Primality Testing Using Elliptic Curves

SHAFI GOLDWASSER

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

JOE KILIAN

NEC Research Institute, Princeton, New Jersey

Abstract. We present a primality proving algorithm—a probabilistic primality test that produces short certificates of primality on prime inputs. We prove that the test runs in expected polynomial time for all but a vanishingly small fraction of the primes. As a corollary, we obtain an algorithm for generating large certified primes with distribution statistically close to uniform. Under the conjecture that the gap between consecutive primes is bounded by some polynomial in their size, the test is shown to run in expected polynomial time for all primes, yielding a Las Vegas primality test.

Our test is based on a new methodology for applying group theory to the problem of prime certification, and the application of this methodology using groups generated by elliptic curves over finite fields.

We note that our methodology and methods have been subsequently used and improved upon, most notably in the primality proving algorithm of Adleman and Huang using hyperelliptic curves and in practical primality provers using elliptic curves.

Categories and Subject Descriptors: F.1.2 [**Computation by Abstract Devices**]: Modes of Computation; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Distribution of primes, elliptic curves, group theory, Las Vegas algorithms, primes, prime certification, prime generation

---

The results of this paper appeared first in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, ACM, New York, 1986, pp. 316–329, and then in Chapter 2 of the second author's thesis (KILIAN, J. 1990. Uses of randomness in Algorithms and Protocols. MIT Press, Cambridge, Mass.). This paper is based upon the writeup in the thesis, with major revisions.

Research supported by NSF Postdoctoral Fellowship while at the MIT Laboratory for Computer Science.

The work of S. Goldwasser was supported by ARO grant DAAL 03-86-K-0171 and the National Science Foundation (NSF) grant 86-57527-CCR.

The research of J. Kilian was supported by NSF Postdoctoral Fellowship while at the MIT Laboratory for Computer Science.

Authors' addresses: S. Goldwasser, Computer Science Department, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, e-mail: shafi@wisdom.weizmann.ac.il; J. Kilian, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540.

1. *Introduction*

The written history of distinguishing prime numbers from composites goes back to Eratosthenes who came up with the first recorded algorithm for primality testing, in the 3rd century BC. He showed how to efficiently generate the set of primes from 1 to N in $O(N \ln \ln N)$ arithmetic steps.

Starting in the 17th century, mathematicians (Fermat, Euler, Legendre, and Gauss, to name a few) began to study primality once more. Their work laid the foundation for a new age in primality testing, which began in the 1970's. In this early work (see, for example, Brillhart et al. [1975] and Williams [1978]), factoring and primality testing were intimately related. Consequently, the algorithms were quite slow or worked for numbers of a special form (Brillhart et al. [1988] follows up on this work on factoring numbers of a particular form). Then, using elementary results from number theory, Miller [1976], Solovay-Strassen [1977], and Rabin [1980] developed efficient (polynomial time) algorithms for these problems.

Solovay-Strassen [1977] and Rabin [1980] give randomized primality tests. On an input N, these tests flip a sequence of coins, and compute its answer based on N and the outcome of these coins. If N is composite, the tests will with high probability output a proof (*witness*) that N is composite. If N is prime, they will fail to produce a witness of compositeness, giving probabilistic support to the assertion that N is prime, but no definitive proof.

Miller [1976] gives a deterministic polynomial-time algorithm for primality testing based on the Extended Riemann Hypothesis (ERH). On input N, the algorithm searches for a proof that N is composite. If it finds one, it stops and reports that N is composite, along with its proof of compositeness. If it doesn't find a proof of compositeness, the algorithm reports that either N is prime or the ERH is false. Hence, a proof of the ERH implies the existence of an efficient deterministic primality test; unfortunately, this proof is not currently within reach.

Adleman et al. [1983] and Cohen-Lenstra [1984] give nearly polynomial-time deterministic primality tests that do not rely on any unproven assumptions. They require $k^{\theta(\ln \ln k)}$ computational steps on an input N of length k. Furthermore, they do not provide any succinct proof of the primality number of a number it declares prime.

Given the previous success at producing proofs of compositeness, a natural question is whether one can produce short proofs of primality. We call such a short proof a *certificate of primality*. Pratt [1975] has shown that such a short certificate of primality always exists (and hence that primes are in NP), but while his method is constructive it requires one to factor large integers and thus does not run in polynomial time. Wunderlich [1983] discusses a heuristic that will efficiently find certificates for some primes; however, the set of primes certifiable in this manner is sparse, and indeed has not been proven to be infinite. However, it turns out that these techniques, albeit in much more general form, are useful in the efficient generation of certificates of primality for most (probably all) primes. This is the topic of our work.

We present a simple methodology for applying group theory to the problem of prime certification. We use this methodology, in conjunction with the theory of

elliptic curves, to develop an algorithm for prime certification. This algorithm has the following three properties.

(1) Given an input of length k, the algorithm produces a certificate of primality that is of length $O(k^2)$, and requires $O(k^4)$ steps to verify.

(2) The algorithm terminates in expected polynomial time on every prime number, provided that the following conjecture is true:

CONJECTURE 1. $(\exists c_1, c_2 > 0)\pi(x + \sqrt{x}) - \pi(x) \geq \dfrac{c_2\sqrt{x}}{\log^{c_1} x}$,

for x sufficiently large.

Here, $\pi(n)$ denotes the number of prime numbers that are less than n. This conjecture is very believable, for reasons that will be discussed later.

(3) There exist constants c_1 and c_2 such that for all k sufficiently large, the algorithm will terminate in expected $c_1 k^{11}$ time for all but at most,

$$\frac{2^k}{2^{k^{c_2/\ln\ln k}}},$$

of the inputs. In other words, the algorithm can be proved to run quickly on all but a vanishingly small fraction of the prime numbers.

A corollary to the above result is a method to efficiently generate large certified primes. Previous to our work, no method was known which provably produced more than a finite number of certified primes. Since we can certify most primes as prime, we can use the following simple algorithm to generate a k-bit certified prime with close to uniform distribution.

(1) Uniformly generate a random k-bit integer, n. Using a standard probabilistic test, attempt to prove it composite. If the attempt succeeds, repeat Step (1).

(2) Using our test, attempt to quickly (using only k^c steps, for some constant c) find a certificate of primality. If this succeeds, output n with its certificate. Otherwise, go to Step (1).

In other words, we randomly generate probable primes until we find one we can quickly certify. Since we can certify nearly all primes in expected polynomial time, and a random k-bit number will be prime with probability $O(1/k)$ (by the prime-number theorem), the above algorithm will terminate in expected polynomial time. The distribution on k-bit certified primes will be statistically very close to the uniform distribution on k-bit primes.

We note that the primality test we will describe has by now (subsequent to its appearance in conference proceedings [Goldwasser and Kilian 1986]) been implemented and incorporated in other algorithms (see below). We thus emphasize here the full and rigorous proof that for almost all primes the algorithm will terminate in expected polynomial time. This proof entails a careful analysis of the trade-off between the frequency of small intervals with no primes in them, and the number of primes that depend on these intervals for certification. The proof need not resort to any unproven assumptions on the distribution of primes in small intervals.

1.1. TECHNIQUES USED. Perhaps the most interesting aspect of our algorithm is the techniques it uses. In addition to using previous probabilistic primality tests as subroutines to guide our search for the primality certificate, we need to resort to the theory of elliptic curves and algorithms that compute the number of points on such curves over finite fields, and to the best known results on the density of primes in small intervals. We detail these usages below.

1.1.1. *Previous Primality Tests used in the Algorithm*. We use the previous state of the art in primality testing, both the randomized algorithms, the deterministic algorithms, and Pratt's proof that primes have short certificates. These three results are used in the following different ways.

Both Pratt's existential result and Wunderlich's heuristic successively reduces the primality of p to the primality of a set of smaller primes, $\{q_i\}$, by considering the order of elements of the group Z_p^*. We apply similar ideas, using groups generated by considering elliptic curves over Z_p, to reduce the primality of p to the primality of a significantly smaller prime q. For this step to be useful, it is important to be sure that q is indeed prime; this may be determined efficiently and with high confidence using the probabilistic tests of Solovay–Strassen and Miller–Rabin. Finally, we stop the recursion when q is small enough so that the deterministic algorithms of Adleman–Pomerance–Rumely and Cohen–Lenstra only require polynomial time in the size of the original input.

