Annals of Mathematics, **160** (2004), 781–793

PRIMES is in P

By Manindra Agrawal, Neeraj Kayal, and Nitin Saxena*

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

1. Introduction

Prime numbers are of fundamental importance in mathematics in general, and number theory in particular. So it is of great interest to study different properties of prime numbers. Of special interest are those properties that allow one to determine efficiently if a number is prime. Such efficient tests are also useful in practice: a number of cryptographic protocols need large prime numbers.

Let PRIMES denote the set of all prime numbers. The definition of prime numbers already gives a way of determining if a number n is in PRIMES: try dividing n by every number $m \leq \sqrt{n}$—if any m divides n then it is composite, otherwise it is prime. This test was known since the time of the ancient Greeks—it is a specialization of the *Sieve of Eratosthenes* (ca. 240 BC) that generates all primes less than n. The test, however, is inefficient: it takes $\Omega(\sqrt{n})$ steps to determine if n is prime. An efficient test should need only a polynomial (in the size of the input $= \lceil \log n \rceil$) number of steps. A property that *almost* gives an efficient test is Fermat's Little Theorem: for any prime number p, and any number a not divisible by p, $a^{p-1} = 1 \pmod{p}$. Given an a and n it can be efficiently checked if $a^{n-1} = 1 \pmod{n}$ by using repeated squaring to compute the $(n-1)^{\text{th}}$ power of a. However, it is not a correct test since many composites n also satisfy it for some a's (*all* a's in case of *Carmichael* numbers [Car]). Nevertheless, Fermat's Little Theorem became the basis for many efficient primality tests.

Since the beginning of complexity theory in the 1960s—when the notions of complexity were formalized and various complexity classes were defined—

*The last two authors were partially supported by MHRD grant MHRD-CSE-20010018.

this problem (referred to as the *primality testing* problem) has been investigated intensively. It is trivial to see that the problem is in the class co-NP: if n is not prime it has an easily verifiable short certificate, viz., a nontrivial factor of n. In 1974, Pratt observed that the problem is in the class NP too [Pra] (thus putting it in NP $\cap$ co-NP).

In 1975, Miller [Mil] used a property based on Fermat's Little Theorem to obtain a deterministic polynomial-time algorithm for primality testing assuming the *Extended Riemann Hypothesis* (ERH). Soon afterwards, his test was modified by Rabin [Rab] to yield an unconditional but randomized polynomial-time algorithm. Independently, Solovay and Strassen [SS] obtained, in 1974, a different randomized polynomial-time algorithm using the property that for a prime n, $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} \pmod{n}$ for every a ($\left(\frac{}{}\right)$ is the Jacobi symbol). Their algorithm can also be made deterministic under ERH. Since then, a number of randomized polynomial-time algorithms have been proposed for primality testing, based on many different properties.

In 1983, Adleman, Pomerance, and Rumely achieved a major breakthrough by giving a deterministic algorithm for primality that runs in $(\log n)^{O(\log\log\log n)}$ time (all the previous deterministic algorithms required exponential time). Their algorithm was (in a sense) a generalization of Miller's idea and used higher reciprocity laws. In 1986, Goldwasser and Kilian [GK] proposed a randomized algorithm based on elliptic curves running in expected polynomial-time, on almost all inputs (*all* inputs under a widely believed hypothesis), that produces an easily verifiable short certificate for primality (until then, all randomized algorithms produced certificates for compositeness only). Based on their ideas, a similar algorithm was developed by Atkin [Atk]. Adleman and Huang [AH] modified the Goldwasser-Kilian algorithm to obtain a randomized algorithm that runs in expected polynomial-time on all inputs.

The ultimate goal of this line of research has been, of course, to obtain an unconditional deterministic polynomial-time algorithm for primality testing. Despite the impressive progress made so far, this goal has remained elusive. In this paper, we achieve this. We give a deterministic, $O^{\sim}(\log^{15/2} n)$ time algorithm for testing if a number is prime. Heuristically, our algorithm does better: under a widely believed conjecture on the density of Sophie Germain primes (primes p such that $2p+1$ is also prime), the algorithm takes only $O^{\sim}(\log^{6} n)$ steps. Our algorithm is based on a generalization of Fermat's Little Theorem to polynomial rings over finite fields. Notably, the correctness proof of our algorithm requires only simple tools of algebra (except for appealing to a sieve theory result on the density of primes p with $p-1$ having a large prime factor—and even this is not needed for proving a weaker time bound of $O^{\sim}(\log^{21/2} n)$ for the algorithm). In contrast, the correctness proofs of earlier algorithms producing a certificate for primality [APR], [GK], [Atk] are much more complex.

