Fast Generation of Provable Primes Using Search in Arithmetic Progressions

Preda Mihailescu

Union Bank of Switzerland, CH 8021 Zürich

Abstract. Many cryptographic algorithms use number theory. They share the problem of generating large primes with a given (fixed) number n of bits. In a series of articles, Brandt, Damgard, Landrock and Pomerance address the problem of optimal use of probabilistic primality proofs for generation of cryptographic primes. Maurer proposed using the Pocklington lemma for generating provable primes. His approach loses efficiency due to involved mechanisms for generating close to uniform distribution of primes. We propose an algorithm which generates provable primes and can be shown to be the most efficient prime generation algorithm up to date. This is possible at the cost of a slight reduction of the set of primes which may be produced by the algorithm. However, the entropy of the primes produced by this algorithm is assymptotically equal to the entropy of primes with random uniform distribution. Primes are sought in arithmetic progressions and proved by recursion. Search in arithmetic progressions allows the use of Eratosthenes sieves, which leads finaly to saving 1/3 of the psuedo prime tests compared to random search.

1 Introduction

Primality testing has a long history and has undergone a radical development during the last decade. We shall not go into details of this development but begin with mentioning that prime tests split into compositeness proofs (which are polynomial) and primality proofs, which are more complex. The fastest known primality proof [8] which has been implemented by Atkins and Morain [16], is polynomial, but in practice slower than superpolynomial algorithms such as the cyclotomy test [4].

Algorithms proposed for the specific purpose of generating primes for cryptographic use are different due to the technological motivation behind them. Efficiency and simplicity are the main concerns and - as a consequence of the domain of application - "security" of primes is an issue of concern too. Security of a prime p basically means that the discrete logarithm base p and factorization of p out of a product of usually two primes should be, if possible, harder than average. The last condition raised the wish to have large prime factors in $p \pm 1$; this was formalized in conditions for so called "Gordon secure primes" [10]. It was obvious that with the information about factors of $p \pm 1$ available for a Gordon prime, a proof by Pocklington's lemma was almost for free. The first to make use of this observation was Maurer in [14]. By the date of Maurer's paper,

Y.G. Desmedt (Ed.): Advances in Cryptology - CRYPTO '94, LNCS 839, pp. 282-293, 1994.

Gordon primes were de facto outdated by the ECM ([12]) factoring method: Gordon primes were designed to reduce the odds of $p \pm 1$ factoring methods [19], but cannot provide any specific protection against elliptic curve factoring. This facts are reflected in [21]. For technological applications, Maurer's algorithm lacks both the ease of implementation and the efficiency requirements - being, by the authors statement ([15]), p19 60% slower than Rabin - Miller tests which are also simpler to implement. Maurer does state in [15],p29, that his algorithm is more efficient than probable primes, but this statement is not supported by facts and is in contradiction with the rest of the paper. The paper [14] is important for having first indicated that for the specific purpose of cryptographic use, provable primes were worth considering. We propose an algorithm that seeks primes in arithmetic sequences for which a factorization of the ratio is known - this implies a recursive use of the algorithm. For the trial division phase we use Eratosthenes sieve methods, which lead to an increase of the optimal length of trial division; optimal trial division was also evaluated in [14], [2]. In the last paper, the authors also consider means of speeding up Maurer's method and notice that the 'almost' uniform distribution required by Maurer has a negative impact on the performance. An elegant analysis of the information entropy of primes produced by "incremental search" - and proved with Rabin - Miller tests - is provided in [1]. Incremental search is a special case of arithmetic progressions - when the ratio is 2. We shall adapt some of the methods of that article for the analysis of our alogithm. Probabilistic methods like Rabin - Miller may falsely declare composites for primes and the probability for this event decreases with the iteration of the same test for different bases. In [3] the authors show that this probability also depends on the size of the primes produced; tables for the error probability $q_{t,k}$ that a composite with k - binary digits is declared prime after t independent Rabin - Miller tests are quoted also in [1]; the tables show that the probability of failure of the Rabin - Miller test after t rounds is substantially lower than the 4^{-t} initially proved by Rabin [20]. These results suggest to the authors the conclusion that "using the Rabin test remains in many cases the most practical approach" [1],p.1. We show that recursive primality proofs combined with search in arithmetic progressions is preferable to Rabin - Miller tests, also from the point of view of efficiency, in the range of sizes of primes covered by the tables in [1] and probably for primes up to at least 1000 bits. For larger primes, the difference becomes negligeable.

