Almost All Primes Can be Quickly Certified

Shafi Goldwasser^{*} EECS Department Joe Kilian^{*} Department of Mathematics

and Laboratory for Computer Science Massachusetts Institute of Technology

ABSTRACT

This paper presents a new probabilistic primality test. Upon termination the test outputs "composite" or "prime", along with a short proof of correctness, which can be verified in deterministic polynomial time. The test is different from the tests of Miller [M], Solovay-Strassen [SS], and Rabin [R] in that its assertions of primality are certain, rather than being correct with high probability or dependent on an unproven assumption.

The test terminates in expected polynomial time on all but at most an exponentially vanishing fraction of the inputs of length k, for every k. This result implies:

• There exist an infinite set of primes which can be recognized in expected polynomial time.

• Large certified primes can be generated in expected polynomial time.

Under a very plausible condition on the distribution of primes in "small" intervals, the proposed algorithm can be shown to run in expected polynomial time on every input. This

* Research supported in part by NSF Grant 8509905DCR

© 1986 ACM 0-89791-193-8/86/0500/0316 \$00.75

condition is implied by Cramer's conjecture.

The methods employed are from the theory of elliptic curves over finite fields.

1. INTRODUCTION

1.1 Testing Primality: Brief Review

Distinguishing prime numbers from composites has intrigued mathematicians as early as about 274 B.C, when the sieve algorithm of Eratosthenes has been allegedly recorded. Much progress has been made on this problem since the 17th century by Fermat, Euler, Legendre and Gauss.

With the arrival of fast computational devices new algorithmic ideas based on the work of Fermat and Gauss were proposed and implemented (see [D],[BLS]). These algorithms mostly relied on factoring and thus where impractical for even moderate size inputs.

The interest in primality in complexity theory was invoked by the exciting primality tests of Miller[M], Solovay and Srassen [SS], and Rabin [R].

Miller's algorithm [M] is a deterministic polynomial time procedure, which when answering "composite" gives a proof of correctness, and when answering "prime" does not. The assertions of primality made by the algorithm are always correct if if the Extended Riemann Hypothesis (ERH) is true. However, if the ERH is false, the numbers declared prime may still be composite. Thus, the ERH is not used to bound the running time of the algorithm, but to vouch for the correctness of the answer.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

The probabilistic primality tests of Solovay-Strassen [SS] and Rabin [SS], essentially perform a probabilistic search for a proof of compositeness. The failure of this search, provides circumstantial evidence that the number is not composite. These algorithms always terminate in polynomial time on every input. Upon termination they declare the input either *composite* or *probably prime*. When a number is declared "composite", a short (verifiable in deterministic polynomial time) proof (certificate) of compositeness is provided. When a number is declared "probably prime", then it is a prime with very high probability, but no certainty is provided.

The fastest deterministic algorithm known is due to Adleman, Pomerance and Rumley [APR] (followed by Choen-Lenstra[CL]) and runs in time $O(k^{c \log k})$ on inputs of length k. The answers of this algorithm are always correct. Unfortunately, it is not only slow but, like its predecessors, does do not provide us with a short certificate (i.e polynomial time verifiable proof) of its assertions of primality.

As discussed above, finding a short certificate of compositeness can be done quickly probabilistically. But, how about short proofs of primality? Although it is not as obvious as in the case of compositeness, Pratt [P] has shown that short proofs of primality do exist (i.e the set PRIMES is in NP). Unfortunately, finding a Pratt-certificate for a given prime involves being able to factor quickly, which is hard.

Partial progress toward finding short proof of primality quickly was made by Furer [F]. He shows a Las Vegas (always correct, probably fast) algorithm distinguishing between n - a product of two primes and the n a prime (provided $n \neq 1$ mod 24).

To summarize, the following questions remain open:

• Is there an infinite set of primes which can be recognized in expected polynomial time ?

• Can random large certified primes be generated in expected polynomial time?

• Is there a probabilistic primality test which is alaways correct and probably fast on every prime input, i.e a Las Vegas primality test ?

1.2 Our Results

In this paper, we propose a probabilistic algorithm which upon termination outputs either "prime" or "composite", along with a short proof (certificate) of correctness. The proof of correctness can be verified by a deterministic polynomial time algorithm.

We prove the following.

Theorem 1: Given any prime p of length k, our algorithm outputs a certificate of correctness of size $O(k^2)$, which can be verified correct in $O(k^4)$ deterministic time.

Theorem 3: For every size k > 0, our algorithm terminates in expected polynomial time on at least $1 - O(2^{-n \log \log n})$ of the prime inputs of length k. Note that the fraction of primes for which we could not prove that the algorithm terminates in expected polynomial time is smaller than any polynomial in k fraction.

Let $\pi(x)$ denote the number of primes smaller than x.

Theorem 2: Our algorithm terminates in expected polynomial time on every input if the following conjecture is true:

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

for sufficiently large x.

π

The above conjecture is implied by the wellknown Cramer's conjecture concerning the maximail size gap between consecutive primes (see section 1.2.3 for details).

Theorems 1 through 3 imply the following.

1.2.1 An infinite set of primes recognized in expected polynomial time

Theorem 2 implies that there exists an infinite set of primes which cen be recognized in expected polynomial time. In addition the set contains almost all primes of length k, for every k > 0.

1.2.2 Generating Large Certified Primes

A trivial application of this result is that we can, given any length k, generate a random prime p of length k, along with a short proof (certificate) that p is prime, such that the distribution of the generated primes is very close to uniform. One can simply run our algorithm on random k digit numbers until one is found that can be proven prime. If the algorithm takes too long on a particular input, it can be restarted on a different random input. The expected number of random numbers to be tried before a certifiably prime one is found is polynomial in k.

Moreover, certified members of special subsets of the prime numbers, can be produced at random. If one wished, for instance, to find an kdigit prime of the form 4t+3, one could randomly try k digit numbers until one was found that was certifiably prime, and also of the form 4t+3. Since there is a large fraction of k digit primes of this form, this algorithm is guaranteed to find one in expected polynomial time. In general, members of any sufficiently dense subset of the prime numbers may be found in this manner.

An application of this technique is in producing random numbers with certified factorization. Bach[B] showed how to generate uniformly distributed numbers of a given length with "known" factorization. However, one could never be certain that the given factorization was complete. Using Bach's algorithm in conjunction with our algorithm, one can randomly generate nearly uniformly distributed numbers with certified, known factorization.

1.2.3 How likely is our conjecture ? Let us restate our conjecture:

$$\exists c_1, c_2 > 0$$

$$\pi(x + \sqrt{x}) - \pi(x) \geq \frac{c_2 \sqrt{x}}{\log^{c_1} x}$$

> 0

for sufficiently large x.

