The Role of Smooth Numbers in Number Theoretic Algorithms

CARL POMERANCE*

The University of Georgia
Athens, Georgia 30602, USA

1 Introduction

A *smooth number* is a number with only small prime factors. In particular, a positive integer is *y-smooth* if it has no prime factor exceeding y. Smooth numbers are a useful tool in number theory because they not only have a simple multiplicative structure, but are also fairly numerous. These twin properties of smooth numbers are the main reason they play a key role in almost every modern integer factorization algorithm. Smooth numbers play a similar essential role in discrete logarithm algorithms (methods to represent some group element as a power of another), and a lesser, but still important, role in primality tests.

In this article we shall survey some of the more interesting theoretical and practical algorithms for factoring, computing discrete logarithms, and primality testing, and will especially highlight the central role of smooth numbers in the subject.

2 A "fundamental lemma"

We begin with a problem that does not appear to have anything to do with our main topic. We shall first see that smooth numbers play an essential role in both the theoretical and algorithmic solution of the problem. We next shall show how the problem is the key ingredient in a robust class of factoring algorithms.

Suppose we choose integers independently and with uniform distribution in the interval $[1, x]$. How many should we choose so that almost surely some nonempty subset of our choices will have a product that is a square? The answer depends on the function $\exp(\sqrt{\ln x \ln \ln x})$, which we shall abbreviate as $L(x)$.

Lemma 2.1. *Let ε be an arbitrarily small positive number. If we choose $L(x)^{\sqrt{2}+\varepsilon}$ integers from $[1, x]$ (independently and with uniform distribution), then as $x \to \infty$, the probability tends to 1 that some nonempty subset product is a square, whereas if we choose $L(x)^{\sqrt{2}-\varepsilon}$ integers, the probability tends to 0.*

A proof of the first statement, which is considerably easier than a proof of the second, is implicit in [BLP, Theorem 10.1], and explicit in [P4, Proposition 4.1].

*) Supported in part by a grant from the National Science Foundation.

A proof of the entire result will be given in a forthcoming paper of the author. For our purposes it will be interesting to see why Lemma 2.1 is true. In addition, there is an algorithmic problem implicit in Lemma 2.1. Namely, if you are actually choosing the random integers and want to explicitly find a subset product that is a square, what is an efficient way to do this?

If we choose a number $n \in [1, x]$ with a large prime factor p, then it is unlikely that p^2 divides n and it will probably be a long wait before we ever see another number $m \in [1, x]$ divisible by p. Thus, it is unlikely that we can use n in a subset product that is a square. That is, the numbers that we can potentially use in the subset product are smooth numbers. Let y be some positive number, which we shall specify shortly. If we have chosen more y-smooth numbers than there are primes up to y, then some nonempty subset of these numbers has a product that is a square. This follows from a simple linear algebra argument. Each y-smooth integer n has an *exponent vector* $\mathbf{v}(n)$ of length the number of primes up to y. Indeed, if $p \le y$ is prime, then the coordinate in $\mathbf{v}(n)$ corresponding to p is the exponent on p in the prime factorization of n. Let $\pi(y)$ denote the number of primes up to y. So if we have more than $\pi(y)$ of these exponent vectors, they must be linearly dependent. In particular, they are dependent over the field $\mathbf{F}_2$ of two elements. A dependency here is represented as a nonempty subset sum being the 0-vector, which corresponds exactly to the corresponding subset product being a square.

Let $\psi(x, y)$ denote the number of y-smooth integers up to x. Then the expected number of choices of random integers in $[1, x]$ to find one y-smooth number is $x/\psi(x, y)$, so that the expected number of choices to find $\pi(y)+1$ such y-smooths is $x(\pi(y) + 1)/\psi(x, y)$. We thus wish to choose y as a function of x so as to minimize this expression. It turns out that the optimal value of y is about $L(x)^{\sqrt{2}/2}$, and the resulting expected number is about $L(x)^{\sqrt{2}}$. This is how the upper bound in Lemma 2.1 is shown.

This proof sketch also serves to suggest how the algorithmic problem implicit in Lemma 2.1 may be efficiently solved. Namely, for $y = L(x)^{\sqrt{2}/2}$, test each choice of a number $n \in [1, x]$ to see if it is y-smooth, discarding those that are not. When $\pi(y) + 1$ successes n have been found, create the exponent vectors $\mathbf{v}(n)$, and with Gaussian elimination over $\mathbf{F}_2$, find a subset product that is a square.

The lower bound in Lemma 2.1 shows us that we cannot do substantially better; that is, smooth numbers are essentially forced upon us. The proof is tricky, but not especially deep, the idea being that for each fixed k, almost surely a subset whose product is a square will contain a number with kth largest prime factor maximal over the subset and k other numbers in the subset, each divisible by a different one of these k large primes. Further, a calculation shows that it is unlikely that we will see such a $(k + 1)$-tuple if we only choose $L(x)^{\sqrt{2k/(k+1)-\varepsilon}}$ numbers. Because k may be taken arbitrarily large, we get our result.