In Section 2, we summarize the basic idea behind our algorithm. In Section 3, we fix the notation used. In Section 4, we state the algorithm and present its proof of correctness. In Section 5, we obtain bounds on the running time of the algorithm. Section 6 discusses some ways of improving the time complexity of the algorithm.

2. The idea

Our test is based on the following identity for prime numbers which is a generalization of Fermat's Little Theorem. This identity was the basis for a randomized polynomial-time algorithm in [AB]:

LEMMA 2.1. *Let* $a \in \mathcal{Z}$, $n \in \mathcal{N}$, $n \geq 2$, *and* $(a, n) = 1$. *Then* n *is prime if and only if*

$$(X + a)^n = X^n + a \pmod{n}. \tag{1}$$

Proof. For $0 < i < n$, the coefficient of x^i in $((X + a)^n - (X^n + a))$ is $\binom{n}{i} a^{n-i}$.

Suppose n is prime. Then $\binom{n}{i} = 0 \pmod{n}$ and hence all the coefficients are zero.

Suppose n is composite. Consider a prime q that is a factor of n and let $q^k || n$. Then q^k does not divide $\binom{n}{q}$ and is coprime to a^{n-q} and hence the coefficient of X^q is not zero (mod n). Thus $((X + a)^n - (X^n + a))$ is not identically zero over Z_n. □

The above identity suggests a simple test for primality: given an input n, choose an a and test whether the congruence (1) is satisfied. However, this takes time $\Omega(n)$ because we need to evaluate n coefficients in the LHS in the worst case. A simple way to reduce the number of coefficients is to evaluate both sides of (1) modulo a polynomial of the form $X^r - 1$ for an appropriately chosen small r. In other words, test if the following equation is satisfied:

$$(X + a)^n = X^n + a \pmod{X^r - 1, n}. \tag{2}$$

From Lemma 2.1 it is immediate that all primes n satisfy the equation (2) for all values of a and r. The problem now is that some composites n may also satisfy the equation for a few values of a and r (and indeed they do). However, we can almost restore the characterization: we show that for appropriately chosen r if the equation (2) is satisfied for several a's then n must be a prime power. The number of a's and the appropriate r are both bounded by a polynomial in $\log n$ and therefore, we get a deterministic polynomial time algorithm for testing primality.

3. Notation and preliminaries

The class P is the class of sets accepted by deterministic polynomial-time Turing machines [Lee]; see [Lee] for the definitions of classes NP, co-NP, etc.

Z_n denotes the ring of numbers modulo n and F_p denotes the finite field with p elements, where p is prime. Recall that if p is prime and $h(X)$ is a polynomial of degree d and irreducible in F_p, then $F_p[X]/(h(X))$ is a finite field of order p^d. We will use the notation $f(X) = g(X) \pmod{h(X), n}$ to represent the equation $f(X) = g(X)$ in the ring $Z_n[X]/(h(X))$.

We use the symbol $O^{\sim}(t(n))$ for $O(t(n) \cdot \text{poly}(\log t(n)))$, where $t(n)$ is any function of n. For example, $O^{\sim}(\log^k n) = O(\log^k n \cdot \text{poly}(\log\log n)) = O(\log^{k+\varepsilon} n)$ for any $\varepsilon > 0$. We use log for base 2 logarithms, and ln for natural logarithms.

$\mathcal{N}$ and $\mathcal{Z}$ denote the set of natural numbers and integers respectively. Given $r \in \mathcal{N}$, $a \in \mathcal{Z}$ with $(a, r) = 1$, the *order of a modulo r* is the smallest number k such that $a^k = 1 \pmod r$. It is denoted as $o_r(a)$. For $r \in \mathcal{N}$, $\phi(r)$ is *Euler's totient function* giving the number of numbers less than r that are relatively prime to r. It is easy to see that $o_r(a) \mid \phi(r)$ for any a, $(a, r) = 1$.

We will need the following simple fact about the lcm of the first m numbers (see, e.g., [Nai] for a proof).

LEMMA 3.1. *Let* LCM(m) *denote the* lcm *of the first m numbers. For* $m \geq 7$:

$$\text{LCM}(m) \geq 2^m.$$

4. The algorithm and its correctness

Input: integer $n > 1$.