A famous conjecture, due to Cramer, concerning the maximal gap between two primes, implies ours. His conjecture can be restated as: for sufficiently large x,

$$\pi(x+\log^2 x)-\pi(x)>0$$

Additional support of our conjecture, is provided by the following facts about the density of the primes.

The Prime Number Theorem tells us that for suff. large x.

$$\pi(x) \to \frac{x}{\log x}$$

The best bound known for the maximal gap between two primes is due to Heath-Brown and Iwanicc[HI]. They show, that for sufficiently large x, there always exist a prime in the interval $[x, x + x^{11/20}]$.

Finally, a theorem by Heath-Brown [HB] implies that there exists constants c_1, c_2 such that for sufficiently large x, the number of intervals $[y, y + \sqrt{y}]$ where $x \leq y \leq 2x$ in which there are less than $\frac{c_1\sqrt{y}}{\log y}$ primes, is less than $x^{5/6}\log^{c_2}x$ [MP].

Heath- Brown's result is also used, in section 4, to show that our algorithm terminates in expected polynomial time on almost all prime inputs.

1.3 New techniques: Elliptic Curves

Most primality tests previously proposed used mathematics developed in days of Fermat and Gauss.

Recently, methods from elliptic curves (see survey by Tate [T]) over finite fields have been used for speeding up problems in computational number theory. This has started with Schoof's[Sc] deterministic algorithm to compute square roots modulo primes. In the same work, Schoof shows a polynomial time algorithm to compute the order of the group generated by an elliptic curve over a finite field. The usage of Schoof's algorithm, is crucial to our primality test.

Lenstra [L] uses elliptic curves to obtain an integer factorization method which uses nearly constant memory, and whose running time is a function of the size of the smallest prime divisor of the integer to be factored. The running time analysis of Lenstra's algorithm depends on a very plausible assumption concerning the distribution of smooth numbers in small intervals, and requires no assumptions about elliptic curves. This is due to a result he proves concerning the distribution of the order of elliptic curves. We make use of this result in the analysis of the running time of our algorithm.

The use of elliptic curves in solving the two oldest problems in number theory: factoring [L] and primality, suggests they may be quite useful in solving other computational number theoretic problems as well.

GUIDELINE TO THE PAPER

In section 2 a very high level sketch of the algorithm is given.

In section 3 the necessary background from the theory of elliptic curves is reviewed, and notation is established.

The description of the algorithm is resumed in section 4. Section 4.1 presents the full version of the algorithm, and some of its technical details are discussed. In section 4.2 correctness of the algorithm is proved. In section 4.3, the expected running time of the algorithm is analyzed using the conjecture mentioned in section 1.2.3. In section 4.4, slight improvements to the main algorithm are proposed.

Section 5 presents a more sophisticated analysis of the running time of our algorithm, and it is proved that the algorithm will terminate in expected polynomial time on nearly all primes.

2. OVERVIEW OF THE ALGORITHM

Our algorithm has some of the flavor of Pratt's[P] nondeterministic algorithm for generating short proofs of primality. Given some prime p, Pratt proves that p is prime by exhibiting some $g \in Z_p^*$ such that $O_p(g) = p - 1$. To prove that $O_p(g) = p - 1$, the prime decomposition of $p - 1 = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ must be exhibited, with proof (i.e. one must recursively show that all the p_i 's are prime). Once this has been accomplished, one can show that $O_p(g) \neq \frac{p-1}{p_i}$ $(1 \le i \le k)$ by a straightforward computation.

Pratt's nondeterministic algorithm is ineffective, since it is hard to generate the prime decomposition of p-1.

We overcome this difficulty by working with elliptic groups mod p instead of Z_p^* . Given any prime modulus, there exist many elliptic groups with varying orders. Moreover, it is easy to generate such groups at random. We exploit this extra degree of freedom in a manner similar to Lenstra's factoring algorithm, as follows.

We randomly pick elliptic groups mod p until one is found whose order has a sufficiently large prime factor, q. Such a group can be utilized to generate a proof that p is prime if q is prime. We then recursively prove that q is prime, finally stopping when the number to be proven prime is sufficiently small.

Using results by Lenstra and others on the distribution of the order of elliptic groups mod p, we can deduce important properties of our algorithm from standard results (and/or well known conjectures) about the distribution of primes in small intervals. Thus, our analysis requires no unproven assumptions about elliptic groups.

3. OVERVIEW OF THE ELEMENTARY THEORY OF ELLIPTIC CURVES

3.1 Definition of Elliptic Curves and Elliptic Groups

Given an arbitrary field \mathbf{F} , we define an elliptic curve, represented in Weierstrauss normal form, to be the set of points (x, y) which satisfy the equation $y^2 = x^3 + Ax + B$, where $A, B \in \mathbf{F}$ and $4A^3 + 27B^2 \neq 0$. For the rest of our discussion, we assume that the characteristic of \mathbf{F} is not 2 or 3.

How to add points on an elliptic curve.

If one adds a special element, I(usually referred to as the point at infinity) to the set of solutions, one gets an abelian additive group, where <math>I is the identity. Addition is defined by the "tangent and chord" method illustrated in figure 1. In figure 1, \mathbf{F} is the real line.

Given points L and M, not equal to I, consider the line connecting them, or, if L=M, the line tangent to the curve at L. If this line is vertical, define L + M to be I. Otherwise, it is

guaranteed to intersect the curve at a third point. Define L+M to be the reflection of this point over the x axis.



figure 1

One can calculate the equation for this line (of the form $y = \lambda x + \beta$), which can then be used to find the third intersection point. Thus, given $L = (x_1, y_1), M = (x_2, y_2)$, we can compute L + M by the following algorithm:

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

The identity element I satisfies I + L = L + I = L.

This addition law clearly generalizes to arbitrary fields whose characteristic is not 2 or 3. Furthermore, it also generalizes to elliptic curves defined over arbitrary rings. However, L + M is not always defined in this case.

In particular, the correct inverse element

 $((2y)^{-1} \text{ or } (x_2 - x_1)^{-1} \text{ as the case may be) must}$ exist in the ring for the addition to be well defined. Working in the ring Z_n it is easy to determine if an inverse exists, and thus easy to determine if L + M is well defined.

How to multiply points by integers.

One must specify just what qM, $(q \ge 0, M \in E_n)$ is (if it is even defined), for the case where n is composite(in fact, the same procedure will be used when n is prime). Our "addition" operation is not necessarily associative in this case. We define qM to be

$$0M = I,$$

qM = (q-1)M + M (for q odd), and $qM = \frac{q}{2}M + \frac{q}{2}M \text{ (for } q \text{ even}\text{).}$

This "repeated doubling" algorithm, analogous to the repeated squaring algorithm for exponentiation in Z_n , allows for efficient computation. Using this definition, scalar multiplication by q can be computed using $O(\log q)$ additions.