The upper and lower bound calculations in Lemma 2.1 depend on an estimate for the number $\psi(x, y)$ of y-smooths up to x: for any fixed, positive real number a, we have $\psi(x, L(x)^a) = xL(x)^{-1/(2a)+o(1)}$ as $x \to \infty$, see [CEP]. For more on the distribution and application of smooth numbers, see [HT] and [V].

3 Combinations of congruences

In this section we shall discuss the connection between Lemma 2.1 and factoring. An old factoring method due to Fermat is to represent the number to be factored as the difference of two squares. For example, one can verify mentally that $8051 = 90^2 - 7^2$, so that $8051 = 83 \times 97$. The problem with this method is that it is often very difficult to find two squares that work.

Instead, one may search for two squares whose difference is a multiple of the number to be factored. If $u^2 \equiv v^2 \bmod n$ and $u \not\equiv \pm v \bmod n$, then the greatest common factor of $u - v$ and n, which may be computed rapidly via Euclid's algorithm, is a nontrivial factor of n.

Assume n is an odd composite that is not a power. Suppose we choose a random integer $A \in [1, n]$ and compute the least positive residue $Q \equiv A^2 \bmod n$. Then Q is "close" to being a random integer in $[1, n]$. If Lemma 2.1 is applicable, we would expect to find a set of such numbers Q, with their product a square, after taking about $L(n)^{\sqrt{2}}$ values of A. Multiplying the corresponding congruences $Q \equiv A^2 \bmod n$ would thus give rise to a congruence of the shape $u^2 \equiv v^2 \bmod n$, from which we would have a chance of factoring n. (It is not certain that such a congruence could split n because it may be that $u \equiv \pm v \bmod n$.)

I have just described the "random squares" factorization method of Dixon. It can be proved that the likelihood of Q being smooth is about the same as a uniformly distributed random integer in $[1, n]$, so that this step in the above heuristic method can be made rigorous. It is also not hard to show that the final congruence is nontrivial with probability at least $1/2$, so the random squares method is a completely rigorous probabilistic factoring algorithm. With special subroutines to determine if the numbers Q are smooth and to do the linear algebra over $\mathbf{F}_2$, the expected running time for the random squares method is $L(n)^{\sqrt{2}+o(1)}$, see [P3].

A simple, but crucial fact about smooth numbers is that large numbers are less likely to be y-smooth than small numbers. In the random squares method we are presented with a stream of random auxiliary numbers Q that we examine for smoothness, discarding the majority that are not, and stopping when we have found sufficiently many that are smooth. If we could alter the stream so that the numbers Q are smaller, then each would have a better chance of being smooth, and we would not have to examine so many.

One simple way to make Q smaller is to replace it with $Q - n$ if it exceeds $n/2$; that is, to use the residue closest to 0 rather than the least positive residue. To make a square, we now would also have to worry about the sign of the product, but this can be easily handled by adding one extra coordinate to the exponent vectors $\mathbf{v}(Q)$ to represent the sign of Q. However, reducing the size from n to $n/2$ is not sufficient to substantially affect the complexity estimate.

If A_i/B_i is the ith convergent in the continued fraction expansion of $\sqrt{n}$, then the residue Q_i of $A_i^2 \bmod n$ that is closest to 0 satisfies $|Q_i| < 2\sqrt{n}$. Further, the numbers A_i, Q_i are easy to find by a simple recursive procedure. If it could be shown that the numbers Q_i are sufficiently random, then indeed we would have a significant improvement over the random squares method, with a complexity of $L(n)^{1+o(1)}$.

In some cases it can be shown that the numbers Q_i are definitely *not* sufficiently random. For example, if the period of the (periodic) continued fraction for $\sqrt{n}$ is too short, then the pairs A_i, Q_i may begin repeating before we have found enough smooth values of Q_i. However, for most numbers n this phenomenon does not occur, and even when it does, looking at the continued fraction for $\sqrt{an}$ for a small integer a seems to solve the problem.

This method, due to Brillhart and Morrison [MB], is known as the continued fraction method. It completely majorizes the random squares method. However, no one has *proved* that it is likely to work. Of course, the number n we are trying to factor does not *know* this! The continued fraction algorithm, like all modern, practical factoring algorithms, does not have a rigorous complexity analysis. However, heuristic analyses help us to compare various methods, and to see which may be worthy of further tinkering.

