Annales Univ. Sci. Budapest., Sect. Comp. 44 (2015) 3–27

PRIMALITY PROOFS WITH ELLIPTIC CURVES: HEURISTICS AND ANALYSIS

Wieb Bosma, Eric Cator
(Nijmegen, The Netherlands)

Antal Járai and Gyöngyvér Kiss
(Budapest, Hungary)

Communicated by Karl-Heinz Indlekofer

(Received September 12, 2014; accepted February 23, 2015)

Abstract. This paper deals with the heuristic running time analysis of the elliptic curve primality proving (ECPP) algorithm of Atkin and Morain. Our aim is to collect assumptions and the fastest possible algorithms to reduce the heuristic running time and to show that under these assumptions and some plausible conditions the heuristic running time can be reduced down to $o(\ln^4 n)$ bit operation for input possible prime n.

1. Introduction

In the work of Atkin and Morain [1] the background and an exact implementation of elliptic curve primality proving (ECPP) algorithm is described. A heuristic running time analysis is given by Lenstra, Lenstra [20] and Morain [22]. They have found that the running time is $O(\ln^{6+\varepsilon} n)$ for any positive ε for input n. Using asymptotically fast methods for multiplication, division, polynomial calculation, etc. it is possible to reduce this down to $O(\ln^{5+\varepsilon} n)$. With a trick attributed to J. O. Shallit (building up discriminants from small primes), it can be expected to run in time $O(\ln^{4+\varepsilon} n)$.

Key words and phrases: ECPP, elliptic curves, primality proving.
2010 Mathematics Subject Classification: 11Y11, 11A51, 14H52, 97R20.

In this paper we are investigating the heuristic running time of ECPP using the fastest known algorithms to compute the various parts and prove that under some conditions the heuristic running time can be reduced down to $o(\ln^4 n)$. Moreover we are summarizing questions and assumptions that are related to this topic.

There are projects of one of the authors, Gy. Kiss, on implementations that are using these assumptions in practice. One of them is finished, the details can be found in the work of Farkas, Kallós, Kiss [9] and Járai, Kiss [13]. The other project is still ongoing. Both implementations are written in Magma computer algebra system [2]. The aim of this second project is to replace the actual ECPP implementation used in Magma.

2. The ECPP algorithm: outline

We give an outline of the ECPP algorithm; some of the necessary definitions and details will be given in the subsequent sections. Input to the algorithm is an odd positive integer n that is large and probably prime. Given such an integer n, the basic ECPP algorithm proceeds roughly in these three stages:

Algorithm 2.0.1. ECPP

(D) *Starting with $n_0 = n$, find a sequence of probable primes $n_0, n_1, \ldots, n_k$, such that n_{i+1} divides the order of an elliptic curve modulo n_i, such that $n_{i+1} > (\sqrt[4]{n_i}+1)^2$, and such that n_k is small (so that primality of n_k can be established easily).*

(F) *For each of the integers n_i with $i = 0, 1, \ldots, k-1$, construct an elliptic curve E_i moduli n_i of order a multiple of n_{i+1}, together with a point P_i of order n_{i+1} on the curve modulo n_i.*

(P) *Verify that n_k is prime, and that P_i is a point on the elliptic curve E_i modulo n_i of order n_{i+1} and that $n_{i+1} > (n_i^{\frac{1}{4}} + 1)^2$ (see Theorem 3.1), for $i = k-1, k-2, \ldots, 0$.*

In this paper we will be almost exclusively concerned with the first step (D). The difficulties will only become clear when we look at what is done in more detail.

The 'Downrun' part (D) of ECPP will be called recursively with input n_i; the main objective is to find n_{i+1}. This is what happens at level i.

Algorithm 2.0.2. Downrun

(D) Select a pair (D, u) of negative discriminant D and integer u such that $n_i + 1 + u$ is the product of small primes and a probable prime n_{i+1} that exceeds $(\sqrt[4]{n_i} + 1)^2$. This is done as follows:

(0) *Select discriminants D suitable for n_i from a list.*

(1) *For these D, reduce the binary quadratic form $n_i x^2 + Bxy + \frac{B^2 - D}{4n_i} y^2$, where B is such that $B^2 \equiv D \bmod n_i$. If this provides ν with $\nu \cdot \overline{\nu} = n_i$, then $u = \nu + \overline{\nu}$ and the pair (D, u) is usable.*

(2) *From all pairs $(D, \pm u)$ of the previous step, select those for which a probably prime q dividing $n_i + 1 - u$ can be found such that $(n_i + 1 - -u)/q$ is the product of small primes only.*

(3) *Select the best possible pair (D, u) from this list, and let $n_{i+1} = q$, the probable prime for which $(n_i + 1 - u)/q$ is the product of small primes only.*

In the following $m(k)$ denotes the complexity of the multiplication of k-bit integers.

3. The theory

Starting point for the application of ECPP will always be a probable prime $n = n_0$; it is assumed that n will be free of small prime factors (in particular 2 and 3), and that n has passed certain compositeness tests (see below). This will make it very likely that n is indeed prime; the objective is to *prove* that.

In ECPP this is done in a recursive fashion, using the following theorem; precisely what (points P and O_E on) elliptic curves modulo n are, will be explained further on.

Theorem 3.1. *Let $n \in \mathbb{N}$ with $\gcd(6, n) = 1$ and E an elliptic curve modulo n. Suppose there exist $m, n' \in \mathbb{N}$ with $n' \mid m$ such that for every prime factor q of n' there exist P on E with $mP = 0_E$ but $\frac{m}{q} P \neq 0_E$. Then:*

$$n' > (n^{\frac{1}{4}} + 1)^2 \quad \Rightarrow \quad n \text{ is prime.}$$

It is good to keep the following in mind.

