# NUMBER-THEORETIC ALGORITHMS

by

Eric Bach

Computer Sciences Technical Report #844

April 1989

CONTENTS

# NUMBER-THEORETIC ALGORITHMS

*Eric Bach*

Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706

To appear in *1989 Annual Review of Computer Science.*

## INTRODUCTION

This paper is a report on algorithms to solve problems in number theory. I believe the most interesting such problems to be those from elementary number theory whose complexity is still unknown. For this reason, I shall concentrate on methods to test primality, to find the prime factors of numbers, and to solve equations in various finite groups, rings, and fields. These problems have the attractive feature that they are easily stated, and frequently can be solved by algorithms that are easy to implement. However, the intuition behind these algorithms and the methods used to analyze them are often anything but elementary; for this reason I describe not only the algorithms but also the underlying mathematics.

It should be pointed out early on that the present review is not an exhaustive survey of the area, and necessarily reflects my own biases and interests. For example, the inclusion of algorithms for polynomials over finite fields may seem inappropriate. However, the polynomials in one variable over a finite field have a well-studied and attractive arithmetic, and it seemed interesting to contrast algorithms for this domain with algorithms for the integers. At other points, the subject impinges on algebraic number theory, but I shall omit the details of such connections, for lack of space. For similar reasons I shall not fully describe the many interesting and important applications of this material. Finally, because the literature on the subject contains a profusion of different computational models, it seemed important to analyze algorithms from one perspective; perhaps a survey of results compiled with this in mind will be useful.

Number theorists in the past thought of their field as a purely theoretical area devoid of potential applications. But recent developments have shown this not to be so: algebraic methods are important in several areas, in which the basic techniques are all number theoretic in nature. As examples one might note computer algebra, algebraic coding theory, cryptography, and pseudo-random number generation.

Already classical number theory contains a wide variety of algorithms for various problems, and mathematicians such as Gauss were well aware of the difference between good and bad algorithms (his *Disquisitiones* is peppered with remarks on this topic). In fact, there is a direct connection from 19th century constructivism, through Hilbert's program and computability theory, to the modern science of algorithms. However, the notion of algorithm complexity became predominant only in the computer age. There are two reasons for this. First, when doing hand calculations, one can often apply clever tricks or some particular knowledge about the instance at hand, so that the worst-case behavior of an algorithm may not be apparent. Second, the relative difference in performance between efficient and inefficient algorithms becomes magnified when fast computers are used.

Because the integers are God-given (at least to Kronecker), number theory has some of the characteristics of a natural science. Some of the earliest algorithm analysts were experimental number theorists, most notably Lehmer, who more than 50 years ago built sieve machines for factoring (1933). This empirical tradition has been carried on to the present day, as evidenced in the results cited by Williams (1982a). A striking example of such computational effort is Odlyzko and te Riele's refutation (1985) of a famous conjecture of Mertens in analytic number theory.

Despite much work on the theory of computation, theorists and practitioners disagree on what features a good algorithm should have. The theory has been largely dominated by considerations of asymptotics, which demand that one get the best possible function for the running time bound (never mind the constants), and by the requirement that every running time be rigorously proved. In contrast, implementors are often comfortable using algorithms that are asymptotically not the best known, and whose observed behavior cannot yet be formally justified.

As might be suspected, research in this area has always shown a major influence from analytic and algebraic number theory and the theory of finite algebraic objects such as Galois fields. In this regard the subject is thoroughly classical; the ideas of "post-modern" algebra such as categories tend to be used only indirectly, to prove results about the simpler structures in which computation takes place.

Several distinctly modern aspects of the area are notable.

First, computational number theory shows a profound influence from combinatorics, via the theory of NP-completeness and its assumption that an algorithm is useful if and only if it runs in polynomial time. This theory has been very helpful in organizing our knowledge about combinatorial optimization: most problems in this domain are now known to be either NP-complete, meaning that they can encode propositional logic, or solvable in polynomial time, by a practical algorithm. Inspired by this success, number theorists now work within the polynomial time paradigm, but the advantage of this viewpoint is less clear than for combinatorics. Although some problems involving Diophantine equations are NP-complete (see the paper of Manders and Adleman (1978)), for more important problems such as factorization this is not known. Number theory seems to lack the sharp boundary between solvable and intractable problems exhibited in combinatorics. For example, subexponential time algorithms are known for difficult problems such as factorization and the discrete logarithm, and for structural reasons these problems are probably not NP-complete.

A second major influence is that of probability theory and the idea of randomness, which has been injected into the area in three ways. First, probability theory is a powerful tool for formulating conjectures about the behavior of number-theoretic functions; such conjectures often lead to practical algorithms. Second, the area of pseudo-random number generation, which has always been influenced by number theory, has undergone a revival with the recent interest in cryptography. The most striking idea in this realm is that computational intractability is good for something: if a number-theoretic problem is not efficiently solvable, then it can be used to construct sequences that appear "featureless" to an algorithm. For such results, see the papers of Blum and Micali (1984) and Yao (1982). Third, the recent notion of randomized algorithm has clarified our ideas concerning tentative methods in computation; one may now deal with such procedures in a formal way.

Finally, and this is a very recent trend, our ideas about number-theoretic algorithms have been irreversibly changed by algebraic geometry. This subject, an outgrowth of classical complex analysis, embraces a general theory of polynomial equations – exactly the sort of equations that would arise in computation using rational operations. In the realm of algebraic complexity theory, this was recognized 15 years ago by Strassen (1973). In number theory, algebraic geometry is especially useful because of the deep connections between the topology of complex manifolds and bounds for the number of solutions of equations over finite fields. Although the ultimate results of this trend remain to be unveiled, the geometric approach has led to important algorithms for primality testing and factorization, and has helped sharpen the analysis of randomized algorithms.

Before going on to a detailed review, I would like to mention other sources for this subject. The primary literature is easily accessible through three important compilations. Dickson (1919) wrote an exhaustive three-volume history of number theory, and the relevant articles in *Mathematical Reviews* from 1940 to 1983 have been complied by LeVeque (1974) and Guy (1984). For longer treatments of number-theoretic algorithms, see the books of Lenstra and Tijdeman (1984), Riesel (1985), Knuth (1981), and Koblitz (1987b). The last two discuss applications in depth, as do the books of Davenport, Siret, and Tournier (1988) and Berlekamp (1968). I shall mention other books and surveys as appropriate; for detailed coverage of recent algorithms, the article of Lenstra and Lenstra (1987) is especially

recommended.

The rest of this paper is organized as follows. The next section presents mathematical background from computation theory, algebra, analytic number theory, algebraic geometry, and probability theory. The next five sections are problem-oriented, and discuss arithmetic, solving equations in finite fields, primality testing, factorization, and discrete logarithms.

## BACKGROUND

This section summarizes the background used in the rest of the paper. To start this from scratch would be impossible, and I presume that a certain amount is known by the reader. For more background on the topics of this section, the books of Aho, Hopcroft, and Ullman (1974), van der Waerden (1970), Davenport (1980), Hartshorne (1977), and Feller (1968) are useful standard references. For number theory *per se,* a comprehensive introduction has been written by Hua (1982).

### *Computation theory*

To compare algorithms, one needs a notion of computation step that does not depend on the particular problem being solved. Although one may formalize this idea in many ways, it is most common to use either a step of a multitape Turing machine computation or an instruction executed by an idealized random-access computer with no bound on the word size. Since number-theoretic algorithms are intended to deal with large integers on real computers, with random access but a fixed word size, neither of these conventions is entirely satisfactory.

In addition, running time estimates serve a dual function: that of prediction (how quickly will my algorithm run?) and comparison (is my time bound an improvement on what is known?). With number-theoretic algorithms, these two goals are often at cross purposes, for a problem size that seems large by one measure (300 bits) may be small by another (10 computer words). In addition, an asymptotically efficient algorithm for a problem may not beat an ordinary method until the problem size is so large that any algorithm would be useless.

In line with the principle that theory should explain and enlighten practice, my emphasis here is on the side of prediction. Therefore, I shall present running times in terms of "naive bit complexity." This phrase refers to a set of assumptions about how arithmetic operations are implemented, together with simple running time bounds for these operations. These assumptions are as follows. All arithmetic on large numbers and polynomials is done by classical methods, the running time bounds count only arithmetic operations on numbers of some fixed size (say, one bit), and all considerations of addressing and space requirements are ignored. Using this model, one can very easily estimate how long a simple computation should take on a given machine.

Although this is the model currently in favor among computational number theorists, two other approaches are common in the literature. One model, used by complexity theorists, replaces classical algorithms for arithmetic with the asymptotically fastest methods but still measures bit operations. Another approach, commonly used in algebraic complexity theory, is similar in spirit to this but counts algebraic operations in a ring or field. Often the ring or field is not specified, but may be constrained to satisfy certain assumptions such as having characteristic zero. Results using these models point out the ultimate limits of computational methods, and they are of considerable interest. However, they are not central to this paper. I shall only summarize them when appropriate, labeling them "asymptotic", and provide references.

So that the reader may easily compare results from the various approaches, *all* running times in this paper will count bit operations. I shall, however, employ slightly different

notations for classical and asymptotic results; in general, bounds in terms of a binary length $l$ are asymptotic.

In a similar spirit I shall also briefly describe results in parallel complexity theory, using the following framework. A feasible parallel algorithm, or NC algorithm, can be implemented by a family of Boolean circuits whose $l$th member has $l$ input bits, $l^{O(1)}$ gates, and paths of length $(\log l)^{O(1)}$ from inputs to outputs. It is customary to impose uniformity conditions that guarantee that the circuits are simple to build; for different approaches to this problem see the papers by Ruzzo (1981) and Allender (1986). A set $L$ is said to be in the complexity class NC if it is recognizable by a logspace uniform circuit family satisfying the above size and depth restrictions. This class may be further stratified into levels; one then speaks of $\mathrm{NC}^k$ sets, which can be recognized by uniform circuit families of depth $O(\log l)^k$.

This notion must be distinguished from that of an "algebraic NC" algorithm, which is a uniform family of straight-line programs, of which the $l$th family member takes $l^{O(1)}$ inputs, does $l^{O(1)}$ operations, and has depth $(\log l)^{O(1)}$. The instructions in such programs are typically arithmetic operations in a given class of rings or fields. For the purposes of the present discussion, the algebraic NC model suffers from the drawback that operations such as inversion in large finite fields, which so far lack NC circuits, are presumed to take one time unit.

Another difficulty with NC algorithms is the large number of circuit elements allowed, and the assumption that any interconnection pattern whatsoever can be used without affecting performance; in practice, these problems are more severe than a lack of uniformity. Nevertheless, no model with a comparable appeal has yet emerged, and without a clear consensus on the issue, it seems justifiable to use the NC framework when presenting parallel algorithms. (For a survey of parallel algorithms, see the article by Karp and Ramachandran (1988)).

For problems that do not have efficient algorithms, it is useful to take a coarser point of view, in which the notion of polynomial time algorithm is primitive. One may define polynomial time in various ways, but the end result does not change; see, for example, the book by Machtey and Young (1978). In such a definition, one always compares running time to the binary length of the input; thus an algorithm with input $n$ runs in polynomial time when its bit complexity is bounded by a power of $\log n$, not a power of $n$. In addition, there is a further simplification: every problem is represented as a decision problem, that is, as a subset of the natural numbers. With appropriate coding we may speak of sets of integers, pairs of integers, and so on. P is the collection of sets having decision algorithms that run in polynomial time.

The following complexity classes have proved to be the most important for number-theoretic problems. NP is the class of sets $L$ such that for some set $M$ in P, and for some positive $d$,

$$x \in L \iff \text{for some } y \text{ with } \log y \leq \log^d x,\ (x,y) \in M$$

Intuitively, a set is in NP if each of its elements has a short, easily checkable, proof of membership in that set. A set $L$ is said to be in RP if for $M \in \mathrm{P}$, $d > 0$,

$$x \in L \Rightarrow \text{for at least half of the } y\text{'s with } \log y \leq (\log x)^d,\ (x,y) \in M$$

and

$$x \notin L \Rightarrow \text{for none of the } y\text{'s with } \log y \leq (\log x)^d,\ (x,y) \in M;$$

from this it follows that $\mathrm{RP} \subset \mathrm{NP}$. $L$ is in ZPP if both it and its complement are in RP. Finally, the class BPP is defined similarly to RP, by replacing "at least half" by "at least

3/4" and "none" by "at most 1/4." These notions were introduced by Gill (1977); the constants in their definitions are somewhat arbitrary. Intuitively, a set is in BPP if it has a reliable probabilistic polynomial time algorithm to test membership, in RP if one can find a short, easily checkable, proof of set membership without being extraordinarily lucky, and in ZPP if there is a probabilistic algorithm to decide membership that is always correct and probably fast. None of these classes are known to equal P.

Although these are names of complexity classes, it is common to signal a result's salient features by informal phrases such as "NP-complete problem" or "BPP algorithm." Efficient randomized algorithms whose output is always correct are called *Las Vegas* procedures; this notion, although used mostly for randomized algorithms, can apply to any heuristic procedure.

*Algebra*

I shall take as known definitions and properties of basic algebraic structures: groups, fields, and rings (presumed commutative). In the sequel $\mathbb{Z}$ will denote the integers, $\mathbb{N} = \{0, 1, 2, \ldots\}$ the natural numbers, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ the rational, real, and complex numbers, and $\mathbb{F}_q$ a finite field with $q$ elements. If the size of the finite field does not matter, it will simply be called $k$. Note that if $p$ is prime then $\mathbb{F}_p$ is the same as $\mathbb{Z}/(p)$. If $A$ is a ring, then $A[X]$ denotes the polynomial ring in one indeterminate, and $A^*$ denotes its group of units.

A fundamental idea of algebra is that of factorization, in which one breaks up a structure into simpler ones. The most useful factorization theorems are the following two.

The *Chinese remainder theorem* asserts that if $A$ is a commutative ring, and $I$ and $J$ are ideals with $I + J = A$ then $A/(IJ) \cong A/I \oplus A/J$. Applying this recursively to the prime factorization $p_1^{e_1} \cdots p_r^{e_r}$ of a positive integer $n$,

$$\mathbb{Z}/(n) \cong \mathbb{Z}/(p_1^{e_1}) \oplus \cdots \oplus \mathbb{Z}/(p_r^{e_r}).$$

A similar result holds for polynomials in $k[X]$; if $f = f_1^{e_1} \cdots f_r^{e_r}$ with the $f_i$'s irreducible and relatively prime, then

$$k[X]/(f) \cong k[X]/(f_1^{e_1}) \oplus \cdots \oplus k[X]/(f_r^{e_r}).$$

In both cases, the group of units is the direct product of the unit groups of the direct summands. By definition, $\varphi(n)$ denotes the size of $\mathbb{Z}/(n)^*$, and $\varphi(f)$ the size of $k[X]/(f)^*$.

The *fundamental theorem of abelian groups*, due to Frobenius and Stickelberger (1879), states that if $G$ is a finite abelian group, then $G$ is the direct product of cyclic groups of prime power order. The number and type of these factors are invariants of the group.

Each of these theorems gives an algebraic object an alternate "data structure." One may either transform a computational problem by converting it to the alternate representation, or just imagine in one's analysis that such a transformation has been made; both techniques are commonly used.

Another use of the Chinese remainder theorem is the following, which has been dubbed the "pretend-field" technique. When doing computations in the finite ring $\mathbb{Z}/(n)$, one may pretend that it is a field, until an inversion operation fails. This is especially useful in factorization, for a nonzero element $x$ of $\mathbb{Z}/(n)$ that is not a unit will have a nontrivial gcd with $n$. When inversion fails in other situations, one just factors the ring, and proceeds with computations in the direct product; such an interruption can happen only a few times.

It is also useful to know some facts about finite fields. I shall only give the basics; for complete coverage, see the book of Lidl and Niederreiter (1984). The most important

properties of these fields are the following. The multiplicative group of a finite field is cyclic. If a finite field has characteristic $p$, then the function taking an element to its $p$th power is $\mathbb{F}_p$-linear, and called the *Frobenius map.* This function is an automorphism and generates the Galois group of a finite field. (In a ring of characteristic $p$, it is an endomorphism but need not be injective.) Similar facts are true about $q$th powers, relative to a subfield of order $q$. The product of an element's conjugates is called its *norm*, and the sum of its conjugates its *trace.*

Because they occur as the multiplicative groups of finite fields, it is useful to know some facts about finite cyclic groups. A cyclic group of order $n = p_1 \cdots p_r$ has a chain factorization

$$1 = G_0 \subset G_1 \subset \cdots \subset G_k = G$$

where $G_i/G_{i-1}$ is cyclic of order $p_i$. A fundamental "exactness" result, due to Euler, states that if $d|n$, then $x$ is a $d$th power if and only if $x^{n/d} = 1$.

In certain situations one seeks the structure of the unit group modulo an integer or polynomial. By the Chinese remainder theorem this reduces to finding the structure of the unit group for each direct factor. This was found by Gauss (1801, §82) for $\mathbb{Z}/(p^e)^*$, but apparently only recently by Claasen (1977) for $k[X]/(f^e)^*$. The first result states that the group of units modulo $p^e$ is cyclic, except when $p = 2$ and $e \geq 3$, in which case it is the direct product of a group of order 2 and one of order $2^{e-2}$. The second case is more complicated. If $k = \mathbb{F}_q$, a field of characteristic $p$, and $f$ is an irreducible polynomial in $k[X]$ of degree $d$, then the unit group of $k[X]/(f^e)$ is the direct product of a cyclic group of order $q^d - 1$ and a group of order $q^{d(e-1)}$. The second group is composed mainly of direct factors of order $p$; roughly speaking, the number of cyclic factors of order $p^i$ decreases exponentially with $i$.

If $p$ is an odd prime the following defines the *Jacobi symbol*:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if } a \text{ and } p \text{ are relatively prime and } a \text{ is a square modulo } p; \\ -1, & \text{if } a \text{ and } p \text{ are relatively prime and } a \text{ is not a square modulo } p; \\ 0, & \text{otherwise;} \end{cases}$$

if $n = p_1 \cdots p_r$ with each $p_i$ an odd prime, then $\left(\frac{a}{n}\right) = \prod_{i=1}^{r} \left(\frac{a}{p_i}\right)$. Squares in the group $\mathbb{Z}/(n)^*$ are called *quadratic residues*; $\left(\frac{x}{n}\right) = 1$ is necessary but not generally sufficient for $x$ to be a quadratic residue modulo $n$. The law of *quadratic reciprocity,* proved by Gauss (1801, §131), states that for odd primes $p$ and $q$, $\left(\frac{p}{q}\right) = \pm\left(\frac{p}{q}\right)$, with the minus sign taken if and only if $p$ and $q$ are congruent to 3 modulo 4; this was later extended by Jacobi (1837) to all odd $p$ and $q$. One may also define a similar symbol in $k[X]$ if the characteristic is odd; the corresponding reciprocity law states that if $f, g \in k[X]$, with $f, g$ monic and of positive degree, then

$$\left(\frac{f}{g}\right) = (-1)^{\frac{(q-1)}{2} \deg f \deg g} \left(\frac{g}{f}\right).$$

This was proved by Dedekind (1857) for prime fields $k$ and extended by Kühne (1902) to cover all finite fields.