The fastest factoring algorithm that has been rigorously analyzed is the class group method (see [LP]). This method uses the group of primitive binary quadratic forms with discriminant either $-n$, or a small multiple of $-n$. Smooth numbers play a key role here as well. The algorithm generates random forms (Q, R, S) in the class group by looking at random words on a small generating set. Corresponding to the prime factorization of Q, we get a factorization of (Q, R, S) into corresponding "prime forms". By accepting only those cases where Q is smooth, we collect relations between the generating forms and the prime forms. When enough such relations are collected, we can use them, again via a linear algebra step over $\mathbf{F}_2$, to construct an "ambiguous form", namely one whose square is the identity. From Gauss, we may use such forms to factor the discriminant, which is exactly what we wish to do. The expected running time of this algorithm is $L(n)^{1+o(1)}$, which is the same as for the simpler continued fraction method discussed above. In contrast though, the class group method is rigorous. It is surely not practical, being majorized in practice by other methods that will be discussed below.

4 Smoothness tests

When presented with an integer $n \le x$, how long does it take to determine if n is y-smooth? If one uses trial division with the primes up to y, it takes about $\pi(y)$ steps to determine if most numbers are y-smooth. In factoring algorithms such as the ones above, the overwhelming majority of the auxiliary numbers Q presented are not y-smooth. If it takes us $\pi(y)$ steps per candidate to discover if a number is y-smooth, then this step of the algorithm dominates all others. It is greatly in our interest to find a smoothness test faster than trial division.

In [P1] an "early abort strategy" is described, which suggests that one give up on the trial division of a particular candidate Q if at various strategic points below y not enough of Q has been partially factored. This method loses some y-smooth numbers, but not many. The average time per number is only about $\sqrt{y}$. In addition, trial division may be replaced with a fast Fourier transform method of Pollard and Strassen (see [P4]), which further reduces the amortized time per candidate to about $y^{1/4}$.

The elliptic curve factoring method of Lenstra [L1], [L2] (see Section 6 below) has the feature that its expected running time to completely factor a number is a small function of the second largest prime factor of the number. In particular, it recognizes y-smooth numbers below x in $O(y^\varepsilon \ln x)$ steps, for any $\varepsilon > 0$. It is with this subroutine as a smoothness test that the complexity estimates of the last section are achieved.

A few comments are in order. The elliptic curve method is not completely rigorous. However, it is possible to show that *most* smooth numbers will be factored quickly with the method. Thus, as with the early abort strategy above, it is not crucial that a few y-smooths may pass unrecognized. Nevertheless, it is of theoretical interest if a smoothness test can be devised that recognizes y-smooth numbers in time about y^ε and that has no exceptions. This is provided in a recent method that is similar to the elliptic curve method, but uses Jacobian varieties of hyperelliptic curves of genus 2 (see [LPP]).

Is it ever practical to use the elliptic curve method as a subroutine to recognize smooth auxiliary numbers? We know of no case where it is. This is largely due to the existence of a far better smoothness test that is applicable when the stream of auxiliary numbers presented happens to be the consecutive values of a polynomial with integer coefficients.

Everyone knows the sieve of Eratosthenes as an efficient method of finding all of the primes up to some point. However, this sieve can also be used to find y-smooth numbers. One sieves with the primes up to y (and possibly their powers), and instead of crossing off the multiples of each prime, one keeps a tally at each number of how frequently it has been "hit". This tally may be done by adding the (single precision) logarithms of the primes that hit, and if the sum exceeds a threshold, the number is reported as being y-smooth.

What makes this sieve work is that the multiples of a given prime p occur in a regular pattern, namely an arithmetic progression of difference p. If instead of the consecutive integers to some point, one has the image of this interval under a polynomial with integer coefficients, one still has regularity for the multiples of p. They now form the union of several arithmetic progressions of difference p, and we may sieve just as efficiently as before. For example, no value of $t^2 + 1$ is divisible by 3, and the multiples of 5 are found in the progressions $t \equiv \pm 2 \bmod 5$.

However, the streams of auxiliary numbers Q described in the previous section are not the consecutive values of a polynomial, and there is no discernible regularity to where the multiples of a given prime p appear. In the next section we shall describe two algorithms that can make use of sieving as a smoothness test.

5 The quadratic sieve and the number field sieve

Say we wish to factor the odd number n, which has been already verified to be composite and not a power. Consider the quadratic polynomial $Q(t) = t^2 - n$. For ε small and $|t - \sqrt{n}| < n^\varepsilon$, we have $|Q(t)| < 3n^{1/2+\varepsilon}$. Thus, the values of $Q(t)$ for t close to $\sqrt{n}$ are relatively small. If it could be assumed that the values of $Q(t)$ for t in this range are about as likely to be smooth as random integers of the same size, then Lemma 2.1 suggests that with $x = 3n^{1/2+\varepsilon}$, before $L(x)^{\sqrt{2}+\varepsilon} < L(n)^{1+2\varepsilon}$ values of t are examined, there will be a nonempty subset such that the product of the corresponding values of $Q(t)$ is a square, say u^2. If v is the product of these values of t, then because $Q(t) \equiv t^2 \bmod n$, we have $u^2 \equiv v^2 \bmod n$. So, as before, the greatest common factor of $u - v$ and n may provide a nontrivial factor of n. This algorithm then is exactly the same as the random squares method and the continued fraction method discussed in Section 3, but now the stream of auxiliary numbers $Q(t)$ are the consecutive values of a polynomial with integer coefficients, so that we may use a sieve as a smoothness test.