2 Outline of the Algorithm

Let $B > 0$ and $n > 0$ be positive integers and $s, c > 1$ real constants. We propose following algorithm for generating provable primes of n digits, together with their certifiactes:

Algorithm AP:
Input: n.
Constants: B, c, s.

Step 1. If $n < B$ return a random prime with n bits (generated by trial division).
Step 2. Produce an integer F with $2^{\epsilon n} < F < 2^{c\epsilon n}$ which is completely factored, by recursive use of this algorithm. The constant ϵ may be $1/2$, or $1/3$, depending on the prime ceritfication used (see bellow).
Step 3. Draw a random number $t \in (2^{n-2}/F, 2^{n-1}/F - sn)$.
Step 4. Find a prime in the arithmetic progression $P = \{N \mid N = N_0 + ia; N_0 = ta + 1; a = 2F; 0 \leq i \leq s\}$.

The primality test for Step 4 splits naturally in three parts which are applied to prime candidates in the progression P until a prime is found - together with a corresponding certificate, or the bound s is passed and the algorithm returns failure.

Part I : **Trial division by primes** $< A$, where A is a given upper bound. calculate $a'_p = a \bmod p, \forall p < A$. calculate $N'_p = N_0 \bmod p, \forall p < A$. initiate an array tab of length s. For all $p < A$, set $tab[i] = 1$ for all solutions of the modular equation $N'_p + ia'_p = 0 \bmod p$. This is obviously an Eratosthenes sieve.
Part II : **Compositeness test** using Rabin - Miller. traverse the table tab starting with $i = 0$ and perform a Rabin - Miller test with base 2, for the i-th elements in the progression P, such that $tab[i] = 0$. Continue if the test declares "composite", go to Part III otherwise.
Part III : **Primality proof** using the Pocklington lemma.

We recall the Pocklington lemma:

Proposition 1. *: Let N be an integer, such that $N - 1 = F \cdot R$, where F is completely factored. Suppose that,*

$$\forall q \mid F, \quad \exists \alpha_q \text{ with } \quad (\alpha_q^{(N-1)/q}, N) = 1 \text{ and } \quad \alpha_q^{N-1} = 1 \text{ mod } N \tag{1}$$

Then all primes $p \mid N$ are of the shape $p = kF + 1$. In particular, if $F > \sqrt{N}$, then N is prime.

Proof. Let $\alpha = \prod_{q|F} \alpha_q$. Then

$$(\alpha^{(N-1)/q}, N) = 1 \quad \text{and} \quad \alpha^{N-1} = 1 \bmod N \quad \forall q \mid F \tag{2}$$

If $r \| N$ is a prime, the above equations hold a fortiori mod r. The multilpicative group mod r has thus a subgroup of index F generated by α thus proving the first assertion: $F \mid (r - 1)$ and, a fortiori, $r \geq F + 1$. If $F > \sqrt{N}$, it follows that $r > \sqrt{N}, \forall r \mid N, r$ prime. This is a contradiction, since N must have at least one prime factor $r \leq \sqrt{N}$. Consequently, N must be prime, which ends the proof.

In part III of the algorithm, for each $q||F$, a base α_q verifying (3.4) is sought among the first primes, starting with 2. If for some base $\alpha_q^{N-1} = 1 \bmod N$ does not hold, N is composite (little Fermat does not hold). If for some q, no base verifying (1) is found before a fixed upper bound R of trials, no primality or compositeness proof can be provided and N is incremented (Part II is proceeded). Finally, if bases α_q are found $\forall q \mid F$, N is prime. The prime certificate consists of the pairs $\{(\alpha_q, q) \mid q \mid F\}$ together with certificates for each prime q, recursively [18]. Providing all these pairs together with the certificates for q being prime gives a proof of N's primality by the above lemma. In [5] a certification method using an $O(n^{1/3})$ factored part of $n-1$ is provided. It requires proving that a certain discriminant is not a perfect square. Suppose $n = Ft + 1$ and $F > \sqrt[3]{n}$, and bases α_q verifying (3.4) have been found for all $q \mid F$. If n is composite, it is built up of two primes $p_i = k_i \cdot F + 1$. Let $n = aF^2 + bF + 1$; by comparing $n = p_1 p_2$ we get $k_1 k_2 = a$ and $k_1 + k_2 = b$. It is then easy to see that n is composite iff $\Delta = b^2 - 4a$ is a perfect square. This again can be checked by verifying if Δ is a perfect square modulo a set of primes $\{p \mid \prod p > F\}$; if this is the case, $(k_i \bmod F) = k_i < F$. Otherwise n is prime. One can take in general $\{p \mid p < logF\}$ as set of primes.

