Contemporary Mathematics
Volume 00, 0000

Discrete Logarithms and Smooth Polynomials

A. M. ODLYZKO

December 27, 1993

ABSTRACT. This paper is a survey of recent advances in discrete logarithm algorithms. Improved estimates for smooth integers and smooth polynomials are also discussed.

1. Introduction

If G denotes a group (written multiplicatively), and $\langle g \rangle$ the cyclic subgroup generated by $g \in G$, then the discrete logarithm problem for G is to find, given $g \in G$ and $y \in \langle g \rangle$, the smallest nonnegative integer x such that $y = g^x$. This integer x is called the discrete logarithm of y to the base g, and is written $x = \log_g y$.

The discrete log problem has been studied by number theorists for a long time. The main reason for the intense current interest in it, though, is that many public key cryptosystems depend for their security on the assumption that it is hard, at least for suitably chosen groups. With the proposed adoption of the NIST digital signature algorithm [**28**] (based on the ElGamal [**10**] and Schnorr [**35**] proposals), even more attention is likely to be drawn to this area.

There are already several surveys of discrete logarithm algorithms and discrete logarithm cryptosystems in the literature [**24, 29**]. The purpose of this brief note is to summarize the significant developments in this area since the McCurley survey [**24**] was published. The main emphasis will be on discrete logarithm problems in finite fields. There is also extensive work on discrete logarithms on elliptic curves. However, this research is presented in detail in the book of Menezes [**25**]. We also do not discuss discrete logarithms on class groups of number fields, for which references can be found in [**4**]. There has been

1991 *Mathematics Subject Classification.* Primary 11Y16, 12E05; Secondary 11T71, 68P25, 94A60.

This paper is in final form and no version of it will be submitted for publication elsewhere

work on the degree to which individual bits of the discrete logarithm might be computable. For references for the most recent results, see [**15**].

All of the fast discrete logarithm algorithms in finite fields rely on finding elements that are smooth. This means that they can be expressed as products of other elements that in some sense are "small." For ordinary integers, smoothness means that the "small" elements are small primes. When the elements of the field are represented as polynomials over a small subfield, the "small" elements are irreducible polynomials of low degrees. In Section 3 we discuss the latest results on smoothness, especially of polynomials.

2. Discrete logarithms

In this section we consider discrete logarithms in finite fields $GF(q)$, where $q = p^n$ for p a prime. We will assume that g is a primitive element in $GF(q)$.

One can always use the Shanks baby steps – giant steps attack [**29**], which computes $\log_g y$ in roughly $r^{1/2}$ steps, where r is the largest prime factor of $q-1$. This algorithm is deterministic, but uses about $r^{1/2}$ space. Space requirements can be decreased at the cost of increasing the running time. There is also a randomized algorithm of Pollard [**30**] which has approximately the same running time of $r^{1/2}$, but uses practically no space. The lesson to be drawn from these algorithms is that to have a secure public key cryptosystem, one needs to choose the field $GF(q)$ carefully. If q is a prime of 512 or 1024 bits, this is not a major concern, as $q-1$ will have a large prime factor with high probability. However, if $q = 2^n$, then the safe choices for n are restricted [**29**].

The Shanks and Pollard attacks apply to all discrete logarithm problems in all groups. However, since the largest prime factor r of $q-1$ can be almost as large as q (say if $q = 2r+1$), these attacks are of exponential complexity in the worst case. However, in all finite fields we now have available subexponential algorithms. They are all based on the index-calculus idea, which was ascribed in [**29**] to Western and Miller, but actually goes back a few decades earlier to Kraitchik [**24**]. This idea is that if

$$\prod_{i=1}^{a} x_i = \prod_{j=1}^{b} y_j \tag{2.1}$$

holds for some elements x_i, y_j of $GF(q)^*$, then

$$\sum_{i=1}^{a} \log_g x_i \equiv \sum_{j=1}^{b} \log_g y_j \pmod{q-1} . \tag{2.2}$$

If we obtain many equations of the form (2.1) (with at least one of them involving an element z such as g, for which $\log_g z$ is known), and they do not involve too many x_i and y_j, then the system (2.2) can be solved. For fields $GF(p)$, p a prime, the simplest way to implement this idea is to take random integers a,