This is the basic quadratic sieve method (see [P1]). Though the values $Q(t)$ are slightly larger than the auxiliary numbers Q in the continued fraction algorithm, sieving is so good a smoothness test that this small defect is not important. When the quadratic sieve method is used today, we do not use only one polynomial, but a family of many polynomials of the form $at^2 + 2bt + c$, where a, b, c are chosen in a certain range and with $b^2 - ac = n$. This idea of Montgomery (see [P2]) mitigates somewhat the growth of the size of the polynomial values, for when the values of one polynomial become large, we switch to another. The multiple polynomial quadratic sieve currently enjoys the record for the factorization of the largest number of no special form and without small prime factors that has ever been factored. This number is the 129-digit composite announced as a cryptographic challenge in Martin Gardner's *Scientific American* column in 1977. It was factored in 1994 by D. Atkins, M. Graff, A. Lenstra, P. Leyland, and a host of others who volunteered time on their workstations.

A word must be said about the linear algebra subroutines used to assemble the congruences into congruent squares at the final stage of the algorithm. To achieve the complexity estimate $L(n)^{1+o(1)}$ for the quadratic sieve (and the earlier algorithms mentioned), one cannot use Gaussian elimination as the linear algebra subroutine. Instead, there are methods due to Wiedemann, Lanczos, and others that are used. These methods exploit the facts that the matrix of exponent vectors is sparse, and that the algebra is done over the field $\mathbf{F}_2$ of two elements. In practice so far, we have largely been able to get by with Gaussian elimination and variations of it. Although it is easy to distribute the sieving stage of the algorithm to many unextraordinary computers, so far it is awkward to do this with the linear algebra stage, and for the record numbers factored these days, supercomputers are used for the matrix. Clearly more work is needed for this step of combination of congruences algorithms. For more on this subject, see [C], [LO], [M], [PS].

The reader may have noticed that many factoring algorithms seem to end up with either rigorous or heuristic complexity of $L(n)^{1+o(1)}$. This is due to the auxiliary numbers that we examine for smoothness, which in the algorithms we have described so far (except for the random squares method) are all bounded by the common expression $n^{1/2+o(1)}$. If we could reduce the size of these numbers that we hope to find smooth, we could reduce the complexity of the algorithm. The *number field sieve* allows us to do just this. In this algorithm the auxiliary numbers are about $\exp(c(\ln n)^{2/3}(\ln\ln n)^{1/3})$. Putting this bound in for x in Lemma 2.1 suggests that the complexity of the number field sieve is about $\exp((4c/3)^{1/2}(\ln n)^{1/3}(\ln\ln n)^{2/3})$. In the original version of the number field sieve, invented by Pollard in the late 1980's, only numbers near a high power are factored, and in this case, the number c turns out to be $(16/3)^{1/3}$. This method was later generalized to arbitrary numbers n by Buhler, H. Lenstra, and the author, but at the cost of increasing the number c to $(64/3)^{1/3}$.

In sum, the number field sieve is asymptotically fast because it achieves a dramatic reduction in the number of digits of the auxiliary numbers: they have about the 2/3 *power* of the number of digits of n, as opposed to about half the number of digits of n in the quadratic sieve. How then does the number field sieve work?

Suppose f, g are irreducible, monic polynomials over $\mathbf{Z}$ for which there is an integer m with $f(m) \equiv g(m) \equiv 0 \bmod n$. Say α, β are complex numbers with $f(\alpha) = g(\beta) = 0$. Consider the substitution homomorphisms $\phi : \mathbf{Z}[\alpha] \to \mathbf{Z}/(n)$, $\psi : \mathbf{Z}[\beta] \to \mathbf{Z}/(n)$, where $\phi(\alpha) = \psi(\beta) = m + (n)$. Thus, for any integers a, b we have the congruence $\phi(a + b\alpha) \equiv \psi(a + b\beta) \bmod n$. It is via this family of congruences as a, b vary over small (coprime) integers that we hope to assemble our congruent squares. But we actually will construct the squares in the rings $\mathbf{Z}[\alpha]$, $\mathbf{Z}[\beta]$, which is no loss because the homomorphic image of a square is a square. Not only is there no loss, there can be a substantial gain.