1.1.2. *The Theory of Elliptic Curves*. Given a prime $p \geq 5$ and a pair (A, B) where $A, B \in GF(p)$ and $4A^3 + 27B^2 \not\equiv 0 \bmod p$, we consider solutions (x, y) to the equation

$$y^2 \equiv x^3 + Ax + B \bmod p.$$

These sets of ordered pairs, when augmented by an extra point I, are the points of an *elliptic curve* over $GF(p)$. There is a natural addition operation under which the points of an elliptic curve form an Abelian group. Elliptic curves have been studied extensively from the standpoint of pure mathematics, and have been recently used in the development of algebraic algorithms.

Our algorithm uses Schoof's [1985] deterministic polynomial time algorithm for computing the number of points on an elliptic curve. The analysis of our algorithm uses a theorem of Lenstra [1987] concerning the distribution of the orders of elliptic curves.

We note that elliptic curves have been used earlier in the context of primality testing [Bosma 1985; Chudnovsky and Chudnovsky 1986].

1.1.3. *Results on the Density of Primes in Small Intervals*. The running-time analysis of our algorithm depends on the frequency of primes in intervals of the form $[x, x + \sqrt{x}]$, that is, on the value of $\pi(x + \sqrt{x}) - \pi(x)$. The Prime Number Theorem states that for sufficiently large x, $\pi(x)$ will approach $x/\ln x$, suggesting (but not implying) our conjecture (with $c_1 = 1$). A famous, widely believed conjecture of Cramer states that for sufficiently large x, $\pi(x + \ln^2 x) - \pi(x) > 0$, implying our conjecture, with $c_1 = 2$.

While no one has been able to prove our conjecture for all numbers, Heath-Brown [1978] have shown that our conjecture is true for most intervals. One of their technical lemmas implies the following result (communicated to us by H. Maier and C. Pomerance).

THEOREM [HEATH-BROWN]. *Call an integer y sparse if there are less than $\sqrt{y}/2\lfloor ln\ y \rfloor$ primes in the interval $[y, y + \lfloor \sqrt{y} \rfloor]$. Then there exist a constant α such that for sufficiently large x,*

$$|\{y : y \in [x, 2x],\ y \text{ is sparse}\}| < x^{5/6} \ln^{\alpha} x.$$

Heath-Brown's theorem allow us to analyze our algorithm for uniformly distributed inputs.

1.2. SUBSEQUENT RESEARCH. Our methodology has been used in two more recent algorithms. First, and foremost, Adleman and Huang [1987; 1992] have developed an algorithm that is guaranteed to find short certificates for all prime numbers. To do this, they first sharpen the analysis of an extended version of our algorithm [Goldwasser and Kilian 1986] to bound above the fraction of "bad" k-bit primes, which the elliptic curve based algorithms could not quickly certify, down to $2^{-\Omega(k)}$. Another exposition of this result will be given in Lenstra et al. [to appear]. This by itself is not of great interest, but turns out to be crucial to their next, much larger step. They then apply our methodology to a different class of groups, those generated by hyperelliptic curves. This yields an algorithm which first reduces the proof of primality for a prime p to a proof of primality for a sufficiently randomized prime q. Second, the sharpened version of the elliptic curve algorithm is used to prove that q prime. It can be shown that q is sufficiently random so that it will be certifiable with high probability.

Unfortunately, both the algorithm presented here and the algorithm of Adleman–Huang are quite slow in practice. Our algorithm takes $O(k^{11})$ expected time on most k-bit primes and the Adleman–Huang is even slower.

Our algorithm may be speeded up by using faster algorithms for computing the number of points on an elliptic curve over $GF(p)$. In practice, Schoof's algorithm has been made significantly more efficient [Atkin 1986a; 1988; 1992; Elkies 1998]. A survey of these results, many still unpublished, is given in Schoof [1995]. The current record for these techniques is the computation of the size of a group modulo a 500-digit prime (c.f. Morain [1995]).

Furthermore, Atkin [1986b] has developed a variant of our method, in which groups and their order are picked at the same time, that runs much quicker in practice. This is due to the fact that Schoof's algorithm for computing the number of points on the curve need not be run. This algorithm has been further improved in Kaltofen et al. [1989] and Atkin and Morain [1993]. Further discussions of elliptic curves and primality testing may be found in Morain [1990]. This class of algorithms has been used to certify primes of over 2,000 digits. We note that for numbers of a few thousand digits, a superpolynomial-time algorithm based on cyclotomy, due to Mihăilescu [1994], appears to be faster in practice; however, verifying these proofs is not much faster than generating them.

Unfortunately, the modification necessary to improve these algorithms' running times has frustrated attempts at rigorous analysis. A rigorous algorithm which is provably fast (in the practical sense of the word) still eludes us, as does a polynomial-time deterministic primality test. As partial progress on the latter problem, deterministic algorithms have been found that prove primality for infinite sets of primes [Pintz et al. 1989; Konyagin and Pomerance 1997].

Along a different line of research, Pomerance has used these techniques to prove the existence of very short certificates of primality [Pomerance 1987].

A more detailed discussion of using elliptic curves to finding small factors is (or will be) given in Lenstra [1993; to appear; to appear].

1.2.1. *Outline of the paper.* In Section 2, we give a quick introduction to elliptic curves. In Section 3, we give our new primality criterion and primality proving algorithm. In Section 4, we analyze the running time of the main step of our algorithm, as a function of the number of primes in certain small intervals. In Section 5, we show that our algorithm produces certificates for all primes in expected polynomial time, modulo a number-theoretic conjecture. We then extend this argument to show that our algorithm produces certificates for almost all primes in expected polynomial time. This last theorem depends on no unproven assumptions.

2. *An Introduction to Elliptic Curves*

For those unfamiliar with the basic theory of elliptic curves, we present a brief introduction to this field; more complete introductions appear in, for example, Silverman [1986], Tate [1974], and Lenstra and Lenstra [1987].

2.1. Definition. First, we define an elliptic curve, represented in *Weierstrass normal form*.

Definition 1. Let $\mathcal{F}$ be a field whose characteristic is not 2 or 3. An elliptic curve is an ordered pair (A, B), where $A, B \in \mathcal{F}$, and $4A^3 + 27B^2 \neq 0$.

Definition 2. Let $\mathcal{F}$ be a field whose characteristic is not 2 or 3, and let (A, B) be an elliptic curve over $\mathcal{F}$. We define the points of (A, B) to be the set of ordered pairs (x, y) such that $y^2 = x^3 + Ax + B$, and an additional element, I, called "the *point at infinity*." We denote these points by $E_{A,B}(F)$. If $F = GF(p)$, we use the abbreviation $E_{A,B}(p)$ to denote $E_{A,B}(GF(p))$.

2.2. Adding Points on an Elliptic Curve. There is a natural way of defining addition for the points on an elliptic curve. First, we define $a + I = I + a = a$ (I the identity). For the rest of this discussion, we write $L = (x_1, y_1)$ and $M = (x_2, y_2)$.

For elliptic curves over the reals, we can interpret our addition operation as illustrated in Figure 1 (this is known as the "tangent and chord" method). For the "general case", given points L and M, we first consider the line connecting L and M, and locate the third intersection point of the line with the points on curve (A, B). We then reflect this third point over the x-axis, and define the resulting point as $L + M$.

Some degenerate cases remain. If $L = M$, instead use the line tangent to the elliptic curve at L. If L and M are on a vertical line, we define $L + M = I$. Finally, if the line L and M fails to intersect the curve any other point, it can be shown that the line will be tangent to the curve at one of the two points of intersection. We treat this tangency as a double point of intersection, and use it as the "third" point.

Expressing these geometric operations algebraically, the resulting algorithm is given in Figure 2. This algorithm works for arbitrary fields such that $2, 3 \neq 0$.