3 Optimal Trial Division and Run Time Analysis

We assume a machine with wordlength B, the time for a short integer multiplication is t and let $m = n/B$. The time involved in the trial division is: $T_t = t((1+\epsilon)m + \log A)\frac{A}{\log A}$ and is build up of $(1+\epsilon)m$ short divisions for calculating a' and N' and $\log A$ divisions for calculating a modular inverse, for each prime; we shall write $T_t = tm(1+\epsilon+\delta)\frac{A}{\log A}$, with $\delta = \frac{\log A}{m} \to 0$ for $m \to \infty$. Note that the trial divisions are performed only once and do not have to be repeated for all n candidates. Assuming that the prime candidates for AP are equally distributed among the residue classes mod p for all small test primes p, the fraction of prime candidates after this step is, by Mertens' theorem $f = \frac{e^{-\gamma}}{\log A}$. The second part then takes in average:

$$T_R = t \cdot f \cdot d \cdot w^2 \cdot m^4 = t \cdot \frac{f \cdot d \cdot n^4}{w^2}, \tag{3}$$

where the length of the expected seek interval is $d \cdot n$, and naive multiplication is used for exponentiation. We shall see that $d = 1$, so we subsequently neglect this factor. Optimal table length A may be found by writing $g_P(A) = T_t(A) + T_R(A)$ and taking the derivative. The optimal value found is $A_P = \frac{kn^2m}{\log(n^2m)}$, with a constant $k = \frac{e^{-\gamma}}{1+\epsilon+\delta}$ close to 0.4. As a term of comparison, optimal table length in [14] is $O(nm)$. Actually, Maurer's argument goes like this: adding a prime A to the table saves a pseudoprime test with probability $1/A$, so optimal length is reached when the expected saving T_e/A balances the expected extra division which he sets to $1 \cdot T_d$; the argument is incorrect, since both values should be averaged over the whole seek interval. It does however yield a good

estimate of the optimal value $A_R = \frac{2e^{-\gamma}}{\alpha(A)} nm$, with $\alpha(A)$ defined in [14], as found by taking the derivative of the adequate function g_R (indices P stay for progression search, R for random search). In [2] the optimal table length is even smaller: $O\left(\frac{nm}{\log(nm)}\right)$. An alternative to the sieve method consists in computing the current remainders $N_c = N_i + ia \bmod p$. This yields the same assymptotics as A_P, with δ replaced by a constant $\alpha = 2wt_a$ where t_a is the ratio between the time for a short addition and a short multiplication. Beside the smaller constant, this approach has the disadvantage that the remainders mod p must be stored (three values for each test prime p). Comparing A_P to A_R shows that the use of the Eratosthenes sieve sensibly increases optimal table length, leading to the elimination of more candidates in the trial division phase. One can evaluate this performance improvement: since the number of pseudoprime tests is in average fn both for random search and for search in progressions, in the last case the exponentiations drop by a factor of $f_p = \frac{\log A_R}{\log A_P} = O\left(\frac{\log(nm)}{\log\left(n^2 m/\log(n^2 m)\right)}\right) \to \frac{2}{3}$ for $n \to \infty$. This improvement appears to be more substantial than what can be saved in a probabilistic proof approach by better error estimates, like in [3]. For $n = 1024$ the number of primes for the trial division phase is about 2^{15} (with $A \approx 2^{19}$) and this may be argued to be too large for some machines. Using the incremental difference storing for the primes, this requires 32 KB of memory which should be afordable on most current PC's. The saving factor is 0.72 and it results in about 10 pseudo prime tests less! It is theoretically intersting to observe that $A_P = O\left(\frac{n^4}{(\log n)^2}\right)$ can be achieved by using a divide and conquer method. The idea is following: group the test primes in $A/(\log A\sqrt{m})$ products of length $\sqrt{m}$ and take first the remainders modulo these products and then the remainders of these remainders modulo the single primes; this yields: $T_i = 2t\sqrt{m}(1+\epsilon+\delta)\frac{A}{\log A}$. Iterating this method leads to the stated result, when the factor $2\sqrt{m}$ is replaced by $(\log m) \cdot m^{1/\log m} = O(\log m)$. This means taking products of length $m/2, m/4, ..., 2$. Of course, the building of the factors belongs to a precomputation phase and the storage space explodes to $O(n^2 m^2)$ - the method is not meant for implementation! Note that the base 2 is a suitable base for the Pocklington lemma, being a q-th power nonresidue with probability $(1-1/q)$. This is remarcable, since exponentiation base 2 is particularly efficient ([2]).We shall consider here the version of the algorithm where F is built up of a single prime. With this, the time for part III is $1-\epsilon$ exponentiations, since for proving that 2 is an adequate F-th power nonresidue, one only raises to the power $(n-1)/F$, the exponent thus having length $n(1-\epsilon)$. With probability $1/F$ a further base should be tested: we do not take this event into account, since its probability is exponentially low. If $\epsilon = 1/3$, checking that Δ is not a perfect square takes $\sum \log p = O(\log F) = O(n)$ short opertations, which are negligeable. We now want to motivate our claim that this algorithm is more efficient than pseudo - prime algorithms, for the range of values of n of current interest. Of course, the sieve method can also be used with pseudo - primes in incremental search, although this method was not proposed before. Some implementation might also use suboptimal tablelength, due to limited memory.