From an abstract viewpoint, reciprocity laws solve the following problem: given a ring $A$ and an irreducible polynomial $f$ in $A[X]$, explain how $f$ factors when $A$ is reduced modulo a prime ideal. This is the motivating problem for a major area of algebraic number theory, called class field theory. Some results of this theory are useful in algorithms; the book of Ireland and Rosen (1982) provides an introduction to higher reciprocity laws, and the volume edited by Cassels and Fröhlich (1967) is a more technical introduction to the subject.

For a thorough introduction to algebraic number theory, see the book of Narkiewicz (1974). For a discussion of the computational aspects of algebraic numbers, see the book of Zimmer (1972).

*Analytic number theory*

The fundamental fact of analytic number theory is the prime number theorem, due independently to Hadamard (1896) and de la Vallée Poussin (1896). It states that $\pi(x)$, the number of primes less than or equal to $x$, is asymptotic to $x/\log x$.

This theorem has been generalized in various ways. De la Vallée Poussin went on to show that if $a$ and $q$ are positive numbers, relatively prime, then $\pi_{a,q}(x)$, the number of primes not exceeding $x$ and congruent to $a$ modulo $q$, is asymptotic to $x/(\varphi(q)\log x)$. Consequently, there are infinitely many such primes. Čebotarev's density theorem (1926) leads to similar estimates for the number of prime ideals with given splitting properties in a ring of algebraic integers. Because every prime ideal of such a ring is associated with an ordinary prime number, this last result may be used to estimate the density of prime numbers with desirable properties.

Besides questions of density, one may be interested in the first prime in an arithmetic progression modulo $q$. This was shown by Linnik (1944) to be $O(q^C)$ for some $C > 0$. Fogels (1962) found a corresponding result for algebraic number fields.

Unfortunately, in this subject there is a large gap between what is currently provable and what is apparently true. For example, there is ample empirical and heuristic evidence to support the belief that

$$\pi(x) = \mathrm{li}(x) + x^{1/2+o(1)}$$

where $\mathrm{li(x)} = \int_2^x dx/\log x$. This is true if and only if the zeta function has no zeroes to the right of the line $\mathrm{Re}(s) = 1/2$, which is the famous hypothesis of Riemann (1859). Since this is a precise statement, one can investigate it computationally by proving that all zeroes up to a given height lie on this line. Culminating investigations by many researchers, Van de Lune, te Riele, and Winter (1986) showed that the Riemann hypothesis holds for the first 1.5 billion zeroes of the zeta function. Odlyzko (1987) has examined some blocks of extremely large zeroes, which are similarly placed.

For primes in progressions, the corresponding estimate is that for each relatively prime $a$ and $q$,

$$\pi_{a,q}(x) = \frac{1}{\varphi(q)}\mathrm{li}(x) + x^{1/2+o(1)};$$

this is equivalent to the assertion that Dirichlet $L$-functions, generalizations of the zeta function, satisfy the Riemann hypothesis. This conjecture was apparently first stated by Piltz (1884); similar conjectures apply to more exotic $L$-functions appearing in algebraic number theory. It is common for number theorists to derive results assuming the truth of the Riemann hypothesis or some of its generalizations. Such assumptions usually go under the collective (and vague) name of the *Extended Riemann Hypothesis* (ERH). If true, the ERH would have many important consequences. However, it has not been checked as thoroughly as the ordinary Riemann hypothesis; to date the most comprehensive computations are those of Spira (1969), who examined zeroes of Dirichlet $L$-functions for moduli $q \leq 24$. Lagarias and Odlyzko (1979) report computations involving more general $L$-functions.

In some sense the ERH encapsulates certain observed facts about the behavior of number-theoretic functions, and this alone gives it value as a heuristic. In addition, there are many results that can be sharpened if Riemann hypotheses are assumed. Below I list some important consequences of the ERH.

First, if the ERH holds, then the multiplicative group of $\mathbb{Z}/(n)$ is generated by the numbers up to $O(\log n)^2$; this was first proved by Ankeny (1952) for $n$ prime and later extended by Montgomery (1971) to all $n$. A similar result due to Lagarias, Montgomery, and Odlyzko (1979) holds in the algebraic number case. Here one defines groups that generalize the group of units modulo $n$; it would follow from the ERH that they also are generated by small prime ideals.

Second, every integer $a$ not equal to $\pm 1$ or a perfect square is believed to be a primitive root for infinitely many primes; this is Artin's conjecture. Artin also found heuristic estimates of the densities, depending on $a$, of such primes; computations by Lehmer and Lehmer (1962) showed some of these estimates to be in error, but the corrections have been "channeled" by Lang and Tate (1965). Hooley (1967) has shown that Artin's conjecture would follow from the ERH.

Finally, it seems to be true that every prime has a small primitive root; assuming the ERH, Wang (1961) showed that the least primitive root modulo $p$ is $O(\log p)^8$. Perhaps a similar bound holds for algebraic number fields, but I am unaware of any such result.

When designing algorithms, it is useful to have numerical versions of results in analytic number theory. Rosser and Schoenfeld (1962) gave explicit versions of the prime number theorem, and Schoenfeld (1976) gave corresponding bounds that assumed the Riemann hypothesis. Explicit bounds were given by McCurley (1984a, 1984b) for the density of prime numbers in arithmetic progressions. To my knowledge there are no unconditional numerical estimates for Čebotarev's density theorem, except for the special cases cited in this paragraph.

The best published constant $C$ in Linnik's theorem is $C = 16$, due to Wang (1986). (A value of 15, for which a proof apparently never appeared, is cited in Hua's book (1982)). Such a result, however, merely estimates the exponent in Linnik's theorem and does not provide explicit bounds for the least prime in a progression. Oesterlé (1979) announced explicit versions of Čebotarev's theorem that assumed the ERH; for the rational case his results specialize to yield

$$|\pi_{a,q}(x) - \frac{1}{\varphi(q)}\mathrm{li}(x)| \le \sqrt{x}(2\log q + \log x)$$

for $x \ge 2$. As noted by Chowla (1934), a bound of this quality would imply that the constant $C$ in Linnik's theorem is arbitrarily close to 2. Presumably, then, the least prime in a progression modulo $q$ is $q^{2+o(1)}$, and an explicit bound if desired could be computed from the above formula. According to heuristic arguments and empirical evidence of Wagstaff (1979), even this is an overestimate; he argued that the least prime in any progression modulo $q$ is likely less than $\phi(q)\log q \log\phi(q)$.

The case of Ankeny's theorem is especially interesting, as algorithms such as Miller's prime test (1976) cannot even be specified without a concrete version of this result. Bach* (1985) showed, assuming the ERH, that if $G$ is a nontrivial subgroup of the integers modulo $n$, then there is some $x$ outside $G$ with $x < 2(\log n)^2$; generalizations to algebraic number theory appear in a further paper (1989a).

For the analysis of recent primality algorithms it is important to know not only the average behavior of primes, but also their local distribution. That is, one wishes to know the growth rate of $d_n$, the difference between the $n$th and the $n-1$st prime. Unfortunately, even with Riemann hypotheses the results on this problem do not agree with observations.

* This self-referential style should aid readability.

Various bounds of the form $d_n = n^{\alpha+o(1)}$ are known. In the paper of Halberstam, Lou, and Yao (1989), $\alpha = 6/11$ is claimed by the last two authors; $\alpha = 1/2$ would follow from the Riemann hypothesis, as shown by Cramér (1921). But computations such as those of Brent (1973) suggest that $\alpha = 0$; this is implied by heuristics such as that of Cramér (1937), who gave a probabilistic argument that $d_n = (\log n)^{2+o(1)}$. Were this or a similar conjecture true, the distance from a number $x$ to the smallest prime greater than $x$ would be, for $x$ large, bounded by a polynomial in $\log x$.

Finally, the factorization patterns of numbers show a statistical regularity that can be stated informally as follows: the factors of a random $l$-bit number have the same asymptotic distribution as the cycle lengths of a random permutation on $l$ letters. In various algorithmic settings, it is important to find numbers all of whose prime factors are less than a certain bound, say $M$; a number is called *M-smooth* if it has this property. Dickman (1930) investigated a function $\rho(x)$ that asymptotically gives the probability that a random number $n$ is $n^{1/x}$-smooth, as a function of $x$. His heuristic arguments were made rigorous by Knuth and Trabb Pardo (1976), who also investigated distributions for the relative length of the $k$th largest factor of a number.

Because Dickman's function is useful in the analysis of algorithms, I summarize some facts about it here. First, the rule of thumb that $\rho(x) \cong x^{-x}$ is useful in rough calculations; for ranges of practical interest, $5 \leq x \leq 10$, say, it is surprisingly accurate. Although reasonable approximations for $\rho$ in terms of the exponential integral were given by de Bruijn (1951), if one needs accurate values one should simply compute it numerically. Good algorithms to do this were published by van de Lune and Wattel (1969). Of course, one usually needs not the rho function but a probability of smoothness. In estimating such probabilities asymptotically, results of Canfield, Erdös, and Pomerance (1983) are quite useful. These results do not provide explicit smoothness estimates; however, from computations of Odlyzko in Schnorr and Lenstra's paper (1984) it can be concluded that the use of $\rho(x)$ probably suffices for practical purposes.

Several important results in analytic number theory have analogies in the polynomial ring $k[X]$. The prime number theorem, which implies that the fraction of $n$-bit primes is roughly inversely proportional to $n$, becomes Dedekind's result (1857) that the number of irreducible monic polynomials of degree $n$ in $\mathbb{F}_q[X]$ is $q^n/n + O(q^{n/2})$. Artin (1924) proved the result, analogous to the prime number theorem for arithmetic progressions, that if $a$ and $b$ are relatively prime polynomials in $\mathbb{F}_q[X]$, then the number of irreducible monic polynomials of degree $n$ congruent to $a$ modulo $b$ is $q^n/(n\varphi(b)) + O(q^{n\theta})$ for $\theta < 1$; $\theta = 1/2$ is an analog of the ERH. From results of Weil (1974, appendix V) one can show that an analog of Ankeny's theorem holds for $k[X]$. However, it states that groups $k[X]/(f)^*$ are generated by elements of small degree, which is useful only if the degree of $f$ is large relative to the size of $k$; an efficient construction of a generating set for all cases would be interesting and useful. Bilharz (1937) reduced the analog of Artin's conjecture in $k[X]$ to the Riemann hypothesis for finite fields, which is now known. Finally, by analogy with numbers, one may call a polynomial in $k[X]$ smooth if all of its irreducible factors have small degree. Asymptotic smoothness estimates using Dickman's function hold in this case as well; for some useful non-asymptotic bounds see the paper of Odlyzko (1985).

*Algebraic geometry*

Classical algebraic geometry studies the solution sets of polynomial equations in affine or projective space. For number-theoretic applications, it is usual to consider solutions lying in a subfield $k$ of some fixed algebraically closed field; it will suffice here to assume that $k$ is finite. The basic questions the theory tries to answer are: do such equations have solutions

in $k$, how many solutions are there, and do they have any additional structure?

Already by considering $k = \mathbb{F}_2$, one sees that the first two questions are probably not computationally tractable, for the question of the existence of a solution over $\mathbb{F}_2$ is NP-complete (Garey and Johnson (1979) ascribe this to Fraenkel and Yesha), and the problem of computing the number of solutions is complete for the class #P defined by Valiant (1979). Often, however, useful estimates are possible.

Assume that the algebraic set $X$ is a nonsingular $d$-dimensional projective variety and that it is specified by equations with integer coefficients that, interpreted over the complex numbers, define a complex manifold $X'$. Then one can use the Weil conjectures, which include analogs of the Riemann hypothesis for function fields, to estimate the size of $X$. These conjectures were proved by Deligne (1973) and imply that $N_q$, the number of $\mathbb{F}_q$-points on $X$, can be expressed as follows:

$$N_q = \alpha_0 q^d + \alpha_1 q^{d-1/2} + \cdots + \alpha_{2d-1} q^{1/2} + \alpha_{2d},$$

where $|\alpha_i|$ is at most $\beta_i$, the $i$th Betti number of the manifold $X'$ (i.e. the dimension of its $i$th de Rham cohomology group). The invariants $\beta_i$ may be computed by topological methods such as those of Hirzebruch (1966).

Most applications to algorithms assume that $X$ is an algebraic curve, that is, $d = 1$. Then $\beta_0 = 1$ (there is one connected component), and $\beta_2 = 1$ (by duality). The middle dimension $\beta_1$ is twice the genus $g$ of the curve. In this instance, the bound becomes

$$|N_q - (q+1)| \leq 2g\sqrt{q},$$

as shown by Weil (1948). If $X$ is nonsingular, and a complete intersection of hypersurfaces whose degrees are $a_1, \ldots, a_r$, then Hirzebruch's techniques give the formula

$$g = (\prod_{i=1}^{r} a_i)[\frac{\sum_{i=1}^{r}(a_i - 1)}{2} - 1] + 1.$$

(If the curve is singular, then one may use this number – the arithmetic genus – in an upper bound for $N_q$). Special cases of this formula imply the following results on plane curves.

A line or a conic section is always nonsingular, and has genus 0. Therefore the number of solutions to an irreducible degree 2 equation in two variables may be computed exactly.

A nonsingular plane curve of degree 3 has genus 1 and is called an *elliptic curve.* It is usual to take such curves in the standard form

$$y^2 = x^3 + ax + b$$

and include an extra point at infinity, given by (0:1:0) in projective coordinates. Elliptic curves have an abelian group structure, which can be specified by requiring that $P+Q+R = 0$ when $P$, $Q$, and $R$ are collinear, and that (0:1:0) is the identity of the group. Formulas for the addition law follow by elementary analytic geometry; they express the sum rationally using coordinates of the addends and the parameters $a$ and $b$ defining the curve. For these formulas and more facts about the arithmetic of elliptic curves, see the article of Tate (1974) or the book of Silverman (1986).

Curves of genus greater than 1 do not have a naturally defined group structure, but such curves are associated with algebraic groups called *abelian varieties.* For an introduction to this topic, see the book of Cornell and Silverman (1986) and the references therein.

*Probability*

I shall assume that the notions of elementary probability theory such as random variables, expected values, and so on are known. Below I summarize some less well-known facts that are useful in algorithm design.

Many theorems in algebra can fruitfully viewed in probabilistic terms. For example, if $\phi$ is a homomorphism mapping a finite group $G$ onto another group $H$ and $x$ is chosen at random from $G$, then $\phi(x)$ will be a random element of $H$. Since the order of a subgroup divides the order of a group, a random element chosen from a group $G$ will be outside any given proper subgroup of $G$ with probability at least $1/2$. Finally, direct product decompositions give independent random variables; for example, if $n = pq$ with $p$ and $q$ relatively prime, then by the Chinese remainder theorem, if $x$ is chosen at random modulo $n$, its residues modulo $p$ and $q$ are independent and uniformly distributed. A similar statement holds for the fundamental theorem of abelian groups.

The Extended Riemann Hypothesis corresponds to the almost sure behavior of a simple probabilistic model for primes, as follows. Let $a$ and $q$ be relatively prime, and for each positive integer $n$ congruent to $a$ modulo $q$, call $n$ "prime" with probability $1/(\varphi(q)\log n)$. Then an estimate analogous to $\pi_{a,q}(x) - \mathrm{li}(x)/\varphi(q) = x^{1/2+o(1)}$ holds with with probability 1. Of course, the primes are not chosen at random, but this suggests that the ERH should be useful as a guide to their behavior.

The distribution of smooth numbers also has a probabilistic interpretation, which one could call the "random bisection" process. Imagine that to choose a random number $n$, one first picks a prime factor $p$ whose length is uniformly chosen from the interval $(0, \log n)$, then repeats with $n$ replaced by $n/p$, and so on until all prime divisors are chosen. This model intuitively explains several results in analytic number theory; for example, the probability that a number near $n$ is prime is roughly inversely proportional to $\log n$, and a random number near $n$ should typically have $O(\log\log n)$ prime factors. The first result is implied by the prime number theorem, and the second by the Erdös-Kac theorem (1940), according to which the number of prime factors of a random number $n$ is roughly normally distributed with mean and variance $\log\log n$. Using random bisection one can see, heuristically at least, that the "probability" $\rho(x)$ that $n$ is $n^{1/x}$-smooth should satisfy the relation

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

with the initial condition $\rho(x) = 1$ for $0 < x < 1$. This is Dickman's recurrence (1930) for the function $\rho$.

Finally, for plane curves over $\mathbb{F}_p$, Weil's bound corresponds to the rough idea that a point $(x, y) \in \mathbb{F}_p^2$ "decides" to be on the curve with probability $1/p$.

However, caution is advised in the use of heuristic probability arguments, as the following examples show. An appealing probabilistic model of an elliptic curve over $\mathbb{F}_p$ results from imagining that for each $x$, $x^3 + ax + b$ is a quadratic residue modulo $p$ independently with probability $1/2$. Were this true, the number of points on the curve would, for $p$ large, have a binomial distribution with mean and variance close to $p$. Therefore, there would be a small probability for such a curve to have more than $p + 2\sqrt{p} + 1$ points, but this is impossible. The presumably correct distribution, conjectured independently by Tate (1965) and Sato, is more subtle and follows from considering measures on Lie groups; for this connection see the book of Serre (1968).

One may also consider the least quadratic nonresidue modulo a prime $p$ heuristically. Were $(\frac{x}{p})$ an independent random variable for each $x$, equally likely to be $\pm 1$, then the

least nonresidue would be almost surely $O(\log p)$. Although this conjecture matches a lower bound implied by Linnik's theorem, it is false: Graham and Ringrose (1988) have shown that the least nonresidue modulo $p$ is $\Omega(\log p \log\log\log p)$, and Montgomery (1971) showed that $\Omega(\log p \log\log p)$ follows from the ERH.

