O(n^{1/3}(\log n)^{2/3})\}$. Nevertheless, the algorithm of this section is the fastest we know of now with a rigorous analysis.

REFERENCES

1. L. M. Adleman, A subexponential algorithm for the discrete logarithm problem with applications to cryptography, Proc. 20th IEEE Found. Comp. Sci. Symp. (1979), 55-60.

2. L. M. Adleman, C. Pomerance, and R. S. Rumely, On distinguishing prime numbers from composite numbers, Annals Math. 117(1983), 173-206.

3. K. Alladi, The Turán-Kubilius inequality for integers without large prime factors, J. Reine Angew. Math. 335 (1982), 180-196.

4. N. G. de Bruijn, The asymptotic behaviour of a function occurring in the theory of primes, J. Indian Math. Soc. (N.S.) 15(1951), 25-32.

5. N. G. de Bruijn, On the number of positive integers $\le x$ and free of prime factors $> y$, Nederl. Akad. Wetensch. Proc. Ser. A 54(1951), 50-60.

6. D. Coppersmith, Fast evaluation of logarithms in fields of characteristic two, IEEE Trans. Inform. Theory IT-30(1984), 587-594.

7. D. Coppersmith, A. M. Odlyzko, R. Schroeppel, Discrete logarithms in GF(p), Algorithmica 1(1986), 1-15.

8. J. D. Dixon, Asymptotically fast factorization of integers, Math. Comp. 36(1981), 255-260.

9. T. ElGamal, A subexponential-time algorithm for computing discrete logarithms over $GF(p^2)$, IEEE Trans. Inform. Theory, to appear.

10. J. B. Friedlander and J. C. Lagarias, On the distribution in short intervals of integers having no large prime factor, J. Number Theory, to appear.

11. A. Hildebrand, On the number of positive integers $\le x$ and free of prime factors $> y$, J. Number Theory 22(1986), 289-307.

12. H. W. Lenstra, Jr., Factoring integers with elliptic curves, preprint.

13. H. Maier, On integers free of large prime factors, unpublished manuscript.

14. M. A. Morrison and J. Brillhart, A method of factoring and the factorization of F_7, Math. Comp. 29(1975), 183-205.

15. A. M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, in "Advances in Cryptology" (Pro. Eurocrypt '84), Springer Lecture Notes in Computer Science 209(1985), 224-314.

16. C. Pomerance, Analysis and comparison of some integer factoring algorithms, in "Computational Methods in Number Theory: Part I", H. W. Lenstra, Jr. and R. Tijdeman, eds., Math. Centre Tract 154(1982), 89-139.

17. C. P. Schnorr and H. W. Lenstra, Jr., A Monte Carlo factoring algorithm with linear storage, Math. Comp. 43(1984), 289-311.

18. M. Seysen, A probabilistic factorisation algorithm with quadratic forms of negative discriminant, Math. Comp., to appear.

19. A. E. Western and J. C. P. Miller, "Tables of Indices and Primitive Roots", Royal Society Mathematical Tables, vol. 9, Cambridge Univ. Press, 1968.

20. D. Wiedemann, Solving sparse linear equations over finite fields, IEEE Trans. Inform. Theory IT-32(1986), 54-62.

21. M. E. Hellman and J. M. Reyneri, Fast computation of discrete logarithms in GF(q), in "Advances in Cryptography: Proceedings of CRYPTO '82, D. Chaum, R. Rivest, and A. Sherman, eds., pp. 3-13, Plenum Press, 1983.