Computing the order of elliptic groups.

Our algorithm heavily relies on the ability to determine the order of an elliptic group modulo some prime in polynomial time. An algorithm due to Schoof[Sc] computes the order of an elliptic group mod p in $O(\log^9 p)$ time.

3.2 Notation and Conventions

Define $E_n(A, B)$ (where $gcd(4A^3 + 27B^2, n) =$ 1) as the set of points (x, y) over Z_n satisfying $y^2 = x^3 + Ax + B$, along with a special element *I*. For *p* prime, $E_p(A, B)$ is an elliptic group.

Define $N_p(A, B)$ to be the order of $E_p(A, B)$. For notational convenience, we abbreviate these as N_p and E_p when the choice of A and B is clear or unimportant. We write the order of an element M in E_p as $O_{E_p}(M)$.

3.3 The Group Structure of Elliptic Curves over GF(p) for p prime.

We use two well known properties of elliptic curves over GF(p):

(1) The order of the group generated by an elliptic curve over GF(p) is equal to p+1-t, where $|t| \leq 2\sqrt{p}$ (the Riemann Hypothesis for Finite

Fields).

(2) An elliptic group E_p will be isomorphic to $Z_{m_1} \times Z_{m_2}$ for some m_1, m_2 , where $m_2|m_1$.

Our algorithm works by picking random curves (i.e. randomly choosing $A, B \in \mathbb{Z}_p$) until one is found whose order belongs to a special subset of the natural numbers. An important result due to Lenstra[L] allows us to relate the probability of picking a curve of the right order with the density of this subset in the interval $[p - \sqrt{p}, p + \sqrt{p}]$.

Theorem(Lenstra [L]): For $S \subseteq N$, define S'_p to be S intersected with the interval $[p-\sqrt{p}, p+\sqrt{p}]$. Then

$$(\forall p, prime)(\forall S)(\exists c > 0)$$

 $prob(N_p(A, B) \in S) \geq \frac{c \cdot (|S'_p| - 2)}{\sqrt{p} \log p}.$

Here, A, B are picked uniformly (subject to $4A^3 + 27B^2 \neq 0 \mod p$) from Z_p .

This theorem is crucial to the analysis of our algorithm. It enables us to relate the running time of our algorithm to the distribution of primes in small intervals, without any mention of elliptic curves.

3.4 The Structure of Elliptic Curves over Z_n for composite n.

While elliptic curves over Z_n do not form groups, there is a natural projection from E_n to E_p , where p > 3 is prime, and p|n. Given a point $x \mod n$, define $(x)_p$ to be x taken mod p. Given a point $M = (x, y) \in E_n(A, B)$, define $(M)_p \in$ $E_p((A)_p, (B)_p)$ as $((x)_p, (y)_p)$. Define $I_p = I$, the identity element in E_p . Note that $4A^3 + 27B^2 \neq$ 0 mod p, since $gcd(n, 4A^3 + 27B^2) = 1$. Thus, $E_p((A)_p, (B)_p)$ is well defined.

This projection is well behaved, as shown by the following lemma:

Definition 1: Given points $L, M \in E_n, L + M$ is well defined if, when computing this sum, all the required inverse elements exist in the ring Z_n . Likewise, qM, for q an integer, is well defined if all the additions required in the computation of

qM by the procedure described in section 3.1 are well defined.

Lemma 1: Given points $L, M \in E_n, p > 3$, p prime, and p|n, if L + M is well defined, then $(L + M)_p = (L)_p + (M)_p$.

This lemma is useful because once we verify that some L + M is defined mod n (by just going through the calculation), we can then make statements about $(L)_p + (M)_p$ (for p|n) without knowing what p actually is.

Proof: Lemma 1 follows trivially from a simple case analysis on the rules for computing the sum of two points on the elliptic curve E_n . These rules may be summarized as:

- (1) I+M=M+I=M
- (2) (x, y) + (x, -y) = I
- (3) $(x, y) + (x, y) = (\frac{P_1}{(2y)}, \frac{P_2}{(2y)})$ (for $y \neq 0$)
- (4) $(x_1, y_1) + (x_2, y_2) = (\frac{Q_1}{(x_1 x_2)}, \frac{Q_2}{(x_1 x_2)})$ (for $x_1 \neq x_2$)

Here, P_1, P_2 are fixed polynomials over x, y, A, B, and Q_1, Q_2 are fixed polynomials over x_1, y_1, x_2, y_2, A, B .

If either L or M is equal to I, then lemma 1 holds trivially. For the rest of the analysis we assume $L, M \neq I$. This implies $(L)_p, (M)_p \neq I$.

If L + M is of the form (x, y) + (x, -y), then $(L)_p + (M)_p$ will also be of this form, and the lemma will hold.

If L = M = (x, y) (where $y \neq 0$), then $(L)_p = (M)_p = ((x)_p, (y)_p)$. Using rule 3, it is clear that $L + M = (\frac{P_1(x,y,A,B)}{(2y)}, \frac{P_2(x,y,A,B)}{(2y)})$. L + M exists only if (2y, n) = 1, which implies $(y)_p \neq 0$. Therefore, by rule 3, $(L)_p + (M)_p = (\frac{P_1((x)_p, (y)_p, (A)_p, (B)_p)}{(2(y)_p)}, \frac{P_2((x)_p, (y)_p, (A)_p, (B)_p)}{(2(y)_p)})$. Since we are computing the same rational function mod n and mod p, using arguments which are respectively congruent mod p, the two values computed will be congruent mod p. Thus, the lemma will clearly hold.

If $L = (x_1, y_1)$, and $M = (x_2, y_2)$ (where $x_1 \neq x_2$) then $(x_1 - x_2)^{-1}$ exists only if $(x_1)_p \neq (x_2)_p$. The same argument then applies as with the L = M case. Q.E.D.

Corollary 1.1: If $M \in E_n$, and qM = I(as calculated by the algorithm described in 3.2), then if p is prime, p > 3, and p|n, then $(qM)_p = I_p$.

The proof follows trivially from lemma 1.

Note that in E_n , n composite, it is not clear that one gets a consistent answer if one computes qM in several different ways. Corollary 1.1 allows us to avoid this difficulty.

4. THE ALGORITHM

4.1 Description of the Algorithm.

Our algorithm may be thought of as a primality prover. Given a prime number, p, it tries to generate a proof that p is prime. This is a logical counterpart to current randomized algorithms, which produce proofs of compositeness. Given some number n, which we wish to test for primality, we can run our algorithm on n in parallel with a compositeness prover.