Essential (for the *correctness* of this theorem) is the theorem of Hasse, stating that the order of the group of points on any elliptic curve modulo a *prime* number p equals $p+1-u$ for some integer u with $|u| \leq 2\sqrt{p}$. If the conditions of Theorem 3.1 are satisfied the number of points on E modulo n cannot be a proper divisor of n'.

Essential for the *use* of this theorem is that all prime factors q of n' are known; keeping the Theorem of Hasse in mind, one searches for a suitable m satisfying $n+1-2\sqrt{n} < m < n+1+2\sqrt{n}$, where suitable means: m the product of 'small' prime factors and a 'large' probable prime n', that is, $n' > (n^{\frac{1}{4}}+1)^2$. Once such m, n' have been found, the algorithm recurses, with n' replacing n.

Essential for the algorithm to *succeed quickly* is the ability to spot at least one suitable m; current methods from analytic number theory cannot guarantee that it can be found in polynomial time.

Thus, recursively, a sequence of probable primes $n_0 = n, n_1, \ldots, n_k$ is determined, with n_k is small enough to be recognized as prime by other means (like table lookup, or exhaustive pseudo-primality tests).

In reversed order, the 'Find curves' part (F) of the algorithm then constructs pairs (E_j, P_j) for $j = k-1, k-2, \ldots, 0$ of an elliptic curve E_j and a point $P_j \in E[\mathbb{Z}/n_j\mathbb{Z}]$ of order (a multiple of) n_{j+1}.

The primality proof thus constructed will serve as a prime certificate that can be verified in polynomial time, by a computation showing that the given points lie on the given curves and have the given orders.

The difficulties in each of the three steps have been obscured here by lack of detail. In the following subsections we will fill in some important details.

3.1. Probable primes

First we explain the notion of probable primes. Not only because the input n_0 to our algorithm will always be probably prime, but because in the recursion we also want to establish that the n_i for $i > 0$ are probably prime.

Theorem 3.2. *Suppose $n > 3$ is odd, and $n-1 = 2^s t$, with t odd and $s \geq 1$. Let $a \in \mathbb{Z}$ satisfy $1 < a < n-1$, and let $b_j \equiv a^{t2^j} \bmod n$ for $0 \leq j \leq s$. Then n is composite if one of the following holds*

- $\gcd(a, n) \neq 1$;
- $b_k \equiv 1 \bmod n$ *and* $b_{k-1} \not\equiv \pm 1 \bmod$ *for some k with $1 \leq k \leq s$;*
- $b_s \not\equiv 1 \bmod n$.

Moreover, if $n > 9$, odd and composite, then for a fraction of at least $\frac{3}{4}(n-1)$ of the possible a one of these conditions is satisfied.

The proof of the first part will be clear: if n is prime, a cannot be a divisor, and by Fermat's theorem $a^{n-1} \equiv 1 \bmod n$; the important additional property is that modulo a prime number n only ± 1 can be square roots of 1 mod n.

The importance of the second part of Theorem 3.2 is that we obtain an efficient probabilistic test for compositeness by taking values for a at random. If one of the conditions is satisfied we say that a is a *witness* to the compositeness of n. If, for some n, after k random choices for a the computations still have shown up no witness to the compositeness of n, one will be fairly confident that n is *probably* prime. In practice often a fixed set of bases (for example $a = 2, 3, 5, 7, 11, 13, 19, 23, 29, 31$) is used.

3.2. Elliptic curves

Definition 3.1. The *projective plane modulo* m, denoted $\mathbb{P}^2(\mathbb{Z}/m\mathbb{Z})$, for a positive integer m, consists of equivalence classes $(x : y : z)$ of triples $(x, y, z) \in \in (\mathbb{Z}/m\mathbb{Z})^3$ satisfying $\gcd(x, y, z, m) = 1$, under the equivalence $(x, y, z) \sim \sim (\lambda x, \lambda y, \lambda z)$ for any $\lambda \in (\mathbb{Z}/m\mathbb{Z})^*$.

An *elliptic E curve modulo* m, for an integer m coprime to 6, is a pair $(a, b) \in (\mathbb{Z}/m\mathbb{Z})^2$ for which $\gcd(4a^3 + 27b^2, m) = 1$. *The set of points* $E[\mathbb{Z}/m\mathbb{Z}]$ on an elliptic curve E modulo m consists of $(x : y : z) \in \mathbb{P}^2(\mathbb{Z}/m\mathbb{Z})$ for which

$$y^2 z = x^3 + axz^2 + bz^3.$$

Define $V = V_m(E)$ to be the set of all $(x : y : 1) \in \mathbb{P}^2(\mathbb{Z}/m\mathbb{Z})$ on the elliptic curve together with $O = (0 : 1 : 0) \in \mathbb{P}^2(\mathbb{Z}/m\mathbb{Z})$. Given (V, a), the *partial addition* operation $\oplus$ computes for any pair $P = (x_p : y_p : z_p)$, $Q = (x_q : y_q : : z_q) \in V$ either an element in $R \in V$ (the sum of P and Q) or a non-trivial factor of m:

(0) If $z_p = 0$ then $R = Q$ and if $z_q = 0$ then $R = P$;

(1) If $x_p = x_q$ and $y_p = -y_q$ then $R = (0 : 1 : 0)$;

(2) If $x_p \neq x_q$ and $y_p = -y_q$ define $v = x_p - x_q$, and $\lambda = (y_p - y_q)$. Now let (x_r, y_r, z_r) be $((y_p - y_q)^2 v - (x_p + x_q)v^2, (y_p - y_q)^3 - (y_p - y_q)(2x_p + +x_q)v^2, v^3)$;

(3) otherwise let $w = y_p + y_q$, and let (x_r, y_r, z_r) be $((3x_p^2 + a)^2 w - (x_p + +x_q)w^2, (3x_p^2 + a)^3 - (3x_p^2 + a)(2x_p + x_q)w^2, w^3)$.