Finally, out of the grab-bag of tricks in finite probability theory, the following should be mentioned. The *birthday problem* asks how many independent samples from $\{1, \ldots, n\}$ are needed to find a duplicate; on the average, one needs $O(\sqrt{n})$. The *geometric distribution* governs the waiting time until the first success, when one flips a coin with success probability $p$; its expected value and standard deviation are $O(1/p)$. Finally, large deviation bounds such as Chernoff's (1952) are useful for improving the success rate of a BPP algorithm. For example, if a recognition algorithm has probability at least $1/2 + \epsilon$ of correctly accepting inputs in some set, and probability at most $1/2 - \epsilon$ of falsely accepting inputs not in the set, then one may run $n$ independent trials and side with the majority; the probability of error is at most $(1 - 4\epsilon^2)^{n/2}$.

## BASIC ALGORITHMS

This section discusses algorithms for the classical problems in arithmetic. Although some questions of asymptotic complexity are unresolved, the sequential algorithms for these problems are generally satisfactory. However, some important problems that are easily solved sequentially do not have good parallel algorithms.

To simplify descriptions, in this section I shall denote the binary length of a number $u$ in $\mathbb{Z}$ by $\log u$, and the length of a polynomial $u$ in $k[X]$, that is, $(\deg u + 1)\log q$, by $\log u$. Also if $u$ and $v$ are polynomials, then $u < v$ means that $u$ has smaller degree than $v$.

### *Basic arithmetic*

Let $u$ and $v$ be positive integers or polynomials in $k[X]$ with $v \leq u$. Then the ordinary algorithms for arithmetic have the following complexities: addition and subtraction take $O(\log u)$ steps, and $u$ and $v$ may be multiplied in $O(\log u \log v)$ steps. The complexity of performing the long division $u = qv + r$ is $O(\log q \log v)$, that is, the cost of a division is roughly proportional to the cost of the corresponding multiplication.

Evidently the time needed for addition and subtraction cannot be reduced in general. Various authors have found nearly linear-time algorithms for integer multiplication; the best such result is that of Schönhage and Strassen (1971), who showed that the product of two $l$-bit numbers may be computed in $O(l \log l \log\log l)$ steps. Their algorithm involves a recursive application of the Fast Fourier Transform (FFT) and is quite intricate. Using Newton's method, the complexity of multiplication can be shown to equal that of approximating reciprocals and square roots; no one has ever found a linear-time algorithm for these problems nor shown that $O(l)$ time is impossible to achieve. Asymptotically, the basic arithmetic operations on $l$-bit polynomials in $k[X]$ may also be performed on $l^{1+o(1)}$ steps.

However, even the simpler multiplication algorithms based on the FFT are not used much in practice, unless enormous numbers are involved; the only such situations I am aware of arise in testing Mersenne and Fermat numbers for primality, and in computing constants such as $\pi$ to record-breaking precision. Haworth (1986), Young and Buell (1988) and Borwein, Borwein, and Bailey (1989) discuss these applications. More useful, perhaps, is an $O(l^{1.59})$ multiplication method published by Karatsuba and Ofman (1962); for example, the designers of Maple, a computer algebra system, found that this method outperformed classical multiplication when $l$ exceeded 1200 (about 360 decimal digits).

Regarding parallel computation, integer addition, subtraction, and multiplication have $\mathrm{NC}^1$ algorithms. Therefore one may add, subtract, or multiply two $l$-bit numbers with

$l^{O(1)}$ gates, in time $O(\log l)$. For Boolean circuit families with such properties see the papers of Sklanksy (1960) and Karatsuba and Ofman (1962). The parallel complexity of integer division is a long standing open problem. Quadratically convergent methods for approximate inversion, such as that used by Anderson and his co-authors (1967), will lead to an $NC^2$ division algorithm. Only recently did Beame, Cook, and Hoover (1986) show that division circuits of polynomially bounded size and logarithmic depth are possible. This result, however, does not completely resolve the problem. Because it is based on residue arithmetic and computation of discrete logarithms, it suffers theoretically from a lack of logspace uniformity, and practically from a large hardware requirement.

For polynomials in $k[X]$, the situation is different. NC algorithms are known that will add, subtract, and multiply such polynomials, but simple examples show that a quotient with remainder in $k[X]$ cannot be thus computed unless the inversion problem in $k$ is in NC. This is only known to be true if the characteristic of $k$ is small.

*Modular arithmetic, powering*

The bounds above lead to complexity estimates for operations in rings of the form $\mathbb{Z}/(u)$ and $k[X]/(u)$. In either case, addition and subtraction take $O(\log u)$ steps, and multiplication and division by units take $O(\log u)^2$ steps. The last result is not obvious; it uses the quadratic time bound for the extended Euclidean algorithm given below.

Asymptotically, one may perform these operations in $l^{1+o(1)}$ steps when $u$ is $l$ bits long. Regarding parallel computation, NC algorithms are known in such rings for addition, subtraction, and multiplication, although for the last result in $k[X]$ $u$ should be monic. Inversion in $k[X]/(u)$ may be done in NC if the characteristic is small.

The usual repeated squaring algorithm leads to the following bound: in either ring one may compute $a^e$ in $O(\log e \log^2 u)$ steps. Since in general $e$ need be no greater than the size of the ring, this gives an $O(\log u)^3$ bound for exponentiation. Arguments based on the characteristic zero case suggest that repeated squaring or a similar technique is necessary, but this has never been proved.

If $u$ is $l$ bits long, then exponentiation in the above rings takes $l^{2+o(1)}$ steps, asymptotically. The parallel complexity of modular exponentiation is an important unsolved problem; no NC algorithm is known for the integers, although von zur Gathen (1987a) has given one for the special case of smooth moduli. Fich and Tompa (1988) gave an NC algorithm to compute powers in $k[X]/(u)$, but the characteristic of $k$ is required to be small. In the remaining cases, the the question is open.

*The greatest common divisor*

The division algorithms for $\mathbb{Z}$ and for $k[X]$ may be used in Euclid's algorithm to find the greatest common divisor (gcd) of two elements. The iteration is as follows. Starting with $u$ and $v$, one repeatedly applies the division algorithm and computes

$$u = qv + r, \qquad r < v,$$

$$v = q'r + r', \qquad r' < r,$$

$$\ldots$$

until some remainder vanishes; the last divisor is the gcd. The elements $q, q', \ldots$ are called the *quotients* of the algorithm; they are also the quotients of the ordinary continued fraction expansion of $u/v$.

Lamé (1844) showed that the Euclidean algorithm applied to positive integers $u$ and $v$ with $v \leq u$ stops after $O(\log u)$ division steps. From this result it is easy to conclude that the

bit complexity of Euclid's algorithm is $O(\log u)^3$, but its true bit complexity is $O(\log u)^2$. Intuitively, this holds because not all the quotients can be large, and large quotients reduce the intermediate values considerably. This bound seems to have been first proved by Collins (1969), who also showed a corresponding estimate for $k[X]$.

Since the Euclidean algorithm in $\mathbb{Z}$ is crucial to many computations, several authors have improved it, notably Lehmer (1938). To use his method, one computes the continued fraction expansion of a single-precision approximation to $u/v$, using an "interval arithmetic" scheme to decide which quotients are correct. These quotients may be used to produce new values of $u$ and $v$, from which the process continues. An alternative is the so-called "binary method," due to Stein (1967). This algorithm first determines the number of 2's in the gcd, then uses a subtractive algorithm in the ring $\mathbb{Z}[1/2]$, in which 2 is a unit and may be removed by shifting. However, neither of these methods reduce the running time of Euclid's algorithm by more than a constant factor.

Since any Euclidean domain is a principal ideal domain, if $d$ is the greatest common divisor of $u$ and $v$ in $\mathbb{Z}$ or $k[X]$, then there are $a$ and $b$ for which $au + bv = d$. If $v \leq u$, they do not exceed $u$ in absolute value or degree, and may be found in $O(\log u)^2$ steps by an extension to the Euclidean algorithm. When $u$ and $v$ are relatively prime, one may thus invert $v$ modulo $u$ in $O(\log u)^2$ steps.

The greatest common divisor algorithm leads to the following question, which was studied by Bach, Driscoll, and Shallit (1988): given a set of numbers or polynomials, what is the "best" factorization obtainable using *only* repetitions of the Euclidean algorithm? This is the factor refinement problem. The result is well-defined and may also be computed in $O(\log u)^2$ steps, where $u$ is the product of the inputs; in the case of multiple inputs, the proof involves amortized analysis.

Schönhage (1971) showed that the time needed to compute the gcd of two $l$-bit integers and the continued fraction of their quotient is, asymptotically, $l^{1+o(1)}$; the same suffices for expressing the gcd linearly in terms of the inputs. Using this result and an algorithm of Moenck (1973), the same asymptotic bounds hold for $l$-bit polynomials in $k[X]$.

In $\mathbb{Z}$ no one knows an NC algorithm for the greatest common divisor. The most practical parallel method is perhaps that of Brent and Kung (1983), who showed that the gcd of two $l$-bit positive integers may be computed in $O(l)$ time units using $O(l)$ circuit elements, each of which computes a particular six-bit Boolean function. Asymptotically, this result has been improved by Chor and Goldreich (1985), who showed that the gcd of two $l$-bit positive integers may be computed in $O(l/\log l)$ parallel time with $l^{O(1)}$ bit operations. Adleman and Kompella (1988) found that $O(\log l)$ depth is possible, with a subexponential number of gates.

In $k[X]$ computation of gcd's may be reduced to linear algebra; using this technique Borodin, von zur Gathen, and Hopcroft (1982) gave an algebraic NC algorithm for the problem. However, it is not a true NC algorithm, unless the characteristic of $k$ is small.

*Characters*

Since the multiplicative group of a finite field is cyclic, one may decide if an element $x$ of $\mathbb{F}_p^*$ is a square by raising it to the $(p-1)/2$th power and checking for 1. Although this algorithm takes $O(\log p)^3$ steps, it is not optimal. Perhaps this was sensed by Gauss, who called the method "impractical" (1801, §106).

A much better approach uses the reciprocity law for Jacobi symbols, together with the observation that $(\frac{-1}{n})$ and $(\frac{2}{n})$ depend only on $n$ modulo 8. Various algorithms of this type are known, which all take $O(\log n)^2$ steps to evaluate $(\frac{x}{n})$; the time bound seems to have been first proved by Collins and Loos (1982). Shallit (1989) has examined the worst-case

behavior of these methods, of which the best is an algorithm due to Lebesgue (1847); it is based on the least-remainder Euclidean algorithm and removes powers of 2 at each step. Using the Dedekind-Kühne reciprocity law, one can also compute $(\frac{g}{f})$ in $k[X]$ in $O(\log f)^2$ steps, as shown by Bach (1988b). These results all imply that for any $q$, the quadratic character in $\mathbb{F}_q$ may be evaluated in $O(\log q)^2$ steps.

When $m$ divides $q-1$ one may use Euler's criterion and compute $x^{(q-1)/m}$ to see if an $x$ in $\mathbb{F}_q^*$ is an $m$th power. This will take $O(\log q)^3$ steps, but as the case $m=2$ shows, better methods may be possible. So far none seem to be known.

Asymptotically, the Jacobi symbol of two $l$-bit numbers may be found in $l^{1+o(1)}$ steps by combining Gauss's algorithm (1817), which computes $(\frac{x}{n})$ from the continued fraction of $x/n$, with Schönhage's nearly linear-time continued fraction algorithm. The quadratic symbol in $k[X]$ has similar asymptotic complexity. Little is known about the parallel complexity of this problem beyond Fich and Tompa's result (1988) that an NC algorithm exists for quadratic residuacity in finite fields of small characteristic.

*Linear equations*

Even in its computational aspects, linear algebra is a vast area; I shall only mention a few results here that are useful in designing number-theoretic algorithms.

Linear equations in $\mathbb{Z}/(u)$ or $k[X]/(u)$ may be solved by the usual techniques of elimination, for which the cost will be $O(d^3 \log^2 u)$ for a $d \times d$ system. However, with recently improved methods for factorization and discrete logarithms, linear algebra has become a theoretical bottleneck in such procedures. The equations that one needs to solve in these applications are sparse, and Wiedemann (1986) found a useful randomized algorithm for solving such systems. Applied to an $d \times d$ system of linear equations in $\mathbb{F}_q$ with $w$ nonzero coefficients, his method takes $O(dw \log^2 q)$ steps. By a pretend-field technique, a similar result holds in finite rings, as shown by Lenstra and Lenstra (1987).

The system of congruences

$$x_1 \equiv a_1 \pmod{u_1}, \ldots, x_r \equiv a_r \pmod{u_r}$$

where $u_1, \ldots, u_r$ are pairwise relatively prime is particularly important. In $\mathbb{Z}$ or $k[X]$, it may be solved in $O(\log u)^2$ steps, where $u$ denotes the product of the moduli $u_i$; this time also suffices to compute the residue modulo each $u_i$. This result is due to Collins (1969); it shows that both directions of the isomorphism given by the Chinese remainder theorem are computable in quadratic time. If the moduli are not relatively prime, the same bound suffices to compute a solution or to show that none exists.

Asymptotically, both isomorphisms in the Chinese remainder theorem are computable in $l^{1+o(1)}$ steps, when $u$ is $l$ bits long. NC algorithms to do this are not generally available; such procedures are known for $\mathbb{Z}$ only when the moduli are small, and for $k[X]$ only when $k$ has small characteristic.

In general, a system of linear equations in $\mathbb{Z}$ or $k[X]$ can be tested for solvability in polynomial time; the same is true for computing a generating set for the free module of solutions to such a system. A survey of efficient algorithms for such problems has been given in the book of Schrijver (1986). Since an algorithm to test linear equations for solvability can decide relative primality, and one to solve such equations can be modified to compute the gcd, NC algorithms for such equations are not known, except for $k[X]$, in the special case when the characteristic of $k$ is small.

PROBLEMS IN FINITE FIELDS

In this section I discuss algorithms to count and find the solutions to various equations over finite fields. As motivation for such problems, the polynomial ring $k[X]$ can be thought of as a simpler analog to $\mathbb{Z}$, in which factoring is easy and the Riemann hypothesis has been proved (what more do you want?). As with basic arithmetic, results about $k[X]$ separate into two categories, depending on the characterisic of $k$. If this characteristic is small (e.g. $k = \mathbb{F}_2$) then problems such as polynomial factorization are solvable deterministically in polynomial time. If this characteristic is large, random polynomial time algorithms are the only generally applicable methods.

In this section I shall assume that $k$ is a finite field with $q$ elements and characteristic $p$, where $q = p^n$. For running time bounds when $q$ is not prime, $k$ is presumed to be realized as $\mathbb{F}_p[X]/(f)$, where $f$ is an irreducible monic polynomial.

*Square roots in finite fields*

This section discusses algorithms to compute square roots in $k$. I shall assume that $k$ has odd characteristic, for otherwise a square root, which always exists, may be found by inverting the matrix for the Frobenius automorphism. As noted by Berlekamp, Rumsey, and Solomon (1967), this takes $O(\log q)^2$ steps once the inverse matrix is found.

By modern standards, the earliest efficient algorithm for this problem was given by Tonelli (1891) for the case $k = \mathbb{F}_p$. (It also appears in papers by Shanks (1972), and Adleman, Manders, and Miller (1977)). It uses the fact that $k^*$ is cyclic, of order $2^s t$ with $t$ odd. Then there is a chain factorization

$$H = G_0 \subset G_1 \subset \cdots \subset G_s = k^*,$$

$H$ being the subgroup of order $t$. If $g \in k$ is not a square (half of the nonzero elements have this property), then since $g$ generates $G/H$, there is an $e$, $0 \le e < 2^s$, and an $h$ in $H$ for which $a = g^e h$. The bits of $e$ may be computed one by one, for if a number $e_i$ is formed from the $i$ least significant bits of $e$, the next bit is zero if and only if $ag^{-e_i} \in G_{s-i-1}$. Then $g^{e/2}h^{(t+1)/2}$ is a square root of $a$. Since $s = O(\log q)$, the expected time for the algorithm is $O(\log q)^4$. (Note that an algorithm for power residuacity more efficient than Euler's criterion would improve this.) However, if $s \le 2$, the algorithm can be made deterministic; for explicit formulas, which are given for $k = \mathbb{F}_p$ but work for any $k$, see the survey article of Lehmer (1969).

A more efficient algorithm was found by Cipolla (1903). (See also the paper of Lehmer (1969)). To compute a square root of $a$, one finds a field element $x$ for which $x^2 - 4a$ is not a square; since most other values of $x$ correspond to two nonzero $y$ with $y^2 = x^2 - 4a$, the genus zero Weil bound implies that one may choose $x$ at random and be successful about half of the time. Then $k[T]/(T^2 - xT + a)$ is a finite field of order $q^2$, in which the norm of $T$ equals $a$, so $T^{(q+1)/2}$ is a square root of $a$, computable in $O(\log q)^3$ steps. If $q = p^n$, this running time may be improved to $O((\log p + n)\log^2 q)$, by reducing the problem to a square root computation in $\mathbb{F}_p$.

Both of these algorithms, as well as the methods based on polynomial factoring, fail with probability about $1/2$. Peralta (1986) has found an algorithm that performs well when $q - 1 = 2^s t$ and $s$ is large; his algorithm takes $O(\log q)^3$ steps and has error probability roughly $1/2^{s-1}$. For all of these methods, however, one can tell in $O(\log q)^2$ steps whether one's random choice will work, so Peralta's method appears to be only of theoretical interest.

The deterministic complexity of this problem is still unresolved. If the ERH holds, $p$ has a quadratic nonresidue less than $2(\log p)^2$, which can be easily found using the Jacobi symbol algorithm. Then the Tonelli algorithm in $\mathbb{F}_p$ would take $O(\log p)^4$ steps, as shown

by Adleman, Manders, and Miller (1977). For computing square roots in $k$, $O(\log q)^4$ steps would also suffice, because one can deterministically reduce the problem to factorization of a quadratic polynomial over $\mathbb{F}_p$ within this time bound. Perhaps a deterministic algorithm using $O(\log q)^3$ steps can be found, by combining Cipolla's or Peralta's algorithm with the ERH, but none is known even for $\mathbb{F}_p$.

Asymptotically, computing square roots in finite fields does not seem to be any easier than polynomial factoring in general; if $q$ has $l$ bits, then an expected time of $l^{2+o(1)}$ steps is the best known. No one has found an NC algorithm for this problem, except for the case where $k$ has small characteristic.