We thus do not consider the advantages of trial division as specific for AP. The advantage of our algorithm compared to pseudo - prime algorithms will thus relay in the shorter proof, the disadvantage, in recursion. It is these two factors we shall have to compare. If R is the number of Rabin - Miller tests performed until a pseudoprime is found and T_e is the time for an n-bit exponentiation, the time for the proof is, as seen above, $(1-\epsilon)T_e$ and the time involved in recursion is upper bounded by $T_r = \left(\frac{\epsilon^4}{1-\epsilon^4}\right) \cdot RT_e$. The total overhead for proving a prime is thus: $T_r = \left(\frac{\epsilon^4}{1-\epsilon^4} \cdot R + (1-\epsilon)\right) \cdot T_e$. With $n = 384$ - which is currently standard for RSA - for a weak implementation, with $\epsilon = 1/2$ and $A = 1000$ - thus $R \approx 20$, the overhead is $T_p \approx \frac{11}{6} T_e$. If $\epsilon = 1/3$, $T_p < T_e$, even without increasing the tables. With optimal tables, $T_p < T_e$, up to more than 1000 bits. A standard implementation of our algorithm needs thus less than the time for one additional exponentiation as overhead for a primality proof, for $n \leq 1000$ bits. Compared to that, a Rabin - Miller test with probability of error $< 2^{-48}$ requires three additional exponentiations for 384 bits and two for 600 bits; no values for $n = 1000$ are provided in [1]. Extrapolating the values from [1], two Rabin - Miller tests for $n = 1000$ bits will provide an error probability p_e with $2^{-60} < p_e < 2^{-48}$. For primes with more than 1000 bits our algorithm may become more expensive than a pseudo - prime algorithm, while keeping the advantage of providing a primality proof. It must be noted that for large primes, the number R of Rabin tests which eliminate composite candidates increases (linearly with n) and the overhead for the primality proof becomes negligeable: a good approximation of R is $R(n, A) = \frac{n\, e^{-\gamma} \log 2}{\log A} \approx 0.392 \frac{n}{\log A}$, the logarithm being the natural one. From this and the estimate for the overhead T_p above, it follows,

$$f(n) = \frac{T_p}{R\, T_e} = \frac{\epsilon^4}{1-\epsilon^4} + \frac{0.392(1-\epsilon)\log A}{n} \approx \frac{1}{80} + \frac{2/3}{n/\log(n)}. \qquad (4)$$

and $f(n) < 2\%$ for $n > 1000$. Thus AP is never more than 2% slower than probabilistic primes methods, if at all!