(4) If $\gcd(z_r, m) = 1$, then $(x_r, y_r, z_r) \sim R = (x'_r : y'_r : 1) \in V$ and we say that $R = P \oplus Q$ is the *sum* of P and Q in V; if $d = \gcd(z_r, m) \neq 1$ then we have found a non-trivial factor of m.

In the special case that $m = p$ is a prime number, the following important special properties hold. First of all, the partial addition algorithm will produce a sum $R = P \oplus Q$ for every pair $P, Q \in E[\mathbb{Z}/p\mathbb{Z}]$, and the operation is then called the *addition* algorithm, and the sum usually denoted $R = P + Q$. Moreover, due to Hasse and Deuring [7] the following holds; see also [4] and [27].

Theorem 3.3. *Let $p > 3$ be a prime number. The set $E[\mathbb{Z}/p\mathbb{Z}]$ forms a finite abelian group under addition, with zero element $O = (0 : 1 : 0)$. The number of elements $N = \#E[\mathbb{Z}/p\mathbb{Z}]$ of this group satisfies the bounds $p + 1 - 2\sqrt{p} < < N < p + 1 + 2\sqrt{p}$. Conversely, for every N between these bounds there exists an elliptic curve modulo p with order N.*

Remark 3.1. Moreover, it can be shown that for a prime divisor p of arbitrary m coprime to 6 and the discriminant of E, the sum R produced by the partial addition algorithm for any two points P, Q on an elliptic curve $E_{a,b}$ modulo m, has the property that R_p (obtained by reducing the coordinates of R modulo p) is the sum of the points P_p and Q_p in the group $E_{\bar{a},\bar{b}}[\mathbb{Z}/p\mathbb{Z}]$, where $\bar{a} \equiv a \bmod p$, and $\bar{b} \equiv b \bmod p$. So $R \bmod p = (P \bmod p) + (Q \bmod p) \in E_{\bar{a},\bar{b}}[\mathbb{Z}/p\mathbb{Z}]$.

Using the partial addition algorithm repeatedly, it is of course possible to obtain a partial multiplication algorithm, which computes either $k \cdot P$ or finds a divisor of m, for any positive integer k, given any $P \in V$ and any a as before. However, there are various ways to speed up this computation of $k \cdot P$, using partial doubling, and the fact that it is not necessary to keep track of the y-coordinate. Also, writing points in Montgomery representation may speed up the algorithm (see for example [3]).

In the next sections we will occasionally be sloppy, and write about the sum and multiples of points on elliptic curves modulo n; we mean the result of application of the partial addition and multiplication algorithms, which in unlikely exceptional cases (when a probable prime is in fact composite) means that a divisor of n could be produced, rather than a point. In practice this does not happen, as probable primes will be prime.

3.3. Complex multiplication

Although there exists an efficient algorithm to compute the order of the group of points on an elliptic curve over a finite field (usually referred to as SEA; [26]), the complexity of this algorithm is too high in practice. Therefore, Atkin and Morain [1] devised an alternative way to find suitable elliptic curves, by reduction of special curves defined in characteristic zero.

Suppose that the probable prime $n = n_i$ is given; we drop the subscript i in this section. We describe next how to constructs candidate numbers $n + 1 - u$,

with $|u| < 2\sqrt{n}$ as cardinalities of elliptic curves modulo n. The trick is to use the reduction modulo n of elliptic curves in characteristic zero for which the number of points is known, since they admit *complex multiplication*. This works as follows.

Let D be a negative fundamental discriminant; this means that $D < 0$, that it is a discriminant (an integer with $D \equiv 0, 1 \bmod 4$) and fundamental in the sense that D/f^2 is not a discriminant, for any integer f. If $n = \nu\bar{\nu} = u^2 - Dv^2$ in the ring $\mathcal{O} = \mathbb{Z}[\frac{D+\sqrt{D}}{2}]$, the ring of integers of the imaginary quadratic field $\mathbb{Q}(\sqrt{D})$, then it is easy to write down coefficients a, b of an elliptic curve with complex multiplication by $\mathcal{O}$ having $(v-1)(\bar{\nu}-1) = n+1-u$ and one having $(-v-1)(-\bar{\nu}-1) = n+1+u$ points when taken modulo n, if n is prime indeed. (The cases $D = -3, -4$ are special, as the roots of unity in the complex multiplication field give rise to more curves and more easily constructed curves in fact;). So we find candidate numbers $n + 1 \pm u$, provided we can find the decomposition $n = \nu\bar{\nu}$. [20]

We may, and will, assume that n is odd and coprime to D. Next suppose that we can find b such that $b^2 \equiv D \bmod n$; by adding n to b if necessary, we can then also achieve $b^2 \equiv D \bmod 4n$. Then consider $Q(x, y) = nx^2 + bxy + \frac{b^2-D}{4n}y^2$, which will be a binary quadratic form of discriminant D, which is primitive. This is equivalent to some reduced form $R(x, y)$, and a solution to $n = \nu\bar{\nu}$ exists if and only if $Q(x_0, y_0) = n$ for some integers x_0, y_0, which will be the case if and only if $R(x, y)$ is the unit binary form of discriminant D (being either $x^2 - \frac{D}{4}y^2$ or $x^2 - x + \frac{1-D}{4}y^2$, depending on D modulo 4).

The reduction time using binary reduction algorithm is proportional to $(\ln n)^2$, but this can be decreased to $O\big(m(\ln n) \ln\ln n\big)$ with the algorithm of Cornacchia ([5], p. 34) combined with controlled euclidean descent of Schönhage [31].