$1 \le a \le q-1$, compute $u \equiv g^a \pmod{p}$, $1 \le u \le p-1$, and check whether

$$u = \prod p_i \ , \tag{2.3}$$

where the p_i are all primes satisfying $p_i < B$ for some bound B. (When (2.3) holds, we say that u is smooth with smoothness bound B.) For most values of a, u will not satisfy (2.3), and so will be discarded. Since (2.3) is equivalent to

$$a \equiv \sum \log_g p_i \pmod{p-1} \ , \tag{2.4}$$

after collecting slightly more equations like (2.3) than than there are primes $< B$, we can expect to be able to solve the system of equations of the form (2.4) for the $\log_g p_i$. Individual discrete logarithms are then computed using the database of the $\log_g p_i$ in a separate phase that is similar to the procedure above.

When the field is $GF(2^n)$, and the elements are represented as polynomials in $GF(2)[x]$ modulo an irreducible polynomial $f(x)$ of degree n, the same general procedure as is outlined above can be applied. After choosing a random integer a, $1 \le a \le q-1 = 2^n - 1$, we compute the polynomial $u(x)$ such that $g^a \equiv u(x) \pmod{f(x)}$ over $GF(2)$, and check whether

$$u(x) = \prod f_i(x) \ , \tag{2.5}$$

where the $f_i(x) \in GF(2)[x]$ are irreducible over $GF(2)$ and have low degrees.

The outline above of how index-calculus algorithms operate in fields $GF(p)$ and $GF(2^n)$ gives the basic idea, but does not yield efficient algorithms as presented. However, it is important to remember that all index-calculus algorithms consist of three phases, just as outlined above. The first one is the generation of equations of the form (2.2). It is always the most time-consuming phase. (However, it can usually be done on a distributed network of workstations, since there is little need for communication, so huge computing power can be brought to bear on this problem, just as in integer factorization.) The second phase is the solution of those equations. This phase takes much less computing time than the first one, but it has to be done on a single processor (possibly a massively parallel one), and so causes some difficulty. Finally, once the equations are solved, the third phase computes individual logarithms using the data from the first two phases. We will not say anything about the third phase, since it is always considerably faster than the first one. (However, in the most recent and fastest algorithms, the third phase has been getting increasingly complicated.)

The sketch of the index-calculus approach presented above yields algorithms with running time bounded by

$$\exp(C(\log q)^{1/2}(\log\log q)^{1/2}) \tag{2.6}$$

for some constant $C > 0$, provided that either $q = p$, a prime, or $q = 2^n$ (the two cases that are still of greatest cryptographic significance). However, the simplest methods lead to large values of C, resulting in impractical algorithms.

Hence considerable effort had been expanded by many researchers (see [**24, 29**] for references) on trying to lower the value of C. For prime fields $GF(p)$, the best algorithms that were known until recently were the three presented in [**9**], which had running times

$$\exp((1+o(1))(\log p)^{1/2}(\log\log p)^{1/2}) \text{ as } p \to \infty\ . \tag{2.7}$$

On the other hand, for fields $GF(q)$ with $q = 2^n$, Coppersmith [**6**] invented an algorithm with running time bounded by

$$\exp((c+o(1))(\log q)^{1/3}(\log\log q)^{2/3}) \text{ as } q = 2^n \to \infty \tag{2.8}$$

for a certain constant $c = 1.5874\ldots$. (Actually, the $c+o(1)$ term can be replaced by $c(n)+o(1)$, where $c(n)$ is an explicit oscillating function of n that is bounded above by $1.5874\ldots$ and bounded away from 0.) This was a major breakthrough, and represented the first practical integer factorization or discrete logarithm algorithm that did not have a running time of the form (2.6). (Schnorr [**34**] had earlier proposed an integer factorization algorithm with a similar running time, but it applied only to factoring large blocks of consecutive integers, and was totally impractical for factoring individual integers.)