A naive version of our algorithm may be informally described as follows:

Step 1: Given some number p, which we wish to prove prime, we start generating random elliptic groups mod p. This is done by simply choosing $A, B \in \mathbb{Z}_p$ at random, rejecting choices where $4A^3 + 27B^2 \equiv 0 \mod p$. The random group will be denoted by $E_p(A, B)$. We compute the order of this group using a deterministic algorithm due to Schoof[Sc]. Schoof's algorithm takes $O(\log^9 p)$ steps.

Next, a standard probabilistic primality testing algorithm ([R,SS]) is used to determine if $N_p(A, B)$ is of the form 2q, where q is prime. The probability of making a mistake can be made exponentially small. This operation is relatively inexpensive, and is clearly dominated by the time required to determine the order of the group in the first place. If N_p is not of the form 2q, q prime, we repeat step 1.

Step 2: Once such a group is found, we randomly select points on the curve (excluding the identity) until a point of order q is found. We can pick random points by picking $x \in Z_p$ at random, and taking y to be a random square root of $x^3 + Ax + B$, if one exists(repeating the process if $x^3 + AX + B$ is a quadratic nonresidue). Clearly, (x, y) cannot be the identity. Note that square roots mod p(p prime) may be efficiently computed in $O(\log^3 p)$ expected time by probabilistic algorithms ([AMM],[B]).

Step 3: We exhibit the curve (represented by A and B), q, and the point found of order q. We then recursively prove that q is prime. We stop when the number to be proven prime is sufficiently small.

This gives the general flavor of the algorithm. However, there are some technical details which still need to be addressed. We proceed to describe the complete algorithm.

Let PP(x) be a probabilistic primality test, such as the ones proposed by Miller-Rabin[R] or Solovay-Strassen[SS]. We require that PP errs with exponentially low probability, and always identifies multiples of 2 or 3 as composite.

Prove(p)

 $p_0 = p; i = 0$ $lowerbound = 2^{(\lg p)^{c/\lg \lg \lg p}}$ $certificate = \emptyset$ while $p_i \geq lowerbound$ do repeat Randomly choose $A, B \in \mathbb{Z}_{p_i}$ Compute $N_{p_i}(A, B)$ until $(4A^3 + 27B^2, p) = 1$ and $PP\left(\frac{N_{p_i}(A, B)}{2}\right) =$ "probably prime" $q = \frac{N_{p_i}(A, B)}{2}$ **repeat** Randomly choose $M \in E_{p_i}(A, B)$ until $M \neq I$ and qM = Ii=i+1 $p_i = q$ Append (M, p_i, A, B) to certificate if p_i is composite then run Prove(p) again $/* p_i$ is small enough to be tested deterministically by [APR]*/ return(certificate)

end

We run this procedure in parallel with a deterministic immplementation of Pratt's algorithm for producing a short certificate of primality. If Pratt's algorithm finishes first, then his certificate will be output instead, and Prove(p) is aborted.

We proceed to discuss in greater detail three key issues: (1) Giving a criteria for when p_i is small enough to be deterministically checked for primality, (2) The problem of needing to test $\frac{N_p(A,B)}{2}$ for primality without a provably correct polynomial test, and (3) Showing how the certificate output by Prove(p) can be used to prove that p is indeed prime. This last point is dealt with in section 4.2.

•Using a deterministic algorithm to stop early:

We need to establish when a number is small enough that its primality may be verified deterministically in polynomial time. Using a deterministic test due to Adleman, Pomerance, and Rumley[APR] one can test n for primality (and thereby prove n is prime) in $(\lg n)^{c' \lg \lg \lg n}$ steps. Therefore, one can test a number less than $2^{(\lg n)^{c' \lg \lg \lg n}}$ for primality in time polynomial in $\lg n$.

Thus, the algorithm can keep track of the size of the original input number, and stop as soon as the current number to be proved prime goes below the bound stated above. No real cost is incurred by switching to the deterministic algorithm at this point.

The reason for stopping at this point is to increase the fraction of primes we can show (without any assumptions) are proven prime by this algorithm. Switching to the deterministic test will not speed the algorithm up.

• Creating a proof that is always correct while using a primality test that makes mistakes:

In order to determine if $\frac{N_p(A,B)}{2}$ is prime, we use a randomized algorithm with an exponentially low probability of failure. Naively, one might suppose that this destroys any hope of creating a certificate that is guarenteed to be correct. A failure of the primality tester could conceivably lead to generating an incorrect certificate, or send the procedure into an infinite loop. However, the fact that an actual certificate is being generated, instead of merely a yes/no answer, remedies this problem. As will be shown in section 3.2, the certificate is easily checked for validity (else it could hardly be called a certificate!). If the procedure gives a certificate that turns out to be invalid, the procedure can be run again until it outputs one that is valid.

Since the algorithm is run in parallel with Pratt's algorithm, the procedure is guarenteed to terminate with a certificate in expected time "merely" exponential in $\log p$. The primality testing algorithms used can be made to have such a small probability of error that, even with an exponential time penalty for making an error, the affect on the expected running time will be negligible.

4.2 Proof of correctness.

We can show how one can use the output of Prove(p) to infer that p is indeed prime. This will prove the correctness of our algorithm.

Lemma 2: For all *n* not divisable 2 or 3, if $\exists M, q, A, B$ such that $q > n^{\frac{1}{2}} + 1 + 2n^{\frac{1}{4}}$, *q* is prime, $(n, 4A^3 + 27B^2) = 1, M \neq I, M \in E_n(A, B)$, and qM = I, then *n* is prime.

Proof: (by contradiction) Suppose n was composite. Then $\exists p < \sqrt{n}, p$ prime s.t. p|n.

If qM = I then $qM_p = I_p$ by corollary 1.1. Thus, $O_{E_p}(M_p)|q$. However, $O_{E_p}(M_p) \leq N_p \leq p+1+2\sqrt{p} < n^{\frac{1}{2}}+1+2n^{\frac{1}{4}} < q$. Since q is prime, we have $O_{E_p}(M_p) = 1$. This implies $M_p = I_p$, which implies M = I, a contradiction. Q.E.D.

We can now show how to use the output of the algorithm to prove p is prime.

Theorem 1: For any prime p, given the output of Prove(p), the primality of p can be verified in $(\log p)^4$ steps.