3.4. Modular square roots

This implies that we will be able to construct curves with $n + 1 \pm u$ points provided we can solve $b^2 \equiv D \bmod n$ and the form Q reduces to the unit form; the latter is equivalent to the ideal $(n, \frac{b-\sqrt{D}}{2})$ being principal in $\mathcal{O}$. If the ideal class is assumed to be random, this happens with probability $\frac{1}{2h}$, where $h = h(D)$ is the ideal class number of $\mathcal{O} = \mathcal{O}_{\sqrt{D}}$ [20]. To ensure that the first condition holds we choose D such that the Jacobi symbols $(D|n)$ is equal to 1.

What one does in practice, for given n, is to compute the Legendre symbols $(n|p)$ for primes p below some bound, and build up discriminants as products of small primes (and -4 or ± 8) such that the condition holds. Note that an odd negative discriminant can be written uniquely as the product $D = \prod_j p_j^*$,

where $p_j^* = \pm p_j$ for prime numbers p_j and the signs of p_j^* are chosen such that $p_j^* \equiv 1 \bmod 4$. The condition $(D|n)$ for the Jacobi symbol of D then easily translates to conditions on $(n|p_j)$ depending on n mod 4. Similarly if D contains a factor -2 or -4. One expects roughly half the primes from the list to satisfy the condition. For those we compute a modular square root.

Up to a limit s, for any prime number p for which $(n|p) = 1$, the square roots of p or $-p$ is calculated. The small primes are stored in a precalculated list, containing half the difference of consecutive odd primes. Calculating one square root with algorithm of Tonelly and Shanks ([5], p. 32) requires roughly the same time as the probabilistic primality test for n.

Indeed, the algorithm has a probabilistic part which find a generator of $(\mathbb{Z}/n\mathbb{Z})^*$ and a deterministic part using $2\lceil \lg(n) \rceil + e^2$ multiplication modulo n, where e is the largest exponent for which $2^e | n-1$. Because we know that n is odd, we expect the value e for this exponent to occur with probability $2^{-(e+1)}$. The expected number of multiplications is $O(\ln n)$. Heuristics suggest that the running time of the probabilistic part can be neglected compared to the running time of the deterministic part. The total running time of one square root operation is $O\big(m(\ln n) \ln n\big)$.

3.5. Factorization

Elliptic curves modulo n with order $n + 1 \pm u$ can now be found. These orders will be trial divided with primes up to a limit t. Only $u \bmod p$ is calculated, because $n + 1 \bmod p$ is precalculated. Division by the divisor found is performed, and the quotient is stored. The necessary time is proportional to $t \ln n$.

A 'batch trial division' using fast multiplication and fast gcd can be applied to remove divisors from $10^4 \ldots 10^6$ up to an upper limit b, say up to 10^8. This trick was suggested by one of the authors several years ago and seem to have appeared independently by several authors; Pollard, Strassen, D. J. Bernstein, for example; see [10], p. 353. The product of curve orders is calculated: first products of pairs, then products of quadruplets, etc. Prime products will be read from a file. It is divided by the product of curve orders, then the gcd of the remainder and the product of curve orders is calculated. This is small. It is distributed to partial products of curve orders, etc., product of quadruplets, product of pairs, and finally to curve orders. The surprise is that the dependence on $\log n$, the size of curve orders, is very weak. The algorithm seems to be superior to other factorization procedures useful here, as Pollard ρ and $p-1$, the elliptic curve method, etc., for factors below 10^8, but further investigation is necessary to find the best limit.

3.6. Iteration

One of the difficulties in optimizing the Downrun algorithm lies in the choices to be made at a given stage i, for some known probable prime n_i. In this section we will attempt to describe a simple model for this process.

To avoid numerous indices, we drop the i again and assume we are dealing with the probable prime n. Naively speaking, the process consists of traversing a decision *tree*: at the node for n we create a set of descendants (corresponding to elliptic curve orders) and choose from those the one we consider to be best. At this descendant node we repeat. The set of descendants is created from a much larger set of candidates: to become descendant the order of the elliptic curve needs to be suitable: smooth with one big probable prime divisor (more on that below).

Theorem 3.4. *Under reasonable assumptions, namely that*

- *the probability that a given curve order is suitable is small and independent of the particular curve, and*
- *the probabilities are independent for different curves, and do not vary too much,*

the number of descendants at some node is asymptotically distributed according to a Poisson-λ distribution, with λ equal to the average fraction of suitable curves among all elliptic curves.

Let us sketch the proof.

Let X_j (for $1 \leq j \leq N$) be stochastic variables, representing the probability that the i-th curve order m_j is suitable for the next step in the primality proof. The first assumption is that the success probability

$$p_j = \mathbb{P}(X_j = 1)$$

is small, $p_j \ll 1$; we will write $p_j = \frac{u_j}{N}$. Assuming that the X_j are independent, the expected number of successes will equal

$$\mathbb{E}(\sum_{j=1}^{N} X_j) = \sum_{j=1}^{N} \frac{u_j}{N} = \overline{u},$$

and we will also assume that this average $\overline{u}$ converges to some limit μ as $N \to \infty$.

To show that the distribution of $\sum X_j$ converges to a Poisson distribution, it suffices to show for a real variable t that $\mathbb{E}(e^{t\sum_{j=1}^{N} X_j})$ converges to $\mathbb{E}(e^{tY})$ for a stochastic variable Y with Poisson-λ distribution.