In the last few years there have been several important developments in discrete logarithm algorithm. The most important and mathematically the deepest has been the invention of the number field sieve integer factoring algorithm. Starting with an idea of Pollard (motivated, curiously enough, by a discrete logarithm method), this technique was developed by Buhler, H. Lenstra, Pomerance, and a large group of other researchers. We do not present any details nor do we try to give proper credits for all the work that was involved. The basic source for information about the number field sieve is the book [**21**] edited by A. Lenstra and H. Lenstra, which contains complete references and many of the papers related to the number field sieve. (However, developments in this area are rapid, and there are new papers circulating, such as [**27**], and new ones in preparation.) We only mention the most important facts about this subject. There are two basic forms of the number field sieve. The special number field sieve can factor integers q of special forms (such as $q = 2^n + 1$) in time of the form (2.8) with $c = 1.5262\ldots$. The general number field sieve factors arbitrary integers with running time bound of the form (2.8) with $c = 1.9229\ldots$. (There is also a variant due to Coppersmith [**7**] that achieves $c = 1.9018\ldots$, but it is probably not practical.) The special number field sieve has been used to factor integers of 523 bits in about 200 mips-years. The general number field sieve has not yet been used to obtain any factorization records, but it probably will become the method of choice for integers over 120 decimal digits (approximately 400 bits). The current record for factoring a generally hard integer is held by the multiple polynomial quadratic sieve with the two large primes variation [**22**], which has factored a 120 decimal digit integer in about 800 mips-years. There is a project going on right now to factor the original 1977 challenge integer of

Rivest, Shamir, and Adleman. This integer has 129 decimal digits, and the quadratic sieve is expected to factor it in about 6000 mips-years. Because most of this giant computation can be carried out using the idle time on thousands of workstations around the world, this project is likely to succeed within a few more months.

The number field sieve was extended to compute discrete logarithms in fields $GF(p)$ with p prime by Gordon [**13**]. He obtained a running time of the form (2.8) with $c = 2.0800\ldots$, and a lower value of c for some special primes p. Schirokauer [**32, 33**] has lowered this to $c = 1.9229\ldots$. No implementations of number field sieve discrete logarithm algorithms have been reported yet. It is not clear whether they will be practical in the near future, as they are more complicated than the integer factoring version of the number field sieve.

All the algorithms mentioned so far rely on unproved heuristic assumptions about their running times. Pomerance [**31**] and previous authors have presented probabilistic algorithms whose running time can be analyzed rigorously. For $q = p$, a prime, and $q = 2^n$, they have running times that are bounded by a function of the form (2.6) with high probability, but their constants C are considerably larger than for the heuristic algorithms.

For a long time little was known about the complexity of discrete logarithms in fields $GF(q)$, where $q = p^n$, and neither p nor n is small. The only result available were ElGamal's algorithms [**11, 12**] for $GF(p^n)$ for a fixed n. Also, Coppersmith's algorithm for $GF(2^n)$ can be generalized to $GF(p^n)$ with p growing slowly. Recently, Lovorn [**23, 2**] has obtained rigorous probabilistic algorithms for $GF(p^n)$ with $\log p \leq n^{0.98}$. Even more recently, Adleman and DeMarrais [**1**] have presented a heuristic algorithm that applies to all fields $GF(q)$ and has a running time bound of the form (2.6).

There is a very interesting new algorithm of Semaev [**36**]. It applies to some fields $GF(p^n)$ with $p \leq n^\alpha$ for any constant $\alpha > 0$, and achieves running time of the form (2.8). (This algorithm is heuristic, as are all the known fast algorithms.) It relies on some special algebraic relations, and therefore it requires that either (i) n is odd, $2n+1$ is prime, and the multiplicative order of p modulo $2n+1$ is n or $2n$ or else that (ii) $p^n - 1$ has a prime divisor ℓ such that $\ell \nmid p^k - 1$, $1 \leq k \leq n-1$. This algorithm has not been fully tested nor even analyzed, but it appears to be practical, and might open up new approaches for future improvements.

There have been relatively few implementations of discrete logarithm algorithms. For $GF(p)$, p a prime, the largest problem that has been solved fully is still that for the 192-bit prime used in the Sun NFS [**20**]. The problem for a 224-bit prime was almost completed as well [**20**]. These experiments were modest ones, and used on the order of 1 mips-year of computing time, as opposed to the hundreds or thousands of mips-years used in the modern factoring experiments mentioned earlier. Their main purpose was not so much to break the Sun NFS security system, but to test the methods of solving the large linear system of equations of the form (2.4) [**19**]. (For recent work on linear algebra approaches, see also [**8, 14, 16, 21**].) The computations of [**20**] also proved the feasibility of the algorithms in [**9**] and provide a base from which to extrapolate to obtain an estimate of what size primes can be handled with more computing time. The conclusions of [**20**] still appear valid, so that the 800 mips-years that was devoted to factoring a 400 bit integer should suffice (using the same distributed network of workstations, and with only slightly modified programs) to compute discrete logarithms modulo a prime of over 350 bits. The NIST decision in the revised draft of the proposed digital signature standard to allow for primes between 512 and 1024 bits appears prudent, as the 512 bit primes of the initial draft offer too little protection to guard against advances in either technology or algorithms.