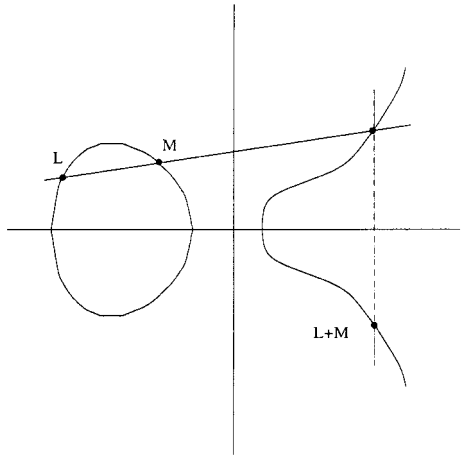

FIG. 1. Pictorial description of addition of points on an elliptic curve.

Algorithm ADD$((x_1, y_1), (x_2, y_2), (A, B))$

if $x_1 = x_2$ and $y_1 = -y_2$ then return(I)
if $x_1 = x_2$ and $y_1 = y_2$ then $\lambda = \frac{3x_1^2 + A}{2y_1}$
if $x_1 \neq x_2$ then $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$
$\beta = y_1 - \lambda x_1$
$x_s = \lambda^2 - x_1 - x_2$
$y_s = -(\lambda x_s + \beta)$
return$((x_s, y_s))$

FIG. 2. Algorithm for adding two points on an elliptic curve.

We define qL, where L is a point and q is an integer, by repeated addition in the natural manner. The value of qL may be efficiently computed via repeated doubling. That is,

$$qL = \begin{cases} L & \text{for } q = 1, \\ (L + L) * q/2 & \text{for } q \text{ even}, \\ L + (q - 1)L & \text{for } q \text{ odd}. \end{cases}$$

2.3. APPLYING ADD OVER Z_N. Algorithm ADD may be formally applied to points L and M on an elliptic curve over the ring Z_n; we use $L + M$ as shorthand for ADD(L, M). However, the requisite inverse elements may not exist, or it may be that $x_1 = x_2$ but $y_1 \neq \pm y_2$, in which case $L + M$ is undefined. These failures simply give a witness that n is composite, and indeed, yield nontrivial factors of n. We next observe that, when defined, ADD(L, M) "makes sense" when the coordinates of the point are taken mod p, where p is a prime divisor of n.

Let $p > 3$, $p|n$ and let $4A^3 + 27B^2 \neq 0 \pmod{p}$ (we can view A and B mod p as well as mod n, since $p|n$). Given $x \in Z_n$ we define x_p to be the natural projection from x to $GF(p)$. Given a point $L = (x, y) \in E_{A,B}(Z_n)$ we define $L_p = (x_p, y_p)$, and we define $I_p = I$. Note that $L_p \in E_{A,B}(p)$.

LEMMA 1. *If $L + M$ is defined, then $(L + M)_p = L_p + M_p$.*

PROOF. The lemma trivially holds if L or M is the identity; for the rest of the proof we write $L = (x_1, y_1)$ and $M = (x_2, y_2)$. First, note that for any rational function R over Z_n, either $R(x_1, x_2, \ldots)$ is undefined or

$$(R(x_1, x_2, \cdots)_p)_p = R((x_1)_p, (x_2)_p, \cdots),$$

Where in computing $R((x_1)_p, (x_2)_p, \ldots)$ the coefficients of R are taken mod p instead of mod n.

Algorithm ADD considers 3 cases:

(1) $x_1 = x_2$ and $y_1 = -y_2$, in which case ADD returns I.
(2) $x_1 = x_2$ and $y_1 = y_2$, in which case ADD returns

$$\left(\frac{P(x_1, x_2, y_1, y_2, A, B)}{(2y_1)^3}, \frac{Q(x_1, x_2, y_1, y_2, A, B)}{(2y_1)^3}\right).$$

(3) $x_1 \neq x_2$, in which case ADD returns

$$\left(\frac{R(x_1, x_2, y_1, y_2, A, B)}{(x_2 - x_1)^3}, \frac{S(x_1, x_2, y_1, y_2, A, B)}{(x_2 - x_1)^3}\right).$$

Here, P, Q, R, and S are polynomials. If (L, M) and (L_p, M_p) both fall into the same case, then ADD will compute the same rational function on x_1, x_2, y_1, y_2 as it computes on $(x_1)_p, (x_2)_p, (y_1)_p, (y_2)_p$, and the lemma follows. It remains to show that whenever (L, M) and (L_p, M_p) fall into different cases, $L + M$ is undefined. This event can happen if either

(1) $x_1 = x_2$, but $y_1 \neq \pm y_2$ or
(2) $x_1 \neq x_2$, but $(x_1)_p = (x_2)_p$

(The other "possibilities" can be eliminated since $a = b$ implies $a_p = b_p$ and $a = -b$ implies $a_p = -b_p$.) In the former case, ADD is undefined. In the latter case, $p|(x_1 - x_2)$, and ADD will thus be unable to compute the inverse of $x_1 - x_2$. □

2.4. THE GROUP STRUCTURE OF CURVES OVER $GF(p)$. We use some classical results about curves over Z_p, as well as some more recent results. First, the set of points of the elliptic curve (A, B) over Z_p form an Abelian group under the point addition operation defined above. This group is isomorphic to $Z^+_{m_1} \times Z^+_{m_2}$ for some m_1, m_2, where $m_1|m_2$ and $Z^+_{m_i}$ denotes the cyclic additive group of integers mod m_i.

We next consider the size of these groups. Given an elliptic curve (A, B), we denote by $\#_p(A, B)$ the number of points on (A, B) over $GF(p)$. For the rest of our discussion, we assume that $p \neq 2, 3$. The well-known *Riemann Hypothesis for Finite Fields* implies that

$$p + 1 - 2\sqrt{p} \leq \#_p(A, B) \leq p + 1 + 2\sqrt{p}.$$

The following theorem of Lenstra [1987] considers the distribution of $\#_p(A, B)$ when (A, B) is uniformly distributed. This result is crucial to our analysis.

THEOREM 1 [LENSTRA]. *Let $p > 5$ be a prime. Let,*

$$S \subseteq [p + 1 - \lfloor \sqrt{p} \rfloor, p + 1 + \lfloor \sqrt{p} \rfloor].$$

If curve (A, B) over Z_p is chosen uniformly, then,

$$prob(\#_p(A, B) \in S) > \frac{c}{\ln p} \cdot \frac{|S| - 2}{2\lfloor \sqrt{p} \rfloor + 1},$$

where c is some fixed constant.

Essentially, the size of a random group is at most $O(1/\ln p)$ times less likely to have a particular property as a randomly selected integer in

$$[p + 1 - \lfloor \sqrt{p} \rfloor, p + 1 + \lfloor \sqrt{p} \rfloor],$$

provided that $|S| > 2$.

Given a curve (A, B) over $GF(p)$, where p is a k-bit prime, there is an algorithm due to Schoof [1985] that deterministically computes $\#_p(A, B)$ in $O(k^9)$ steps. Improvements to Schoof's algorithm may be found in Atkin [1986a; 1988; 1992] and Elkies [1991].

3. *The Primality Proving Algorithm*

We present a new primality criterion using elliptic curves, and use it to create a new algorithm for proving primality.

3.1. A PRIMALITY CRITERION USING ELLIPTIC CURVES. Using Lemma 1, we can prove the following primality criterion. Theorem 2 is the heart of this paper; the remainder shows how to implement, use and analyze it in detail.

THEOREM 2. *Let n be an integer, not divisible by 2 or 3. Let $A, B \in Z_n$, and $(4A^3 + 27B^2, n) = 1$ and let $L \in E_{A,B}(Z_n)$, with $L \neq I$. If $qL = I$, for some prime $q > n^{1/2} + 2n^{1/4} + 1$, then n is prime.*

Formally, qL is shorthand for performing the repeated doubling algorithm described in Section 2.

PROOF. Our proof is by contradiction. If n is composite, then there exists a prime divisor p such that $p \leq \sqrt{n}$ and $p \neq 2, 3$. Furthermore, $4A^3 + 27B \neq 0 \bmod p$. Thus $L_p \in E_{A,B}(p)$ and $qL_p = I$, by repeated application of Lemma 1. Hence, the order of L_p must divide q and since $L_p \neq I$ and q is prime, its order must be equal to q. However, clearly, the order of L_p is at most $\#_p(A, B) \leq p + 2\sqrt{p} + 1 < q$, a contradiction. □

3.2. OVERVIEW OF THE PRIMALITY PROVING ALGORITHM. We focus on the problem of proving that a (prime) number is prime; throughout this discussion, p is prime.