But, using again that the X_j are supposed to be independent, and that $P(X_j = 1) = \frac{u_j}{N}$ and hence $P(X_j = 0) = 1 - \frac{u_j}{N}$:

$$\mathbb{E}(e^{t\sum_{j=1}^N X_j}) = \prod_{j=1}^N \mathbb{E}(e^{tX_j}) = \prod_{j=1}^N \left((1 - \frac{u_j}{N})e^0 + \frac{u_j}{N}e^t\right) =$$

$$= \prod_{j=1}^N \left(1 + \frac{u_j(e^t - 1)}{N}\right),$$

the logarithm of which equals

$$\sum_{j=1}^N \log(1 + u_j\frac{e^t - 1}{N}) = \sum_{j=1}^N \left(\frac{u_j(e^t - 1)}{N} + O(u_j^2/N^2)\right) \to \overline{u}(e^t - 1),$$

for $N \to \infty$, when assuming that u_j^2 does not vary to wildly (more precisely: $\sum_{j=1}^N \frac{u_j^2}{N} = o(N)$). So, $\mathbb{E}(e^{t\sum_{j=1}^N X_j})$ tends to $e^{\mu(e^t-1)}$ as $N \to \infty$.

But for a stochastic variable Y with Poisson-λ distribution we have:

$$\mathbb{E}(e^{tY}) = \sum_{j=0}^\infty \frac{\lambda^j}{j!}e^{-\lambda}e^{tj} = \sum_{j=0}^\infty \frac{(\lambda e^t)^j}{j!}e^{-\lambda} = e^{\lambda(e^t-1)}.$$

Thus, our variable $\sum X_j$, signifying the number of successes, behaves like a Poisson-disctributed stochastic variable with parameter $\lambda = \mu$. This concludes the proof sketch.

Note that the parameter may well depend on the stage at which we are, for example on the *size* of the numbers involved at the current stage.

Continuing our first, naive, approach, let us suppose that at each node in the primality proof the number of successes has Poisson distribution with parameter μ_i, where the index i now makes explicit the dependence on the level. We would like to compute the probability that our 'tree' has an 'infinite' branch, indicating that we will be able to complete the primality proof.

Let p_0 denote the probability that the entire tree has an infinite branch; then

$$p_0 = \mathbb{P}_{\mu_0\mu_1\ldots}(\infty\text{−branch}),$$

depends on the parameters in the nodes at all levels $0, 1, \ldots$. Likewise, once we are at level k, we have

$$p_k = \mathbb{P}_{\mu_k\mu_{k+1}\ldots}(\infty\text{−branch}).$$

But this leads to a recursion:

$$p_k = \left(\sum_{\ell=1}^{\infty} \frac{\mu_k^\ell}{\ell!} e^{-\mu_k} \right) \left(1 - (1 - p_{k+1})^\ell\right) = 1 - e^{-\mu_k p_{k+1}}.$$

as the first factor represents the probability that there is at least 1 descendant at the k-th node, and the second factor expresses the probability that at least one of these children will have a further descendant. Roughly speaking (since all of our trees will be finite) this probability is the probability that at *no* stage of the algorithm we will have to backtrack (due to the absence of further suitable descendants).

If, for simplicity, we assume that all μ_j are equal to some fixed μ, we find that $p = p_0 = p_k = 1 - e^{-\mu p}$. Note that we get a unique solution $p > 0$ for $\mu \geq 1$. It will be clear that p increases with increasing value of μ: the more suitable descendants are created at each step, the larger the probability that we will not have to backtrack at all. This shows the main defect of this naive model: increasing μ at each level will always be beneficial! This is an artefact of not taking the *cost* for creating more descendants into account.

Therefore our second, slightly more sophisticated approach, will be to take two types of cost into account: the cost C for creating an additional descendant (essentially, costs for modular square roots and quadratic form reduction), and the cost K for computing the additional data for a new node (performing factorization of the smooth part and probabilistic prime testing). In fact we introduce a single new parameter giving the ration between C and K into the model.

Currently, extensive simulations are taking place to compute the optimal value for μ to be chosen as a function of this parameter. [19]

3.7. The other parts

We spend a few words on parts (F) and (P) in the outline of ECPP as in Algorithm 1.

Algorithm 3.0.3. Find curves; and Proof

(F) Construct elliptic curves and points on them.

(0) **Hilbert polynomial.** *The first step is the calculation of the Hilbert polynomial (or a related polynomial) for the selected discriminants D using floating point calculations. The resulting coefficients are rounded to integers.*

(1) **Polynomial root.** *The second step is the calculation of a root of the Hilbert polynomial, which has degree $h(D)$, modulo n. This uses the well-known splitting procedure, until a degree 1 part is obtained. To do this, we have to calculate the nth power of a random first-degree polynomial modulo the Hilbert polynomial modulo n. If the more refined version of Atkin and Morain is used, then the degree is $h^*(D) = h(D)/2^{t-1}$, where t is the number of factors of D.*

(2) **Find curve and point on it.** *The third step to find the appropriate elliptic curve from the two possible twists, and an appropriate point on it.*

(P) Partial proof verification, given n, n', P, E.

(0) **Proof step.** *Verify that n' divides n, and compute $n'P$ and nP; then verify that both are on the given elliptic curve E modulo n, and that $n'P \neq O_E$ while $nP = O_E$.*

4. Asymptotic running time analysis

Let us make a more detailed analysis of the running time for each step separately. Of course, the analysis will strongly depend on the complexity for multiplication of integers (which will be of the same order as that for division and squaring), and which will, for any known practical method, exceed the complexity of addition, subtraction, and multiplication by powers of two (shifts). To keep our calculations as far as possible independent from the choice of fast multiplication methods, $m(k)$ shall denote the time that we need to multiply arbitrary k-bit numbers using the chosen multiplication method. The Fast Fourier Transform, and the method of Schönhage and Strassen will make it possible to keep this time as low as $O(k \ln k \ln \ln k)$ even on a multitape Turing machine with enough tape.