Proof: The first quadruple output from Prove(p) is of the form (M, q, A, B), where $(p, 4A^3 + 27B^2) = 1$, $M \in E_p(A, B)$, p not a multiple of 2 or 3(unless p = 2 or 3), and qM = I. These facts can all be verified in $O(\log^3 p)$ time. By lemma 2 it is clear that $q(=p_1)$ prime $\rightarrow p$ prime. Likewise, one can verify that p_i prime $\rightarrow p_{i-1}$ prime. Inductively, it can be verified that p_k prime $\rightarrow p$ prime, where p_k is the second element of the last quadruple. However, p_k can be verified trivially (sublinear in |p|), due to its small size.

Finally, we note that $p_i \leq \frac{p_{i-1}+2\sqrt{p_{i-1}}}{2}$. Thus, k will be $O(\log p)$. Verifying the primality of p thus requires $O(\log p)$ scalar multiplications, for a total of $O(\log^4 p)$ steps.

Finally, note that a certificate used by Pratt's algorithm can also be verified in $O(\log^4 p)$ steps.Q.E.D.

4.3 Analysis of the Expected Running Time Under a Conjecture.

To prove a number p is prime, one must go through $O(\log p)$ iterations of the outer **repeat** loop. In each iteration, the algorithm must (1) Find an elliptic group mod p_i whose order is twice a prime, and (2) Find a point in the group with this prime as its order.

(1) The expected time required to find an elliptic group mod p_i with the right order is equal to the expected number of groups that must be tried before finding one with the right order, multiplied by the expected time needed to test a group for this property. We momentarily defer the question of what expected number of groups must be tried. We denote this quantity by T_{p_i} .

Determining if a group has the right order can be done in $O(\log^9 p_i)$ time: First determine the group's order[Sc], taking $O(\log^9 p_i)$ time, then determine if it is twice a prime (probabilistically), taking $O(\log^4 p_i)$ time.

The expected time required to find a group of the right order is thus $T_{p_i} \cdot O(\log^9 p_i)$.

(2) Picking a random point (x, y) requires picking an expected number of $\frac{2p_i}{N_{p_i}} x$'s before finding an x s.t. $(\exists y) (x, y)$ is on the curve.

Since E_{p_i} is isomorphic to Z_{2q} (for some q), half the points will be of order q. Thus, the expected number of random points that must be chosen before finding one of order q will be 2. Determining if a point is of order q can be done in $O(\log^3 p_i)$ time. Therefore, the expected time to find a point of order q will be $2 \cdot \frac{2p_i}{N_{p_i}} \cdot \log^3 p_i = O(\log^3 p_i)$. Thus, once a group with the desired

Thus, once a group with the desired properties has been found, one can find a point with the desired properties in expected $O(\log^3 p_i)$ time. This is a low order term, overwhelmed by the $T_{p_i} \cdot O(\log^9 p_i)$ running time of (1).

Summing over all the $O(\log p)$ p_i 's, and noting that $p_i < p$, we get an upper bound on the total expected running time which is $O(\log^{10} p)$ (the maximal T_{p_i}).

Applying Lenstra's Result:

We can use Lenstra's result on the distribution of N_{p_i} to give a bound on T_{p_i} which depends solely on the distribution of primes in short intervals. Let S be the set of all numbers which are twice a prime. The set S'_{p_i} will thus be the set of numbers in $[p_i - \sqrt{p_i}, p_i + \sqrt{p_i}]$ which are twice a prime. The cardinality of S'_{p_i} will therefore be the number of primes in $[\frac{p_i - \sqrt{p_i}}{2}, \frac{p_i + \sqrt{p_i}}{2}]$, which is just $\pi(\frac{p_i + \sqrt{p_i}}{2}) - \pi(\frac{p_i - \sqrt{p_i}}{2})$. By Lenstra's theorem, and the definition of T_{p_i} ,

$$T_{p_i} = O\left(\frac{\sqrt{p_i}\log p_i}{\pi(\frac{p_i + \sqrt{p_i}}{2}) - \pi(\frac{p_i - \sqrt{p_i}}{2}) - 2}\right).$$

If one assumes the asymptotic distribution of primes holds in these intervals, one gets a heuristic bound of $O(\log^2 p_i)$ for T_{p_i} . This yields a heuristic bound of $O(\log^{12} p)$ for the expected running time of our algorithm. In fact, our algorithm almost certainly runs in $O(\log^{11} p)$ time, but this requires a further assumption on the distribution of N_{p_i} .

The interval $\left[\frac{p-\sqrt{p}}{2}, \frac{p+\sqrt{p}}{2}\right]$ can be essentially rewritten as $\left[p', p' + \sqrt{2p'}\right]$, where $p' = \frac{p-\sqrt{p}}{2}$ (we may be off by a tiny amount at one of the endpoints, but this doesn't matter). This transformation allows a slightly cleaner formulation of our next result:

Theorem 2: There exists a probabilistic primality test which is always correct and terminates in expected polynomial time on all inputs if

$$(\exists c)(\exists k)\frac{\pi(n+\sqrt{2n})-\pi(n)}{\sqrt{2n}} \geq \frac{1}{c\log^k n}.$$

Proof: By the same argument as above, our algorithm will prove p prime in expected time $O(\log^{11+k} p)$. Our algorithm may then be run in parallel with a standard compositeness prover (e.g. any of the usual randomized primality testers). Q.E.D.

4.4 A Generalization of the Algorithm.

As stated, the algorithm searches for elliptic groups whose order of the form 2q, where q is prime. This condition is overly restrictive, and can be relaxed to that of having an order that is of the form aq, where $q > p^{\frac{1}{2}} + 2p^{\frac{1}{4}}$, and q is prime.

Heuristically, this should speed up the algorithm for two reasons:

(1) There are more groups which fit this criteria. Thus, fewer groups must be tried before a suitable one can be found.

(2) Fewer iterations of the outermost **repeat** loop will be required, since q may go down by more than a factor of 2 at each iteration.

The first reason is of some theoretical interest. As will be seen in section 5, mathematicians have come close to proving the conjecture needed to show that our algorithm runs in expected polynomial time. This slight generalization of the algorithm requires a correspondingly weaker conjecture.

The second effect is of less interest, since even if only one iteration was needed the algorithm would still take $O(\log^{11} p)$ time.

5. EXPECTED POLYNOMIAL RUN-NING TIME ON NEARLY ALL PRIMES

In this section we prove that our algorithm terminates in expected polynomial in k time, on "almost all" of the input primes of length k.

In section 3.5, an upper bound for the running time of the algorithm on input p was shown to be $O(\log^{10} p)$ (the maximal T_{p_i})) where T_{p_i} is the expected number of elliptic groups to be picked mod p_i till one of order twice a prime is found and p_i is the prime used in the *i*th iteration of the algorithm.