We use our primality criterion to reduce the primality of p to the primality of a new prime, q, where $q \leq p/2 + o(p)$, and recursively prove that q is prime. For technical reasons, we eventually stop when the number to be proven prime is sufficiently small that it may be deterministically verified as prime. If too much time passes, the algorithm times out and starts over from scratch.

Algorithm GENERATE-CURVE(p)

1. Uniformly generate (A, B) until $(4A^3 + 27B^2, p) = 1$.
2. Compute $\#_p(A, B)$ using Schoof's algorithm. If this number is odd, go to Step 1. Otherwise, set $q = \#_p(A, B)/2$.
3. Run the probabilistic test of [41] (or [37]) on q for $2k$ trials (where p has k bits). If one of the trials ever outputs "composite", go to Step 1. If $2, 3|q$, go to Step 1.
4. Return$((A, B), q)$

FIG. 3. Algorithm for generating a curve of order $2q$, where q is prime.

3.3. THE BASIC REDUCTION STEP. Our basic reduction works as follows: Given a prime p, we construct a curve (A, B) over Z_p, and a point L on this curve with a prime order q, where $q \approx p/2$. We use our primality criterion to reduce the primality of p to the primality of q.

We first uniformly choose A, B such that $(4A^3 + 27B^2, p) = 1$ and its size $\#_p(A, B) = 2q$ for some prime q. We do this by uniformly choosing a pair (A, B), checking that $(4A^3 + 27B^2, p) = 1$, computing its size, $\#_p(A, B)$, using Schoof's algorithm, and checking that $\#_p(A, B) = 2q$, where q is a prime. We check q for primality using a standard primality testing algorithm, with an exceedingly small probability of error, say, $1/p$, where p is the number we initially wished to prove prime. We repeat the above steps until (A, B) until it passes both checks.

More generally, one may allow $\#_p(A, B) = rq$, where r is smooth or otherwise easy to factor out of $\#_p(A, B)$, and q is sufficiently large. Such considerations only complicate the analysis presented here, without giving stronger results. However, a more sophisticated analysis [Adleman and Huang 1992; Lenstra et al. to appear] does indeed give stronger results.

Using probabilistic primality tests introduces a small probability of error into our algorithm. However, as we will see later, it will be possible to correct such errors before they can cause an incorrect output.

We give the curve generation algorithm in Figure 3.

To choose L, we first independently and uniformly choose $x \in Z_p$ until $x^3 + Ax + B$ is a quadratic residue, then compute $y = \sqrt{x^3 + Ax + B}$, using the algorithm of Adleman et al. [1977], and uniformly choose which of the two square roots to take. Note that when x is chosen independently and uniformly, $x^3 + Ax + B$ is a quadratic residue with some constant probability; hence only a constant expected choices of x are needed. Once $x^3 + Ax + B$ is known to be a quadratic residue, then only an expected polynomial root-extraction algorithm is needed to ensure that L is generated in expected polynomial time.

We give the point selection algorithm in Figure 4, and the full algorithm for the main step in Figure 5.

3.4. THE COMPLETE PRIMALITY PROVING ALGORITHM. The full algorithm essentially iterates the main reduction algorithm until the prime to be proven is so small that it may be verified prime in time polynomial in k, the number of bits of the original prime number to be certified. We also have an abort condition, so

Algorithm SELECT-POINT$(p, q, (A, B))$

1. Choose x uniformly from Z_p until $z = x^3 + Ax + B$ is a quadratic residue.
2. Compute $y = \sqrt{z}$, uniformly choosing which square root of z to take. Set $L = (x, y)$.
3. Compute qL by repeated doubling. If $qL \neq I$, then go to Step 1. Otherwise, return(L).

FIG. 4. Algorithm for choosing a point on (A, B) of order q.

Algorithm MAIN-STEP(p)

1. Compute $(A, B), q \leftarrow \text{GENERATE} - \text{CURVE}(p)$ and $L \leftarrow \text{SELECT} - \text{POINT}(p, q, (A, B))$.
2. Return$((A, B), L, q)$.

FIG. 5. Main reduction step of the primality-proving algorithm.

that the algorithm will restart after sufficiently many steps have taken place. This mechanism handles the exponentially rare case when a mistake by the probabilistic primality test causes the algorithm to get stuck trying to prove a composite number prime. We give the complete algorithm in Figure 6. For the exposition of this algorithm, we let constant C to be defined as a positive constant such that the Cohen–Lenstra primality test takes $O(k)$ time on inputs of size

$$2^{(k)^{C/\lg \lg k}}.$$

It is easily verified that such a positive constant does exist.

3.5. CHECKING THE CERTIFICATE. We now show that the output of Prove-Prime(p) constitutes a certificate of p's primality, that can be deterministically checked in time $O(|p|^4)$. Our deterministic checker works as follows: On input

$$p, ((A_0, B_0), L_0, p_1), \ldots, ((A_{i-1}, B_{i-1}), L_{i-1}, p_i),$$

the algorithm first checks that p_i is small enough to be rapidly verified prime using the algorithm of Cohen and Lenstra [1984] (and hence does not need a certificate of primality), and aborts if this is not the case. It then verifies that p_i is prime, and aborts if it is not the case. Then, for $j = 1, \ldots, i - 1$, it verifies that

—p_j is not divisible by 2 or 3,
—(A_j, B_j) is a curve over Z_{p_j},
—$p_{j+1} > p_j^{1/2} + 2p_j^{1/4} + 1$, and
—$L_j \neq I_{p_j}$, $q_j L_j = I_{p_j}$.

We give this algorithm in Figure 7.

The following theorem shows that the output of our primality prover is indeed a certificate of primality.

THEOREM 3. *If algorithm* CHECK$(p,$ certificate$)$ *accepts, then p is prime. For a prime p, let* certificate *be an output of algorithm* PRIME-PROVE(p). *Then check* $(p,$ certificate$)$ *will accept in $O(|p|^4)$ deterministic time.*

Algorithm PROVE-PRIME(p)

1. Let $i = 0$, $p_0 = p$ and LOWERBOUND $= \max\left(2^{k^{C/\lg\lg k}}, 37\right)$ (where p is k bits long).
2. While $p_i >$ LOWERBOUND, do
$$(A_i, B_i), L_i, p_{i+1} \leftarrow \text{Main-Step}(p_i),$$
and set $i = i + 1$. Go to Step 1 if any p_i is divisible by 2 or 3.
3. Using the deterministic algorithm of [15, 9], check if p_i is prime. If p_i is not prime, then go to Step 1. Otherwise,
$$\text{Return}((A_0, B_0), L_0, p_1), \ldots, ((A_{i-1}, B_{i-1}), L_{i-1}, p_i).$$

* If, since starting Step 1, more than $k^{\lg k}$ steps have been run, abort and go to Step 1.

FIG. 6. The Primality Proving Algorithm.

Algorithm CHECK $(p, ((A_0, B_0), L_0, p_1), \ldots, ((A_{i-1}, B_{i-1}), L_{i-1}, p_i))$

1. Abort if $p_i > max\left(2^{k^{C/\lg\lg k}}, 37\right)$ (where p is k bits long). Otherwise, test p_i for primality, using the algorithm of [15].
2. Define $p_0 = p$. For $j \in [0, i-1]$, check that
   - p_i is not divisible by 2 or 3,
   - $(4A_j^3 + 27B_j^2, p_j) = 1$,
   - $p_{j+1} > p_j^{1/2} + 2p_j^{1/4} + 1$, and
   - $L_j \neq I_{p_j}, p_{j+1}L_j = I_{p_j}$.

   If any of these conditions do not hold, abort. Otherwise, accept p as prime.

FIG. 7. Algorithm for checking certificates of primality.

Note that this theorem makes no guarantee as to how quickly, if ever, prime-prover will output a certificate for p, merely that such a certificate will be valid.