In the rest of the paper $\ln^k n$ shall denote $(\ln n)^k$, $\ln \ln^k n$ shall denote $(\ln \ln n)^k$, and so on.

In the following, we use three parameters, that may depend on the size of n, and the choice of which will determine the running time. We have already seen $b(n)$ above, which is the smoothness bound in the factorization of the elliptic curve orders. The second will be $d(n)$, a bound on the size of the discriminant used, and the third is $s(n)$, a smoothness bound on the factors of the discriminants.

D_1: Selection of discriminants. We need the square root of D modulo n for several discriminants D. To keep the number of square root calculations reasonable, we will consider only discriminants D with absolute value below a bound $d(n)$ that are $s(n)$-smooth for some smoothness bound s.

As a standard example, we may choose $d(n) \asymp (\ln n)^2$. To obtain a lower estimate for the number of fundamental discriminants below $d(n)$ (in absolute value), we consider only the classes of -3, -4, -7, -8, -11 and -15 from the residue classes mod 16, because any number from these classes which is free from a square of any odd primes is necessarily a fundamental discriminant. (Not hard to see that only these are the fundamental discriminants.) We estimate the density of such numbers. The density of square free numbers is $\prod_{p\in\mathbb{P}}(1-1/p^2)$ having reciprocal

$$\prod_{p\in\mathbb{P}}\big(1-\frac{1}{p^2}\big)^{-1} = \prod_{p\in\mathbb{P}}\big(1+\frac{1}{p^2}+\frac{1}{p^4}+\cdots\big) = \sum_{n\in\mathbb{N}^+}\frac{1}{n^2} = \frac{\pi^2}{6}.$$

Considering that the density of odd square free numbers in general is $(6/\pi^2)\cdot(4/3) = 8/\pi^2$ and they reside only in six residue classes modulo 16, asymptotically we obtain that there are at least

$$3d(n)/\pi^2$$

fundamental discriminants below $d(n)$. To compute the modular square root of all these discriminants will be too time-consuming. Instead, we only consider the fundamental discriminants which are $s(n) \asymp d(n)^c$ smooth for some $0 < < c < 1$. Then we only have to calculate a square root modulo n for each prime below $s(n)$ for which $(n|p) = 1$. That is, for half the primes below $d(n)^c$, i.e., $O\big(d(n)^c/\ln d(n)\big)$ primes. So if $d(n) = O(\ln^2 n)$ and $m(k) = O(k \ln k \ln\ln k)$ then to determine the modular square roots we need $O(\ln^3 n \ln\ln\ln n)$ bit operations if $c = 1/2$ and $o(\ln^3 n)$ bit operations if $c < 1/2$. Still we may ([18], 4.5.4) suppose that the number of discriminants which can be factored is $\asymp \ln^2(n)$.

To check all the fundamental discriminants up to $d(n)$ whether they are $d(n)^c$ smooth and to find their factorizations we may use simple trial division by all primes up to this limit. This takes at most $O\big(d(n)^{1+c}\ln^3 d(n)\big)$ bit operations using the classical division procedure.

So, as we will see, with an appropriate choice of $c < 1/2$ the total amount of time for this step can be neglected compared with the amount of time to the other steps.

D_2: Modular square roots. For a given discriminant D first we check whether the conditions $(D|n) = 1$ and $(n|p) = 1$ for all odd prime factors of D are satisfied. The probability that these conditions are satisfied is 2^{-t} where t is the number of 'prime' factors of D; here -4 or -8 is considered as a 'prime'

factor but -1 is not. Because of the Erdős–Kac theorem [14] the average number of the factors for naturals up to x has in limit normal distribution with expected value $\ln\ln x$ and variance $\ln\ln x$, we obtain that this probability is

$$2^{-\ln\ln d} = (e^{-\ln\ln d})^{\ln 2} = 1/\ln^{\ln 2} d(n).$$

If $\ln d(n) \asymp \ln\ln n$, this probability is $\asymp 1/\ln\ln^{\ln 2} n$. For example, this is the case for any choice of $d(n) \asymp \ln^{\alpha}(n)\ln\ln^{\beta}(n)$ with $\alpha > 0$. The time to calculate the Jacobi symbols can be neglected, because $(n|p) = 1$ is already checked, $(4|n)$ and $(\pm 8|n)$ are precalculated. Moreover, we have to calculate the square roots of the discriminants. This needs time

$$O\big(m(\ln n)d(n)/\ln^{\ln 2} d(n)\big)$$

with the following simple trick: we precalculate the square root modulo n of all $\pm k$ for each k up to, say, $\sqrt{d(n)}$ which is the product of primes having $(n|p) = 1$. After this, at most three modular multiplications give the square root of D modulo n; indeed if D is written as the product of four factors, then the product of the least two is not greater then $\sqrt{d(n)}$.

$\mathbf{D}_3$: Reduction of quadratic forms. On the remaining D's the reduction algorithms of quadratic forms or the Cornacchia algorithm is applied [5]. The Cornacchia algorithm consists of three parts:

(0) Computing Kronecker symbol $\left(\frac{D}{n}\right)$.

In our case this was done previously in $\mathbf{D}_1$.

(1) Computing the square root of D.

In our case this was done previously in $\mathbf{D}_2$.

(2) Euclidean Algorithm.

Here we can use the controlled Euclidean descent algorithm of Schönhage [31], which needs time $O\big(m(\ln n)\ln\ln n\big)$.

Thus the running time can be decreased down to

$$O\big(m(\ln n)\ln\ln n\big).$$

Hence the total running time in our case is

$$O\big(m(\ln n)d(n)\ln\ln n/\ln^{\ln 2} d(n)\big).$$

$\mathbf{D}_4$: Factorization. This step is to remove small factors and find large factors of elliptic curve cardinalities.