1. If ($n = a^b$ for $a \in \mathcal{N}$ and $b > 1$), output COMPOSITE.
2. Find the smallest r such that $o_r(n) > \log^2 n$.
3. If $1 < (a, n) < n$ for some $a \leq r$, output COMPOSITE.
4. If $n \leq r$, output PRIME.[1]
5. For $a = 1$ to $\lfloor \sqrt{\phi(r)} \log n \rfloor$ do

   if ($(X + a)^n \neq X^n + a \pmod{X^r - 1, n}$), output COMPOSITE;
6. Output PRIME.

<u>Algorithm for Primality Testing</u>

THEOREM 4.1. *The algorithm above returns* PRIME *if and only if n is prime.*

In the remainder of the section, we establish this theorem through a sequence of lemmas. The following is trivial:

LEMMA 4.2. *If n is prime, the algorithm returns* PRIME.

Proof. If n is prime then steps 1 and 3 can never return COMPOSITE. By Lemma 2.1, the `for` loop also cannot return COMPOSITE. Therefore the algorithm will identify n as PRIME either in step 4 or in step 6. $\square$

The converse of the above lemma requires a little more work. If the algorithm returns PRIME in step 4 then n must be prime since otherwise step 3 would have found a nontrivial factor of n. So the only remaining case is when the algorithm returns PRIME in step 6. For the purpose of subsequent analysis we assume this to be the case.

The algorithm has two main steps (2 and 5): step 2 finds an appropriate r and step 5 verifies the equation (2) for a number of a's. We first bound the magnitude of the appropriate r.

LEMMA 4.3. *There exists an $r \leq \max\{3, \lceil \log^5 n \rceil\}$ such that $o_r(n) > \log^2 n$.*

Proof. This is trivially true when $n = 2$; $r = 3$ satisfies all conditions. So assume that $n > 2$. Then $\lceil \log^5 n \rceil > 10$ and Lemma 3.1 applies. Let r_1, r_2, $\ldots$, r_t be all numbers such that either $\mathrm{o}_{r_i}(n) \leq \log^2 n$ or r_i divides n. Each of these numbers must divide the product

$$n \cdot \prod_{i=1}^{\lfloor \log^2 n \rfloor} (n^i - 1) < n^{\log^4 n} \leq 2^{\log^5 n}.$$

By Lemma 3.1, the lcm of the first $\lceil \log^5 n \rceil$ numbers is at least $2^{\lceil \log^5 n \rceil}$ and therefore there must exist a number $s \leq \lceil \log^5 n \rceil$ such that $s \notin \{r_1, r_2, \ldots, r_t\}$. If $(s, n) = 1$ then $\mathrm{o}_s(n) > \log^2 n$ and we are done. If $(s, n) > 1$, then since s does not divide n and $(s, n) \in \{r_1, r_2, \ldots, r_t\}$, $r = \frac{s}{(s,n)} \notin \{r_1, r_2, \ldots, r_t\}$ and so $\mathrm{o}_r(n) > \log^2 n$. $\square$

Since $\mathrm{o}_r(n) > 1$, there must exist a prime divisor p of n such that $\mathrm{o}_r(p) > 1$. We have $p > r$ since otherwise either step 3 or step 4 would decide about the primality of n. Since $(n, r) = 1$ (otherwise either step 3 or step 4 will correctly identify n), $p, n \in Z_r^*$. Numbers p and r will be fixed in the remainder of this section. Also, let $\ell = \lfloor \sqrt{\phi(r)} \log n \rfloor$.

---

[1]Lemma 4.3 shows that $r \leq \lceil \log^5 n \rceil$, so that Step 4 is relevant only when $n \leq 5{,}690{,}034$.