We define a member of $\mathbf{Z}[\alpha]$ to be smooth if its norm to $\mathbf{Z}$ is smooth. However, the norm function masks the proliferation of prime ideals that may lie over a rational prime. Taking this into account, and adding some extra information afforded by a few random quadratic characters (to get over the obstructions of possibly complicated class groups, unit groups, and quotient groups of $\mathbf{Z}[\alpha]$, $\mathbf{Z}[\beta]$ in their maximal orders), our above method using exponent vectors allows one to construct squares. Finding the square roots of these squares is not as simple as before, but it is a tractable problem. The auxiliary numbers we wish to find smooth are the products of the norms to $\mathbf{Z}$ of $a + b\alpha$ and $a + b\beta$, where a, b run over small coprime integers. This is a polynomial in a and b, and we may use a sieve as a smoothness test. The size of these auxiliary numbers depends on the largest coefficients of f and g and their degrees.

One way to construct the polynomials f and g is to first pick d, the degree of f, next pick $m = [n^{1/d}]$, and write n in the base m, so that $n = m^d + c_{d-1}m^{d-1} + \cdots + c_0$. Then we let $f(t) = t^d + c_{d-1}t^{d-1} + \cdots + c_0$ and $g(t) = t - m$. There are other strategies too, and in particular it is not essential that the polynomials be monic. For more on the number field sieve see [LL] and [P4].

The largest number of no special form that the number field sieve has factored has 119 digits, a recent accomplishment of Contini, Dodson, A. Lenstra, and Montgomery. It is likely though that this will change soon. The very favorable heuristic complexity estimate has concentrated much attention on the number field sieve, people are beginning to find the improvements necessary to make it a practical algorithm, and it is thought that before long it will replace the quadratic sieve as the champion method for numbers of no special form.

6 The elliptic curve method

The elliptic curve method of H. Lenstra uses smooth numbers in an intrinsically different way than the previous factorization methods discussed. Based on a beautiful method of Pollard to discover those prime factors p in a number for which $p - 1$ is smooth, it makes use of the following observation. If G is a finite group (written additively), then there is a simple algorithm to test if the order of an element $g \in G$ is a y-smooth number below x. Indeed, let M be the least common multiple of the y-smooth numbers below x and form Mg. This calculation can be done in $O(\pi(y)\ln x)$ group operations by the repeated doubling method. Then the order of g is a y-smooth number below x if and only if Mg is the identity.

This observation is used as follows. Suppose $p < q$ are prime factors of the number n we wish to split. Let a, b be integers with $4a^3 + 27b^2$ coprime to n and let $P = (x_0, y_0)$ be an integer point on the elliptic curve $E: \ y^2 = x^3 + ax + b$. Let $E(p)$, $E(q)$ be the elliptic curve groups mod p and mod q, respectively. If P mod p has y-smooth order in $E(p)$, but P mod q does not have y-smooth order in $E(q)$, then we can use this to split n. Indeed, let M be the least common multiple of the y-smooth numbers below $(n^{1/4}+1)^2$. We cannot directly work in the groups $E(p)$ and $E(q)$ because we do not know p and q. However, we can try to add points on E modulo n. If the addition law breaks down it is because we are trying to invert a nonzero, noninvertible residue modulo n. But Euclid's algorithm, which is used for inversion, would in this case split n. The addition law breaks down when we try to add two points R, S such that $R+S$ is the identity modulo some factor of n, but not the identity modulo some other factor of n. This is exactly what happens when we try to compute MP modulo n, because this is the identity in $E(p)$ and it is not the identity in $E(q)$.

We can attempt to do this calculation even if we do not know beforehand that P has y-smooth order in $E(p)$, but not in $E(q)$. If it works we have split n. If it does not work, we have the option of trying again with a larger value of y, or more interestingly, trying again with another triple a, b, P. We can easily generate many such triples by choosing a, x_0, y_0 at random, and solving for b.

This then is the elliptic curve method. If the prime p has sufficiently many smooth numbers near it in the "Hasse interval" $((\sqrt{p}-1)^2, (\sqrt{p}+1)^2)$, then it can be shown rigorously that the method is expected to find p as a prime factor of numbers n divisible by p. It is conjectured that this interval does contain enough smooth numbers, but it has not been proved. It is interesting that in the longer interval $((\sqrt{p}-1)^4, (\sqrt{p}+1)^4)$, we can prove that there are many smooth numbers, which is why the *hyperelliptic* curve method can be rigorously analyzed — see [LPP].

An important contrast between the elliptic curve method and combination of congruences methods, is that in the latter we need to be able to find many smooth numbers for success, but each auxiliary number is quickly dealt with. In the elliptic curve method we are successful if just one auxiliary number (which is hidden from us) is smooth, but it takes a fair amount of time for each trial. The two opposite effects balance out. In the worst case the number n has its least prime factor near $\sqrt{n}$, and so the numbers we hope to find smooth are also near $\sqrt{n}$. So in the worst case, the elliptic curve method takes $L(n)^{1+o(1)}$ steps. However, most numbers are not in the worst case, so that the elliptic curve method can be considerably faster. Thus, when presented with a number to factor, one usually tries the elliptic curve method before attempting the quadratic sieve or the number field sieve.