4 Analysis of the Algorithm

This algorithm has two different and independent reasons for producing non uniformly distributed primes. The first is that while searching primes in arithmetical progressions, primes at the end of long gaps are chosen with higher probability. This very phenomenon was investigated by Brandt and Darmgard in [1], for progressions with ratio 2. Their results relay upon the prime r-tuple conjecture of Hardy - Littlewood. This is an assympotoic formula for the number $\pi_{\mathbf{d}}(N)$ of positive integers $n \leq N$ such that $n + d_1, n + d_2, \ldots, n + d_r$ are simultaneously primes. The vector $\mathbf{d}$ is the r-tuple $(d_1, d_2, \ldots, d_r)$. The conjecture says:

$$\pi_{\mathbf{d}}(N) \sim S_{\mathbf{d}} \frac{N}{(\log N)^r} \qquad \text{for } N \to \infty. \qquad (5)$$

with $S_{\mathbf{d}} = \prod_p \frac{p^{r-1}\left(p-\nu_{\mathbf{d}}(p)\right)}{(p-1)^r}$, provided $S_{\mathbf{d}} \neq 0$ and where $\nu_{\mathbf{d}}(p)$ is the number of distinct residue classes modulo p occupied by the numbers in the r-tuple $\mathbf{d}$. Gallagher [9] uses this conjecture to prove that:

$$\sum_{\mathbf{d}\in H'} S_{\mathbf{d}} \sim h^r \quad \text{for } h \to \infty \quad \text{where } H' = \{(d_1, d_2, \ldots, d_r) \mid d \in N, d_i \neq d_j\} \tag{6}$$

In order to use Gallagher's result for primes in arithmetic progressions with ratio F, we must assume the extended Riemann conjecture and $F = o(\sqrt{n})$, for proving the necessary distribution of primes in arithmetic progressions [6]. With this, Gallagher's result can be generalized to following statement. The number $Q_k(h, N)$ of integers $n < N$ for which the interval $(n, n + hF)$ contains exactly k primes in the progression $1 + tF$ verifies:

$$Q_k(h, N) \sim N\frac{e^{-\lambda}\lambda^k}{k!} \quad \text{for } N \to \infty,\ h \sim \lambda\frac{\varphi(F)}{F}\log N \tag{7}$$

and lemma 5 from [1] can be restated:

Proposition 2. *Let $G_h(x)$ denote the number of primes p in the progression $1 + tF$, such that $p < x$ and $p - q \leq hF$, where q is the prime preceding p in the progression. Assuming the r-tuple conjecture, the ERH and with $F = o(\sqrt{n})$, for any constant λ,*

$$G_{\lambda\varphi(F)\log(x)/F}\ (x) = \frac{xF}{\varphi(F)\ \log(x)}\left(1 - e^{-\lambda}\right) \cdot (1 + o(1)). \tag{8}$$

The factor $\varphi(F)/F$ stems from the fact that $\varphi(F)$ out of F residue classes modulo F contain primes. The proof of this lemma uses (6) and Dirichlet's theorem on distribution of primes in arithmetic progressions [6]. Let M_k be the set of primes with k bits and for $k \leq B$, let $M'_k = M_k$. For $k > B$, we put: $M'_k = \{p \in M \mid p = tq + 1, q \in M'_{[k/3]+1}\}$. Then M'_k is a subset of the primes that can be produced using AP. Let H''_k be the uncertainty about primes sought in arithmetic porgressions by AP and H'_k be the uncertainty of randomly chosen primes in M'_k. Using (7) and methods like in [1], one can prove that:

$$\frac{H''_k}{H'_k} \sim 1 \quad \text{for } k \to \infty. \tag{9}$$

It can also be shown with the same means that the expected length of the seek interval for primes in M'_k is k. This result is not trivial. The average distance between two primes in an arithmetic progression is k, but a prime laying at the end of an "average length interval" is found after only $k/2$ trials, since the starting point of the seek-sequence may lay anywhere between this prime and its predecessor. However, the expected value of the length of the seek interval is not $k/2$ but k and this is a consequence of the fact that primes at the end of longer intervals are chosen with higher probability. So, incidentaly, the average number of candidates to be tested for primality is the same in random choice as

in progressions, asuming the r-tuple conjecture! In [2], this result is stated for incremental search, as an empiric evidence. The paper [1] provides the tools for such a proof, but does not address the topic. We state a theorem on arithmetic progressions, which contains the empirical result in [2] as a subcase, when the ratio of the progression is $F = 2$.