The number field sieve discrete logarithm algorithm does not appear to be practical yet, but this may be due more to lack of research in this area than any inherent difficulties. Integer factorization algorithms have attracted much more attention, and the number field sieve factoring algorithms appear to be competitive with the best older algorithms around 400 bits. Therefore it seems prudent, in assessing security of discrete logarithm cryptographic schemes, to assume that the number field sieve will be effective in the computation of discrete logs as well. Some concern has been expressed about the possible use of trapdoor primes. These are primes that are constructed so that the special number field sieve can be applied to them, but only if one knows the secret details (which consist of a representation of the prime as the value of a polynomial with small coefficients at some integer). However, the advantages that a trapdoor prime designer can obtain over somebody who does not know the secret construction (and therefore presumably has to use the general number field sieve algorithm) is too small to worry about.

In fields $GF(2^n)$, the $GF(2^{127})$ discrete logarithm problem (which was implemented in some early cryptosystems) was solved by Coppersmith [**6**] using his algorithm, and subsequently by Blake et al. [**3**] using an older and less efficient method. The latest results in this area are those of Gordon and McCurley [**14**] who implemented the Coppersmith algorithm on a massively parallel processor. They have done most of the computations needed to compute discrete logarithms in $GF(2^{503})$, and project that with the next generation of parallel machines even $GF(2^{593})$ might be achievable.

3. Smooth polynomials

The analysis of the running times of all known index calculus algorithms relies on knowledge of the distribution of smooth integers, smooth ideals, or smooth polynomials. There is a huge literature on smooth integers, since they are of interest in many other applications besides discrete logarithm and factorization algorithms. A detailed survey of this area is presented in [**18**]. Distribution of smooth ideals is less well understood. For recent results, see [**5**]. To some extent problems about smooth ideals can be transformed into problems about smooth integers.

The distribution of smooth polynomials (over finite fields) is much easier to study than that of smooth integers. However, for a long time there was not much interest in this topic. If q is a prime power, let $N_q(n,m)$ be the number of monic polynomials of degree n over $GF(q)$, the finite field with q elements, all of whose irreducible factors have degrees $\leq m$. The author [**29**] obtained estimates of $N_2(n,m)$ for $n \to \infty$, $n^{1/100} \leq m \leq n^{99/100}$, by applying the saddle point method. This estimate was sharpened and extended to $N_q(n,m)$ for $q > 2$ by R. Lovorn [**23**]. More recently, K. Soundararajan [**37**] has obtained estimates of $N_q(n,m)$ that cover the full range of values of q, n, and m.

Let $I(k) = I_q(k)$ denote the number of monic polynomials of degree k over $GF(q)$ that are irreducible over $GF(q)$. Then it is known that

$$I(k) = \frac{1}{k} \sum_{d|k} \mu(d) q^{k/d} \ , \tag{3.1}$$

where $\mu(d)$ is the Möbius μ-function. Because of the uniqueness of factorization of a polynomial into irreducible polynomials, we have

$$N_q(n,m) = [z^n] f_m(z) \ , \tag{3.2}$$

where

$$f_m(z) = f_{q,m}(z) = \prod_{k=1}^{m} (1 - z^k)^{-I(k)} \ . \tag{3.3}$$

The reason that the enumeration of smooth polynomials is much easier than that of smooth integers is that $f_m(z)$ is much simpler and easier to analyze than the corresponding function for integers.

The function $f_m(z)$ is rational, but has to be treated in different ways for different ranges of values of m. It has poles at k-th roots of unity for all $k \leq m$, but by far the highest order pole is at $z = 1$, which is the dominant singularity. For m small we can expect the singularity at 1 to be small, and the partial fraction expansion to give good estimates. That is indeed the case, and it can be shown that for $m \leq (\log n)(\log \log n)^{-1}$, say, this method works. (However, Soundararajan [**37**] does not utilize this technique, and instead estimates $N_q(n,m)$ by working with recurrences for this function.) For higher values of m, the singularity at $z = 1$ becomes large, and the saddle point method is used to estimate $N_q(n,m)$. An interesting feature of this analysis is that the ranges of values of m where the partial fraction and saddle point methods apply overlap, so there is no gap in the estimates.