7 Discrete logarithms and the search for smoothness

Given a cyclic group $G = \langle g \rangle$ (written multiplicatively), and an element h in G, the discrete logarithm problem is to find an integer n with $g^n = h$. In this problem the representation of the group G is of paramount importance. For example, suppose p is a prime. Then $(\mathbf{Z}/(p))^*$ is a cyclic group of order $p-1$, as is the additive group $\mathbf{Z}/(p-1)$. However, solving the discrete logarithm problem in the latter group is a triviality — one uses Euclid's algorithm to solve a linear congruence. But the discrete logarithm problem for the former group is hard, or at least apparently so.

One can find discrete logarithms in the group $(\mathbf{Z}/(p))^*$ by an algorithm similar to the random squares method discussed in Section 3. With g the cyclic generator on which logs are based, consider random powers g^m. Elements of the group are of course residue classes; say we represent these residue classes by their least positive member. That is, we represent group elements by positive integers less than p. If g^m is represented by a smooth integer, we keep it, and otherwise, we discard it. If we can find sufficiently many independent "relations", where a power of g is congruent mod p to a y-smooth number, we can use linear algebra (over the ring $\mathbf{Z}/(p-1)$) to solve for the logs of the primes q up to y. Once this pre-calculation is done, it is now fairly simple to find the log of an element h. Namely, consider $g^m h$, where again m is a random integer. If this is represented by a y-smooth number, say $\prod q_i^{a_i}$, where the q_i's run over the primes up to y, then $\log_g h$ is $-m + \sum a_i \log_g q_i$. To minimize the expected running time we take $y = L(p)^{\sqrt{2}/2}$. Then the running time of the first phase of this algorithm (to compute the logs of all of the primes up to y) is about $L(p)^{\sqrt{2}}$ and the running time to compute an individual log is about $L(p)^{\sqrt{2}/2}$. See [P3] for more details.

Can these ideas be generalized to the multiplicative group of a finite field $\mathbf{F}_q$? In particular, what would it mean to call a member of $\mathbf{F}_q^*$ "smooth"? If $q = p^k$, where k is large, then the usual representation of $\mathbf{F}_q$ is $\mathbf{F}_p[x]/(f)$, where f is an irreducible polynomial in $\mathbf{F}_p[x]$ of degree k. We may represent a group element as the member of the residue class of least degree. Because $\mathbf{F}_p[x]$, like $\mathbf{Z}$, is a Euclidean domain, we may give a definition of a smooth element. Say a polynomial is smooth if it factors completely into low degree irreducibles. There is a theory of the distribution of smooth polynomials in $\mathbf{F}_p[x]$ that is analogous to the distribution of smooth integers — see [Lo1], [O], [So]. We thus obtain a rigorous discrete logarithm algorithm analogous to the one above.

When $q = p^k$ with $k > 1$ and k small, the above representation of $\mathbf{F}_q^*$ is not particularly useful for computing discrete logarithms. Indeed, say $k = 2$. Then every residue class representative has degree 0 or degree 1, and so everything is smooth. Instead, we represent the field as $\mathcal{O}_K/(p)$, where K is an algebraic number field of degree k over the rationals and for which the prime p remains inert. In this case we call a field element smooth, if a canonical representative of the residue class has smooth norm. A problem is how to define a canonical representative. This is solved in the case $k = 2$ in [Lo2] where a rigorous algorithm is described.

Although we have not found a way to use something resembling the elliptic curve method or the quadratic sieve to compute discrete logarithms in $\mathbf{F}_q^*$, we have found a way to use analogs of the number field sieve — see [A], [G], [S]. As with factoring, the analysis is heuristic. Whether these algorithms are practical is unclear.

There are of course many other groups around. For example, one may consider a prime p for which the elliptic curve group $E(p)$ (see Section 6) is cyclic. Does it make sense to say an element of $E(p)$ is smooth? No one has thought of a way to make sense of this (except for some very special cases), and for this reason, we know of no fast ways to compute discrete logs in elliptic curve groups. These groups have been proposed as vehicles for public key cryptography precisely because we have no notion of smoothness for them.

8 Smooth numbers and primality testing

The central problem in primality testing is to decide if a given input is prime or composite. This problem is generally considered much easier than factoring composites. One of the simpler ideas in the subject involves Fermat's "little theorem": $a^p \equiv a \bmod p$ for all integers a. It is computationally easy to compute the residue of $a^p \bmod p$, and if this is not a, then p has been proved composite.