Lenstra's result (see section 2.2.3) implies that $T_{p_i} = O(\frac{\sqrt{p_i \log p_i}}{|S_{p_i}|-2})$ where S_{p_i} is the set of numbers which are of the form twice a prime. (Namely, $|S_{p_i}|$ is the number of primes in the interval $[\frac{p_i - \sqrt{p_i}}{2}, \frac{p_i + \sqrt{p_i}}{2}]$). It is not known whether for every large enough x, the cardinality of S_x is large (or even that S_x is not empty). However, it has been shown that for almost all x the cardinality of S_x is greater than $O(\frac{\sqrt{x}}{\log x})$. This will be sufficient for our analysis.

5.1 The Density of Primes in Almost All Small Intervals

We make use of a theorem due to Heath-Brown [HB] concerning the density of primes in small intervals.

For integers a, b let $\#_p[a, b]$ denote the number of primes x satisfying $a \leq x \leq b$. Let $\iota(a, b) \leftarrow 1$ if $\#_p[a, b] \leq \frac{b-a}{2\lfloor \log a \rfloor}$ and 0 otherwise.

Theorem (Heath-Brown)¹: \exists alpha, such that for sufficiently large x,

$$\sum_{x \leq a \leq 2x} \iota(a, a + \sqrt{a}) \leq x^{\frac{5}{6}} \log^{\alpha} x.$$

5.2 Estimating the Probability of Polynomial Running Time on a Random Input Prime

Let $PR_k = \{p | \lfloor \log p \rfloor = k, p \text{ prime} \}$ denote the set of primes of length k.

The following running time analysis considers a probability space defined by the random choices in PR_k of the initial prime input to the algorithm, and the random coin tosses made by the algorithm itself.

The following random variables are used in the proof of this section's main theorem.

 $P_0(k)$ is a random variable denoting the input to the algorithm – a random prime in PR_k .

¹ This theorem, communicated to us by Maier and Pomerance [MP], is actually not explicitly stated in [IIB] but is implied by one of its technical lemmas.

 $N_i(k)$ for $0 \le i \le k$ is a random variable denoting the order of the "good" elliptic group found in the *i*th iteration of (the main while loop) of the algoithm on input $P_0(n)$. (A "good" elliptic group is one whose order is twice a prime). If a "good" elliptic curve is not found in the *i*th iteration, then set $N_j(k) = 0$ for all $i \le j$.

Recall, that in the *i*th iteration of the algorithm, a "good" elliptic group is not guranteed to be ever actually found. The probabilistic primality test PP which taged the elliptic group as "good" may have been wrong, or the *i*th iteration may simply never find a "good" group. If either of these cases takes place, we assume for simplicity that the algorithm timesout and fails, and have set $N_j(k) = 0$ for all $i \leq j$. (In practice, the algorithm will not timeout and fail, but will be rerun again on input $P_0(k)$.)

 $P_{i+1}(k) \leftarrow \frac{N_i(k)}{2}$ for $1 \le i \le k$ denotes the value of the prime p_i used in the *i*th iteration of (the main while loop of) the algorithm.

A few comments are in order.

Comment 1: By the properties of the orders of elliptic curves, $N_i(k) \in [P_i(k) \pm \sqrt{P_i(k)}]$, and thus $P_i(k) \in [\frac{P_{i-1}(k)}{2} \pm \frac{\sqrt{P_{i-1}(k)}}{2}]$.

Comment 2: Since the main while loop of the algorithm terminates as soon as $P_i(k) \leq 2^{(\log P_0(k))^{\log \log k}}$, an upper bound on the number of iterations *i* made by the while loop is $B = k - k^{\log \log k}$.

Comment 3: For simplicity of the analysis, we shall assume that our algorithm fails and the $N_j(k)$'s are set to 0 for all $j \ge i$, as soon as the number of elliptic groups picked at iteration *i* and tested for "good" order exceeds $\log^3 P_i(k)$. (Clearly, bounding the probability of failure for this simplified version of the algorithm, will bound the probability of failure in the actual algorithm). We are ready to state theorem 3.

Theorem 3: For all k, the probability that the Primality Proving Algorithm terminates in expected time $O(k^{12})$ on random input prime of length k is greather than

$$1 - O(2^{-k \log \log k}).$$

Proof: Note that by the comments 1-3 made above, the probability that the Primality Proving Algorithm terminates in expected time $O(k^{12})$ on random input prime of length k equals the

$$\Pr(\forall 1 \leq i \leq B, P_i(k) \text{ is prime }).$$

Thus, it will suffice to show that

 $\Pr(\exists i \text{ such that } P_{i+1}(k) = 0, P_i(k) \text{ is prime }) \leq$

$$O(2^{-k^{\log\log k}}).$$

The following facts 1 through 4 will be useful in the calcualion.

fact 1: Let c = 7. Then, $\forall i \leq k - 6$,

$$P_i(k) \in [\frac{P_0(k)}{2^i} \pm c\sqrt{\frac{P_0(k)}{2^i}}].$$

Proof (by induction on *i*) is ommitted. fact 2: Let $\epsilon > 0$. For sufficiently large k, for

$$\sum_{2^k \leq a \leq 2^{k+1}} \iota(a \pm \frac{\sqrt{a}}{2}) < 2^{(\frac{5}{6}+\epsilon)k}.$$

proof: Follows directly from Heath-Brown's theorem stated above. Q.E.D.(fact 2)

Define $\Upsilon(x) = 1$ if $\exists a \in \{x \pm j\sqrt{x}, -7 \le j \le +7\}$ such that $\iota(a \pm \frac{\sqrt{a}}{2}) = 1$ and 0 otherwise, where $\iota(a \pm b) = \iota(a - b, a + b)$.

Informally, the meaning of Υ is that if $\Upsilon(t) = 0$, then for all numbers x in $[t \pm 7\sqrt{t}]$ one can quickly find a prime in the interval $[x \pm \sqrt{x}]$. fact 3: Let $\epsilon \ge 0$. For sufficiently large k,

$$\sum_{1 \leq t \leq 2^{k+1}} \Upsilon(t) \leq O(2^{(\epsilon+\frac{5}{2})k}).$$

proof: By the definitions of Υ and ι it follows that

2*

$$\sum_{\substack{2^k \le t \le 2^{k+1}}} \Upsilon(t) \le$$
$$\sum_{\substack{2^k \le t \le 2^{k+1}}} \sum_{a \in \{t+j\sqrt{t}, -7 \le j \le 7\}} \iota(a \pm \frac{1}{2}\sqrt{a}).$$

But, a positive $\iota(a)$ will contribute 1 to at most 30 t's. Thus,

$$\sum_{2^k \leq t \leq 2^{k+1}} \Upsilon(t) \leq$$

 $N_i(k)$ for $0 \leq i \leq k$ is a random variable denoting the order of the "good" elliptic group found in the *i*th iteration of (the main while loop) of the algoithm on input $P_0(n)$. (A "good" elliptic group is one whose order is twice a prime). If a "good" elliptic curve is not found in the *i*th iteration, then set $N_j(k) = 0$ for all $i \leq j$.