Finally, Schoof (1985) found a deterministic algorithm to factor quadratic polynomials modulo $p$; if the polynomial is fixed, his procedure runs in polynomial time, but the degree is large. Nevertheless, this resolved a long-standing problem about writing a prime $p$ congruent to 1 modulo 4 as a sum of squares; since such an expression can be found directly once one knows a square root of $-1$ modulo $p$, it has a deterministic polynomial-time solution. (For other algorithmic problems concerning sums of squares, see Rabin and Shallit (1986)).

*Polynomial factorization*

The length of a polynomial in $k[X]$ depends on three parameters: its degree, the degree of the extension $k/\mathbb{F}_p$, and the characteristic $p$. It is therefore difficult to unequivocally identify the best ways to factor such polynomials. I shall merely describe methods that are conceptually simple and work well with classical arithmetic. As a rule of thumb, most of them need about $O(\log f)^3$ steps to split $f$.

These algorithms often use linear algebra, which is not available for factoring integers. More precisely, let $f$ be a polynomial in $k[X]$; then the Frobenius transformation $x \mapsto x^q$ is a $k$-linear map on $k[X]/(f)$, for which a matrix is easily found by computing $X^q, (X^q)^2, (X^q)^3, \cdots$ modulo $f$. This will take $O((\deg f + \log q)\log^2 f)$ steps; a similar technique to find the matrix of the absolute Frobenius $x \mapsto x^p$ will take $O((\deg f + n + \log p)\log^2 f)$ steps. Once its matrix is found, the transformation may be computed in $O(\log f)^2$ steps.

Now let $f$ be a polynomial to be factored. If $f'$ is the derivative of $f$, $g = \gcd(f, f')$ is a divisor of $f$. By applying factor refinement to $f = (f/g)\cdot g$, one can express $f$ as a product of squarefree polynomials (one has to carefully consider factors of the form $h^p$). If $k = \mathbb{F}_p$, or matrices for the Galois group are available, this will take $O(\log f)^2$ steps; in any case this can be done deterministically in polynomial time, so I shall assume henceforth that $f$ is squarefree.

Berlekamp (1967) gave the following algorithm to factor $f$. Letting $\phi(x) = x^q$ be the Frobenius map on $k[X]/(f)$, one computes the kernel $B$ of $\phi - 1$ using linear algebra. By the structure of $k[X]/(f)^*$, the dimension of $B$ is $r$, the number of distinct irreducible factors of $f$. Then if $g$ is a nonconstant element in $B$, for some $\alpha \in k$, $\gcd(g - \alpha, f)$ splits $f$. The time required to split $f$ is $O((\deg f + q)\log^2 f)$.

When $k$ is fixed, this algorithm runs in polynomial time. Berlekamp (1970) also found a deterministic polynomial time reduction to factoring polynomials over prime fields, so factoring a polynomial over $k$ is easy if $k$ has small characteristic. Berlekamp's proof is complicated; a simpler method can be had by adapting a method of Zassenhaus (1969) as follows. Let $\phi(x) = x^p$ be the absolute Frobenius on $k[X]/(f)$. One finds a nonconstant $g$ in the kernel of $\phi - 1$, and considers the ideal in $\mathbb{F}_p[Y]$ consisting of polynomials $F$ for which $F(g) \equiv 0 \pmod f$. This ideal is generated by a polynomial $G$, whose degree does not exceed that of $f$; it may be computed by finding an $\mathbb{F}_p$-linear dependence among $1, g, g^2, \ldots$ modulo $f$. $G$ will have all its roots in $\mathbb{F}_p$, and if $a$ is such a root, $\gcd(g - a, f)$

splits $f$. The time to compute $G$ is dominated by the time to find the matrix for $\phi$; it is $O((\log p + \deg f + n)\log^2 f)$.

For large fields randomization is convenient; the resulting algorithms are the only generally useful ones if the characteristic is large. Under the present assumptions, the fastest such method seems to be the following one, given by Cantor and Zassenhaus (1981). One considers $k[X]/(f)$ as an $\mathbb{F}_p$-linear space, and finds the kernel of $\phi - 1$, $\phi$ being the map $x \mapsto x^p$. Then one chooses a random element $g$ of the kernel, that is, a random linear combination of its basis elements, and computes $\gcd(g^{(p-1)/2} - 1, f)$; this will split $f$ with probability at least $1/2$. This is a Las Vegas algorithm; using the geometric distribution, the expected time until $f$ is split is $O((n \cdot \deg f + \log p)\log^2 f)$.

Another important method, which will not always factor $f$ completely but is often used as a subroutine, is that of distinct degree factorization. (Its origins are obscure.) Starting from the observation that

$$\prod_{t \in \mathbb{F}_q} (X - t) = X^q - X,$$

one finds the gcd of $f$ and $X^{q^i} - X$ for larger and larger values of $i$, thus extracting the product of all linear factors, then the product of all quadratic factors, and so on. If one uses a matrix for the Frobenius map instead of a power calculation, the method takes $O((\deg f + \log q)\log^2 f)$ steps.

All the running times above assume classical arithmetic. Asymptotically, they can be improved by replacing Gaussian elimination by faster linear algebra techniques such as that of Coppersmith and Winograd (1987). In most cases, however, it is even better to avoid linear algebra altogether. Rabin (1980b) designed a factorization method based on root finding; although not competitive by classical standards, it is fast asymptotically. Using ideas from this procedure, Ben-Or (1981) showed that the expected number of steps needed to completely factor an $l$-bit polynomial in $k[X]$ is $l^{2+o(1)}$. The best result on the parallel complexity of this problem is due to Borodin, von zur Gathen, and Hopcroft (1982), who gave NC algorithms to factor polynomials in $k[X]$ when the characteristic of $k$ is small.

Finally, the above results imply that the primality problem for $k[X]$ is solvable in deterministic polynomial time. Although results from which one could easily decide the irreducibility of a polynomial in $k[X]$ were given by Petr (1937), the first explicit algorithm running in polynomial time is apparently that of Butler (1954).

*Deterministic factorization methods*

No one knows a deterministic polynomial time procedure for factoring polynomials over finite fields; the best asymptotic result, due to Shoup (1988b), is that $f$ in $\mathbb{F}_p[X]$ may be factored in $p^{1/2+o(1)}(\deg f)^{2+o(1)}$ steps. However, algorithms to compute square roots in finite fields suggest an efficient deterministic procedure should exist, assuming the ERH. Not even such a conditional result is known; the best results in this direction apply either to special classes of polynomials or to special finite fields, and are discussed below.

Adleman, Manders, and Miller (1977) showed that if $d$ is a prime divisor of $p - 1$, then the Tonelli algorithm generalizes to compute $d$th roots modulo $p$ in deterministic polynomial time. Once a $d$th power nonresidue is found, the running time depends linearly on $d$, although this may be improved to $d^{1/2}$ using discrete logarithm techniques. Their algorithm extends without difficulty to computing $d$th roots in any finite field that has a $d$th root of unity; such a generalization was given by Mignotte (1980).

Assuming the ERH, Huang (1985) found efficient deterministic algorithms to factor cyclotomic polynomials modulo $p$, and more generally, to factor polynomials with abelian

Galois groups. He also found a partial solution to the problem of quickly constructing generating sets in large finite fields. His results on polynomial factoring have been generalized considerably by Evdokimov (1986), who found a deterministic polynomial time algorithm to factor polynomials with solvable Galois groups, and Rónyai (1989), who found algorithms with similar performance to factor polynomials whose splitting fields have small degree. The analysis of both methods assumes the ERH.

Rónyai (1988a) also took another approach. Starting from the observation that Tonelli's deterministic algorithm may be used to factor quadratic polynomials, which always have few factors, he found an efficient factoring method for "rough" polynomials. More precisely, he showed that the problem of factoring polynomials in $\mathbb{F}_p[X]$ with a bounded number of irreducible factors has, assuming ERH, a deterministic polynomial time solution.

Influenced by some methods of integer factorization, von zur Gathen (1987b) and independently Mignotte and Schnorr (1988) devised algorithms to factor polynomials modulo $p$, when $p-1$ is smooth. Their methods use Ankeny's theorem to find a generating set for $\mathbb{F}_p^*$, which in this case can be done in polynomial time assuming the ERH. (Later Rónyai (1988b) showed that finding certain roots of unity in $\mathbb{F}_p$ would suffice.) Bach and von zur Gathen (1988) have extended these results. The class of primes $p$ to which these generalizations apply is complicated, but it includes all primes for which $p+1$ is smooth; in particular, independently of any hypotheses, there is a deterministic polynomial time algorithm to factor polynomials modulo Mersenne primes. It would be interesting to replace $p+1$ by an arbitrary cyclotomic polynomial, but no such result is known. Rónyai (1989) has recently also found elliptic curve analogues of these results.

*Construction of finite fields*

The simplest way to specify a finite field of order $p^n$ when $n > 1$ is to choose an irreducible polynomial of degree $n$, although one may use other realizations for purposes of efficiency.

In his paper originating the theory of these fields, Galois (1830) recommended obtaining such a polynomial by trial and error. This is reasonable: about $1/d$ of the monic polynomials of degree $d$ are irreducible, so about $d$ trials should suffice. Rabin (1980b) analyzed such a method, using distinct-degree factorization to test for irreducibility. At least in the classical framework, some improvements can be made, as pointed out by Calmet and Loos (1980). To use their algorithm, one tests for repeated factors with $\gcd(f, f')$; if this succeeds, one uses Berlekamp's algorithm to find the number of distinct irreducible factors of $f$. As shown by Carlitz (1932), a random polynomial is squarefree with probability about $1-1/q$; the first step is especially likely to eliminate reducible polynomials if $q$ is small. Such an irreducibility test requires $O((d+\log q)\log^2 f)$ steps; using the geometric distribution, the expected time to find an irreducible polynomial is $d$ times this.

The best asymptotic result on this problem is due to Ben-Or (1981), who found an expected time of $l^{2+o(1)}$ steps to generate an irreducible $l$-bit polynomial of specified degree in $k[X]$. His algorithm also appears to be superior classically to that of Calmet and Loos when $k$ is small.

For various reasons a direct method would be useful; the possible existence of a deterministic polynomial time algorithm to synthesize finite fields is also of theoretical interest. Here the relevant results are the following. Independently, Evdokimov (1986) and Adleman and Lenstra (1986) showed that if the ERH holds, one can generate an irreducible polynomial of degree $n$ modulo $p$ in time $(n\log p)^{O(1)}$. Recently, Shoup (1988a) found an unconditional deterministic polynomial time reduction from generating irreducible polynomials to factoring polynomials; combined with Berlekamp's deterministic factorization algorithm, his result shows that the problem is in P when $p$ (or more generally the characteristic of

the field) is bounded.

Finally, although any two finite fields of the same order are isomorphic, one might wish to quickly find the isomorphism. This problem reduces to polynomial factoring, so in practice it is easily solvable with randomization. Evdokimov (1986) showed that a fast deterministic method would follow from the ERH, and recently Lenstra (1988) has shown that the problem has a deterministic polynomial time solution independently of any hypothesis. His method, based on finite ring theory, promises to be of great interest.

*Algorithms for elliptic curves*

The recent geometric trend in computational number theory can be traced to one source: a deterministic polynomial-time algorithm of Schoof (1985) for computing $N_p$, the number of points on an elliptic curve $E$ defined modulo an odd prime $p$.

Since $E$ is an abelian group, the Frobenius automorphism $\phi$, which raises each coordinate to the $p$th power, belongs to $E$'s endomorphism ring. If $N_p = p + 1 + t$, then in this ring, $\phi$ satisfies the equation

$$\phi^2 + t\phi + p = 0$$

(because of this relation, $-t$ is sometimes called the "trace of Frobenius"). Schoof's algorithm finds $t$ modulo many small prime values of $l$, and then recombines these results using the Chinese remainder theorem to produce $t$.

To do this, one reduces the above identity modulo $l$, as follows. Let

$$E[l] = \{P \in E : lP = 0\},$$

where the points $P$ are allowed to have coordinates in the algebraic closure of $\mathbb{F}_p$. Then

$$E[l] = \mathbb{F}_l \times \mathbb{F}_l,$$

where $\mathbb{F}_l$ denotes the finite field of $l$ elements; furthermore, $E[l]$ is invariant under the action of $\phi$. Consider $\phi$ now as a linear operator $\phi_l$ on this vector space; it must be true that

$$\phi_l^2 + t\phi_l + p = 0.$$

One now searches for a $t$ in $\mathbb{F}_l$ satisfying this identity. The search strategy is not fancy; one just tries every member of $\mathbb{F}_l$. However, there are some clever "data structures" involved, which use the following trick: a finite set of points on a curve, such as $E[l]$, may be represented by a polynomial that intersects the curve at those points. Using such a polynomial representation of $E[l]$, one can decide if $\phi_l^2 + t\phi_l + p$ annihilates $E[l]$ by deciding if, for a "general" point $P$ in $E[l]$, the three points $\phi_l^2(P)$, $t\phi_l(P)$, and $pP$ are collinear, which reduces to a determinant calculation.

To analyze this algorithm, one uses the rational form of the addition law to show that the group operation can be done in $O(\log p)^2$ steps; doing this, Schoof found a running time of $O(\log p)^9$. Later Lenstra observed that this could be replaced by $O(\log p)^8$; for a proof see the report of van der Lingen (1987). Recently Atkin has devised some practical improvements to the algorithm, using the order of $\phi_l$ in the 2-dimensional projective linear group over $\mathbb{F}_l$ to exclude false values of $t$; according to Schoof (1988), these improvements make the algorithm practical for primes $p$ up to 50 decimal digits.

Asymptotically at least, the running time can be improved; Lenstra also showed that if $p$ is $l$ bits long, then $N_p$ may computed in $l^{5+o(1)}$ steps. Nothing seems to be known about the parallel complexity of this problem, and it seems doubtful that an NC algorithm could be found without finding such an algorithm for exponentiation modulo $p$.

Besides leading to algorithms for primality testing and factoring, this algorithm has sparked an interest in constructive aspects of algebraic curves and abelian varieties. Other ways of representing elliptic curves were studied by Chudnovsky and Chudnovsky (1985), and Montgomery (1987). Schoof's algorithm has been generalized by Pila (1987), who showed that if $C$ is a smooth projective curve defined over $\mathbb{Q}$, then there is a polynomial time algorithm to compute the number of $\mathbb{F}_p$-points of $C$. Unfortunately the degree of the polynomial depends severely on the curve $C$. Finally, Cantor (1987) has given explicit addition formulas for algebraic groups associated with certain hyperelliptic (genus 2) curves, and Adleman and Huang (1987) have announced a generalization of Schoof's point counting techniques to these varieties.

*Limited-randomness algorithms*

Weil's theory of algebraic curves over finite fields is also useful for a deeper study of randomized algorithms in number theory, which explains why these methods work so well in practice and might lead ultimately to deterministic polynomial time methods. This study starts from the observation that most of the efficient randomized techniques require numbers with specified multiplicative properties; for example, the Tonelli algorithm to compute square roots modulo $p$ uses a quadratic nonresidue. The Weil theory implies that such numbers are "well mixed". For example, if $f$ is a polynomial, and $\chi$ a nontrivial character on $\mathbb{F}_p^*$, then the joint distribution of $\chi(f(x))$ for different, but correlated, values of $x$ approximates stochastic independence. Therefore, the successive random trials in an randomized algorithm need not be fully independent for the algorithm to be useful, and it is interesting to see just how far the requirement for randomness can be reduced.

The following analogy seems appropriate. Consider an urn containing red and white balls in equal proportions, in which one wishes to find a red ball. Knowing nothing about the distribution of the colors, one should repeatedly choose balls at random, for any other choice strategy will do poorly on some distribution. If, however, one believes the balls to be well mixed, then it is useful to choose a random ball and a few of its neighbors. This corresponds to choosing a random seed and using successive iterates from a pseudo-random number generator for the random trials. Indeed, if the mixing is good throughout, one might profitably use a strategy that is altogether deterministic; this is the intuitive idea of algorithms based on the ERH, which implies that a particular search plan will be successful.

As an illustration of this type of analysis, consider the Cipolla algorithm to compute a square root of $a$ modulo $p$. If one chooses a seed $x$ and simply uses the first $t$ numbers in order, starting from $x$, for the random trials, then the algorithm will fail when

$$x^2 - 4a, (x+1)^2 - 4a, \ldots, (x+t-1)^2 - 4a$$

are all nonzero squares modulo $p$. Put another way, the algorithm fails when $x$ lies under the algebraic curve given by the $t$ equations

$$y_1^2 = x^2 - 4a, y_2^2 = (x+1)^2 - 4a, \ldots, y_t^2 = (x+t-1)^2 - 4a.$$

Since there are $t$ equations of degree 2, the curve's arithmetic genus is $O(t2^t)$, and Weil's theorem, plus a counting argument, shows that the probability of choosing a "bad" $x$ is $1/2^t + O(t/\sqrt{p})$. The optimal value of $t$ is $O(\log p)$; for this choice the simple incrementing strategy will fail with probability $O(p^{-1/2} \log p)$, which is exponentially small.

Similar results hold for the other randomized algorithms presented in this section; in every case one can tailor a deterministic sequence generator to the algorithm's needs and show that a number of random bits equal to the input length suffice for exponentially small

error probability. Analyses of square root algorithms were given by Bach (1987, 1988b), polynomial factoring algorithms by Bach and Shoup (1988), and algorithms to generate irreducible polynomials by Shoup (1988a).

Surprisingly, this type of analysis is not improved by more complicated random number generators. Although all of the results still hold when incrementing is replaced by successive use of a multiplicative congruential sequence, this modification does not lead to sharper bound. It would be of interest to extend these results to pseudo-random sequences computed in structures different than those provided by the original problem.

## PRIMALITY TESTING

Since Fermat, it has been known that if $p$ is prime, then any integer $a$ relatively prime to $p$ must satisfy $a^{p-1} \equiv 1$ modulo $p$. This gives a simple necessary condition for $p$'s primality that is checkable in $O(\log p)^3$ steps. However, it is not foolproof; in particular, there are composite numbers, first found by Carmichael (1912), on which this test fails for every $a$. (The smallest Carmichael number is 561, although a more interesting one is 1729.) Much of the recent work on primality testing may be viewed as an attempt to replace Fermat's condition by something equally fast, yet more reliable.