Step 5 of the algorithm verifies ℓ equations. Since the algorithm does not output COMPOSITE in this step, we have:

$$(X+a)^n = X^n + a \pmod{X^r - 1, n}$$

for every a, $0 \le a \le \ell$ (the equation for $a = 0$ is trivially satisfied). This implies:

$$(X+a)^n = X^n + a \pmod{X^r - 1, p} \tag{3}$$

for $0 \le a \le \ell$. By Lemma 2.1, we have:

$$(X+a)^p = X^p + a \pmod{X^r - 1, p} \tag{4}$$

for $0 \le a \le \ell$. From equations 3 and 4 it follows that:

$$(X+a)^{\frac{n}{p}} = X^{\frac{n}{p}} + a \pmod{X^r - 1, p} \tag{5}$$

for $0 \le a \le \ell$. Thus both n and $\frac{n}{p}$ behave like prime p in the above equation. We give a name to this property:

Definition 4.4. For polynomial $f(X)$ and number $m \in \mathcal{N}$, we say that m is *introspective* for $f(X)$ if

$$[f(X)]^m = f(X^m) \pmod{X^r - 1, p}.$$

It is clear from equations (5) and (4) that both $\frac{n}{p}$ and p are introspective for $X + a$ when $0 \le a \le \ell$.

The following lemma shows that introspective numbers are closed under multiplication:

LEMMA 4.5. *If m and m' are introspective numbers for $f(X)$ then so is $m \cdot m'$.*

Proof. Since m is introspective for $f(X)$ we have:

$$[f(X)]^{m \cdot m'} = [f(X^m)]^{m'} \pmod{X^r - 1, p}.$$

Also, since m' is introspective for $f(X)$, we have (after replacing X by X^m in the introspection equation for m'):

$$\begin{aligned}[f(X^m)]^{m'} &= f(X^{m \cdot m'}) \pmod{X^{m \cdot r} - 1, p} \\ &= f(X^{m \cdot m'}) \pmod{X^r - 1, p} \text{ (since } X^r - 1 \text{ divides } X^{m \cdot r} - 1).\end{aligned}$$

Putting together the above two equations we get:

$$[f(X)]^{m \cdot m'} = f(X^{m \cdot m'}) \pmod{X^r - 1, p}. \qquad \square$$

For a number m, the set of polynomials for which m is introspective is also closed under multiplication:

LEMMA 4.6. *If m is introspective for $f(X)$ and $g(X)$ then it is also introspective for $f(X) \cdot g(X)$.*

Proof. We have:

$$\begin{aligned}[f(X) \cdot g(X)]^m &= [f(X)]^m \cdot [g(X)]^m \\ &= f(X^m) \cdot g(X^m) \pmod{X^r - 1, p}.\end{aligned}$$

□

The above two lemmas together imply that every number in the set $I = \{(\frac{n}{p})^i \cdot p^j \mid i, j \geq 0\}$ is introspective for every polynomial in the set $P = \{\prod_{a=0}^{\ell}(X+a)^{e_a} \mid e_a \geq 0\}$. We now define two groups based on these sets that will play a crucial role in the proof.

The first group is the set of all residues of numbers in I modulo r. This is a subgroup of Z_r^* since, as already observed, $(n, r) = (p, r) = 1$. Let G be this group and $|G| = t$. G is generated by n and p modulo r and since $\mathrm{o}_r(n) > \log^2 n$, $t > \log^2 n$.

To define the second group, we need some basic facts about cyclotomic polynomials over finite fields. Let $Q_r(X)$ be the r^{th} cyclotomic polynomial over F_p. Polynomial $Q_r(X)$ divides $X^r - 1$ and factors into irreducible factors of degree $\mathrm{o}_r(p)$ [LN]. Let $h(X)$ be one such irreducible factor. Since $\mathrm{o}_r(p) > 1$, the degree of $h(X)$ is greater than one. The second group is the set of all residues of polynomials in P modulo $h(X)$ and p. Let $\mathcal{G}$ be this group which is generated by elements $X, X+1, X+2, \ldots, X+\ell$ in the field $F = F_p[X]/(h(X))$ and is a subgroup of the multiplicative group of F.

The following lemma proves a lower bound on the size of the group $\mathcal{G}$. It is a slight improvement on a bound shown by Hendrik Lenstra Jr. [Len], which, in turn, improved a bound shown in an earlier version of our paper [AKS].[2]

LEMMA 4.7 (Hendrik Lenstra Jr.). $|\mathcal{G}| \geq \binom{t+\ell}{t-1}$.

Proof. First note that since $h(X)$ is a factor of the cyclotomic polynomial $Q_r(X)$, X is a primitive r^{th} root of unity in F.