PROOF. Suppose that CHECK accepts an input of the form,

$$p, ((A_0, B_0), L_0, p_1), \ldots, ((A_{i-1}, B_{i-1}), L_{i-1}, p_i).$$

Then clearly, p_i must be prime. Furthermore, by Theorem 2, the checks made for each value of j ensures that if p_{j+1} prime, then p_j is prime. Thus, if check accepts, then,

$$p_i \text{ prime} \Rightarrow p_{i-1} \text{ prime} \Rightarrow \cdots \Rightarrow p_0 = p \text{ prime}.$$

Thus, p must be prime.

We now show that CHECK will always accept a certificate, of the above form, presented to it by PROVE-PRIME. We first note that by the definition of PROVE-PRIME, p_i will be prime. By the definition of GENERATE-CURVE, we have $(4A_j^3 + 27B_j^2, p_j) = 1$. From the definition of GENERATE-CURVE, and the fact that $\#_{p_j}(A_j, B_j) \geq p_j + 1 - 2\sqrt{p_j}$, we have

$$p_{j+1} \geq \frac{p_j + 1 - 2\sqrt{p_j}}{2}$$

$$> p_j^{1/2} + 2p_j^{1/4} + 1,$$

for $p_j > 37$. By the definition of PROVE-PRIME, $p_j > 37$, unless $p \leq 37$, in which case it is easily verified that the output of PROVE-PRIME will be accepted by CHECK. Finally, by the definition of SELECT-POINT, $L_j \neq I_{p_j}$, and $p_{j+1}L_j = I_{p_j}$. Therefore, check will accept.

To compute how many steps are required for a k-bit prime, we first note that $p_{j+1} = p_j/2 + o(p_j)$, and therefore $i = O(\lg p) = O(k)$. For each value of j the checking procedure must perform a constant number of simple arithmetic operations, a single GCD computation, and must multiply a point L_j by an integer q_j. This all can be done in $O(k^3)$, so the total running time of the checking algorithm is $O(k^3) \cdot O(k) = O(k^4)$ steps.

4. *Analyzing the Main Step*

We now analyze the running time of MAIN-STEP in terms of the number of primes in an appropriate interval around $p/2$. Define $S(p)$ by

$$S(p) = \left\{ q : q \in \left[\frac{p + 1 - \lfloor \sqrt{p} \rfloor}{2}, \frac{p + 1 + \lfloor \sqrt{p} \rfloor}{2} \right], q \text{ prime.} \right\}.$$

LEMMA 2. *Let $p > 5$ be a k-bit prime, and suppose that $|S(p)| = O(\sqrt{p}/\lg^c p)$. Then algorithm* MAIN-STEP(p) *will run for expected $O(k^{c+8})$ steps before it terminates.*

PROOF. We bound the time required by GENERATE-CURVE; the SELECT-POINT procedure takes comparatively little time. Our procedure for finding a curve (A, B) of order $2q$ will take expected time equal to the expected time necessary to generate and test a single curve, multiplied by the expected number of curves it must try. The time necessary to test a curve is dominated by Schoof's algorithm which takes $O(|p|^8)$ steps. The expected time necessary to generate a curve, compute $(4A^3 + 27B^2, p)$ and to run the probabilistic primality tests are lower order polynomials in $|p|$.

We now bound the expected number of curves (A, B) we must try. For prime p, and for any value of A, there are at most two bad values of B, $\pm\sqrt{-4A^3/27} \bmod p$. Thus, with overwhelming probability, a randomly chosen (A, B) will constitute an elliptic curve. To bound the number of curves we must test before coming up with one whose order is twice a prime, we use Lenstra's theorem to relate this number to the size of the set $S(p)$.

LEMMA 3. *Let $p > 5$ be a prime, and let (A, B) be chosen uniformly from curves over Z_p. Let $S(p)$ be defined as above. Then*

$$prob(\#_p(A, B) \text{ is twice a prime}) > \frac{c}{\lg p} \cdot \frac{|S(p)| - 2}{2\lfloor \sqrt{p} \rfloor + 1},$$

where c is some fixed constant.

PROOF. There is a trivial bijection between numbers in the interval

$$[p + 1 - \lfloor \sqrt{p} \rfloor, p + 1 + \lfloor \sqrt{p} \rfloor]$$

which are twice a prime, and elements of $S(p)$. Applying Lenstra's theorem immediately gives the desired bound. □

By taking the reciprocal of this bound on the probability, and a simple calculation, we have that GENERATE-CURVE takes only $O(k^{c+9})$ expected steps.

It remains to verify that SELECT-POINT requires much less than $O(k^{c+9})$ expected steps. Assume that $E_{A,B}(p)$ is of order $2q$, where q is a prime. Recall that group $E_{A,B}(p)$ is isomorphic to a product of cyclic additive groups, $Z_{m_1} \times Z_{m_1}$, where $m_1 | m_2$. Since $E_{A,B}(p)$ is of size $2q$, we have $m_1 m_2 = 2q$, and hence $m_1 = 1$, $m_2 = 2q$, for $q > 2$. Thus, $E_{A,B}(p)$ will in fact be isomorphic to Z_{2q}. Since Z_{2q} has $q - 1$ points of order q, so must $E_{A,B}(p)$. Now, note that these points are paired: since (x, y) and $(x, -y)$ are inverse, $q(x, y) = I$ iff $q(x, -y) = I$. Thus, there are at least $(q - 1)/2$ values of x such that choosing x and $y = \sqrt{x^3 + Ax + B}$ will give a point on the curve of order q. Thus, the expected running time of SELECT-POINT is $2p/(q - 1) = O(1)$ times the amount of time it takes to randomly choose x, compute $y = \sqrt{x^3 + Ax + B}$, and check that $q(x, y) = I$. Checking that $z = x^3 + Ax + B$ is a quadratic residue, and then computing a square root of z using the algorithm of Adleman et al. [1977] naively takes $O(|p|^4)$ time. Note that the algorithm of [8] requires a quadratic nonresidue. One can simply choose an element of $GF(p)$ at random until one finds one, with a 1/2 probability of success each time; this step is a lower order contribution to the overall running time. Similarly, it takes $O(k^3)$ steps to add two points and $O(\lg q) = O(k)$ steps to check that $qL = I$ using repeated doubling. Thus, at most $O(k^4)$ expected steps are naively required. These naive running times can be improved, but suffices to show that the time required by SELECT-POINT is a low-order term. □

5. *Analysis of the Primality Proving Algorithm*

In the previous section, we exhibited our primality proving algorithm, and demonstrated that it produced legitimate certificates of primality. We also gave the running-time analysis of the main step of the algorithm, as a function of the number of primes in certain intervals.

In this section, we analyze how long it takes for the entire algorithm to produce proofs of primality. We show that, modulo a conjecture on the distribution of prime numbers, the algorithm will always halt in expected polynomial time. We then extend this argument to show that the algorithm will produce

proofs of primality, in expected polynomial time, for all but a vanishing fraction of the prime numbers. This latter theorem does not depend on any conjectures.

5.1. Analysis Based on a Conjecture. Using the machinery of the previous sections, it is straightforward to analyze the running time of our algorithm under an assumption about the distribution of primes. In the next section, we consider a relaxed, provable version of this assumption, under which we can show that our algorithm runs fast on most prime inputs.

Theorem 4. *Suppose that*,

$$(\exists c_1, c_2 > 0)\pi(x + \sqrt{x}) - \pi(x) \geq \frac{c_2\sqrt{x}}{\log^{c_1} x}.$$

then algorithm PROVE-PRIME(p) *will terminate in expected time* $O(|p|^{c_1+9})$ *for* p *sufficiently large*.

Proof. For ease of exposition, we assume that the probabilistic primality tester used subroutine never incorrectly identifies a composite number as prime, and that the time-out feature is never invoked. We then observe that dropping these assumptions doesn't significantly affect the analysis.

Let us simplify the expression for $S(p_j)$. Setting $x = (p_j + 1 - \lfloor\sqrt{p_j}\rfloor)/2$, and $y = (p_j + 1 + \lfloor\sqrt{p_j}\rfloor)/2$, we have,

$$S(p_j) = \{q \in [x, y], q \text{ prime}\}.$$

For $p_j > 37$, $y > x + \sqrt{x}$, and thus there must be $\Omega(\sqrt{x}/\log^{c_1} x)$ primes in $S(p_j)$. Therefore, by Corollary 2, GENERATE-CURVE(p_j) will take expected $O(|p_j|^{c_1+8}) \leq O(k^{c_1+8})$ steps (where $|p_j|$ denotes the number of bits of p_j). Thus, the algorithm will, in this optimistic scenario, run in expected $O(k^{c_1+9})$ time.