From a complexity theory viewpoint, the problem has the following status. Evidently, the set of composite numbers is in NP; Pratt (1975) showed that the same holds for the set of primes. Miller (1976) showed that the set of primes is in P if the ERH is true, and by a result of Solovay and Strassen (1977), the set of composite numbers is in RP. That is, there is an efficient randomized algorithm to prove composite numbers composite, and provide strong statistical evidence that prime numbers are prime. The existence of a zero error (Las Vegas) random polynomial time algorithm for primality remained open for many years; it was recently resolved by Adleman and Huang (1987) using complicated mathematical tools, showing that the primes are in ZPP. I shall examine each of these results in turn.

In this section, I assume that $p$ is odd, but not necessarily prime.

### *Tests based on Fermat's theorem*

It has long been known that the group of integers modulo $p$ is cyclic of order $p-1$ if and only if $p$ is prime. Knowing the factorization of $p-1$ and a generator $g$ of this group, one may prove $p$ prime by proving that every prime divisor $p'$ of $p-1$ is prime and verifying that $g^{(p-1)/p'} \not\equiv 1$ modulo $p$ for each such $p'$. This leads to a recursive primality algorithm, requiring a factorization for each number tested. Pratt pointed out that an NP algorithm can "guess" the generators and the factorizations, since a number's factorization is roughly as long as the number itself. In recent work, Pomerance (1987b) has investigated the question of how long a certificate of primality needs to be.

Miller's test works as follows. One picks a number $a$, and evaluates $a^{p-1}$ modulo $p$ by the usual repeated squaring algorithm, computing $b = a^m$ for odd $m$ first, then $b^2, b^4$, and so on. If $p$ is prime, the sequence of powers of $b$ will end with 1 and will never contain a square root of 1 different than $\pm 1$ modulo $p$. The first requirement is Fermat's test, and the second checks for Carmichael numbers; if the computation satisfies both, one reports that $p$ is prime. A number $a$ that causes the algorithm to falsely assert the primality of a composite number $p$ is called a *liar* for $p$.

When $p$ is composite, some proper subgroup of $\mathbb{Z}/(p)^*$ contains all the liars. So, if the ERH is true, the explicit Ankeny theorem implies that one can prove $p$ prime by running Miller's test for each $a < 2(\log p)^2$. This would lead to an $O(\log p)^5$ method to test primality. However, even this hypothetical bound is probably an overestimate. For example, Pomerance, Selfridge, and Wagstaff (1980) showed that to test values of $p$ up to 32 bits long, one may use $a \leq 11$; the ERH bound would require $a \leq 983$.

Many other modifications and improvements have been made to Fermat's test. For example, Solovay and Strassen (1977) proved that when $p$ is composite, a randomly chosen $a$ fails to satisfy $(\frac{a}{p}) \equiv a^{(p-1)/2}$ with probability at most $1/2$; this test was also given by Lehmer (1976) without the probability estimate, and Vélu (1978) in a deterministic version using the ERH. In retrospect, an even better probability estimate holds for Miller's test; Rabin (1980a) showed that its chance of failure is, for $p$ large, at most $1/4$. These results lead to simple randomized primality tests using $O(\log p)^3$ steps. (Minor variants, with almost identical running times, were published by Atkin and Larson (1982) and Lehmann (1982); the last can err on all inputs.)

Monier (1980) computed the precise probability of error of both algorithms, which depends on the structure of $\mathbb{Z}/(p)^*$. He also found, as did Pomerance, Selfridge, and Wagstaff, that any liar for Miller's test is also a liar for Solovay and Strassen's test. Therefore for any $p$ the error probability for Miller's test is at most that of Solovay-Strassen's. (See the paper of Erdös and Pomerance (1986) and references therein for more results on liars.)

Several interesting open problems are connected with these primality tests. For example, although the error probabilities for both tests are bounded by simple constants, it is unknown if either bound is tightly approached for infinitely many composite $p$. It is unknown if there are infinitely many Carmichael numbers, although this is widely believed. Finally, although the empirical data of Pomerance, Selfridge, and Wagstaff suggest that no fixed number of bases $a$ will suffice in either algorithm, no one has ever proved this.

One may also consider a limited-randomness approach to randomized primality testing. Bach (1987) showed that if one takes a random seed $a$ modulo $p$ and tries Miller's test using bases $a, a+1, \ldots, a+t$, with $t = O(\log p)$, then the error probability is $p^{-1/4+o(1)}$. (This result is only asymptotic, as the exponent converges to $-1/4$ rather slowly.) Surprisingly, there is no such exponentially small error probability known for the test of Solovay and Strassen.

The results cited above imply that, asymptotically, $l$-bit numbers can be tested for primality in $l^{4+o(1)}$ steps if the ERH is true, or in $l^{2+o(1)}$ steps with a randomized algorithm. Not much is known about the parallel complexity of primality testing; the best sequential algorithms use exponentiation modulo $p$, which does not have an NC algorithm. Adleman and Kompella (1988) have given a randomized parallel algorithm to test an $l$-bit number for primality in time $(\log l)^{O(1)}$, using a subexponential number gates.

*Algorithms to prove primality*

The improvements to Fermat's test suggest that we now have a great algorithm to test primality; the only problem is that its answer cannot be wholly trusted. To correct this problem, many algorithms to verify primality have been designed, but until relatively recently, such verifications presented a difficult problem. It is to the credit of research in the last ten years that one can now check primality by algorithms that, heuristically at least, run in polynomial time. Below I shall only survey these later developments; for primality algorithms not covered here, the reader should consult the papers of Williams (1978) and Lenstra (1981).

A major step toward the goal of quickly proving primality was taken by Adleman, Pomerance, and Rumely (1983), who showed that a deterministic algorithm could test the primality of $p$ in $(\log p^{\log\log\log p})^{c+o(1)}$ steps. (Note that such a result is not sensitive to the distinction between bit operations and algebraic operations.) They also presented a randomized version of their algorithm, for which the same expected time is needed to prove $p$ prime; for this version they conjectured that $c = 1/\log 2$. I shall not present any details of their algorithms, but only summarize them as follows. By performing many pseudoprime

tests, which generalize Fermat's criterion to rings of algebraic integers, one may collect enough information about $p$'s possible nontrivial divisors to restrict them to a small set. Extracting information about $p$'s possible divisors in this way involves an ingenious use of class field theory, through explicit reciprocity laws in cyclotomic fields. Once the possible divisors of $p$ have been reduced in number, they may be enumerated and tested one by one; if none divide $p$, then $p$ is prime.

As originally stated this algorithm was not practical, as it required finite ring computations that, although insignificant asymptotically, became burdensome in practice. An important modification to the test was made by Cohen and Lenstra (1984), who simplified the rings in which the algorithm works, allowing it to be of practical use. With these modifications, Cohen and Lenstra (1987) reported routinely testing numbers up to about 200 digits in size. However, even the improved method is one of exclusion, and to verify a primality proof that it produces one must in essence repeat the computation.

The more recent primality algorithms provide much shorter proofs that are easier to check; they rely on the theory of elliptic curves and their higher-dimensional generalizations, called abelian varieties. Goldwasser and Kilian (1986) found an elegant method that may be thought of as an elliptic curve version of Pratt's results on primality. They based their algorithm on the following ideas. If $p$ is prime, elliptic curves modulo $p$ have roughly $p$ points, and if enough numbers of the form $2p'$, $p'$ prime, lie in the appropriate range, one can choose a random curve, compute its order by Schoof's algorithm, and hope to find one of order $2p'$. A group element of order $p'$, whose existence is easy to verify once one knows that $p'$ is prime, cannot exist unless $p$ is prime. This observation reduces the primality of $p$ to the primality of $p'$, and since $p'$ is roughly $p/2$, this recursion stops quickly.

Theoretically, the main difficulty with this algorithm is in proving that a short interval such as

$$(\frac{p+1}{2} - \sqrt{p}, \frac{p+1}{2} + \sqrt{p})$$

contains enough primes; to date, the best analytic results on this matter do not guarantee the existence of *any*. If a conjecture such as Cramér's (1937) were true, however, this algorithm would show that the set of primes is in RP, hence in ZPP since the composites are already in RP. But nothing close to Cramér's presumed bound is known, and Goldwasser and Kilian were only able to demonstrate that their algorithm quickly proves the primality of all numbers outside a set of density zero.

Practically, the main difficulty with this algorithm is the use of Schoof's procedure, which takes $O(\log p)^8$ steps. For example, combining Goldwasser and Kilian's heuristic running time bound with Cramér's conjecture, the expected time needed to prove $p$'s primality is presumably $(\log p)^{12+o(1)}$, which is too great for practical use.

An algorithm that avoids the theoretical difficulties given above has been announced by Adleman and Huang (1987). Their idea is to replace elliptic curves by two-dimensional abelian varieties associated with curves of genus 2. Over $\mathbb{C}$, such varieties are homeomorphic to the product of two tori, and therefore by Künneth's formula have Betti numbers $\beta_i = \binom{4}{i}$. Thus the number of points modulo $p$ on such varieties is $p^2 + O(p^{3/2})$, and analytic number theory now does supply a proof that the appropriate interval is rich enough in primes. This reduces the primality of $p$ to the primality of a larger number $p'$, which is about $p^2$. In a "multiply-and-conquer" procedure perhaps unique to computer science, the process is iterated three times to produce another prime $r$ near $p^8$. However, the new prime $r$ is distributed over a large enough interval that it has a reasonable chance of being proved prime by the Goldwasser-Kilian procedure.

This algorithm resolves a major open question in complexity theory and shows that the set of primes is in ZPP. Unfortunately, it is not practical. Although it asserts the existence of a constant $c$ such that primality can be proved with high probability in $O(\log p)^c$ steps, no one has ever estimated $c$, and a presumed constant factor of $8^{12}$ for the final prime test alone certainly gives one pause.

A solution to the practical difficulties associated with the Goldwasser-Kilian algorithm was devised by Atkin. His algorithm avoids Schoof's procedure in the following way. Letting $K_{-D}$ denote the quadratic field $\mathbb{Q}(\sqrt{-D})$, one finds a value of $D$ for which $p$ splits into two principal ideals of degree 1 in $K_{-D}$'s ring of integers. This is done by trying values of $D$ in order, and performing computations in $K_{-D}$'s class group. These latter techniques are standard in algebraic number theory and I shall not go into them here; for such algorithms, see the paper of Schoof (1984). The difficult issue to resolve is how long the search for such a $D$ should take. Once an appropriate value of $D$ and a generator $\pi$ for the ideal is known, one forms $q = p + 1 + (\pi + \bar{\pi})$ and hopes that $q$ factors as $sp'$, where $s$ is small and $p'$ is apparently prime. Only at this point, when the factorization is successful and the extra work will be worthwhile, does one construct an elliptic curve. This uses techniques from class field theory, relating certain extensions of $K_{-D}$ to elliptic curves, that allow one to produce a curve with $q$ points modulo $p$. (For a brief introduction to this topic, see Serre's article in Cassels and Fröhlich (1967)). The recursion then proceeds as in the original algorithm, using the primality of $p'$ to prove $p$ prime. Lenstra and Lenstra (1987) have given a heuristic argument that the expected time for Atkin's method to prove $p$ prime is $(\log p)^{6+o(1)}$. Asymptotically, it is conjectured that $l^{4+o(1)}$ steps suffice for an $l$-bit number; modifications leading to the latter bound are due to Shallit. Atkin's algorithm, which is described in detail by Morain (1988), is the most practical algorithm for proving the primality of large numbers. Morain's implementation allowed him to find a primality proof for $[(1+\sqrt{2})^{1901}) + (1-\sqrt{2})^{1901})]/2$, a number of 728 digits.

Finally, another long-standing problem regarding the complexity of the set of primes has been settled. The question deals with the so-called "polynomial immunity" of the set of primes, and asks if there an infinite set of prime numbers in P. This has long been suspected, as Mersenne numbers, which have the form $2^p - 1$, $p$ prime, have a deterministic polynomial time algorithm for primality, given by Lehmer (1930). However, no one knows if there are infinitely many Mersenne primes, although heuristic arguments such as that of Shanks (1978) suggest this. Pintz, Steiger, and Szemerédi (1988) found another infinite set of primes with a deterministic polynomial-time primality test, thus showing that the primes are not polynomially immune.

## INTEGER FACTORIZATION

Besides being inherently attractive to number theorists, the problem of factorization has attracted attention in computer science for two reasons. First, it is currently much easier to generate a pair of large primes than to factor their product, and many cryptographic schemes have been proposed whose security rests on this fact. Second, integer factorization offers an intriguing example of a problem that seems to lie midway between P and NP.

### *The complexity of factoring*

Before going on to factorization algorithms, it should be explained why this problem is intriguing in a structural sense. The decision problem associated with factorization involves recognizing pairs $(x, n)$ where $n$ is a number with some divisor less than $x$. Evidently, the set of such pairs is in NP (guess a divisor), and by Pratt's results it is also in co-NP (guess a primality proof for each divisor). Hence, if this set were NP-complete, the surprising consequence that NP equals co-NP would follow.

It is, however, widely suspected that factorization is computationally intractable, in the sense of having no polynomial time algorithm. If true, this would have consequences for randomized complexity in general, not just for number theory. Yao (1982), taking this intractability as a suitably formalized hypothesis, showed that any set in RP would be recognizable deterministically in subexponential time. Boppana and Hirschfeld (1985) extended this result to BPP (analogous results hold for the discrete logarithm problem as well).

Although factoring seems hard, there is an efficient way to generate random instances of the factorization problem. Using the random bisection idea, Bach (1988a) showed that, assuming the ERH, there is an efficient procedure to generate a random integer of a given length, together with its factorization.

Factorization is also interesting because no one can prove that deterministic algorithms for this problem are as fast as randomized ones. The best proved running time for a deterministic algorithm to factor $n$ is $n^{1/4+o(1)}$, due to Shanks (1971); if the ERH holds, this bound can be improved to $n^{1/5+o(1)}$, as shown by Schoof (1984). These running times are exponential in the length of $n$; there are, however, randomized algorithms that take expected time bounded by a small power of

$$L(n) = e^{\sqrt{\log n \log\log n}}.$$

The best proved running time for a randomized factoring algorithm is $L(n)^{\sqrt{4/3}+o(1)}$, due to Vallée (1989), although A. Lenstra (1987) has shown that $L(n)^{1+o(1)}$ would follow from the ERH.

Finally, factorization differs from hard optimization problems such as the traveling salesman problem in that the solution is immediately checkable. Therefore, one may be satisfied to use an algorithm whose running time estimates are only supported by heuristic arguments, and not rigorously proved. Most of the useful factorization algorithms are of this type.

The possibility of error, which plagues primality testing, is not a problem in factorization. As a practical matter, primality tests are faster than factoring algorithms, and random bisection suggests that prime factors are usually much smaller than their products. Theoretically, it follows from the form of Pratt's primality certificates that any algorithm for factoring integers can be made error-free without drastically affecting its performance; this was pointed out by Tompa (1983).

*Methods of factoring*

As with any algorithmic problem, factorization may be done in many ways, and it is useful to have a gamut of algorithms, ranging from naive (but simple to implement) to sophisticated (but difficult to implement). To show that even naive methods could be useful, it helps to distinguish two versions of the factorization problem, as pointed out by Pomerance (1988). In "natural number" factorization, one wishes to to factor a number chosen for reasons unrelated to the difficulty of factoring. Because the instance is not expected to be unusually difficult, one should try simple methods first; details of such advice have been given by Brillhart, Montgomery, and Silverman (1988). In "unnatural number" factorization, one wishes to factor a number chosen so as to guarantee that the problem will be onerous. One such situation is in decoding messages encoded by a scheme such as Rivest, Shamir and Adleman's (1978); since such coding techniques rely on the purported difficulty of factoring, a cryptanalyst can be assured that any particular factorization required is, by design, not easy. Another such situation is in factoring the "benchmark" numbers tabulated by Brillhart

and his co-authors (1988); such numbers help gauge the performance of factoring algorithms, precisely because they have resisted factorization for so long. For unnatural numbers simple methods will surely fail, and one should proceed directly to the most sophisticated method.

Below I shall restrict discussion to a limited set of factorization algorithms. For surveys of factoring, information about other methods, and more references, see the articles of Guy (1975), Williams (1983, 1984) and Wunderlich (1988).

The oldest factorization method is that of trial division: one simply tries all primes in order until the divisors appear. Because a list of primes up to $m$ can be constructed in $m^{1+o(1)}$ steps (for results on this problem see Pritchard (1983)), and any composite number $n$ must have a prime divisor less than or equal to $\sqrt{n}$, this method takes time $n^{1/2+o(1)}$ in the worst case. One may improve the method's average behavior by noting that whenever a divisor is discovered, one should test its cofactor for primality. Then the running time of the method becomes roughly proportional to the second largest factor of $n$. The random bisection idea suggests why even with this improvement, the algorithm is unlikely to be have a subexponential running time. Knuth and Trabb Pardo (1976) have computed the median running time; it is about $n^{0.21}$.

Pollard (1975) published an important heuristic algorithm called the "rho" method, which relies on pseudo-random properties of the iteration $z \leftarrow z^2 + c$. Over an algebraically closed field, this iteration is equivalent to the logistic map that has been extensively studied in nonlinear dynamics; for this connection see the survey of Lauwerier (1986). Pollard's algorithm is based on the following intuitively appealing argument.

Suppose that numbers produced by this iteration modulo $n$ were independent and uniformly distributed samples from the set $\{0 \ldots n-1\}$. Then if $p$ is a prime divisor of $n$, one expects by the birthday problem that two of the iterates would agree modulo $p$ after $O(\sqrt{p})$ iterations; since the iteration function is a polynomial, corresponding iterates from this point on would also agree. A space-efficient cycle detection trick due to Floyd will find such a pair, which can be detected because its difference will have a nontrivial gcd with $n$. (For improved versions of the algorithm see the papers of Brent (1980) and Gold and Sattler (1983)).

Since every number $n$ has a prime divisor less than $n^{1/2}$, the running time of the method should be $n^{1/4+o(1)}$. Unfortunately, this has never been proved, although experimental evidence supports such a belief. The best rigorous result is the following: Bach (1988c), using the Weil theory, proved that when $z$ and $c$ are chosen at random modulo $n$, a factor $p$ will be detected with probability at least $\Omega(p^{-1} \log^2 p)$.

Several important factorization algorithms use group theory, particularly the idea that the factorization of $n$ is related to the decomposition of various groups associated with $n$.