The number of the elliptic curves $e(n)$ is twice the number of the remaining good discriminants. Small prime factors of the cardinalities of the group of points on these curves modulo n are removed with a simple trial division up to a limit $t(n)$. This takes time

$$O\big(e(n)t(n)\ln n\big)$$

altogether. If $t(n) = O(1)$, then this time can be neglected.

We now concentrate on finding larger factors by various methods.

(a) Further trial division. If we use only simple trial division method to remove small factors up to the bound $b(n)$, then this takes time

$$O\big(e(n)b(n)\ln n\big).$$

(b) Batch trial division. One of the most practical methods is our 'batch trial division' up to a limit $b(n)$. If $b(n) \geq e(n)\ln n$, then dividing the product of primes up to $b(n)$ (which is the magnitude of $e^{b(n)}$) to parts having the same size as the products of curve orders, we obtain

$$\frac{b(n)}{e(n)\ln n}$$

parts. Then we take the gcd of each part with the product of curve orders. This takes time

$$O\left(\frac{b(n)}{e(n)\ln n} m\big(e(n)\ln n\big)\ln\big(e(n)\ln n\big)\right)$$

and size

$$O\big(e(n)\ln n\big)$$

of core space. A somewhat better solution may be to divide the product of primes (as it is read from disc in binary form, starting with MSB) with the product of curve orders, and then taking the gcd of the remainder and the product. This also needs

$$O\big(e(n)\ln n\big)$$

core space, but only

$$O\left(m\big(e(n)\ln n\big)\big(\frac{b(n)}{e(n)\ln n} + \ln\big(e(n)\ln n\big)\big)\right)$$

time.

(c) Pollard ϱ. The so-called Pollard ϱ method can be applied. To find factors up to $b(n)$ roughly, we need $O\big(\sqrt{b(n)}\big)$ iterations, and each iteration needs $O\big(m(\ln n)\big)$ bit operations. We also need gcd operations, which take $O\big(m(\ln n)\ln\ln n\big)$ bit operations, but it seems to be enough to use a gcd operation only after $\ln\ln n$ iterations. This way the total running time will be

$$O\big(e(n)\sqrt{b(n)}m(\ln n)\big).$$

(d) Pollard $p-1$. The $p-1$ method of Pollard can also be applied. Here we can choose parameters so that all factors $p < b(n)$ are found for which $p-1$ is $\sqrt{b(n)}$-smooth. Hence, it will be almost sure to find a prime $p \ll b(n)$, but around $p \approx b(n)$ only with probability ≈ 0.34 ([18], 4.5.4 about the limit distribution of size of largest prime factor). The running time is similar to that of the previous method of Pollard.

(e) ECM. We also may use the elliptic curve factorization method to find the small factors of the elliptic curve cardinalities. By Lenstra and Lenstra [20], 4.3, p. 698 the optimal choice to find factors below $b(n)$ to take the "smoothness bound" s to $L_{b(n)}(1/\sqrt{2})$ and to use $L_{b(n)}(1/\sqrt{2})$ elliptic curves, where $L_k(\beta)$ defined by

$$L_k(\beta) = e^{\beta\sqrt{\ln(k)\ln\ln(k)}}.$$

In this case we shall find prime factors below $b(n)$ with large probability. The total time needed for $e(n)$ curve cardinalities is

$$O\left(e(n)\ln b(n)m(\ln n)L_{b(n)}(\sqrt{2})\right).$$

$\mathbf{D}_5$: Miller-Rabin test. We have to test $e(n)$ remaining unfactored parts with the fast probabilistic Miller-Rabin test ([5], p. 415). A fixed number of bases (for example $2, 3, 5, 7, 11, 13, 19, 23, 29, 31$) is used. This takes time

$$O\big(e(n)m(\ln n)\ln n\big).$$

$\mathbf{F}_1$: Hilbert polynomial. To complete the proof certificate, the first step is the calculation of the Hilbert polynomial. This running time can be neglected.

$\mathbf{F}_2$: Polynomial root. The second step is the calculation of a root of the Hilbert polynomial, which has degree $h(D)$. The degree certainly $\leq 2\sqrt{d(n)}\ln d(n)$, but we may hope $\asymp \sqrt{d(n)}$ [20]. If the more refined version of Atkin and Morain is used, then the degree is $h^*(D)$, so we may hope the degree $\asymp \sqrt{d(n)}/\ln^{\ln 2} d(n)$. So we have two cases:

(a) In a simple implementation the running time for one step is

$$O\big(m\big(h(D)\ln n\big)\ln n\big).$$

(b) In a more refined implementation the running time for one step is

$$O\big(m\big(h^*(D)\ln n\big)\ln n\big).$$

F_3: Find elliptic curve. The third step is to find an appropriate elliptic curve. The number of elliptic curves to try is only two, and the expected number of points to try is also bounded. Hence this step needs

$$O\big(m(\ln n)\ln n\big)$$

time. This can be neglected.

P_1: Proof. One proof verification step needs

$$O\big(m(\ln n)\ln n\big)$$

time. This can be neglected.

5. Heuristics

The key to the running time analysis is the following observation: if we are able to find all prime factors less than $b(n)$ of the number $m = n + 1 \pm s$, an elliptic curve modulo n with m points is a suitable candidate for the 'Downrun' algorithm (D) precisely if the *second largest* prime factor of m is less than b. The probability that this happens, if we use for example trial division, where we guarantee that the number m is completely factored below $b(n)$ (under randomness assumptions for m) is supposedly approximately

$$e^\gamma \frac{\ln b(n)}{\ln n} \approx 1.7811 \frac{\ln b(n)}{\ln n}.$$

Hence it is reasonable to assume that if we have altogether $e(n)$ such numbers m, then the number of successes has a probability distribution with mean approximately equal to

$$\lambda = e^\gamma \frac{\ln b(n)}{\ln n} e(n).$$

Because for each negative discriminant $D \le -7$ the probability of the success is

$$\frac{1}{2h(D)},$$

where $h(D)$ denotes the ideal class number, and each successful D results two different m's, we expect

$$e(n) \approx \sum_{-d(n)\le D\le -7} \frac{1}{h(D)}.$$

Four important theoretical questions appear:

Question 1. *What can be said about the asymptotic behavior of the function*

$$\bar{e}(n) = \sum_{-d(n)\le D\le -7} \frac{1}{h(D)},$$

where we sum the ideal class numbers over $s(n)$-smooth negative discriminants D?