We now account for the possibility of a bad event: the algorithm timing-out or not detecting a composite. In each case, we assume that the algorithm runs for the maximum number of steps, denoted M, and then restarts. Let ρ denote the probability of a bad event, and E denote the expected number of steps the algorithm takes conditioned on no bad event occurring. Then by a straightforward analysis, the total expected time of the algorithm will be bounded above by

$$(\rho + \rho^2 + \cdots)M + E \leq O(\rho M + E),$$

when $\rho \leq 1/2$. Now, M is bounded by $k^{\lg k}$ by design and from the above analysis, $E = O(k^{c_1+9})$. It remains to bound ρ. The algorithm will never make more than M calls to the primality test, and the failure rate on any individual test is at most 1/LOWERBOUND, so the total probability of a primality test failing is $M \cdot 2^{-k^{C/\lg\lg k}}$, which is insignificant ($\ll 1/M^2$, for large p). If the algorithm doesn't make a mistake in its primality test, then by Markoff's inequality, the algorithm will take more than M steps with probability at most E/M. Hence, the algorithm will run for at most $O((E/M + o(1/M^2))M + E) = O(E)$ expected steps. Indeed, this analysis is quite weak; the increase in the expected time from these effects is much smaller. Also, note that if for small p a more reasonable choice of time-out

limits and error thresholds for the ordinary probabilistic primality tests were made, then this analysis would work for all p. □

5.2. PROVING OUR ALGORITHM FAST FOR MOST PRIMES. The scenario in the previous section is optimistic. It assumes that whenever one is attempting to show a number p prime, there will always be sufficiently many primes in the interval

$$\left[\frac{p+1-\lfloor\sqrt{p}\rfloor}{2}, \frac{p+1+\lfloor\sqrt{p}\rfloor}{2}\right].$$

That is, $S(p)$ is assumed to be sufficiently large. This is almost certainly the case for all primes, but it is currently beyond our ability to prove this fact. However, it has been shown that intervals that contain a sparse number of primes are rare. We have the following result, which is implied by a technical lemma of Heath-Brown [1978] (communicated to us by H. Maier and C. Pomerance).

THEOREM 5 [HEATH-BROWN]. *Call an integer y sparse if there are less than $\sqrt{y}/2\lfloor \ln y\rfloor$ primes in the interval $[y, y+\lfloor\sqrt{y}\rfloor]$. Then there exist a constant α such that for sufficiently large x,*

$$|\{y : y \in [x, 2x],\ y \text{ is sparse}\}| < x^{5/6}\ln^{\alpha} x.$$

We use this result to show that our algorithm is fast for most prime numbers. Let $BAD(k, T)$ denote the set of k-bit primes p such that PROVE-PRIME fails to output a certificate of primality for p in expected T steps. We prove the following theorem:

THEOREM 6. *There exist $c_1, c_2 > 0$ such that for k sufficiently large,*

$$BAD(k, c_1 k^{11}) \leq \frac{2^k}{2^{k^{c_2/\lg\lg k}}}$$

PROOF. Given a prime p, we denote by $P_i(p)$ the set of all intermediate primes that can be generated in step i of the algorithm. In other words, $P_i(p)$ consists of all primes that could conceivably be equal to p_i in the certificate generated for p. Thus, for instance,

$$P_0(p) = \{p\},\ P_1(p) \subseteq \left[\frac{p+1-\lfloor 2\sqrt{p}\rfloor}{2}, \frac{p+1+\lfloor 2\sqrt{p}\rfloor}{2}\right], \ldots$$

These are the only primes that need be considered for proving p prime. If it is the case that $S(p_i)$ is $O(\sqrt{p_i}/\ln p_i)$ for $p_i \in P_i(p)$, then by the same analysis as in the proof of Theorem 4, PROVE-PRIME(p) will terminate in expected time $O(k^{11})$. The rest of this proof consists of showing that this will be true for most primes.

Our proof proceeds in three stages. First, we bound the range in which $P_i(p)$ falls, and use this bound to derive a simple criterion which implies that $S(p_i)$ is large for $p_i \in P_i(p)$. Next, we use a result of Heath-Brown, and a simple combinatoric argument, to show that our criterion will fail for only a relatively

small number of values of $P_i(p)$. Finally, we use this result to bound the number of primes for which our algorithm is slow.

5.2.1. *Characterizing* $P_i(p)$. We note that for every certificate, $p_{i+1} = p_i/2 + o(p_i)$. This would suggest $P_i(p)$ should be clustered around $p/2^i$, as follows:

LEMMA 4. *Let p be a prime, and let* $p/2^i$ *be sufficiently large. Then any element of* $P_i(p)$ *lies in the range*,

$$\left(\frac{p}{2^i} - 7\sqrt{\frac{p}{2^i}}, \frac{p}{2^i} + 7\sqrt{\frac{p}{2^i}}\right).$$

Remark. The value of 7 that we obtain can be improved on. However, we only need to establish that some constant exists.

PROOF. Our proof is by induction on i. For $i = 0$, the lemma clearly holds. We can bound the largest and smallest elements of $P_i(p)$ in terms of the largest and smallest elements of $P_{i-1}(p)$. Specifically, we have

$$\max(P_i(p)) \leq \frac{\max(P_{i-1}(p)) + 1 + 2\sqrt{\max(P_{i-1}(p))}}{2}, \quad \text{and,}$$

$$\min(P_i(p)) \geq \frac{\min(P_{i-1}(p)) + 1 - 2\sqrt{\min(P_{i-1}(p))}}{2}.$$

By inductive hypothesis, we have,

$$\max(P_i(p)) \leq \frac{p/2^{i-1} + 7\sqrt{p/2^{i-1}} + 1 + 2\sqrt{p/2^{i-1} + 7\sqrt{p/2^{i-1}}}}{2}.$$

We can simplify the above expression considerably. We note that,

$$x + 7\sqrt{x} < (1 + o(1))x,$$

for x sufficiently large,

$$\frac{p/2^{i-1}}{2} = \frac{p}{2^i},$$

and,

$$\sqrt{\frac{p}{2^{i-1}}} = \sqrt{2} \cdot \sqrt{\frac{p}{2^i}}.$$

These simplifications yield,

$$\max(P_i(p)) < \frac{p}{2^i} + \left(\frac{7}{\sqrt{2}} + \sqrt{2} + o(1)\right)\sqrt{\frac{p}{2^i}} + 1$$

$$< \frac{p}{2^i} + 7\sqrt{\frac{p}{2^i}},$$

For $p/2^i$ sufficiently large. The lower bound is similarly established. □

5.2.2. *A Condition under which $S(p_i)$ Will be Large*. We can use Lemma 4 to give a simple condition under which we can guarantee that $S(p_i)$ will be large for all $p_i \in P_i(p)$. To facilitate the discussion, we first define a parameterized family of intervals, $\mathcal{I}_i(p)$.

Definition 3. Let $\mathcal{I}_i(p)$ denote the set of intervals of the form

$$\left[\frac{p_i + 1 - \lfloor\sqrt{p_i}\rfloor}{2}, \frac{p_i + 1 + \lfloor\sqrt{p_i}\rfloor}{2}\right],$$

where $p_i \in P_i(p)$.

That is, $\mathcal{I}_i(p)$ is the set of intervals which are important in our primality proving algorithm's search for p_{i+1}. If we can show that every interval in $\mathcal{I}_i(p)$ has a large number of primes, then we are guaranteed that our algorithm will always be able to quickly generate p_{i+1}.

Note that the above definition uses a $\sqrt{p}$ instead of a $2\sqrt{p}$ that one might expect given the Riemann hypothesis for finite fields. This is due to our definition of $\mathcal{S}(p_i)$, and more fundamentally due to the fact that Lenstra's result only holds for the smaller interval.

To bound the number of primes in each interval in $\mathcal{I}_i(p)$, we consider a constant-sized set of intervals, $\mathcal{C}_i(p)$, as follows.