The simplest such algorithm is the so-called "$p-1$" method, first published by Pollard (1974). It is easiest to explain in the special case where $n = pq$, a product of two distinct primes. First, one chooses a random integer modulo $n$; it may be assumed relatively prime to $n$, for otherwise $n$ is split. As the Chinese remainder theorem gives a factorization of $\mathbb{Z}/(n)^*$ into two cyclic groups, of orders $p-1$ and $q-1$, it may be imagined that one has simply chosen the components in each factor separately. Knowing a multiple $E$ of $p-1$, one computes $y = x^E$ modulo $n$. In the first component, $y$ will be 1, and in the other component, one hopes this is not the case. If the second component is not 1, one splits $n$ with $\gcd(y-1, n)$. The entire process takes $O(\log E \log^2 n)$ steps.

The difficulty of this method, as with all such algorithms, is in finding a suitable value of $E$. One approach is to choose a bound $B$ and hope that $p-1$ splits entirely as a product of primes less than $B$. If one takes $E$ to be the product of all such primes, raised to appropriate powers, then the prime number theorem implies that $\log E = O(B \log n)$,

which is satisfactory if $B$ is small. However, it is atypical for $p-1$ to be smooth.

Faced with this problem, researchers generalized the method in various ways, which in hindsight involve more general and more exotic groups. For example, since the multiplicative group of any finite field is cyclic, one may use the subgroup of order $p+1$ contained in $\mathbb{F}_{p^2}^*$, into which a random element may be placed by dividing it by its conjugate. For a description of this algorithm, based on recurrent sequences, see the paper of Guy (1975).

For this algorithm it is crucial to compute the Frobenius automorphism of $\mathbb{F}_{p^2}$ differently than as a $p$th power, and the usual definition of this field via quadratic extensions provides such a technique. Extending this to $\mathbb{F}_{p^k}$ is the principal difficulty in devising an arbitrary "cyclotomic" method that can extract a factor $p$ of $n$ when $\Phi_k(p)$, the $k$th cyclotomic polynomial evaluated at $p$, is smooth. A solution to the problem, for which one needs the ERH, was found by Bach and Shallit (1989). Although this result unifies the description of several factoring methods and is the jumping-off point for further work in polynomial factorization, it is only of interest theoretically; the useful algorithms in the family are the $p \pm 1$ methods.

Of course, one is not restricted to using unit groups of finite fields in this approach. A major step forward in this regard was taken by H. Lenstra (1987), who realized that algebraic groups such as elliptic curves would serve the same purpose and also have the desirable property that their order could be randomized within the range $(p+1-2\sqrt{p}, p+1+2\sqrt{p})$. His algorithm is quite elegant and nicely illustrates the pretend-field technique.

It works as follows. To factor $n$, one selects coefficients $a$ and $b$ at random and considers the curve defined by the the affine equation $y^2 = x^3 + ax + b$. If the right hand side of this equation has distinct roots modulo some prime divisor $p$ of $n$ (this is likely and in any case easy to verify), then one may attempt to annihilate a point in the elliptic curve group by multiplying it by a suitably chosen product of primes less than $B$. In general, the solution sets of such equations modulo $n$ do not have a group structure, but since the group operations are rational, one can pretend that $n$ is prime, which will have the effect of doing operations in the elliptic curve modulo $p$. However, the identity element of the curve modulo $p$ is a point at infinity, which cannot be produced by rational operations unless one divides by zero. It is intuitively unlikely that this division by zero will happen simultaneously for all prime divisors of $n$; more often, some division will fail by discovering a non-unit different from modulo $n$, and at this point one splits $n$.

It now remains to choose the smoothness bound $B$. The tradeoff is between making it large (which allows more curves to be useful) and between making it small (which makes the test for each curve less onerous). Without going into details, the best value for $B$ leads to a heuristic bound of $L(p)^{\sqrt{2}+o(1)}$ for the time required to extract a prime factor $p$ of $n$. Notably, this estimate uses only heuristic assumptions about the density of smooth numbers near $p$, and no unproved hypotheses about elliptic curves.

The elliptic curve algorithm shares with trial division and the rho method the desirable property that its running time is sensitive to small prime factors, which get removed first. The best known factoring algorithm, the quadratic sieve of Pomerance (1984, 1985) does not have this property but instead has an expected running time depending only on the size of $n$. As it will not perform better on numbers with small divisors, it is ordinarily only used after some time has been spent on other methods.

The quadratic sieve is the latest development in a group of algorithms that may be termed the *quadratic-residue* family, which all do the following three steps. First, one generates many quadratic residues modulo $n$ together with their square roots. Second, by factoring some of these residues using primes less than $B$, one finds congruences of the

form $\prod_{p<B} p^{e_p} \equiv r^2$. Third, using linear algebra on the exponents modulo 2, one combines the congruences multiplicatively to find $x$ and $y$ with $x^2 \equiv y^2$, and tries to split $n$ with $\gcd(x \pm y, n)$.

Pomerance (1984, 1987a) has reviewed and analyzed many of the known variants; I shall concentrate here on the quadratic sieve as recent experience has shown that this version runs most quickly in practice. In particular, the quadratic sieve seems to have now supplanted the continued-fraction algorithm of Morrison and Brillhart (1975) as the method of choice for large number factorization.

The basic idea of the quadratic sieve is to generate quadratic residues using a polynomial such as

$$f(X) = (X + \lfloor \sqrt{n} \rfloor)^2 - n$$

whose values must be squares modulo $n$. If $x$ is small then $f(x)$ will be about $\sqrt{n}$. For the second phase, one could simply plug in $x$ and factor the result; a much better method results from noting that if $p$ divides $f(x)$, then $p$ also divides $f(x + ip)$ for all integers $i$. To take advantage of this, one starts with a list of all values of $f(x)$ for $x$ in some range, computes the roots of $f$ modulo $p$, then uses these roots to identify, in every interval of length $p$, two $x$-values for which $p$ divides $f(x)$. After this is done for all primes $p$ less than $B$, values of $f(x)$ with enough divisors are the desired residues. In practice, prime powers must also be taken into account, and it is convenient to start with an array containing single-precision values of $\log f(x)$, subtracting $e \log p$ to mark a residue as divisible by $p^e$. At the end of this process, array elements that are close to zero are identified and the corresponding values of $f$ are factored.

As in the analysis of the elliptic curve algorithm, one must choose an appropriate value of $B$, and estimate the running time. Here practice and theory diverge somewhat. In practice, the running time of the algorithm is dominated by the cost of the sieving phase, as a simple elimination procedure suffices to then find $x$ and $y$ with $x^2 \equiv y^2$. One should therefore just choose $B$ to minimize the expected time of the first phase, and if this is done the running time for sieving is conjectured to be $L(n)^{1+o(1)}$. However, in analyzing this version of the algorithm under plausible heuristic assumptions, the time for Gaussian elimination cannot be ignored and thus $L(n)^{\sqrt{9/8}+o(1)}$ steps would be needed to factor $n$. Theoretical improvements in the elimination procedure, based on ideas of Wiedemann (1986), allow one to modify the algorithm so that under reasonable assumptions, it has a running time of $L(n)^{1+o(1)}$.

Remarkably, the running time of the quadratic sieve algorithm equals that of the elliptic curve algorithm in the special case when $n$ has two factors of equal size. More remarkably, there are other factorization methods with the same performance; see the papers of Schnorr and Lenstra (1984), Coppersmith, Odlyzko, and Schroppel (1986), and A. Lenstra (1987).

Perhaps this agreement is evidence that about $L(n)$ steps are needed to factor $n$ in general, but no one has ever proved such a statement. If true, it would imply that P $\neq$ NP. More to the point, perhaps, is the observation that all of the algorithms cited are based on similar techniques, and the same tradeoffs must be resolved in optimizing their performance. Under fairly general assumptions, any algorithm using these techniques will have a running time bounded by a power of $L(n)$ (see, e.g., the paper of Bach (1989b)); why the power needs to be 1 is much less clear.

Out of all the methods that run in time near $L(n)$, quadratic sieve performs the best in practice. It is superior because a typical step in its execution is a single-precision subtraction, compared with other algorithms whose typical step is a group multiplication. Although negligible asymptotically, these group multiplications typically take $O(\log n)^2$ steps, and

cannot be done with one machine instruction. In addition, the quadratic sieve is attuned to the architecture of vector machines; on such computers it can be sped up substantially. It can also be easily parallelized, by letting each processor work on a different interval of $x$-values. Finally, there is also the possibility of using different polynomials; this variant, discussed by Pomerance (1985), has been used in all recent record-breaking factorizations.

The record for factorization is currently held by Lenstra and Manasse (1989), who recently factored $(2^{353}+1)/3$, a 106-digit number, using the quadratic sieve. This computation is notable because it involves the cooperation of research groups at many sites, who have contributed otherwise idle time from their machines.

Recent papers by the following authors contain implementation details of factoring algorithms: Brent and Pollard (1981) on the rho method, Williams (1982b) and Montgomery and Silverman (1988) on the $p \pm 1$ methods, Brent (1985) and Montgomery (1987) on the elliptic curve method, and Davis, Holdridge, and Simmons (1985), Gerver (1986), and Silverman (1987) on the quadratic sieve. In connection with the last algorithm, ideas developed for the continued fraction method are very useful; Pomerance and Wagstaff (1983) and Wunderlich (1985) have written recent reports on this topic.

*Equations in rings*

Various equation-solving problems in finite rings are related to factorization. Usually these problems are solvable in random polynomial time when $n$ is prime and therefore, using the Chinese remainder theorem and Hensel's lemma, when the factors of $n$ are known; it is interesting to ask if one needs to factor $n$ to solve them. In some cases this is true. For example, Rabin (1979) showed that an algorithm to solve the congruence $x^2 \equiv a$ modulo $n$ could be used to factor $n$ in random polynomial time, and based an encryption scheme on this fact.

Rivest, Shamir, and Adleman's cryptographic scheme (1978) relies for its security on the following conjecture: when $e$ is relatively prime to $\varphi(n)$, the congruence $x^e \equiv a$ cannot be solved modulo $n$ without factoring $n$ or doing some other computation equivalent in difficulty. In support of this conjecture, Rivest, Shamir, and Adleman showed that several cryptanalytic techniques applicable to their method were equivalent to factorization in difficulty. For example, knowing $n$ and $e$, one could decode messages if one knew $\varphi(n)$, but Miller (1976) showed that knowledge of $\varphi(n)$ allows one to easily factor $n$. However, such arguments rely on "guilt by association" and do not prove that a method to find $e$th roots modulo $n$ could be used to factor $n$. No one has ever shown this, nor shown that such a method could be useful for any other intractable problem.

The quadratic residuacity problem asks if for some $x$, $x^2 \equiv a$ modulo $n$. If $n$ is composite, no one knows a polynomial time algorithm to solve it, although it is easily solved by computing a Jacobi symbol when $n$ is prime or the factors of $n$ are known. It is similarly an open question if the factorization of $n$ is required, but one relationship between quadratic residuacity and factorization has long been known. If $a$ is a quadratic residue modulo $n$, then for each $p$ dividing $n$, $a$ is a square modulo $p$, and quadratic reciprocity may be used to restrict the factors of $n$ to certain arithmetic progressions. Although important in the era of hand computation, this technique is less useful today, because it is apparently infeasible to recover the factors using only such restrictions. If, however, the problem were intractable, it would be useful in cryptography; Goldwasser and Micali (1984) designed an encryption method whose security relies on such an assumption.

Surprisingly, some equations modulo composites are efficiently solvable. For example, Pollard and Schnorr (1987) presented an efficient algorithm to solve $x^2 - dy^2 \equiv a$ modulo $n$, and their result was generalized to certain rings of algebraic numbers by Adleman, Estes, and

McCurley (1987). At a first glance, there is no reason why this problem should be any easier than the others mentioned above; indeed, a cryptographic scheme was proposed based on its supposed intractability. Although the problems discussed in the last two paragraphs seem hard, they have not received the same intense examination as has factorization. As Pollard and Schnorr together have shown, beliefs about intractability, no matter how strongly held, are only conjectures.

## DISCRETE LOGARITHMS

Let $G$ be a finite cyclic group generated by $g$. If $g^x = a$, then $x$ is called the *discrete logarithm* or *index* of $a$. One also may consider groups that are not cyclic, but I shall not do this here. If $G$ has order $m$, then

$$G \cong \mathbb{Z}/(m),$$

where the isomorphism depends on a particular choice of the generator $g$. Repeated squaring provides an efficient technique for computing the reverse direction of this isomorphism; the discrete logarithm problem takes an element $a$ of $G$ and asks to which element $x$ in $\mathbb{Z}/(m)$ it corresponds. In general, no efficient methods are known for this problem; fast algorithms are only available if $G$ has special structure.

The case $G = \mathbb{Z}/(p)^*$ when $p$ is a large prime is interesting to complexity theorists because, taking $m = p - 1$ and $\{1, \ldots, p-1\}$ as residue classes modulo $m$, exponentiation is apparently a bijection on $G$ whose inverse cannot be efficiently computed. For this reason, the discrete logarithm problem is of interest in cryptography, where its uses were first pointed out by Diffie and Hellman (1978). Other applications and algorithms for the problem were surveyed by Odlyzko (1985); as this latter paper is especially thorough, I shall limit myself here to brief algorithm descriptions and a few later references.

The most general methods for the problem are "canonical" in the sense that they use only the group operations; however, their running times are proportional to some power of the group's order. For reasonably encoded groups, such running times are exponential. In several important cases, though, methods are known with subexponential running times, equal to or better than those of the best factorization algorithms. However, these methods require the group to be specified as part of a larger structure.

For the discrete logarithm problem to make sense computationally, one needs explicit versions of the group operations, as well as a canonical form by which one can test equality. In this section, I assume that such computations can be done in $(\log |G|)^{O(1)}$ steps; this is true of all special cases discussed below.

### *General methods*

Shanks (1971) designed an algorithm that works in any finite group. To use it, one chooses a $t$ so that $|G| \leq t^2$, and seeks a solution to $g^x = a$ of the form $x = x_0 + x_1 t$ with $0 \leq x_i < t$. By computing the $2t$ elements $g^{x_0}$ and $ag^{-x_1 t}$ and looking for a match (one can either sort or use hashing), $x$ can be found in $|G|^{1/2+o(1)}$ steps. This algorithm suffers from the defect of requiring space for $2t$ elements; a heuristic algorithm with similar apparent performance, but for which $|G|$ must be known, has been given by Pollard (1978).

Pohlig and Hellman (1978) extended Shanks's algorithm to be more efficient if $|G|$ is smooth. Their technique factors $G$ into a direct product of cyclic groups $H$ of prime power order, each of which can be further decomposed into a chain

$$1 = H_0 \subset H_1 \subset H_2 \subset \cdots \subset H_k = H.$$

where $|H_i/H_{i-1}|$ is prime, say $p$. To compute a discrete logarithm, one first reduces the problem to the computation of a discrete logarithm in each direct factor $H$, using the Chinese remainder theorem. In each $H$, one computes the index in base $p$ one digit at a time, similarly to Tonelli's square root algorithm. Since each digit can be found by solving a discrete logarithm problem in $H_1$, a group of order $p$, the running time is $p^{1/2}(\log|G|)^{O(1)}$, $p$ being the largest prime factor of $|G|$.

*Subexponential methods*

In certain groups one can use the *index-calculus* family of algorithms, which have much better running times. To use these algorithms $G$ must be specifiable as a "congruence group" of some ring $A$. This means that $A$ has unique factorization (or more generally, unique ideal factorization) and that $G$ is a quotient group of the free Abelian group generated by the primes, modulo a collection of multiplicative identities. Certain "exceptional" primes may be omitted. For example, one forms $\mathbb{Z}/(p)^*$ by taking the primes in $\mathbb{Z}$ relatively prime to $p$, modulo products of these primes that are congruent to 1 modulo $p$. If $G$ is a congruence group, then factorizations in $A$ lead to identities in $G$, which can be exploited to find discrete logarithms.

The basic idea in index-calculus algorithms is to build up a "database" of small primes whose logarithms are known. To do this, one computes a random power of the generator $g$ in the group, lifts this power to the ring, and tries to factor it using a set of small primes. Letting $m$ denote the order of the group, each successful factorization gives a linear equation in $\mathbb{Z}/(m)$ for the logarithms of the small primes. When enough such linear equations are collected, one solves them for the logarithms of the primes, which then form the database. To find the logarithm of an arbitrary element $a$, one multiplies it by a random power of $g$ to produce another element $b$. If $b$ factors over the prime base, this gives the discrete logarithm of $b$, hence that of $a$.

This technique is reminiscent of the quadratic residue family of integer factoring methods, as it repeatedly tries to find smooth elements and factor them. It should therefore be no surprise that the tradeoffs in selecting the prime base are similar, as are the running times.

Below I indicate groups to which the index-calculus method applies, together with the best current estimates of the time needed to find a discrete logarithm. All the methods have the important property that after the first logarithm is found, others can be computed more quickly, in time roughly the square root of that needed to find the first one. Typically, the best running times are only heuristic, and are not supported by rigorous analyses.

Coppersmith, Odlyzko, and Schroeppel (1986) gave an index-calculus algorithm to compute discrete logarithms in $\mathbb{Z}/(p)^*$; here the ring $A$ is $\mathbb{Z}$. Under plausible but unproved assumptions, their method requires $L(p)^{1+o(1)}$ steps. Pomerance (1987a) found another randomized algorithm in this case whose complexity is provably $L(p)^{\sqrt{2}+o(1)}$.

Coppersmith (1984) also extended the index-calculus idea to compute discrete logarithms in $\mathbb{F}_{2^n}^*$, which is a congruence group of the ring $\mathbb{F}_2[X]$. Heuristically, his algorithm requires roughly $K(n)^{c(n)+o(1)}$ steps, where $K(n) = \exp(n^{1/3}\log^{2/3} n)$ and $c(n) \leq 1.41$ (not $\sqrt{2}$). The best rigorous running time in this group is also due to Pomerance (1987a); his method computes a discrete logarithm in expected time $M(n)^{\sqrt{2\log 2}+o(1)}$, where $M(n) = e^{\sqrt{n\log n}}$.

The discrete logarithm problem in the multiplicative group of a non-prime finite field is not well understood. Coppersmith's algorithm extends to compute logarithms in $\mathbb{F}_{p^n}^*$, where $p$ is small. ElGamal (1986) has found index-calculus methods to compute logarithms

in $\mathbb{F}_{p^m}^*$ when $m$ is fixed (he takes $A$ to be a ring of algebraic integers), but his methods apparently do not generalize if $p$ and $m$ vary.