We now show that any two distinct polynomials of degree less than t in P will map to different elements in $\mathcal{G}$. Let $f(X)$ and $g(X)$ be two such polynomials in P. Suppose $f(X) = g(X)$ in the field F. Let $m \in I$. We also have $[f(X)]^m = [g(X)]^m$ in F. Since m is introspective for both f and g, and $h(X)$ divides $X^r - 1$, we get:

$$f(X^m) = g(X^m)$$

in F. This implies that X^m is a root of the polynomial $Q(Y) = f(Y) - g(Y)$ for every $m \in G$. Since $(m, r) = 1$ (G is a subgroup of Z_r^*), each such X^m is a

[2]Macaj [Mac] also proved this lemma independently.

primitive r^{th} root of unity. Hence there will be $|G| = t$ distinct roots of $Q(Y)$ in F. However, the degree of $Q(Y)$ is less than t by the choice of f and g. This is a contradiction and therefore, $f(X) \neq g(X)$ in F.

Note that $i \neq j$ in F_p for $1 \leq i \neq j \leq \ell$ since $\ell = \lfloor \sqrt{\phi(r)} \log n \rfloor < \sqrt{r} \log n < r$ and $p > r$. So the elements X, $X+1$, $X+2$, ..., $X+\ell$ are all distinct in F. Also, since the degree of h is greater than one, $X + a \neq 0$ in F for every a, $0 \leq a \leq \ell$. So there exist at least $\ell + 1$ distinct polynomials of degree one in $\mathcal{G}$. Therefore, there exist at least $\binom{t+\ell}{t-1}$ distinct polynomials of degree $< t$ in $\mathcal{G}$. □

In case n is not a power of p, the size of $\mathcal{G}$ can also be upper bounded:

Lemma 4.8. *If n is not a power of p then $|\mathcal{G}| \leq n^{\sqrt{t}}$.*

Proof. Consider the following subset of I:

$$\hat{I} = \{(\frac{n}{p})^i \cdot p^j \mid 0 \leq i, j \leq \lfloor \sqrt{t} \rfloor\}.$$

If n is not a power of p then the set $\hat{I}$ has $(\lfloor \sqrt{t} \rfloor + 1)^2 > t$ distinct numbers. Since $|G| = t$, at least two numbers in $\hat{I}$ must be equal modulo r. Let these be m_1 and m_2 with $m_1 > m_2$. So we have:

$$X^{m_1} = X^{m_2} \pmod{X^r - 1}.$$

Let $f(X) \in P$. Then,

$$\begin{aligned}[f(X)]^{m_1} &= f(X^{m_1}) \pmod{X^r - 1, p} \\ &= f(X^{m_2}) \pmod{X^r - 1, p} \\ &= [f(X)]^{m_2} \pmod{X^r - 1, p}.\end{aligned}$$

This implies

$$[f(X)]^{m_1} = [f(X)]^{m_2}$$

in the field F. Therefore, $f(X) \in \mathcal{G}$ is a root of the polynomial $Q'(Y) = Y^{m_1} - Y^{m_2}$ in the field F.[3] As $f(X)$ is an arbitrary element of $\mathcal{G}$, the polynomial $Q'(Y)$ has at least $|\mathcal{G}|$ distinct roots in F. The degree of $Q'(Y)$ is $m_1 \leq (\frac{n}{p} \cdot p)^{\lfloor \sqrt{t} \rfloor} \leq n^{\sqrt{t}}$. This shows $|\mathcal{G}| \leq n^{\sqrt{t}}$. □

Armed with these estimates on the size of $\mathcal{G}$, we are now ready to prove the correctness of the algorithm:

Lemma 4.9. *If the algorithm returns* PRIME *then n is prime.*

[3] This formulation of the argument is by Adam Kalai, Amit Sahai, and Madhu Sudan [KSS].

Proof. Suppose that the algorithm returns PRIME. Lemma 4.7 implies that for $t = |G|$ and $\ell = \lfloor \sqrt{\phi(r)} \log n \rfloor$:

$$\begin{aligned} |\mathcal{G}| &\geq \binom{t+\ell}{t-1} \\ &\geq \binom{\ell + 1 + \lfloor \sqrt{t} \log n \rfloor}{\lfloor \sqrt{t} \log n \rfloor} && (\text{since } t > \sqrt{t} \log n) \\ &\geq \binom{2\lfloor \sqrt{t} \log n \rfloor + 1}{\lfloor \sqrt{t} \log n \rfloor} && (\text{since } \ell = \lfloor \sqrt{\phi(r)} \log n \rfloor \geq \lfloor \sqrt{t} \log n \rfloor) \\ &> 2^{\lfloor \sqrt{t} \log n \rfloor + 1} && (\text{since } \lfloor \sqrt{t} \log n \rfloor > \lfloor \log^2 n \rfloor \geq 1) \\ &\geq n^{\sqrt{t}}. \end{aligned}$$

By Lemma 4.8, $|\mathcal{G}| \leq n^{\sqrt{t}}$ if n is not a power of p. Therefore, $n = p^k$ for some $k > 0$. If $k > 1$ then the algorithm will return COMPOSITE in step 1. Therefore, $n = p$. □

This completes the proof of Theorem 4.1.