Definition 4. Let $\mathcal{C}_i(p)$ be defined as the set of intervals,

$$\left\{[x_j, x_j + \lfloor\sqrt{x_j}\rfloor] : x_j = \left\lfloor\frac{p}{2^{i+1}} + \frac{j}{3} \cdot \sqrt{\frac{p}{2^{i+1}}}\right\rfloor, j \in [-22, 22]\right\}.$$

(The value of 22 can probably be reduced, but we only need the fact that some constant exists.)

LEMMA 5. *Let p be a prime, and let $p/2^i$ be sufficiently large. Then every interval in $\mathcal{I}_i(p)$ contains an interval in $\mathcal{C}_i(p)$.*

Thus, if every interval in $\mathcal{C}_i(p)$ has enough primes then every interval in $\mathcal{I}_i(p)$ has enough primes.

PROOF. Let $[x, y] \in \mathcal{I}_i(p)$. We have,

$$y = x + (\sqrt{2} + o(1))\sqrt{x}.$$

By the same argument as in Lemma 4, we have,

$$\frac{p}{2^{i+1}} + 7\sqrt{\frac{p}{2^{i+1}}} \geq x \geq \frac{p}{2^{i+1}} - 7\sqrt{\frac{p}{2^{i+1}}}.$$

Thus, for some $j \in [-21, 21]$, $x_{j-1} \leq x \leq x_j$, where x_j is as in Definition 4. We claim that $[x, y]$ contains the interval $[x_j, x_j + \lfloor\sqrt{x_j}\rfloor] \in \mathcal{C}_i(p)$; we must show that $y \geq x_j + \lfloor\sqrt{x_j}\rfloor$.

First, it is easily verified that $x, x_j = (1 + o(1))p/2^{i+1}$ (where the $o(1)$ term goes to 0 as $p/2^i$ grows sufficiently large). Next, $x_j - x \leq x_j - x_{j-1} = \sqrt{p/2^{i+1}}/3 + O(1)$ (the $O(1)$ compensates for the rounding. Hence, $(x_j + \lfloor\sqrt{x_j}\rfloor) - x = (4/3 + o(1))\sqrt{p/2^{i+1}}$. However, $y - x = (\sqrt{2} + o(1))\sqrt{p/2^{i+1}}$, hence $y \geq x$ for $p/2^i$ sufficiently large. □

5.2.3. *A Further Property of* $\mathcal{C}_i(p)$. Lemma 5 is crucial to our analysis. Instead of having to show that $|\mathcal{I}_i(p)| = O(\sqrt{p/2^{i+1}})$ intervals all have sufficiently many primes, we need only show that,

$$|\mathcal{C}_i(p)| = O(1),$$

intervals have sufficiently many primes (for most primes p). We first extend our notion of sparseness to $\mathcal{C}_i(p)$.

Definition 5. Let p be a prime. We say that $\mathcal{C}_i(p)$ is *sparse* if any of the intervals in $\mathcal{C}_i(p)$ is sparse.

Heath-Brown shows that only a vanishing fraction of the intervals of the form $[x, x + \lfloor\sqrt{x}\rfloor]$ will not have enough primes. But if these "bad" intervals appear in most sets $\mathcal{C}_i(p)$ they could destroy a disproportionate number of primes. The following lemma bounds this effect:

LEMMA 6. *Let x be sufficiently large. Then an interval of the form*,

$$[x, x + \lfloor\sqrt{x}\rfloor],$$

can be in $\mathcal{C}_i(p)$ *for at most* $c \cdot 2^i$ *different values of* p, *where* c *is some constant*.

PROOF. It suffices to show that there are, for each value of $k \in [-22, 22]$, only $O(2^i)$ values of p which satisfy the equation $\lfloor f_k(p)\rfloor = x$, where,

$$f_k(p) = \frac{p}{2^{i+1}} + \frac{k}{3} \cdot \sqrt{\frac{p}{2^{i+1}}}.$$

We first eliminate the integer rounding by noting that

$$\lfloor z\rfloor = x \Rightarrow x - 1 \leq z \leq x + 1.$$

We therefore have to show that there are only $O(2^i)$ integers p which satisfy,

$$x - 1 \leq f_k(p) \leq x + 1.$$

Therefore, if $[x, x + \lfloor\sqrt{x}\rfloor]$ is in $\mathcal{C}_i(p)$ and $\mathcal{C}_i(p')$, then $|f_k(p') - f_k(p)| \leq 2$. We will use this fact to show that p and p' must be near to each other in value,

which will in turn give us the desired bound. For the rest of the proof, we assume without loss of generality that $p \leq p'$.

Let us consider f_k in the continuous domain. For all $p > 0$, the derivative $f'_k(p)$ is at least $2^{-(i+1)}$. Since f_k is clearly monotone increasing, we have $f_k(p') - f_k(p) \leq 2$. By elementary calculus, we have,

$$f_k(p') - f_k(p) \geq \frac{p' - p}{2^{i+1}},$$

from which we can derive,

$$p' - p \leq 2 \cdot 2^{i+1}.$$

This clearly implies that only $c \cdot 2^i$ solutions exist, for some constant c. □

5.2.4. *The Final Calculation*. We now bound the number of primes for which our algorithm will fail. First, we argue that the number of k-bit primes p such that $\mathscr{C}_i(p)$ will be sparse will be small, where c is a positive constant.

LEMMA 7. *Let 2^{k-i} be sufficiently large. At most, $2^k/2^{(1/7)(k-i)}$ k-bit primes p are such that $\mathscr{C}_i(p)$ is sparse.*

Remark. Here, 1/7 may be replaced by any number less than 1/6.

PROOF. First, we note that if $[x, x + \lfloor\sqrt{x}\rfloor]$ is in $\mathscr{C}_i(p)$ for $p \in [2^{k-1}, 2^k]$, then $x \in [2^{k-i-2}, 2^{k-i+1}]$. This follows from the definition of $\mathscr{C}_i(p)$ and the bounds on p. We now use Heath-Brown's theorem on the intervals $[2^{k-i-2}, 2^{k-i-1}]$, $[2^{k-i-1}, 2^{k-i}]$, and $[2^{k-i}, 2^{k-i+1}]$, and sum the results. This bounds the number of sparse intervals in

$$\bigcup_{p \in [2^{k-1}, 2^k]} \mathscr{C}_i(p),$$

to at most,

$$\sum_{j=0}^{2} 2^{(5/6)(k-i-j)} \log^{\alpha} 2^{k-i-j} \leq c_1 \cdot 2^{(5/6)(k-i)} \log^{\alpha} 2^{k-i},$$

for some constant c_1. By Lemma 6, each sparse interval of this form is in $\mathscr{C}_i(p)$ for at most $c_2 2^i$ different values of p, where c_2 is some constant. Thus, at most

$$(c_2 \cdot 2^i)(c_1 \cdot 2^{(5/6)(k-i)} \log^{\alpha} 2^{k-i}) = c_1 c_2 \frac{2^k \log^{\alpha} 2^{k-i}}{2^{(1/6)(k-i)}}$$

$$\leq \frac{2^k}{2^{(1/7)(k-i)}},$$

for 2^{k-i} sufficiently large. □

We now upper-bound the number of k-bit primes that PROVE-PRIME will not quickly certify as prime. In order for a prime p to not be quickly certified, as per

the analysis of Theorem 4, it must be the case that $\mathcal{C}_i(p)$ is sparse for some value of i. Furthermore, the value of i must be sufficiently small that PROVE-PRIME(p) could, with nonzero probability, proceed for i steps without p_i being so small as to be verified deterministically. We denote by i_k the greatest number of reduction steps the algorithm could possibly go through on a k-bit prime. Using Lemma 7, we bound the number of k-bit primes that could conceivably not be quickly certified by

$$|\{p \in [2^{k-1}, 2^k], p \text{ isn't quickly certified}\}| \leq \sum_{j=1}^{i_k} \frac{2^k}{2^{(1/7)(k-i)}},$$

$$\leq \frac{c \cdot 2^k}{2^{(1/7)(k-i_k)}},$$

for some constant c, by standard properties of geometric series.