Buchmann and Williams (1988) and McCurley (1988a) have considered the discrete logarithm problem in class groups of quadratic fields (these groups are generally not cyclic). For applications of the discrete logarithm problem in $\mathbb{Z}/(n)^*$, see the papers of Shmuely (1985) and McCurley (1988b).

No subexponential time algorithm is known for the discrete logarithm problem in elliptic curve groups and abelian varieties. For this reason, Miller (1986) and Koblitz (1987a,1989) have argued that these groups might be useful in cryptography.

The discrete logarithm problem seems generally difficult unless the group $G$ has smooth order. It would be interesting to know if the smoothness of any other numbers related to the group can be of help. In particular, even the following problem appears unsolved: if $p+1$ is smooth, can one easily compute discrete logarithms modulo $p$? Another intriguing unanswered question asks if the complexity of the discrete logarithm problem in $\mathbb{Z}/(p)^*$ equals that of the factorization problem. Such a question is natural because algorithms to solve these problems have similar running times. However, no answer to it is known, beyond the observation that algorithms for both problems use related techniques, which result in nearly identical tradeoffs when one attempts to optimize their running times. It would be of considerable interest to show formally that this similarity is necessary.

*Acknowledgements*

I thank Gaston Gonnet for letting me use his statistics on integer multiplication in Maple. The example of non-randomness in elliptic curves was given by Michael Larsen. Many of the comments on the quadratic sieve are due to Carl Pomerance. I also thank Jim Hafner and Sid Graham for help chasing down recent analytic results, and Joao Meidanis, Jeffrey Shallit, Victor Shoup, and Jonathan Sorenson for comments on the draft. Finally, I thank Bryan So for help with Wang (1986).

The support of the National Science Foundation, via grant DCR-8552596, is gratefully acknowledged.

Adleman, L.M., Estes, D.R., McCurley, K.S. 1987. Solving bivariate quadratic congruences in polynomial time. *Math. Comput.,* 48:17-28.

Adleman, L.M., Huang, M.-D. 1987. Recognizing primes in random polynomial time. *Proc. 19th Ann. ACM Symp. Theory Comput.,* pp. 462-469. New York: ACM Press.

Adleman, L.M., Kompella, K. 1988. Using smoothness to achieve parallelism. *Proc. 20th Ann. ACM Symp. Theory Comput.,* pp. 528-538. New York: ACM Press.

Adleman, L.M., Lenstra, H.W., Jr. 1986. Finding irreducible polynomials over finite fields. *Proc. 18th Ann. ACM Symp. Theory Comput.,* pp. 350-355. New York: ACM Press.

Adleman, L.M., Manders, K., Miller, G.L. 1977. On taking roots in finite fields. *Proc. 18th Ann. Symp. Found. Comput. Sci.,* pp. 175-178. New York: IEEE Press.

Adleman, L.M., Pomerance, C., Rumely, R.S. 1983. On distinguishing prime numbers from composite numbers. *Ann. Math.,* 117:173-206.

Aho, A.V., Hopcroft, J.E., Ullman, J.D. 1974. *The Design and Analysis of Computer Algorithms.* Reading: Addison-Wesley.

Allender, E.W., 1986. Characterizations of PUNC and precomputation. In *Proc. 13th Ann. Coll. Aut. Lang. Prog.,* pp. 1-10. Berlin: Springer-Verlag. [Final version to appear in *J. ACM* as P-uniform circuit complexity.]

Anderson, S.F., Earle, J.G., Goldschmidt, R.E., Powers, D.M. 1967. The IBM System/360 Model 91: the floating-point execution unit. *IBM J. Res. Dev.,* 11:34-53.

Ankeny, N.C. 1952. The least quadratic non residue. *Ann. Math.,* 55:65-72.

Artin, E. 1924. Quadratische Körper im Gebeit der höheren Kongruenzen. *Math. Z.,* 19:153-246.

Atkin, A.O.L., Larson, R.G. 1982. On a primality test of Solovay and Strassen. *SIAM J. Comput.,* 11:789-791.

Bach, E. 1985. *Analytic Methods in the Analysis and Design of Number-Theoretic Algorithms,* Cambridge: MIT Press.

Bach, E. 1987. Realistic analysis of some randomized algorithms. *Proc. 19th Ann. ACM Symp. Theory Comput.,* pp. 453-461. New York: ACM Press. [Final version to appear, *J. Comput. Sys. Sci.*]

Bach, E. 1988a. How to generate factored random numbers. *SIAM J. Comput.,* 17:179-193.

Bach, E. 1988b. A note on square roots in finite fields. *Tech. Rep. 795,* Comput. Sci. Dept., Univ. Wisconsin.

Bach, E. 1988c. Toward a theory of Pollard's rho method. *Inform. Comput.,* to appear.

Bach, E. 1989a. Explicit bounds for primality testing and related problems. *Math. Comput.,* to appear.

Bach, E. 1989b. Intractable problems in number theory. *Proc. CRYPTO '88,* to appear. Berlin: Springer-Verlag.

Bach, E., Shallit, J. 1989. Factoring with cyclotomic polynomials. *Math. Comput.,* 52:201-219.

Bach, E., Shoup, V. 1988. Factoring polynomials with fewer random bits. *J. Symb. Comput.,* to appear.

Bach, E., Driscoll, J., Shallit, J. 1988. Factor refinement. Manuscript.

Bach, E., von zur Gathen, J. 1988. Deterministic factorization of polynomials over special finite fields. *Tech. Rep. 799,* Comput. Sci. Dept., Univ. Wisconsin.

Beame, P.W., Cook, S.A., Hoover, H.J. 1986. Log depth circuits for division and related problems. *SIAM J. Comput.,* 15:994-1003.

Ben-Or, M. 1981. Probabilistic algorithms in finite fields. *Proc. 22nd Ann. Symp. Found. Comput. Sci.*, pp. 394-298. New York: IEEE Press.

Berlekamp, E.R. 1967. Factoring polynomials over finite fields. *Bell Sys. Tech. J.* 46:1853-1859.

Berlekamp, E.R. 1968. *Algebraic Coding Theory.* New York: McGraw-Hill.

Berlekamp, E.R. 1970. Factoring polynomials over large finite fields. *Math. Comput.* 24:713-735.

Berlekamp, E.R., Rumsey, H., and Solomon, G. 1967. On the solution of algebraic equations over finite fields. *Inform. Contr.*, 10:553-564.

Bilharz, H. 1937. Primdivisoren mit vorgegebener Primitivwurzel. *Math. Ann.*, 114:476-492.

Blum, M., Micali, S. 1984. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13:850-863.

Boppana, R.B., Hirschfeld, R. 1985. Pseudorandom generators and complexity classes. In *Advances in Computing Research 5,* ed. S. Micali, to appear.

Borodin, A., von zur Gathen, J., Hopcroft, J. 1982. Fast Parallel Matrix and GCD Computations. *Inform. Contr.*, 52:241-256.

Borwein, J.M., Borwein, P.B., Bailey, D.H. 1989. Ramanujan, modular equations, and approximations to pi or how to compute one billion digits of pi. *Am. Math. Monthly,* 96:201-219.

Brent, R.P. 1973. The first occurrence of large gaps between successive primes. *Math. Comput.*, 27:959-963.

Brent, R.P. 1980. An improved Monte Carlo factorization algorithm. *BIT,* 20:176-184.

Brent, R.P. 1985. Some integer factorization algorithms using elliptic curves. *Tech. Rep. CMA-R32-85,* Comput. Sci. Lab., Australian Nat. Univ.

Brent, R.P., Kung, H.T. 1983. Systolic VLSI arrays for linear-time GCD computation. In *VLSI 83,* ed. F. Anceau, E.J. Aas, pp. 145-154. Amsterdam: Elsevier.

Brent, R.P. 1973. Factorization of the eighth Fermat number, *Math. Comput.*, 65:627-630.

Brillhart, J., Lehmer, D.H., Selfridge, J.L., Tuckerman, B., Wagstaff, S.S., Jr. 1988. *Factorizations of* $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to High Powers.* Providence: American Mathematical Society.

Brillhart, J., Montgomery, P.L., Silverman, R.D. 1988. Tables of Fibonacci and Lucas factorizations. *Math. Comput.*, 50:251-260.

Buchmann, J., Williams, H.C. 1988. A key-exchange system based on imaginary quadratic fields. *J. Cryptology,* to appear.

Butler, M.C.R. 1954. On the reducibility of polynomials over a finite field. *Quart. J. Math. Oxford,* 5:102-107.

Calmet, J., Loos, R. 1980. An improvement of Rabin's probabilistic algorithm for generating irreducible polynomials over GF(p). *Inf. Process. Lett.*, 11:94-95.

Canfield, E.R., Erdoš, P., Pomerance, C., 1983. On a problem of Oppenheim concerning "Factorisatio Numerorum," *J. Number Th.*, 17:1-28.

Cantor, D., Zassenhaus, H. 1981. A new algorithm for factoring polynomials over finite fields. *Math. Comput.*, 36:587-592.

Cantor, D.G. 1987. Computing in the Jacobian of a hyperelliptic curve. *Math. Comput.*, 48:95-101.

Carlitz, L. 1932. The arithmetic of polynomials in a Galois field. *Am. J. Math.*, 54: 39-50.

Carmichael, R.D. 1912. On composite numbers $P$ which satisfy the Fermat congruence $a^{P-1} \equiv 1 \bmod P$. *Am. Math. Monthly,* 19:22-27.

Cassels, J.W.S., Fröhlich, A. 1967. *Algebraic Number Theory.* London: Academic Press.

Čebotarev, N. 1926. Die Bestimmung der Dichtigkeit einer Menge von Primzahlen, welche zu einer Substitutionsklasse gehören. *Math. Ann.* 95:191-228.

Chernoff, H. 1952. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Stat.*, 23:493-507.

Chor, B., Goldreich, O. 1985. An improved parallel algorithm for integer GCD. *Algorithmica,* to appear.

Chowla, S. 1934. On the least prime in an arithmetical progression. *J. Indian Math. Soc.*, 1:1-3.

Chudnovsky, D.V., Chudnovsky, G.V. 1985. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Tech. Rep. RC 11262,* IBM Watson Research Center, Yorktown Heights, New York.

Cipolla, M. 1903. Un metodo per la risoluzione della congruenza di secondo grado. *Rend. Accad. Sci. Fis. Mat. Napoli,* 9:154-163.

Claasen, H.L. 1977. The group of units in $GF(q)[x]/(a(x))$. *Indag. Math.*, 39:245-255.

Cohen, H., Lenstra, H.W. Jr. 1984. Primality testing and Jacobi sums. *Math. Comput.*, 42:287-330.

Cohen, H., Lenstra, A.K. 1987. Implementation of a new primality test. *Math. Comput.*, 48:103-121.

Collins, G.E. 1969. Computing time analyses for some arithmetic and algebraic algorithms. *Proceedings of the 1968 Summer Institute on Symbolic Mathematical Computations,* pp. 195-231. IBM Federal Systems Center.

Collins, G., Loos, R. 1982. The Jacobi symbol algorithm. *SIGSAM Bull.*, 16:12-16.

Coppersmith, D. 1984. Fast evaluation of logarithms in fields of characteristic two, *IEEE Trans. Inf. Theory,* 30:587-594.

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

Coppersmith, D., Winograd, S. 1987. Matrix multiplication via arithmetic progressions. *Proc. 19th Ann. ACM Symp. Theory Comput.*, pp. 1-6. New York: ACM Press.

Cornell, G., Silverman, J.H. 1986. *Arithmetic Geometry.* Berlin: Springer-Verlag.

Cramér, H. 1921. Some theorems concerning prime numbers. *Ark. Mat. Astron. Fys.*, 15(5):1-15.

Cramér, H. 1937. On the order of magnitude of the difference between consecutive prime numbers. *Acta Arith.*, 2:23-46.

Davenport, H. 1980. *Multiplicative Number Theory.* Berlin: Springer-Verlag.

Davenport, J.H., Siret, Y., Tournier, 1988. *Computer Algebra.* London: Academic Press.

Davis, J.A., Holdridge, D.B., Simmons, G.J. 1985. Status report on factoring (at the Sandia national labs). *Proc. EUROCRYPT '84,* pp. 183-215. Berlin: Springer-Verlag.

de Bruijn, N.G. 1951. The asymptotic behavior of a function occurring in the theory of primes, *J. Indian Math. Soc.* 15:25-32.

de la Vallée Poussin, C.-J. 1896. Recherches analytiques sur la théorie des nombres premiers. *Ann. Soc. Sci. Bruxelles,* 20:183-397.

Dedekind, R. 1857. Abrifs einer Theorie der höhern Congruenzen in Bezug auf einen reellen Primzahl-Modulus. *J. Reine Angew. Math.*, 54:1-26.

Deligne, P. 1973. La conjecture de Weil. I. *I.H.E.S. Publ. Math.*, 43:273-307.

Dickman, K. 1930. On the frequency of numbers containing prime factors of a certain relative magnitude. *Ark. Mat. Astr. Fys.*, 22A(10):1-15.

Dickson, L.E. 1919. *History of the Theory of Numbers.* Washington: Carnegie Institution. [Reprinted by Chelsea, 1971.]

Diffie, W., Hellman M., 1978. New directions in cryptography, *IEEE Trans. Inf. Theory,* 22:644-654.

ElGamal, T. 1986. On computing logarithms over finite fields. *Proc. CRYPTO '85,* pp. 396-402. Berlin: Springer-Verlag.

Erdös, P., Kac, M. 1940. The Gaussian law of errors in the theory of additive functions. *Amer. J. Math.,* 62:738-742.

Erdös, P., Pomerance, C. 1986. On the number of false witnesses for a composite number. *Math. Comput.,* 46:259-279.

Evdokimov, S.A. 1986. Efficient factorization of polynomials over finite fields and generalized Riemann hypothesis, manuscript.

Feller, W. 1968. *An Introduction to Probability Theory and Its Applications.* New York: Wiley.

Fich, F., Tompa, M. 1988. The parallel complexity of exponentiating polynomials over finite fields. *J. ACM,* 35:651-667.

Fogels, I. 1962. On the distribution of prime ideals. *Acta Arith.,* 7:255-269.

Frobenius, G., Stickelberger, L. 1879. Über Gruppen von vertauschbaren Elementen. *J. Reine Angew. Math.,* 86:217-262.

Galois, E. 1830. Sur la théorie des nombres. In *Écrits et Mémoires Mathématiques d'Évariste Galois,* ed. R. Bourgne, J.-P. Arza, pp. 112-128. Paris: Gauthier-Villars.

Garey, M.R., Johnson, D.S. 1979. *Computers and Intractability.* San Francisco: Freeman. 338 pp.

Gauss, C.F. 1801. *Disquisitiones Arithmeticae.* [English translation published by Springer-Verlag, 1986.]

Gauss, C.F. 1817. Theorematis fundamentalis in doctrina de residuis quadraticis demonstrationes et ampliationes novae, *Werke,* v. 2. [German translation in *Untersuchen über Höhere Arithmetik,* Chelsea, 1981.]

Gerver, J.L. 1986. Factoring large numbers with a quadratic sieve. *Math. Comput.,* 41:287-294.

Gill, J.T., III., 1977. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.,* 6:675-695.

Gold, R., Sattler, J. 1983. Modifikationen des Pollard-Algorithmus. *Computing,* 30:77-89.

Goldwasser, S., Micali, S. 1984. Probabilistic encryption. *J. Comput. Sys. Sci.,* 28:270-299.

Goldwasser, S., Kilian, J. 1986. Almost all primes can be quickly certified. *Proc. 18th Ann. ACM Symp. Theory Comput.,* pp. 316-329. New York: ACM Press.

Graham, S.W., Ringrose, C.J. 1988. Lower bounds for least quadratic non-residues. manuscript.

Guy, R.K. 1975. How to factor a number. *Proc. 5th Manitoba Conf. Numer. Math.,* pp. 49-89.

Guy, R.K. 1984. *Reviews in Number Theory 1973-83.* Providence: American Mathematical Society.

Hadamard, J. 1896. Sur la distribution des zéros de la fonction $\zeta(s)$ et ses consequences arithmétiques. *Bull. Soc. Math. France,* 24:199-220.

Halberstam, H., Lou, S., Yao, Q. 1989. A new upper bound on the linear sieve. In *Number Theory, Trace Formulas, and Discrete Groups,* ed. K.E. Aubert, E. Bombieri, D. Goldfield, pp. 331-341. London: Academic Press.

Hartshorne, R. 1977. *Algebraic Geometry.* Berlin: Springer-Verlag.

Haworth, G.M. 1986. Primality-testing Mersenne numbers (II). *Am. Math. Soc. Abstr.,* 7:224-225.

Hirzebruch, F. 1966. *Topological Methods in Algebraic Geometry.* Berlin: Springer-Verlag.

Hooley, C. 1967. On Artin's conjecture. *J. Reine Angew. Math.*, 225:209-220.

Hua, L.K. 1982. *Introduction to Number Theory.* Berlin: Springer-Verlag.

Huang, M.-D. 1985. Riemann hypothesis and finding roots over finite fields. *Proc. 17th Ann. ACM Symp. Theory Comput.*, pp. 121-130. New York: ACM Press.

Ireland, K., Rosen, M. 1982. *A Classical Introduction to Modern Number Theory.* Berlin: Springer-Verlag.

Jacobi, C.G.J. 1837. Über die Kreistheilung und ihre Anwendung auf die Zahlentheorie. *Monatsber. Akad. Wiss. Berlin,* 127-136.

Karp, R.M., Ramachandran, V. 1988. A survey of parallel algorithms for shared-memory machines. *Tech. Rep. 88/408,* Comp. Sci. Div., Univ. Calif., Berkeley. [To appear in *Handbook of Theoretical Computer Science,* North-Holland.]

Karatsuba, A., Ofman, Y. 1962. Multiplication of multidigit numbers by automata. *Dokl. Akad. Nauk SSSR,* 145:293-294. [English translation in *Sov. Phys. Dokl.* 7:595-596, 1963.]

Knuth, D.E. 1981. *The Art of Computer Programming; Volume 2: Seminumerical Algorithms.* Reading: Addison-Wesley.

Knuth, D.E., Trabb Pardo, L. 1976. Analysis of a simple factorization algorithm. *Theor. Comput. Sci.,* 3:321-348.

Koblitz, N. 1987a. Elliptic curve cryptosystems, *Math. Comput.* 48:203-209.

Koblitz, N. 1987b. *A Course in Number Theory and Cryptography.* Berlin: Springer-Verlag.

Koblitz, N. 1989. A family of Jacobians suitable for discrete log cryptosystems. *Proc. CRYPTO '88,* to appear. Berlin: Springer-Verlag.