Theorem 3. *Let $P_k = \{p = \lambda F + u\} \cap M_k$ be the set of k-bit numbers in an arithmetic progression, with $(u, F) = 1$. Assuming the r-tuple conjecture, the extended Riemann conjecture and if $F = o(\sqrt{2}^k)$, then the expected search length for finding a prime by incremental search in P_k, with random starting point is $E(l) = \log(2^k)\frac{\varphi(F)}{F}$.*

Proof. Let $l(p) = \frac{p-q}{F}$ be the distance of the prime p to its prime predecessor q in P_k. The probability that a prime p is chosen is $P(p) = \frac{l(p)}{N/(2F)}$: in fact, $l(p)$ out of $N/(2F)$ possible startpoints lead to the choice of p, so $E(l) = \sum_{p \in P_k} P(p)\frac{l(p)}{2} = \sum_{p \in P_k} \frac{l^2(p)}{N/F}$. Let $\mathbf{L} = \{0 < \lambda_1 < \lambda_2 < \ldots < \lambda_m < \infty\}$ be a partition and k such that the term $o(1)$ in Proposition 2 is $< 1/m^2$. Then

$$E(l) = \sum_{i=0}^{m} \sum_{l\lambda_i < l(p) < l\lambda_{i+1}} \frac{l^2(p)}{N/F} + \sum_{l\lambda_{i+1} < l(p)} \frac{l^2(p)}{N/F}, \quad \text{with } l = \frac{\varphi(F)\log N}{F}. \tag{10}$$

By Proposition 2, the number of primes with $l\lambda_i < l(p) < l\lambda_{i+1}$ is

$$\frac{NF}{2\varphi(F)\log N}\left(e^{-\lambda_i} - e^{-\lambda_{i+1}}\right)(1+\delta), \tag{11}$$

with $\delta < 2/m^2$. It follows that $E(l) = (\log N)\frac{\varphi(F)}{2F} f(\mathbf{L}) + O(1/m)$, where $f(\mathbf{L})$ is a Riemann sum of the function $h(x) = x^2 e^{-x}$; letting m grow together with k, so that the condition on the error term stays valid, $f(\mathbf{L})$ will converge to the integral $\int_0^\infty f(x) = 2$, which finishes our proof.

This result needs some comments. It shows that introducing small factors in F will decrease the expected seek length; this seems to suggest a means to reduce the number of pseudoprime tests in our algorithm. However, if $p \mid F$ then $(1 + tF, p) = 1$, so p will be ineffective in the trial division stage. Overall, what one gains in the seek interval length, one loses in the trial division. One may thus aswell choose F to be a prime and this explains our statement about the expected seek interval. The second reason for nonuniform distribution is more dramatic: certain primes with n bits are not produced at all (their probability is 0). We finally want to prove that this does not affect the entropy in the sense that the uncertainty about a prime randomly chosen in M_k' is assymptotically equal to the uncertainty of a prime randomly chosen in M_k. For random choice, it is easy to see that the entropy is $H_k = \log(\|M_k\|)$ resp. $H_k' = \log(\|M_k'\|)$. We need to

give an estimate of $||M'_k||$. By definition, $M'_k = \{p = 2tq + 1 | q \in M'_{[k/3]+1}\} \cap M_k$ and it follows that:

$$\frac{2^{\frac{2k}{3}-3}}{k \log 2} \leq \frac{||M'_k||}{||M'_{k_1}||} \leq \frac{2^{\frac{2k}{3}-2}}{k \log 2}, \tag{12}$$

where $k_i = \left\lfloor \frac{k_{i-1}}{3} \right\rfloor + 1, k_0 = k, i > 0$. By induction we have: $\frac{k}{3^i} + \frac{1}{2} \leq k_i \leq \frac{k}{3^i} + \frac{3}{2}$ and we let m be the least index for which $k_m < B$, so that $||M'_m|| = \frac{2^{k_m - 1}}{k_m \log 2}$. This anchors the recursion and we have, after some manipulations of the above inequalities:

$$\frac{2^k}{(8 \log 2\sqrt{3} k B)^m} \leq ||M'_k|| \leq \frac{2^k}{(2 \log 2\sqrt{k} B)^m}, \tag{13}$$

By definition of m we have $m = \left\lfloor \frac{\log(k/B)}{\log 3} \right\rfloor + 1$. Putting $x = 2^k$, (13) implies:

$$H'_k = \log x - (\log \log x)^2 \left(\frac{1}{2 \log 3} + o(1) \right). \tag{14}$$

Since $H_k = \log\left(\frac{x}{2 \log x}\right)$, we have a forteriori $\frac{H'_k}{H_k} \sim 1$ for $k \to \infty$; together with (12), this yields:

$$\frac{H''_k}{H_k} \sim 1 \qquad \text{for } k \to \infty, \tag{15}$$

which is a measure for the distribution of the primes produced by AP.