The saddle point method stops working when m gets large. The problem is that for n fixed and m increasing, the saddle point decreases away from 1 towards

$1/q$. Now for any fixed z,

$$\lim_{m\to\infty} f_m(z) = \prod_{k=1}^{\infty}(1-z^k)^{-I(k)} = \sum_{n=0}^{\infty} q^n z^n = (1-zq)^{-1} \ , \tag{3.4}$$

and the limiting function has a first order pole at $z = 1/q$, and thus has a small singularity. Therefore for m large, say $m > n(\log\log n)(\log n)^{-1}$, Soundararajan estimates $N_q(n,m)$ via recurrences.

The results of [**37**] imply that if q is fixed, and we let $r_0 = r_0(n,m)$ be the unique solution in $(0,1)$ to the equation

$$r_0 \, \frac{f_m'}{f_m}(r_0) = n \ , \tag{3.5}$$

and let

$$b(r_0) = \left(\frac{f_m'}{f_m}(r)\right)'\Bigg|_{r=r_0} \ , \tag{3.6}$$

then for $(\log n)(\log\log n)^{-1} \le m \le 3n(\log\log m)(\log n)^{-1}$,

$$N_q(n,m) = (1+o(1))\frac{f_m(r_0)r_0^{-n}}{\sqrt{2\pi b(r_0)}} \ \text{as} \ n\to\infty \ . \tag{3.7}$$

If $1 \le m \le (\log n)(\log\log n)^{-1}$, then

$$N_q(n,m) = (1+o(1))\frac{n^{\sum\limits_{k\le m} I(k)}}{\left(\sum\limits_{k\le m} I(k)\right)!}\prod_{k\le m} k^{-I(k)} \ \text{as} \ n\to\infty \ . \tag{3.8}$$

Finally, if $3n(\log\log n)(\log n)^{-1} \le m \le n$, then

$$N_q(n,m) = (1+o(1))q^n\rho(n/m) \ \text{as} \ n\to\infty \ , \tag{3.9}$$

where $\rho(x)$ is the de Bruijn-Dickman function. ■

References

1. L. M. Adleman and J. DeMarrais, A. subexponential algorithm for discrete logarithms over all finite fields, *Math. Comp. 61* (1993), 1–16.
2. R. Lovorn Bender, Rigorous, subexponential algorithms for discrete logarithms in $GF(p^2)$, to be published.
3. I. F. Blake, R. Fuji-Hara, R. C. Mullin, and S. A. Vanstone, Computing logarithms in fields of characteristic two, *SIAM J. Alg. Disc. Methods* **5** (1984), 276–285.
4. J. A. Buchmann and S. Düllmann, On the computation of discrete logarithms in class groups, pp. 134–139 in *Advances in Cryptology — CRYPTO '90*, A. J. Menezes and S. A. Vanstone, eds., Lecture Notes in Computer Sic. #537, Springer, 1991.
5. J. A. Buchmann and C. S. Hollinger, On smooth ideals in number fields, to be published.
6. D. Coppersmith, Fast evaluation of discrete logarithms in fields of characteristic two, *IEEE Transactions on Information Theory* **30** (1984), 587–594.
7. D. Coppersmith, Modifications to the number field sieve, *J. Cryptology 6* (1993), 169–180.