5. Time complexity analysis and improvements

It is straightforward to calculate the time complexity of the algorithm. In these calculations we use the fact that addition, multiplication, and division operations between two m bits numbers can be performed in time $O^{\sim}(m)$ [vzGG]. Similarly, these operations on two degree d polynomials with coefficients at most m bits in size can be done in time $O^{\sim}(d \cdot m)$ steps [vzGG].

THEOREM 5.1. *The asymptotic time complexity of the algorithm is* $O^{\sim}(\log^{21/2} n)$.

Proof. The first step of the algorithm takes asymptotic time $O^{\sim}(\log^3 n)$ [vzGG].

In step 2, we find an r with $\mathrm{o}_r(n) > \log^2 n$. This can be done by trying out successive values of r and testing if $n^k \neq 1 \pmod r$ for every $k \leq \log^2 n$. For a particular r, this will involve at most $O(\log^2 n)$ multiplications modulo r and so will take time $O^{\sim}(\log^2 n \log r)$. By Lemma 4.3 we know that only $O(\log^5 n)$ different r's need to be tried. Thus the total time complexity of step 2 is $O^{\sim}(\log^7 n)$.

The third step involves computing the gcd of r numbers. Each gcd computation takes time $O(\log n)$ [vzGG], and therefore, the time complexity of this step is $O(r \log n) = O(\log^6 n)$. The time complexity of step 4 is just $O(\log n)$.

In step 5, we need to verify $\lfloor \sqrt{\phi(r)} \log n \rfloor$ equations. Each equation requires $O(\log n)$ multiplications of degree r polynomials with coefficients of size $O(\log n)$. So each equation can be verified in time $O^{\sim}(r \log^2 n)$ steps. Thus the time complexity of step 5 is $O^{\sim}(r\sqrt{\phi(r)} \log^3 n) = O^{\sim}(r^{\frac{3}{2}} \log^3 n) = O^{\sim}(\log^{21/2} n)$. This time dominates all the others and is therefore the time complexity of the algorithm. □

The time complexity of the algorithm can be improved by improving the estimate for r (done in Lemma 4.3). Of course the best possible scenario would be when $r = O(\log^2 n)$ and in that case the time complexity of the algorithm would be $O^{\sim}(\log^6 n)$. In fact, there are two conjectures that support the possibility of such an r (below ln is the natural logarithm):

Artin's Conjecture. Given any number $n \in \mathcal{N}$ that is not a perfect square, the number of primes $q \leq m$ for which $o_q(n) = q - 1$ is asymptotically $A(n) \cdot \frac{m}{\ln m}$ where $A(n)$ is Artin's constant with $A(n) > 0.35$.

Sophie-Germain Prime Density Conjecture. The number of primes $q \leq m$ such that $2q + 1$ is also a prime is asymptotically $\frac{2C_2 m}{\ln^2 m}$ where C_2 is the twin prime constant (estimated to be approximately 0.66). Primes q with this property are called Sophie-Germain primes.

Artin's conjecture—if it becomes effective for $m = O(\log^2 n)$—immediately shows that there is an $r = O(\log^2 n)$ with the required properties. There has been some progress towards proving Artin's conjecture [GM], [GMM], [HB], and it is also known that this conjecture holds under the Generalized Riemann Hypothesis.

If the second conjecture holds, we can conclude that $r = O^{\sim}(\log^2 n)$:

> By the density of Sophie-Germain primes, there must exist at least $\log^2 n$ such primes between $8 \log^2 n$ and $c \log^2 n (\log \log n)^2$ for a suitable constant c. For any such prime q, either $o_q(n) \leq 2$ or $o_q(n) \geq \frac{q-1}{2}$. Any q for which $o_q(n) \leq 2$ must divide $n^2 - 1$ and so the number of such q is bounded by $O(\log n)$. This implies that there must exist a prime $r = O^{\sim}(\log^2 n)$ such that $o_r(n) > \log^2 n$. Such an r will yield an algorithm with time complexity $O^{\sim}(\log^6 n)$.