5 Cryptographic Security and Related Topics

It has already been remarked [21] that facing recent factorization and discrete logarithm mehtods, concepts of secure primes like the ones of Gordon [10] become irrelevant. Actual questions for security of prime generating algorithms remain following. The distribution of the produced primes, which was measured in the preceding chapter by their entropy. This was shown to be assymptotically equal to the entropy of randomly chosen k-bit primes. The fact that the produced primes are collission free is also a consequence of the above. Finally, the iterated encryption attack for the RSA algorithm seems to suggest that p-1 should have some larger factor ([14]) - although this fact was also relativized by [21]. However, cryptography remains a somewhat subjective field. The author of [23] has been cited for saying that, while knowing that 'secure primes' are not harder to brake, he would still prefer Gordon primes for the systems he uses. An argument for this atitude may be the fact that algorithms against which Gordon primes offer no additional security may be regarded as more difficult to implement, the crew of possible crackers being thus reduced... The particularities of AP are particularly favourable for efficient generation of primes with special structure. This can be required in algorithms like the [7] or the new signature scheme of Nyberg and

Rüppel [17], which require factors of $p-1$ of fixed length. It is also the case for 'secure primes'. We end this chapter with an application of AP for finding Gordon strong primes. Let M be a magnitude considered infeasible for algorithms on computers - say, $M = 2^{64}$. We want to produce primes p, such that, there are primes l and l' with: $l \mid (p-1), l' \mid (p+1)$ and $l, l' > M$. Let u_0 and v_0 be the minimal solutions of the equation $vl' - ul = 1$. Following modification in the definition of the progression P, in step 4 of AP gives a solution to the problem: $P = \{N \Big| N = N_0 + \lambda a; N_0 = ta + 2u_0 l + 1; a = 2l \cdot l'; \lambda \geq 0\}$. The choice of t in step 3 must also be adjusted accordingly , so that $t \in (\frac{2^{n-1}-2u_0 l}{a}, \frac{2^n - 2u_0 l}{a} - sn)$ It is easy to verify that a prime in the progression P has the desired properties.

6 Performance

The simple version of AP has been implemented on a SUN/IPX machine. It uses suboptimal table length ($A = 2^{15}$) and naive multiplication. Following table shows the performance of this implementation, with averaged values (over 100 generated primes) for different prime lengths.

Table 1. Performance **AP**

n (bits)	Runtime (sec)	$r(n) = t(2n)/t(n)$	Seek Interval	Psp tests	SeekInt/ log(x)	SeekInt/ #Psp.Tests
256	2.2		173	9.3	0.98	18.6
332	4.0		238	13.0	1.03	18.5
384	7.0		260	14.0	0.98	18.5
512	17.5	8.4	331	17.8	0.93	18.5
664	49.1	11.8	495	26.9	1.07	18.4
768	70.1	11.0	472	25.7	0.89	18.4
1024	213.8	12.2	662	35.7	0.93	18.5

This table reflects the behaviour of predicitions in practice. The most stable predicted value, is the ratio (length of the seek - interval)/ (number of pseudo prime tests): it balances around 18.5, whereas the expected value is $\frac{\log A}{e^{-\gamma}} \approx 18.52$. The length of the seek interval also balances around $\log(x)$, but the variance is larger. The $O(n^4)$ behaviour of the algorithm is less well reflected, certainly because of the overhead which is independent of the length of the primes found. The ratio $r = t(2n)/t(n)$ - where $t(x)$ is the average time for finding a x-bits prime - has used corrected times, proportional to the relative seek interval. The table reflects a monotonous increase of the apparent specific exponent of the run time. Obviously the data is insufficient and this exponent will approach the expected value 4 for larger primes. Actually this behaviour only confirms the known rule that subquadratic multiplication algorithms are not recommendable for small lengths of the multiplicants.