8. D. Coppersmith, Solving homogeneous linear equations over $GF(2)$ via a block Wiedemann algorithm, *Math. Comp. 62* (1994), to appear.
9. D. Coppersmith, A. Odlyzko, and R. Schroeppel, Discrete logarithms in $GF(p)$, *Algorithmica* **1** (1986), 1–15.
10. T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory 31* (1985), 469–472.
11. T. ElGamal, A subexponential-time algorithm for computing discrete logarithms over $GF(p^2)$, *IEEE Trans. Inform. Theory 31* (1985), 473–481.
12. T. ElGamal, On computing logarithms over finite fields, pp. 396–402 in *Advances in Cryptology — CRYPTO 85*, Lecture Notes in Computer Science #218, Springer, 1986.
13. D. M. Gordon, Discrete logarithms in $GF(p)$ using the number field sieve, *SIAM J. Discr. Math. 6* (1993), 124–138.
14. D. M Gordon and K. S. McCurley, Massively parallel computation of discrete logarithms, in *Advances in Cryptology — CRYPTO 92*, Lecture Notes in Computer Science, Springer, to appear.
15. J. Håstad, A. W. Schrift, and A. Shamir, The discrete logarithm modulo a composite hides $O(n)$ bits, *J. Comp. Sys. Sci. 47* (1993), 376–404.
16. E. Kaltofen, Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems, to be published.
17. N. Koblitz, Elliptic curve cryptosystems, *Math. Comp.* **48** (1987), 203–209.
18. A. Hildebrand and G. Tenenbaum, Integers without large prime factors, to be published.
19. B. A. LaMacchia and A. M. Odlyzko, Solving large sparse linear systems over finite fields, pp. 109–133 in *Advances in Cryptology: Proceedings of Crypto '90*, A. Menezes, S. Vanstone, eds., *Lecture Notes in Computer Science #537*, Springer, 1991.
20. B. A. LaMacchia and A. M. Odlyzko, Computation of discrete logarithms in prime fields, *Designs, Codes, and Cryptography 1* (1991), 46–62.
21. A. K. Lenstra and H. W. Lenstra, eds., *The Development of the Number Field Sieve*, Lecture Notes in Mathematics #1554, Springer, 1993.
22. A. K. Lenstra and M. S. Manasse, Factoring with two large primes, *Advances in Cryptology: Proceedings of Eurocrypt '90* (I. Damgard, ed.), *Lecture Notes in Computer Science*, New York: Springer-Verlag.
23. R. Lovorn, Rigorous, subexponential algorithms for discrete logarithms over finite fields, Ph.D. thesis, Univ. of Georgia, May 1992.
24. K. S. McCurley, The discrete logarithm problem, pp. 49–74 in *Cryptography and Computational Number Theory*, C. Pomerance, ed., *Proc. Symp. Appl. Math.* **42**, Amer. Math. Soc., 1990, pp. 49–74.
25. A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer, 1993.
26. V. Miller, Use of elliptic curves in cryptography, *Advances in Cryptology: Proceedings of Crypto '85*, (H. C. Williams, ed.), *Lecture Notes in Computer Science* **218** (1986), New York: Springer-Verlag.
27. P. L. Montgomery, Square roots of products of algebraic numbers, to be published.
28. National Institute for Standards and Technology, A proposed federal information processing standard for digital signature standard (DSS), draft, August 1991.
29. A. M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, pp. 224–314 in *Advances in Cryptology: Proceedings of Eurocrypt '84*, (T. Beth, N. Cot, I. Ingemarsson, eds.), *Lecture Notes in Computer Science* **209** Springer-Verlag, 1985.
30. J. M. Pollard, Monte Carlo methods for index computations mod p, *Math. Comp. 32* (1978), 918–924.
31. C. Pomerance, Fast, rigorous factorization and discrete logarithm algorithms, pp. 119–143 in *Discrete Algorithms and Complexity*, D. S. Johnson, T. Nishizaki, A. Nozaki, H. W. Wilf, eds., Academic Press, 1987.
32. O. Schirokauer, On pro-finite groups and on discrete logarithms, Ph.D. thesis, Univ. of California, Berkeley, May 1992.
33. O. Schirokauer, Discrete logarithms and local units, *Proc. Royal Soc. London*, to appear.
34. C. P. Schnorr, Refined analysis and improvements on some factoring algorithms, *J. Algo-*

rithms 3 (1982), 101–127.
35. C. P. Schnorr, Efficient signature generation by smart cards, *J. Cryptology 4* (1991), 161–174.
36. I. A. Semaev, An algorithm for evaluation of discrete logarithms in some nonprime finite fields, to be published.
37. K. Soundararajan, Asymptotic formulae for the counting function of smooth polynomials, in preparation.
38. P. Van Oorschot, A comparison of practical public key cryptosystems based on integer factorization and discrete logarithms, pp. 280–322 in *Contemporary Cryptology: The Science of Information Integrity*, G. J. Simmons, ed., IEEE Proc., 1991.

AT&T Bell Laboratories, Murray Hill, New Jersey 07974
E-mail address: amo@research.att.com