There has been progress towards proving this conjecture as well. Let $P(m)$ denote the greatest prime divisor of number m. Goldfeld [Gol] showed that primes q with $P(q-1) > q^{\frac{1}{2}+c}$, $c \approx \frac{1}{12}$, occur with positive density. Improving upon this, Fouvry has shown:

LEMMA 5.2 ([Fou]). *There exist constants $c > 0$ and n_0 such that, for all $x \geq n_0$:*

$$|\{q \mid q \text{ is prime, } q \leq x \text{ and } P(q-1) > q^{\frac{2}{3}}\ \}| \geq c \frac{x}{\ln x}.$$

The above lemma is now known to hold for exponents up to 0.6683 [BH]. Using the above lemma, we can improve the analysis of our algorithm:

THEOREM 5.3. *The asymptotic time complexity of the algorithm is* $O^{\sim}(\log^{15/2} n)$.

Proof. As argued above, a high density of primes q with $P(q-1) > q^{\frac{2}{3}}$ implies that step 2 of the algorithm will find an $r = O(\log^3 n)$ with $o_r(n) > \log^2 n$. This brings the complexity of the algorithm down to $O^{\sim}(\log^{15/2} n)$. □

Recently, Hendrik Lenstra and Carl Pomerance [LP1] have come up with a modified version of our algorithm whose time complexity is provably $O^{\sim}(\log^6 n)$.

6. Future work

In our algorithm, the loop in step 5 needs to run for $\lfloor \sqrt{\phi(r)} \log n \rfloor$ times to ensure that the size of the group $\mathcal{G}$ is large enough. The number of iterations of the loop could be reduced if we could show that a still smaller set of $(X+a)$'s generates a group of the required size. This seems very likely.

One can further improve the complexity to $O^{\sim}(\log^3 n)$ if the following conjecture—given in [BP] and verified for $r \leq 100$ and $n \leq 10^{10}$ in [KS]—is proved:

CONJECTURE 6.1. *If r is a prime number that does not divide n and if*

$$(X-1)^n = X^n - 1 \pmod{X^r - 1, n}, \tag{6}$$

then either n is prime or $n^2 = 1 \pmod{r}$.

If this conjecture is true, we can modify the algorithm slightly to search first for an r which does not divide $n^2 - 1$. Such an r can assuredly be found in the range $[2, 4\log n]$. This is because the product of prime numbers less than x is at least e^x (see [Apo]). Thereafter we can test whether the congruence (6) holds or not. Verifying the congruence takes time $O^{\sim}(r \log^2 n)$. This gives a time complexity of $O^{\sim}(\log^3 n)$.

Recently, Hendrik Lenstra and Carl Pomerance [LP2] have given a heuristic argument which suggests that the above conjecture is false. However, some variant of the conjecture may still be true (for example, if we force $r > \log n$).

Acknowledgments. We wish to express our gratitude to Hendrik Lenstra Jr. for allowing us to use his observation on improving the lower bound on the size of the group $\mathcal{G}$. This has made the proof completely elementary (an earlier version required the density bound of Lemma 5.2), simplified the proof *and* improved the time complexity!

We are also thankful to Adam Kalai, Amit Sahai, and Madhu Sudan for allowing us to use their proof of Lemma 4.8. This has made the proofs of both upper and lower bounds on the size of $\mathcal{G}$ similar (both are now based on the number of roots of a polynomial in a field).

We thank Somenath Biswas, Rajat Bhattacharjee, Jaikumar Radhakrishnan, and V. Vinay for many useful discussions.

We thank Erich Bach, Abhijit Das, G. Harman, Roger Heath-Brown, Pieter Moree, Richard Pinch, and Carl Pomerance for providing us with useful references.

Since our preprint appeared, a number of researchers took the trouble of pointing out various omissions in our paper. We are thankful to all of them. We have tried to incorporate their suggestions in the paper and our apologies if we missed out on some.

Finally, we thank the anonymous referee of the paper whose suggestions and observations have been very useful.

Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, Kanpur, India
E-mail addresses: manindra@cse.iitk.ac.in
nitinsa@cse.iitk.ac.in
kayaln@cse.iitk.ac.in

References

[AH] L. M. Adleman and M.-D. Huang, *Primality Testing and Abelian Varieties over Finite Fields*, *Lecture Notes in Math.* **1512**, Springer-Verlag, New York, 1992.