7 Conclusions

We proposed an algorithm for generating provable primes using incremental search in arithmetic progressions. We showed that trial division can be performed using the Eratosthenes sieve method, which increases the number of prime candidates eliminated by the trial division step by an assymptotic factor of 3/2. Independently of the search approach used, we showed that our algorithm is more efficient than probable prime algorithms for primes of at least up to 1000 bits length, whereas for larger primes the loss in efficiency is not more than 2%, while the primes produced are always provided with a certificate. The advantages of our algorithm are more substantial when the primes produced are required to have special prime divisors of $p \pm 1$, since this feature can be incorporated for free in the algorithm. The advantages of this algorithm are at the cost of a reduction of the set of primes which can be produced and their nonuniform distribution. We proved though that the information entropy of the primes produced by the algorithm is assymptotically equal to the entropy of randomly chosen primes. We finally presented the performance of a - non optimal - implementation of the algortihm on a SUN/IPX machine.

Acknowledgement:
I wish to thank P. Landrock and I.Damgard for valuable discussions, informations and literature.

References

1. Brandt, J.; Damgard, I.: *On Generation of Probable Primes by Incremental Search*, Proceedings CRYPTO'92, Lecture Notes in Computer Science Vol. **740** pp. 358-71
2. Brandt, J.; Damgard I.; Landrock, P.: *Speeding up Prime Number Generation*, Proc. of Asiacrypt 91, Lecture Notes in Computer Science, Vol **739**, pp. 440-50.
3. Damgard I.; Landrock, P.; Pomerance, C.: *Average Case Bounds for the Strong Probable Prime Test*, Math. Comp., Vol **61**, Oct 1993, pp. 177-195,
4. Bosma, W.; VanderHulst, L.: *Primality Test Using Cyclotomy*, Phd, University of Amsterdam, 1990
5. Couvreur, C.; Quisquater, J.J.: *An introduction to fast generation of large primes*, Philips Journal of Research, vol. **37**, pp 231-264, 1982
6. Davenport, Harold: *Multiplicative Number Theory*, Springer 1980, Second Edtion.
7. *Digital Signature Algorithm*, Federal Information Processing Standards Publication XX, 1993 February 1.
8. Goldwasser,S. and Kilian, J.: *Almost all primes can be quickly certified*, Proc. of the 18th Annual ACM Symposium on the Theory of Computing, 1986, pp. 316-329.
9. Gallagher, P.X.: *On the distribution of primes in short interevals*, Mathematika, vol. **23**, 1976, pp. 4-9
10. Gordon, J.: *Strong primes are easy to find*, Advances in Cryptology - EUROCRYPT '84, Lecture Notes in Computer Science, vol. **209**, 1984, pp. 216-223.
11. Knuth, Donald E.: *The Art of Computer Programming*, Addison-Wesley, 1981

12. Lenstra, H.W.: *Factoring Integers With Elliptic Curves*, Annals of Mathematics, Vol. **126**, 1987, pp649-673.
13. Lenstra, A.K. and Lenstra, H.W.,Jr.: *Algorithms in number theory*, Technical Report 87-008, May 1987, University of Chicago, Dept. of Computer Science.
14. Maurer, Ueli M.: *Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters*, Internal Report, Dept. of Comp. Science, Princeton University, 1991.
15. Maurer, Ueli M.: *Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters*, Internal Report, Dept. Informatik, ETH Zrich, 1993.
16. Morain, F.: *Distributed primality proving and the primality of (23539+1)/3*, Advances in Cryptology, EUROCRYPT '90, Lecture Notes in Computer Science, vol. **473**, 1990, pp. 110-123.
17. Nyberg,K.; Rüppel,R.:*A New Signature Scheme Based on the DSA Giving Message Recovery*, Preprint, 1993.
18. Plaisted, D. A.: *Fast verification, testing and generation of large primes*, Theoretical Computer Science, vol **9**, 1979, pp. 1-17.
19. Pollard, J.M.:*Theorems on factorization and primality testing*, Proceedings of the Cambridge Philosophical society, Vol. **76**, 1974, pp. 521-528.
20. Rabin, M.O.: *Probabilistic algorithm for testing primality*, Journal of number theory, vol **12**, 1980, pp. 128-138
21. Rivest, R.: *Are 'Strong' Primes needed for RSA ?*, preprint, 1991
22. Rivest, R.L., Shamir, A., Adleman, L.: *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, Vol. **21**, No. 2, 1978, pp.120-126.
23. Schneier, B.: Applied Cryptography, Wiley & Sons, 1993