This simple test done with $a = 2$ is enough to recognize most composite numbers. However, for any fixed base a there are infinitely many *pseudoprimes to the base a*, namely composite integers n for which $a^n \equiv a \bmod n$. In fact, there are infinitely many *Carmichael numbers*. These are composite integers n for which $a^n \equiv a \bmod n$ for *every* integer a. It had been conjectured that there are infinitely many Carmichael numbers essentially by Carmichael himself when he introduced them in 1910. The proof that there are infinitely many was accomplished in 1992 by Alford, Granville, and the author [AGP], and is based on a 1956 heuristic argument of Erdős. This heuristic method begins by assuming that there are many primes p for which $p - 1$ is y-smooth. In fact, there should be a positive proportion of all primes below y^c with this property, where c is an arbitrary but fixed number. Erdős himself had proved such a result in 1935 for some particular $c > 1$, and recently Friedlander proved it for any $c < 2\sqrt{e}$. With this and other tools, we were able to prove that there are more than $x^{2/7}$ Carmichael numbers up to x, when x is sufficiently large. It is interesting that the Erdős heuristic method in fact suggests that there are more than $x^{1-\varepsilon}$ Carmichael numbers up to x.

There are stronger tests than Fermat's little theorem for which there is no analog of a Carmichael number, and such that on input of a composite number, the test is expected to prove the number composite in only $O(1)$ iterations. One of these is using Selfridge's strong pseudoprime test to random bases, a result of Rabin. From the work of Miller, Bach, and others we know that every odd composite n will fail a strong pseudoprime test to some base less than $2\ln^2 n$, provided that the Riemann hypothesis for Dirichlet L-functions holds. Thus, if this hypothesis holds, we have a deterministic polynomial time primality test. In a sequel to [AGP], the authors show that there are infinitely many odd composites n that pass the strong pseudoprime test for each base up to $(\ln n)^{c/\ln\ln\ln n}$.

Do we have unconditional tests that end up proving a prime input is prime? Surely we should not be satisfied with a probabilistic composite recognition test that fails to recognize our input as composite after several tries.

There are in fact very fast primality proving algorithms. The fastest known deterministic test has complexity $(\ln n)^{c\ln\ln\ln n}$, and so is "almost" polynomial, see [APR]. As with the discussion above on Carmichael numbers, this test uses auxiliary primes p for which $p-1$ is y-smooth. Certain versions of this test are quite practical, see [BH], [CL]. There is a probabilistic test that expects to find a rigorous proof of primality in expected polynomial time. Though not very practical, simpler, but heuristic versions of it have been used on very large primes, see [AH], [AM], [GK], [L2].

The central unsolved problem in primality testing is to see if there is a deterministic, polynomial time algorithm to distinguish between primes and composites. Towards this end, one may ask for a deterministic, polynomial time algorithm that succeeds in proving prime most or many primes up to a bound x. Recently, Konyagin and the author [KP] have described such an algorithm that proves prime more than $x^{1-\varepsilon}$ primes up to x. It is no mystery on which primes the algorithm works. It works on precisely those primes p for which $p-1$ has a large smooth divisor.

The author gratefully acknowledges W. Alford, A. Granville, and H. Lenstra for their helpful critical comments on an earlier draft of this paper.

References

[A] L. M. Adleman, *The function field sieve*, in: Algorithmic Number Theory (L. M. Adleman and M.-D. Huang, eds.), Lecture Notes in Comput. Sci. **877**, Springer-Verlag, Berlin, 1994, 108–121.

[AH] L. M. Adleman and M.-D. Huang, Primality testing and abelian varieties over finite fields, Lecture Notes in Math. **1512**, Springer-Verlag, Berlin and New York, 1992.

[APR] L. M. Adleman, C. Pomerance, and R. S. Rumely, *On distinguishing prime numbers from composite numbers*, Ann. of Math. (2) **117** (1983), 173–206.

[AGP] W. R. Alford, A. Granville, and C. Pomerance, *There are infinitely many Carmichael numbers*, Ann. of Math. (2) **140** (1994), 703–722.

[AM] A. O. L. Atkin and F. Morain, *Elliptic curves and primality proving*, Math. Comp. **61** (1993), 29–68.

[BH] W. Bosma and M.-P. van der Hulst, Primality proving with cyclotomy, Ph.D. dissertation at the University of Amsterdam, Amsterdam, The Netherlands, 1990.

[BLP] J. P. Buhler, H. W. Lenstra, Jr., and C. Pomerance, *Factoring integers with the number field sieve*, in [LL], 50–94.

[CEP] E. R. Canfield, P. Erdős, and C. Pomerance, *On a problem of Oppenheim concerning "factorisatio numerorum"*, J. Number Theory **17** (1983), 1–28.

[CL] H. Cohen and H. W. Lenstra, Jr., *Primality testing and Jacobi sums*, Math. Comp. **42** (1984), 297–330.

[C] D. Coppersmith, *Solving homogeneous linear equations over $GF(2)$ via block Wiedemann algorithm*, Math. Comp. **62** (1994), 333–350.