Kühne, H. 1902. Eine Wechselbeziehung zwischen Functionen mehrer Unbestimmten, die zu Reciprocitätsgesetzen führt. *J. Reine Angew. Math.*, 124:121-131.

Lagarias, J.C., Montgomery, H.L., Odlyzko, A.M. 1979. A bound for the least prime ideal in the Chebotarev density theorem. *Invent. Math.*, 54:271-296.

Lagarias, J.C., Odlyzko, A.M. 1979. On computing Artin L-functions in the critical strip. *Math. Comput.* 33:1081-1085.

Lamé, G. 1844. Note sur la limite du nombre des divisions dans la recherche du plus grand commun diviseur entre deux nombres entiers. *C. R. Acad. Sci. Paris,* 29:867-870.

Lang, S., Tate, J. 1965. Preface to *Emil Artin: Collected Papers,* Berlin: Springer-Verlag.

Lauwerier, H.A. 1986. Two-dimensional iterative maps. In *Chaos,* ed. A.V. Holden, pp. 58-95. Princeton: Princeton University Press.

Lebesgue, V.A. 1847. Sur le symbole $(\frac{a}{b})$ et quelques-unes de ses applications. *J. Math. Pures Appl.,* 12:497-517.

Lehmann, D.J. 1982. On primality tests. *SIAM J. Comput.* 11:374-375.

Lehmer, D.H. 1930. An extended theory of Lucas' functions. *Ann. Math.,* 31:419-448.

Lehmer, D.H. 1933. A photo-electric number sieve. *Am. Math. Monthly,* 40:401-406.

Lehmer, D.H. 1938. Euclid's algorithm for large numbers. *Am. Math. Monthly,* 45:227-233.

Lehmer, D.H., Lehmer, E. 1962. Heuristics, anyone? In *Studies in Mathematical Analysis and Related Topics,* ed. G. Szegö. Stanford: Stanford University Press.

Lehmer, D.H. 1969. Computer technology applied to the theory of numbers. In *Studies in Number Theory,* ed. W.J. LeVeque, pp. 117-151. Washington: MAA.

Lehmer, D.H. 1976. Strong Carmichael numbers. *J. Aust. Math. Soc.,* 21:508-510.

Lenstra, A.K. 1987. Fast and rigorous factorization under the generalized Riemann hypothesis. *Indag. Math.,* to appear.

Lenstra, A.K., Lenstra, H.W., Jr. 1987. Algorithms in number theory. *Tech. Rep. 87-008,* Dept. Comput. Sci., Univ. Chicago. [To appear in *Handbook of Theoretical Computer Science,* North-Holland.]

Lenstra, A.K., Manasse, M. 1989. Factorization record. announcement.

Lenstra, H.W., Jr. 1981. Primality testing algorithms [after Adleman, Rumely, and Williams]. In *Séminaire Bourbaki, Lecture Notes in Mathematics 901,* pp. 576-01 to 576-15. Berlin: Springer-Verlag.

Lenstra, H.W., Jr. 1987. Factoring integers with elliptic curves. *Ann. Math.,* 126:649-673.

Lenstra, H.W., Jr. 1988. Finding isomorphisms between finite fields. manuscript.

Lenstra, H.W., Jr., Tijdeman, R. 1984. *Computational Methods in Number Theory.* Amsterdam: Mathematisch Centrum.

LeVeque, W.J. 1974. *Reviews in Number Theory.* Providence: American Mathematical Society.

Lidl, R., Niederreiter, H. 1984. *Finite Fields.* Cambridge: Cambridge University Press.

Linnik, U.V. 1944. On the least prime in an arithmetic progression. I. The basic theorem. *Mat. Sbornik,* 15:139-178.

Machtey, M., Young, P. 1978. *An Introduction to the General Theory of Algorithms.* New York: Elsevier North-Holland.

Manders, K., Adleman, L. 1978. NP-complete decision problems for binary quadratics. *J. Comput. Sys. Sci.,* 16:168-184.

McCurley, K.S. 1984a. Explicit estimates for the error term in the prime number theorem for arithmetic progressions. *Math. Comput.,* 42:265-285.

McCurley, K.S. 1984b. Explicit estimates for $\theta(x;3,l)$ and $\psi(x;3,l)$. *Math. Comput.,* 42:287-296.

McCurley, K.S. 1988a. Cryptographic key distribution and computation in class groups. In *Proceedings of the NATO Advanced Study Institute on Number Theory and Applications,* to appear. Dordrecht: Reidel.

McCurley, K.S. 1988b. A key distribution system equivalent to factoring. *J. Cryptology,* to appear.

Mignotte, M. 1980. Calcul des racines $d$-ièmes dans un corps fini. *C. R. Acad. Sci. Paris,* 290:205-206.

Mignotte, M., Schnorr, C. 1988. Calcul déterministique des racines d'un polynôme dans un corps fini. *C. R. Acad. Sci. Paris,* 306:467-472.

Miller, G.L. 1976. Riemann's hypothesis and tests for primality. *J. Comput. Sys. Sci.,* 13:300-317.

Miller, V. 1986. Use of elliptic curves in cryptography. *Proc. CRYPTO '85,* pp. 417-426. Berlin: Springer-Verlag.

Moenck, R.T. 1973. Fast computation of GCD's. *Proc. 5th Ann. ACM Symp. Theory Comput.,* pp. 142-151. New York: ACM Press.

Monier, L. 1980. Evaluation and comparison of two efficient probabilistic primality testing algorithms. *Theor. Comput. Sci.,* 12:97-108.

Montgomery, H.L. 1971. *Topics in Multiplicative Number Theory.* Berlin: Springer-Verlag.

Montgomery, P. 1987. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comput.,* 48:243-264.

Montgomery, P.L., Silverman, R.D. 1988. An FFT extension to the p-1 factoring algorithm. Manuscript.

Morain, F. 1988. Implementation of the Goldwasser-Kilian-Atkin Primality Testing Algorithm. *INRIA Tech. Rep.,* Le Chesnay, France.

Morrison, M.A., Brillhart, J. 1975. A method of factoring and the factorization of $F_7$. *Math. Comput.,* 29:183-205.

Narkiewicz, W. 1974. *Elementary and Analytic Theory of Algebraic Numbers.* Warsaw: Polish Scientific Publishers.

Odlyzko, A.M. 1985. Discrete logarithms and their cryptographic significance. *Proc. EUROCRYPT '84,* pp. 224-314. Berlin: Springer-Verlag.

Odlyzko, A.M. 1987. On the distribution of spacings between zeros of the zeta function. *Math. Comput.,* 48:273-308.

Odlyzko, A.M., te Riele, H.J.J. 1985. Disproof of the Mertens conjecture. *J. Reine Angew. Math.,* 357:138-160.

Oesterlé, J. 1979. Versions effectives du théorème du Chebotarev sous l'hypothese de Riemann généralisée. *Soc. Math. France Astérisque* 61:165-167.

Peralta, R. 1986. A simple and fast probabilistic algorithm for computing square roots modulo a prime number. *IEEE Trans. Inf. Theory,* IT-32:846-847.

Petr, K. 1937. Über die Reduzibilität eines Polynoms mit ganzzahligen Koeffizienten nach einem Primzahlmodul. *Časopis Pest. Mat. Fys.,* 66:85-94.

Pila, J. 1987. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Math. Comput.,* to appear.

Piltz, A. 1884. *Über die Haufigkeit der Primzahlen in Arithmetischen Progressionen und über Verwandte Gesetze.* (Habilitationsschrift.) Jena: Neuenhahn.

Pintz, J., Steiger, W.L., Szemerédi, E. 1988. Two infinite sets of primes with fast primality tests. *Proc. 20th Ann. ACM Symp. Theory Comput.,* pp. 504-509. New York: ACM Press.

Pohlig, S., Hellman, M. 1978. An improved algorithm for computing logarithms over GF($p$) and its cryptographic significance, *IEEE Trans. Inf. Theory,* 24:106-110.

Pollard, J.M. 1974. Theorems on factorization and primality testing. *Proc. Camb. Phil. Soc.,* 76:521-528.

Pollard, J.M. 1975. A Monte Carlo method for factorization. *BIT,* 15:331-334.

Pollard, J.M. 1978. Monte Carlo methods for index computation (mod $p$). *Math. Comput.* 32:918-924.

Pollard, J.M., Schnorr, C.P. 1987. An efficient solution of the congruence $x^2 + ky^2 = m$ (mod $n$). *IEEE Trans. Inf. Theory,* IT-33:702-709.

Pomerance, C. 1984. Analysis and comparison of some integer factoring algorithms. See Lenstra and Tijdeman 1984, pp. 89-141.

Pomerance, C. 1985. The quadratic sieve factoring algorithm. *Proc. EUROCRYPT '84,* pp. 169-182. Berlin: Springer-Verlag.

Pomerance, C. 1987a. Fast, rigorous factorization and discrete logarithm algorithms. In *Discrete Algorithms and Complexity: Proceedings of the Japan-US Joint Seminar,* ed. D.S. Johnson, A. Nishizeki, A. Nozaki, H.S. Wilf, pp. 119-143. London: Academic Press.

Pomerance, C. 1987b. Very short primality proofs. *Math. Comput.,* 48:315-322.

Pomerance, C. 1988. Algorithms in number theory. Presented at 4th SIAM Conf. Discr. Math., San Francisco.

Pomerance, C., Selfridge, J.L., Wagstaff, S.S., Jr. 1980. The pseudoprimes to $25 \cdot 10^9$. *Math. Comput.,* 35:1003-1026.

Pomerance, C., Wagstaff, S.S., Jr. 1983. Implementation of the continued fraction integer factoring algorithm. *Congr. Numer.,* 37:99-118.

Pratt, V.R. 1975. Every prime has a succinct certificate. *SIAM J. Comput.,* 4:214-220.

Pritchard, P. 1983. Fast compact prime number sieves (among others). *J. Algorithms,* 4:332-334.

Rabin, M.O. 1979. Digitalized signatures and public-key functions as intractable as factorization. *Tech. Rep. 212,* MIT Lab. Comput. Sci., Cambridge, Mass.

Rabin, M.O. 1980a. Probabilistic algorithm for testing primality. *J. Number Th.*, 12:128-138.

Rabin, M.O. 1980b. Probabilistic algorithms in finite fields. *SIAM J. Comput.*, 9:273-280.

Rabin, M.O., Shallit, J.O. 1986. Randomized algorithms in number theory. *Comm. Pure Appl. Math.*, 39:239-256.

Riemann, B. 1859. Ueber die Anzahl der Primzahlen unter einer gegebenen Grösse. *Monatsber. Akad. Wiss. Berlin*, 671-680. [English translation in H.E. Edwards, *Riemann's Zeta Function*, Academic Press.]

Riesel, H. 1985. *Prime Numbers and Computer Methods for Factorization.* Boston: Birkhauser.

Rivest, R., Shamir, A., Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems, *Comm. ACM* 21:120-126.

Rónyai, L. 1988a. Factoring polynomials over finite fields. *J. Algorithms*, 9:391-400.

Rónyai, L. 1988b. Factoring polynomials modulo special primes. *Combinatorica*, to appear.

Rónyai, L. 1989. Galois groups and factoring polynomials over finite fields. manuscript.

Rosser, J.B., Schoenfeld, L. 1962. Approximate formulas for some functions of prime numbers. *Ill. J. Math.*, 6:64-94.

Ruzzo, W.L. 1981. On uniform circuit complexity. *J. Comput. Sys. Sci.*, 22:365-383.

Schnorr, C.-P., Lenstra, H.W., Jr. 1984. A Monte Carlo factoring algorithm with linear storage. *Math. Comput.*, 43:289-311.

Schoenfeld, L. 1976. Sharper bounds for the Chebyshev functions $\theta(x)$ and $\psi(x)$. II. *Math. Comput.*, 30:337-360.

Schönhage, A. 1971. Schnelle Berechnung von Kettenbruchenentwicklungen. *Acta Inf.*, 1:139-144.

Schönhage, A., Strassen, V. 1971. Schnelle Multiplikation Grosser Zahlen. *Computing*, 7:281-292.

Schoof, R. 1984. Quadratic fields and factorization. See Lenstra and Tijdeman 1984, pp. 235-279.

Schoof, R. 1985. Elliptic curves over finite fields and the computation of square roots mod $p$. *Math. Comput.*, 44:483-494.

Schoof, R. 1988. Elliptic curves. Presented at workshop on Constructive Algebraic Number Theory, Oberwolfach, Germany.

Schrijver, A. 1986. *Theory of Linear and Integer Programming.* New York: Wiley.

Serre, J.-P. 1968. *Abelian l-adic Representations and Elliptic Curves.* New York: Benjamin.

Shallit, J. 1989. On the worst case of three algorithms for computing the Jacobi symbol. manuscript.

Shanks, D. 1971. Class number, a theory of factorization, and genera. In *Proceedings of Symposia in Pure Mathematics 20*, pp. 415-440. Providence: American Mathematical Society.

Shanks, D. 1972. Five number-theoretic algorithms. In *Proc. 2nd Manitoba Conf. Numer. Math.*, pp. 51-70.

Shanks, D. 1978. *Solved and Unsolved Problems in Number Theory.* New York: Chelsea.

Shmuely, Z. 1985. Composite Diffie-Hellman public-key generating systems are hard to break. *Tech. Rep. 356*, Technion, Haifa, Israel.

Shoup, V. 1988a. New algorithms for finding irreducible polynomials over finite fields. In *Proc. 29th Ann. Symp. Found. Comput. Sci.*, pp. 283-290. New York: IEEE Press. [Final version to appear in *Math. Comput.*]

Shoup, V. 1988b. On the deterministic complexity of factoring polynomials over finite fields. *Inf. Process. Lett.*, to appear.

Silverman, J. 1986. *The Arithmetic of Elliptic Curves.* Berlin: Springer-Verlag.

Silverman, R.D. 1987. The multiple polynomial quadratic sieve. *Math. Comput.*, 48:329-339.

Sklansky, J. 1960. Conditional-Sum Addition Logic. *IRE Trans. Electr. Comput.*, EC-9:226-231.

Solovay, R., Strassen, V. 1977. A fast Monte-Carlo test for primality. *SIAM J. Comput.*, 6:84-85. [Erratum in v. 7, p. 118, 1978.]

Spira, R. 1969. Calculation of Dirichlet L-functions. *Math. Comput.*, 23:489-497.

Stein, J. 1967. Computational problems associated with Racah algebra. *J. Comput. Phys.*, 1:397:405.

Strassen, V. 1973. Die Berechnungskomplexität von elementarsymmetrischen Funktionenen und von Interpolationskoeffizienten. *Numer. Math.*, 20:238-51.

Tate, J.T. 1965. Algebraic cycles and poles of zeta functions. In *Arithmetical Algebraic Geometry,* ed. O.F.G. Schilling, pp. 93-110. New York: Harper and Row.

Tate, J.T. 1974. The arithmetic of elliptic curves. *Invent. Math.*, 23:179-206.

Tompa, M. 1983. Probabilistic factoring algorithms can be made errorless. *Tech. Rep. 83-09-01,* Dept. Comput. Sci., Univ. Washington.

Tonelli, A. 1891. Bemerkung über die Auflösung quadratischer Congruenzen. *Gött. Nachr.* 344-346.

Valiant, L.G. 1979. The complexity of computing the permanent. *Theor. Comp. Sci.*, 8:189-201.

Vallée, B. 1989. Provably fast integer factoring with quasi-uniform small quadratic residues. *Proc. 21st Ann. ACM Symp. Theory Comput.,* to appear.

van de Lune, J., te Riele, H.J.J., Winter, D.T. 1986. On the zeros of the Riemann zeta function in the critical strip. IV. *Math. Comput.* 46:667-681.

van de Lune, J., Wattel, E. 1969. On the numerical solution of a differential-difference equation arising in analytic number theory. *Math. Comput.*, 23:417-421.

van der Lingen, J. 1987. Elliptic curves and factorization algorithms. *Tech. Rep. 87-2,* Math. Inst., Univ. Amsterdam.

van der Waerden, B.L. 1970. *Algebra.* New York: Ungar.

Vélu, J. 1978. Tests for primality under the Riemann hypothesis. *SIGACT News,* 10:58-59.

von zur Gathen, J. 1987a. Computing powers in parallel. *SIAM J. Comput.*, 16:930-945.

von zur Gathen, J. 1987b. Factoring polynomials and primitive elements for special primes. *Theor. Comp. Sci.*, 52:77-89.

Wagstaff, S.S., Jr. 1979. Greatest of the least primes in arithmetic progressions having a given modulus. *Math. Comput.*, 33:1073-1080.

Wang, Y. 1961. On the least primitive root of a prime. *Sci. Sinica,* 10:1-14.

Wang, W. 1986. On the least prime in an arithmetic progression. *Acta Math. Sinica,* 26:826-836. [In Chinese.]

Weil, A. 1948. *Sur les Courbes Algébriques et les Variétés qui s'en Déduisent.* Paris: Hermann.

Weil, A. 1974. *Basic Number Theory,* Berlin: Springer-Verlag.

Wiedemann, D.H. 1986. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory,* IT-32:54-62.

Williams, H.C. 1978. Primality testing on a computer. *Ars Combin.*, 5:127-185.

Williams, H.C. 1982a. The influence of computers in the development of number theory. *Comput. Math. Appl.*, 8:75-93.

Williams, H.C. 1982b. A $p+1$ method of factoring. *Math. Comput.*, 39:225-324.

Williams, H.C. 1983. An overview of factoring. *Proc. CRYPTO '83,* pp. 71-80. New York: Plenum Press.

Williams, H.C. 1984. Factoring on a computer. *Math. Intelligencer,* 6:29-36.

Wunderlich, M.C. 1985. Implementing the continued fraction algorithm on parallel machines. *Math. Comput.,* 44:251-260.

Wunderlich, M.C. 1988. Computational methods for factoring large integers. *Abacus,* 5:19-33.

Yao, A. 1982. Theory and application of trapdoor functions. In *Proc. 23rd Ann. Symp. Found. Comput. Sci.,* pp. 80-91. New York: IEEE Press.

Young, J., Buell, D.A. 1988. The twentieth Fermat number is composite. *Math. Comput.,* 50:261-263.

Zassenhaus, H. 1969. On Hensel factorization. I. *J. Number Th.,* 1:291-311.

Zimmer, H.G. 1972. *Computational Problems, Methods, and Results in Algebraic Number Theory.* Berlin: Springer-Verlag.