Recall, that in the *i*th iteration of the algorithm, a "good" elliptic group is not guranteed to be ever actually found. The probabilistic primality test PP which taged the elliptic group as "good" may have been wrong, or the *i*th iteration may simply never find a "good" group. If either of these cases takes place, we assume for simplicity that the algorithm timesout and fails, and have set $N_j(k) = 0$ for all $i \leq j$. (In practice, the algorithm will not timeout and fail, but will be rerun again on input $P_0(k)$.)

 $P_{i+1}(k) \leftarrow \frac{N_i(k)}{2}$ for $1 \leq i \leq k$ denotes the value of the prime p_i used in the *i*th iteration of (the main while loop of) the algorithm.

A few comments are in order.

Comment 1: By the properties of the orders of elliptic curves, $N_i(k) \in [P_i(k) \pm \sqrt{P_i(k)}]$, and thus $P_i(k) \in [\frac{P_{i-1}(k)}{2} \pm \frac{\sqrt{P_{i-1}(k)}}{2}]$.

Comment 2: Since the main while loop of the algorithm terminates as soon as $P_i(k) \leq 2^{(\log P_0(k))^{\frac{1}{\log \log k}}}$, an upper bound on the number of iterations *i* made by the while loop is $B = k - k^{\frac{1}{\log \log k}}$.

Comment 3: For simplicity of the analysis, we shall assume that our algorithm fails and the $N_j(k)$'s are set to 0 for all $j \ge i$, as soon as the number of elliptic groups picked at iteration *i* and tested for "good" order exceeds $\log^3 P_i(k)$. (Charly, bounding the probability of failure for this simplified version of the algorithm, will bound the probability of failure in the actual algorithm). We are ready to state theorem 3.

Theorem 3: For all k, the probability that the Primality Proving Algorithm terminates in expected time $O(k^{12})$ on random input prime of length k is greather than

$$1 - O(2^{-k \log \log k}).$$

Proof: Note that by the comments 1-3 made above, the probability that the Primality Proving Algorithm terminates in expected time $O(k^{12})$ on random input prime of length k equals the

$$\Pr(\forall 1 \leq i \leq B, P_i(k) \text{ is prime }).$$

Thus, it will suffice to show that

$$\Pr(\exists i \text{ such that } P_{i+1}(k) = 0, P_i(k) \text{ is prime }) \leq$$

$$O(2^{-k^{\log \log k}}).$$

The following facts 1 through 4 will be useful in the calcualion.

fact 1: Let c = 7. Then, $\forall i \leq k - 6$,

2*

$$P_i(k) \in [\frac{P_0(k)}{2^i} \pm c\sqrt{\frac{P_0(k)}{2^i}}].$$

Proof (by induction on *i*) is ommitted. fact 2: Let $\epsilon > 0$. For sufficiently large k, for

$$\sum_{\leq a \leq 2^{k+1}} \iota(a \pm \frac{\sqrt{a}}{2}) < 2^{(\frac{5}{6}+\epsilon)k}.$$

proof: Follows directly from Heath-Brown's theorem stated above. Q.E.D.(fact 2)

Define $\Upsilon(x) = 1$ if $\exists a \in \{x \pm j\sqrt{x}, -7 \le j \le +7\}$ such that $\iota(a \pm \frac{\sqrt{a}}{2}) = 1$ and 0 otherwise, where $\iota(a \pm b) = \iota(a - b, a + b)$.

Informally, the meaning of Υ is that if $\Upsilon(t) = 0$, then for all numbers x in $[t \pm 7\sqrt{t}]$ one can quickly find a prime in the interval $[x \pm \sqrt{x}]$. fact 3: Let $\epsilon \ge 0$. For sufficiently large k,

$$\sum_{2^k \le t \le 2^{k+1}} \Upsilon(t) \le O(2^{(\epsilon+\frac{5}{6})k}).$$

proof: By the definitions of Υ and ι it follows that

$$\sum_{\substack{2^k \le t \le 2^{k+1}}} \Upsilon(t) \le$$
$$\sum_{1 \le t \le 2^{k+1}} \sum_{a \in \{t+j\sqrt{t}, -7 \le j \le 7\}} \iota(a \pm \frac{1}{2}\sqrt{a})$$

But, a positive $\iota(a)$ will contribute 1 to at most 30 t's. Thus,

$$\sum_{k\leq t\leq 2^{k+1}}\Upsilon(t)\leq$$

 $\mathbf{2}^{l}$

2*

$$\sum_{\substack{2^{k}-7\sqrt{2^{k}} \le a \le 2^{k+1}+7\sqrt{2^{k+1}}\\ \sum_{\substack{2^{k-1} \le a \le 2^{k+2}\\ 0 \le a \le 2^{k+2}}} 30\iota(a \pm \frac{1}{2}\sqrt{a}) =$$

(By fact 2)

$$O(2^{(\frac{b}{6}+\epsilon)k})$$

Q.E.D. (fact 3). fact 4: For $1 \le i \le B$ such that i < k - 6, $\Pr(P_i(k) = 0 | \Upsilon(\lfloor \frac{P_0(k)}{2^i} \rfloor) = 0, P_{i-1}(k)$ is prime $) \le O(e^{-k^2}).$

proof: For every value $P_{i-1}(k) \in [\frac{P_0(k)}{2^{i-1}} \pm 7\sqrt{\frac{P_0(k)}{2^{i-1}}}]$ the cardinality of the set of integers of the form twice a prime in the interval $[P_{i-1}(k) \pm 2\sqrt{P_{i-1}(k)}]$ (previously referred to as $S_{P_{i-1}(k)}$) is the the numebr of primes in the interval $[\frac{P_{i-1}(k)}{2} \pm \sqrt{P_{i-1}(k)}]$.