[GK] S. Goldwasser and J. Kilian, *Almost all primes can be quickly certified*, Proc. 18th STOC (Berkeley, May 28–30, 1986), ACM, New York, 1986, 316–329. Also see Chapter 2 in J. Kilian, Uses of Randomness in Algorithms and Protocols, The MIT Press, Cambridge, MA, 1990.

[G] D. Gordon, *Discrete logarithms in $GF(p)$ using the number field sieve*, SIAM J. Discrete Math. **6** (1993), 124–138.

[HT] A. Hildebrand and G. Tenenbaum, *Integers without large prime factors*, J. de Théorie des Nombres de Bordeaux **5** (1993), 411–484.

[KP] S. Konyagin and C. Pomerance, *On primes recognizable in deterministic polynomial time*, in: The Mathematics of Paul Erdős (R. L. Graham and J. Nesetril, eds.), Springer-Verlag, to appear.

[LO] B. A. LaMacchia and A. M. Odlyzko, *Solving large sparse linear systems over finite fields*, in: Advances in Cryptology — CRYPTO 90 (A. J. Menezes and S. A. Vanstone, eds.), Lecture Notes in Comput. Sci. **537**, Springer-Verlag, Berlin and New York, 1991, 109–133.

[LL] A. K. Lenstra and H. W. Lenstra, Jr. (eds.), The development of the number field sieve, Lecture Notes in Math. **1554**, Springer-Verlag, Berlin and New York, 1993.

[L1] H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. (2) **126** (1987), 649–673.

[L2] H. W. Lenstra, Jr., *Elliptic curves and number theoretic algorithms*, in: Proc. Int'l Cong. Math., Berkeley, CA, 1986, vol. 1 (A. M. Gleason, ed.), Amer. Math. Soc., Providence, RI, 1987, 99–120.

[LPP] H. W. Lenstra, Jr., J. Pila, and C. Pomerance, *A hyperelliptic smoothness test*, I, in [V], 397–408.

[LP] H. W. Lenstra, Jr., and C. Pomerance, *A rigorous time bound for factoring integers*, J. Amer. Math. Soc. **5** (1992), 483–516.

[Lo1] R. Lovorn, Rigorous, subexponential algorithms for discrete logarithms over finite fields, Ph.D. dissertation at The University of Georgia, Athens, GA, 1992.

[Lo2] R. Lovorn Bender, *Rigorous, subexponential algorithms for discrete logarithms in $GF(p^2)$*, SIAM J. Discrete Math., to appear.

[M] P. L. Montgomery, *A block Lanczos algorithm for finding dependencies over $GF(2)$*, in: Advances in Cryptology — EUROCRYPT 95 (L. C. Guillon and J.-J. Quisquater, eds.), Lecture Notes in Comput. Sci. **921**, Springer-Verlag, Berlin, 1995, 106–120.

[MB] M. Morrison and J. Brillhart, *A method of factoring and the factorization of F_7*, *Math. Comp.* **29** (1975), 183–205.

[O] A. M. Odlyzko, *Discrete logarithms in finite fields and their cryptographic significance*, in: Advances in Cryptology — Proceedings of EUROCRYPT 84 (T. Beth et al., eds.), Lecture Notes in Comput. Sci. **209**, Springer-Verlag, Berlin and New York, 1985, 224–314.

[P1] C. Pomerance, *Analysis and comparison of some integer factoring algorithms*, in: Computational Methods in Number Theory (H. W. Lenstra, Jr. and R. Tijdeman, eds.), Math. Centre Tracts **154/155**, Mathematisch Centrum, Amsterdam, 1982, 89–139.

[P2] C. Pomerance, *The quadratic sieve factoring algorithm*, in: Advances in Cryptology — Proceedings of EUROCRYPT 84 (T. Beth et al., eds.), Lecture Notes in Comput. Sci. **209**, Springer-Verlag, Berlin and New York, 1985, 169–182.

[P3] C. Pomerance, *Fast, rigorous factorization and discrete logarithm algorithms*, in: Discrete Algorithms and Complexity (D. S. Johnson et al., eds.), Academic Press, Orlando, FL, and New York, 1987, 119–143.

[P4] C. Pomerance, *The number field sieve*, in: Mathematics of Computation 1943–1993: A Half-Century of Computational Mathematics (W. Gautschi, ed.), Proc. Sympos. Appl. Math. **48**, Amer. Math. Soc., Providence, 1994, 465–480.

[PS] C. Pomerance and J. W. Smith, *Reduction of huge, sparse matrices over a finite field via created catastrophes*, Experimental Math. **1** (1992), 90–94.

[S] O. Schirokauer, *Discrete logarithms and local units*, in [V], 409–423.

[So] K. Soundararajan, *Asymptotic formulae for the counting function of smooth polynomials*, J. London Math. Soc., to appear.

[V] R. C. Vaughan, ed., *Theory and applications of numbers without large prime factors*, Philos. Trans. Roy. Soc. London **345** (15 November 1993), 327–423.