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

[AB] M. Agrawal and S. Biswas, Primality and identity testing via Chinese remaindering, *Journal of the ACM* **50** (2003), 429–443.

[AKS] M. Agrawal, N. Kayal, and N. Saxena, PRIMES is in P, preprint (http://www.cse.iitk.ac.in/news/primality.ps), August 2002.

[Apo] T. M. Apostol, *Introduction to Analytic Number Theory*, Springer-Verlag, New York, 1997.

[Atk] A. O. L. Atkin, Lecture notes of a conference, Boulder (Colorado), Manuscript, August 1986.

[BH] R. C. Baker and G. Harman, The Brun-Titchmarsh Theorem on average, in *Analytic Number Theory*, *Volume* I (Allerton Park, IL, 1995), *Progr. Math.* **138**, 39–103, Birkhäuser Boston, Boston, MA, 1996.

[BP] R. Bhattacharjee and P. Pandey, Primality testing, Technical report, IIT Kanpur, 2001; available at http://www.cse.iitk.ac.in/research/btp2001/primality.html.

[Car] R. D. Carmichael, Note on a number theory function, *Bull. Amer. Math. Soc.* **16** (1910), 232–238.

[Fou] E. Fouvry, Théorème de Brun-Titchmarsh; application au théorème de Fermat, *Invent. Math.* **79** (1985), 383–407.

[GK] S. Goldwasser and J. Kilian, Almost all primes can be quickly certified, in *Proc. Annual ACM Symposium on the Theory of Computing*, 316–329, 1986.

[GM] R. Gupta and M. Ram Murty, A remark on Artin's conjecture, *Invent. Math.* **78** (1984), 127–130.

[GMM] R. Gupta, V. Kumar Murty, and M. Ram Murty, The Euclidian algorithm for S-integers, *Number Theory* (Montreal, Que., 1985), *CMS Conf. Proc.* **7** (1987), 189–202.

[Gol] M. Goldfeld, On the number of primes p for which $p + a$ has a large prime factor, *Mathematika* **16** (1969), 23–27.

[HB] D. R. Heath-Brown, Artin's conjecture for primitive roots, *Quart. J. Math. Oxford* **37** (1986), 27–38.

[KS] N. Kayal and N. Saxena, Towards a deterministic polynomial-time test, Technical report, IIT Kanpur, 2002; available at http://www.cse.iitk.ac.in/research/btp2002/primality.html.

[KSS] A. Kalai, A. Sahai, and M. Sudan, Notes on primality test and analysis of AKS, Private communication, August 2002.

[Lee] J. V. Leeuwen (ed.), *Handbook of Theoretical Computer Science*, *Volume A*, MIT Press, Cambridge, MA, 1990.

[Len] H. W. Lenstra, Jr., Primality testing with cyclotomic rings, unpublished (see http://cr.yp.to/papers.html#aks for an exposition of Lenstra's argument), August 2002.

[LN] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications*, Cambridge Univ. Press, Cambridge, 1986.

[LP1] H. W. Lenstra, Jr. and C. Pomerance, Primality testing with gaussian periods, Private communication, March 2003.

[LP2] ———, Remarks on Agrawal's conjecture, unpublished (http://www.aimath.org/WWN/ primesinp/articles/html/50a/), March 2003.

[Mac] M. Macaj, Some remarks and questions about the AKS algorithm and related conjecture, unpublished (http://thales.doa.fmph.uniba.sk/macaj/aksremarks.pdf), December 2002.

[Mil] G. L. Miller, Riemann's hypothesis and tests for primality, *J. Comput. Sys. Sci.* **13** (1976), 300–317.

[Nai] M. Nair, On Chebyshev-type inequalities for primes, *Amer. Math. Monthly* **89** (1982), 126–129.

[Pra] V. Pratt, Every prime has a succinct certificate, *SIAM Journal on Computing* **4** (1975), 214–220.

[Rab] M. O. Rabin, Probabilistic algorithm for testing primality, *J. Number Theory* **12** (1980), 128–138.

[SS] R. Solovay and V. Strassen, A fast Monte-Carlo test for primality, *SIAM Journal on Computing* **6** (1977), 84–86.

[vzGG] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge Univ. Press, Cambridge, 1999.

(Received January 24, 2003)