Now, if $\Upsilon(\frac{P_0(k)}{2^i}) = 0$, then $(\exists a)(\exists -7 \leq j \leq +7)$ such that the interval $[a \pm \frac{1}{2}\sqrt{a}]$ is entirely contained in $[\frac{P_{i-1}(k)}{2} \pm \sqrt{P_{i-1}(k)}]$ and $\#_p[a \pm \frac{1}{2}\sqrt{a}] > \frac{\sqrt{a}}{\lfloor \log a \rfloor}$ Thus, the number of primes contained in $[\frac{P_{i-1}(k)}{2} \pm \sqrt{P_{i-1}(k)}]$ is greater than $\frac{\sqrt{a}}{\lfloor \log a \rfloor}$. (Namely, $|S_{P_{i-1}(k)}| = \frac{O(\sqrt{P_{i-1}(k)})}{\log P_{i-1}(k)}$). By Lenstra's theorem the expected number of

By Lenstra's theorem the expected number of elliptic groups picked in the i - 1th iteration of the algorithm till one of "good" order is found is $O(\frac{\sqrt{P_{i-1}(k)\log P_{i-1}(k)}}{|S_{P_{i-1}(k)}|})$. Thus, the probability that in $\log^3 P_{i-1}(k)$ trials a "good" elliptic group is not found is less than $(1 - \frac{1}{\log^2 P_{i-1}(k)})^{\log^3 P_{i-1}(k)} \leq O(e^{-k})$. Q.E.D. (fact 4).

The final calculation now follows. Pr(algorithm does not terminate in expected $O(k^{12})$)

time on random input prime of length k) \leq $\Pr(\exists 0 \leq i \leq B, P_{i+1}(k) = 0, P_i(k) \text{ is prime}) \leq$ $\sum_{i=1}^{B} \Pr(P_{i+1}(k) = 0, P_i(k) \text{ is prime}) \leq$

$$\sum_{i=1}^{B} \Pr(P_{i+1}(k) = 0, P_i(k) \text{ is prime } |\Upsilon(\lfloor \frac{P_0(k)}{2^{i+1}} \rfloor) = 1) \cdot \Pr(\Upsilon(\lfloor \frac{P_0(k)}{2^{i+1}} \rfloor) = 1) + \sum_{i=1}^{B} \Pr(P_{i+1}(k) = 0, P_i(k) \text{ is prime } |\Upsilon(\lfloor \frac{P_0(k)}{2^{i+1}} \rfloor) = 0) \cdot \Pr(\Upsilon(\lfloor \frac{P_0(k)}{2^{i+1}} \rfloor) = 0) \leq 1$$

(the second summation can be bounded by by fact 4, which is negligible with respect the final result, and will be ignored from here on).

$$\sum_{i=1}^{B} \Pr(\Upsilon(\lfloor \frac{P_0(k)}{2^{i+1}} \rfloor) = 1) \leq$$

$$\sum_{i=1}^{B} \sum_{q \in PR_{k}} \Pr(\Upsilon(\lfloor \frac{P_{0}(k)}{2^{i+1}} \rfloor) = 1 \mid P_{0}(k) = q)$$
$$\cdot \Pr(P_{0}(k) = q) =$$
$$\sum_{i=1}^{B} \sum_{q \in PR_{k}} \frac{\Upsilon(\lfloor \frac{q}{2^{i+1}} \rfloor)}{|PR_{k}|} \leq$$

The sum $\sum_{q \in PR_k} \Upsilon(\lfloor \frac{q}{2^{i+1}} \rfloor)$ can be replaced by $\sum_{2^{k-i} \leq t \leq 2^{k-i+1}} \Upsilon(t)2^{i+1}$. This can be done as at most 2^{i+1} of the $q \in PR_k$ will be mapped into the same $t = \lfloor \frac{q}{2^{i+1}} \rfloor$.

Finally, by fact 3, for $\epsilon > 0$

$$\frac{1}{|PR_k|} \sum_{i=1}^{B} \sum_{2^{k-i} \le t \le 2^{k-i+1}} \Upsilon(t) 2^{i+1} \le O(\frac{k}{2^k}) \sum_{i=1}^{B} 2^{i+1} 2^{(\frac{5}{6}+\epsilon)(k-i)} \le O(\frac{k}{2^k}) 2B\left(2^{(\frac{5}{6}+\epsilon)k+\frac{1}{6}B}\right) \le O(2^{-k\frac{\log k}{\log \log k}})$$

6. ACKNOWLEDGEMENTS

We are especially grateful to Oded Goldreich, Johan Hastad and Silvio Micali for their numerous useful suggestions about this research and its writeup.

Carl Pomerance and Helmut Maier pointed us to the needed result on the density of primes in short intervals.

Burt Kaliski, Hendrik Lenstra, Andrew Odlyzko, Albert Meyer, Victor Miller, Ron Rivest, and Richard Zippel made helpful remarks and gave pointers to the literature.

Last but not least, Joe would also like to thank his parents, Leonard and Mary for their support and encouragment. Thanks, Len and Mary!

7. REFERENCES

[APR] Adleman, Pomerance, Rumely, "On Distinguishing prime numbers from composite numbers", to appear. Ext. Abstract 21st FOCS (1980), 387-406.

[BLS] Brillhart, Lehmer, Selfridge, "New Primality7 Criteria and Factorization of 2sup m+1", vol 29, no. 1930 (1975).

[Ba2] Bach Eric, "Lenstra's Algorithm for Factoring with Elliptic Curves (Expose)", notes, February 27th, 1985.

[CL] Choen, Lenstra, "Primality Testing and Jacobi Sums", to appear.

[D] Dickson, "History of the Theory of Numbers", Chelsea Publishing Company, 1952.

[F] Furer, "Deterministic and Las Vegas Primality Testing Algorithms", Proc. of ICALP 1985.

[IIB] Heath- Brown D. R., "The Differences between Consecutive Primes", J. London Math. Soc.
(2), 18 (1978), 7-13.

[L] Lenstra, "Factoring Integers using Elliptic Curves over Finite Fields", to appear.

[M] Miller, "Riemann Hypothesis and test for primality", JCSS 13 (1976), 300-317.

[MP] Maier H., Pomerance C., Personnal Communication.

[P] Plaisted, "Generating Large Prime Numbers".

[P] Pratt, "Every Prime has a Succinct Certificate", SIAM J. of Comp. (1975), 214-220.

[R] Rabin, "Probabilistic Algorithms for Testing Primality", J. of Num. Th. 12, 128-138 (1980).

[Sch] Schoof, "Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p", Math. Computation, Vol. 44, Num 170, April 1985, pp.

[Sh] Shallit, "Lenstra's Elliptic Curve Factoring Algorithm", notes, March 15, 1985.

[Se] Selberg, "On the Normal Density of Primes in Small Intervals, and the Difference between Consecutive Primes", Archiv for Mathematik of Naturvidensakb B. XLVII. Nr. 6. 483-494.

[Sha] Shanks, "On Maximal Gaps between Successive Primes", Math. Computation, Vol. 18, pp. 646-651, 1964.

[SS] Solovay and Strassen, "A fast Monte-Carlo test for Primality", SIAM. J. of Comp. 6 (1977), 84-85.

[T] Tate, "The Arithmetic of Elliptic Curves", Inventiones Math. 23, (1974), 179-206.