We can use Lemma 4 to bound below the value of 2^{k-i_k}. Suppose that, on some k-bit prime p, the primality proving algorithm proceeded for i_k reductions before hitting its last prime, p_{i_k}. Recall that, for k sufficiently large, the algorithm will stop as soon as,

$$p_{i_k} \leq 2^{k^{C/\lg \lg k}}.$$

We also have,

$$p_{i_k-1} \geq 2^{k^{C/\lg \lg k}},$$

or the algorithm would have stopped after the $(i_k - 1)$th reduction. Since, $p_{i_k} \geq p_{i_k-1}/3$, for p_{i_k-1} sufficiently large (to give a very weak bound), we have,

$$p_{i_k} \geq \frac{1}{3} \cdot 2^{k^{C/\lg \lg k}},$$

for k sufficiently large. Since $p_{i_k} = p/2^{i_k} + O(\sqrt{p/2^{p(i)}})$, by Lemma 4, and $k - 1 \leq \lg p \leq k$, we have,

$$\lg p_{i_k} \leq k - i_k + 1,$$

for k sufficiently large. Hence, we have,

$$2^{k-i_k} \leq \frac{1}{6} \cdot 2^{k^{C/\lg \lg k}}.$$

If follows then that

$$\frac{c \cdot 2^k}{2^{(1/7)(k-i_k)}} \leq c' \cdot \frac{2^k}{2^{(1/7)(k-1)^{C/\lg \lg(k-1)}}}, \qquad \text{for some } c' > 0$$

$$\leq \frac{2^k}{2^{(k)^{C'/\lg \lg k}}},$$

for some suitably chosen C'. □

REFERENCES

ADLEMAN, L. M., AND HUANG, M. 1987. Recognizing primes in polynomial time. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (New York, N.Y., May 25–27). ACM, New York, pp. 462–471.

ADLEMAN, L. M., AND HUANG, M. 1992. Primality testing and Abelian varieties over finite fields. In *Lecture Notes in Mathematics*, vol. 1512. Springer-Verlag, New York.

ADLEMAN, L. M., MANDERS, K., AND MILLER, G. L. 1977. On taking roots in finite fields. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*. IEEE, New York, pp. 175–178.

ADLEMAN, L. M., POMERANCE, C., AND RUMELY, R. 1983. On distinguishing prime numbers from composite numbers. *Ann. Math. 117*, 173–206.

ATKIN, A. O. L. 1986a. Schoof's algorithm. Manuscript.

ATKIN, A. O. L. 1986b. Manuscript.

ATKIN, A. O. L. 1988. The number of points on an elliptic curve modulo a prime. Manuscript.

ATKIN, A. O. L. 1992. The number of points on an elliptic curve modulo a prime (II). Manuscript.

ATKIN, A. O. L., AND MORAIN, F. 1993. Elliptic curves and primality proving. *Math. Comput. 61*, 203 (July), 29–68.

BOSMA, W. 1985. Primality testing using elliptic curves. Tech. Rep. 8512. Math. Instituut, Univ. Amsterdam, Amsterdam, The Netherlands.

BOSMA, W., AND VAN DER HULST, M. P. 1990. Faster primality testing. In *Proceedings of EUROCRYPT '89*. Lecture Notes in Computer Science, vol. 434. Springer-Verlag, New York, pp. 652–656.

BRILLHART, J., LEHMER, D. H., AND SELFRIDGE, J. L. 1975. New primality criteria and factorizations of $2^m \pm 1$. *Math. Comput. 29*, 130, 620–647.

BRILLHART, J., LEHMER, D. H., SELFRIDGE, J. L., TUCKERMAN, B., AND WAGSTAFF, JR., S. S. 1988. Factorizations of $b^n + 1$; $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers. *Cont. Math. 2*, 22.

CHUDNOVSKY, D., AND CHUDNOVSKY, G. 1986. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Adv. App. Math. 7*.

COHEN, H., AND LENSTRA, JR., H. W. 1984. Primality testing and Jacobi sums. *Math. Comput. 42*.

ELKIES, N. D. 1991. Explicit isogenies. Manuscript.

ELKIES, N. D. 1998. Elliptic and modular curves over finite fields and related computational issues. In *Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A. O. L. Atkins*, D. A. Buell and J. T. Teitelbaum, eds. AMS/IP Studies in Advanced Mathematics, vol. 7. American Mathematics Society, Providence, R. I., pp. 21–76.

GOLDWASSER, S., AND KILIAN, J. 1986. Almost all primes can be quickly certified. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (Berkeley, Calif., May 28–30). ACM, New York, pp. 316–329.

HEATH-BROWN, D. R. 1978. The differences between consecutive primes. *J. London Math. Soc. 2*, 18, 7–13.

KALTOFEN, E., VALENTE, T., AND YUI, N. 1989. An improved Las Vegas primality test. In *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation* (ISSAC '89) (Portland, Ore., July 17–19), Gilt Gonnet, ed. ACM, New York, pp. 26–33.

KILIAN, J. 1990. Uses of Randomness in Algorithms and Protocols. MIT Press, Cambridge, Mass.

KONYAGIN, S., AND POMERANCE, C. 1997. On primes recognizable in deterministic polynomial time. In *The Mathematics of Paul Erdös*, R. Graham and J. Nešetřil, eds. Springer-Verlag, New York, pp. 177–198.

LENSTRA, JR., H. W. 1987. Factoring, integers with elliptic curves. *Ann. Math. 126*, 649–673.

LENSTRA, A., AND LENSTRA, JR., H. W. 1987. Algorithms in number theory. Tech. Rep. 87-008. Univ. Chicago, Chicago, Ill.

LENSTRA, JR., H. W., PILA, J., AND POMERANCE, C. 1993. A hyperelliptic smoothness test, I. *Philos. Trans. Roy Soc. London Ser. A 345*, 397–408.

LENSTRA, JR., H. W., PILA, J., AND POMERANCE, C. 1999. A hyperelliptic smoothness test, II. Manuscript.

LENSTRA, JR., H. W., PILA, J., AND POMERANCE, C. 1999. A hyperelliptic smoothness test, III. To appear.

MIHĂILESCU, P. 1994. Cyclotomy primality proving—Recent developments. In *Proceedings of the 3rd International Algorithmic Number Theory Symposium* (*ANTS*). Lecture Notes in Computer Science, vol. 877. Springer-Verlag, New York, pp. 95–110.

MILLER, G. L. 1976. Riemann's hypothesis and test for primality. *J. Comput. Syst. Sci. 13*, 300–317.

MORAIN, F. 1990. Courbes elliptques et tests de primalité. Ph.D. dissertation. Univ. Claude Bernard-Lyon I.

MORAIN, F. 1995. Calcul de nombre de points sur une courbe elliptique dans un corps fini: Aspects algorithmiques. *J. Théor. Nombres Bordeaux 7*, 255–282.

PINTZ, J., STEIGER, W., AND SZEMEREDI, E. 1989. Infinite sets of primes with fast primality tests and quick generation of large primes. *Math. Comput. 53*, 187, 399–406.

POMERANCE, C. 1987. Very short primality proofs. *Math. Comput. 48*, 177, 315–322.

PRATT, V. R. 1975. Every prime has a succinct certificate. *SIAM J. Comput. 4*, 3, 214–220.

RABIN, M. 1980. Probabilistic algorithms for testing primality. *J. Numb. Theory 12*, 128–138.

SCHOOF, R. 1985. Elliptic curves over finite fields and the computation of square roots modulo p. *Math. Comput. 44*, 483–494.

SCHOOF, R. 1995. Counting points on elliptic curves over finite fields. *J. Théor. Nombres. Bordeaux 7*, 219–254.

SILVERMAN, J. 1986. The arithmetic of elliptic curves. In *Graduate Texts in Mathematics*, vol. 106. Springer-Verlag, New York.

SOLOVAY, R., AND STRASSEN, V. 1977. A fast Monte-Carlo test for primality. *SIAM J. Comput. 6*, 1, 84–85.

TATE, J. 1974. The arithmetic of elliptic curves. *Invent. Math. 23*, 179–206.

WILLIAMS, H. C. 1978. Primality testing on a computer. *Ars. Combinat. 5*, 127–185.

WUNDERLICH, M. C. 1983. A performance analysis of a simple prime-testing algorithm. *Math. Comput. 40*, 162, 709–714.

RECEIVED FEBRUARY 1991; REVISED MAY 1999; ACCEPTED JUNE 1999