For example, it would be good to know that for some $c < 1/2$ we have $s(n) \asymp d(n)^c$, $\bar{e}(n) \asymp \sqrt{d(n)}$, or at least for $c = 1/2$.

Question 2. *For different factorization methods, other than trial division, such as Pollard ϱ, Pollard $p-1$, the elliptic curve method, etc., where we cannot guarantee that we find all the prime factors below $b(n)$, applying them with various parameters to find prime factors of the cardinalities of $e(n)$ elliptic curves modulo n, what will be the expected number of curve cardinalities completely factored?*

Question 3. *Suppose that, applying the different factorization methods with various parameters to find prime factors, one succeeds in factoring completely the cardinality of a given elliptic curve modulo n, what is the expected value and distribution of the 'gain', defined as* log *of the 'smooth part' of the cardinality?*

For example, using trial division or batch trial division up to a bound $b(n)$, the expected value of the 'gain' is supposed to be $\asymp \ln b(n)$. The heuristic behind this is the following: for each prime p the probability that p divides a given curve order m is $1/p$ and this results in a 'gain' of $\ln p$. Hence the expected value of the total gain is

$$G(b(n)) = \sum_{p\in\mathbb{P}, p\le b(n)} \frac{\ln p}{p} \sim \int_2^{b(n)} \ln x \frac{1}{\ln x}\frac{1}{x}\,dx \sim \ln b(n).$$

By other methods we may still expect that the 'gain' will only differ by a bounded factor, so it remains $\asymp \ln b(n)$.

We may suppose that if the expected number of suitable curves is large enough, then with a fixed positive probability we will succeed. Note that λ is the parameter of the Poisson distribution in 3.6. This gives the condition

$$\lambda = e^{\gamma} \frac{\ln b(n)}{\ln n} e(n) > 1.$$

We have the possibility always to use more effort in a given step or to backtrack to a previous level. We would like to find the optimal solution for this problem too.

Question 4. *What is the optimal choice of the parameters b, s, d?*

It is not clear what the best strategy for traversing the 'tree' of possibilities is; on the one hand, one would want to minimize the time spent on searching for suitable discriminants, while on the other hand the 'gain' should be maximized.

The problem seems similar to some problems in percolation theory. We plan to investigate an optimized choice of parameters in our implementation. Occasionally we may need a backtrack step; some of the data calculated earlier can be reused by backtracking, which complicates the optimization problem.

6. Strategies

Our aim is to verify that with certain choices of parameters $d(n)$, $s(n)$ and $b(n)$, the running time could be reduced down to $o(\ln^4 n)$.

Let $m(k) = O(k \ln k \ln \ln k)$. We suppose that the 'gain' $G\big(b(n)\big) \asymp \ln b(n)$, the expected number of steps $I\big(b(n)\big) \asymp \ln n / G\big(b(n)\big)$, $e(n) \asymp \sqrt{d(n)}$ if $s(n) \asymp$ $\asymp d(n)^c$ for some appropriate $c < 1/2$,

$$h(D) = O(\sqrt{d(n)})$$

and

$$h^*(D) = O\big(\sqrt{d(n)} / \ln^{\ln 2} d(n)\big).$$

From the previous chapters we could see that from running time point of view the most significant parts are D_3, D_4, D_5, F_2a and F_2b. For this reason in the following we will consider only the running time of these parts to determine the running time of one step.

'Strong factorization' strategy. Let us take simply take $d(n) \asymp \ln^2 n$, $h(D) = O\big(\ln n\big)$, $h^*(D) = O\big(\ln n/\ln\ln^{\ln 2} n\big)$. The time of the most critical steps, D_3, D_6 and F_2 is the following:

$\mathbf{D}_3$:

$$O\big(d(n)m(\ln n)\ln\ln n/\ln^{\ln 2} d(n)\big) = O\big(\ln^3 n \ln\ln^{2-\ln 2} n \ln\ln\ln n\big);$$

$\mathbf{D}_5$:

$$O\big(e(n)m(\ln n)\ln n\big) = O\big(\ln^3 n \ln\ln n \ln\ln\ln n\big);$$

$\mathbf{F}_2$a:

$$O\big(m(h(D)\ln n)\ln n\big) = O\big(\ln^3 n \ln\ln n \ln\ln\ln n\big);$$

$\mathbf{F}_2$b:

$$O\big(m(h^*(D)\ln n)\ln n\big) = O\big(\ln^3 n \ln\ln^{1-\ln 2} n \ln\ln\ln n\big).$$

$\mathbf{D}_4$: In the different cases of factoring we choose $b(n)$ in such a way that the running time of the factoring won't exceed the most critical step D_3, so we can take the running time of D_3 as the running time of one step. As the 'gain' $G(b(n)) \asymp \ln b(n)$ and the number of the iterations $I(b(n)) \asymp \ln(n)/G(b(n))$ are depending on $b(n)$, we will determine here also the total running time of the algorithm for each factoring method.

(a): Trial division. If $b(n)$ is between $\asymp \ln n$ and $\asymp \ln n \ln\ln^{2-\ln 2} n \ln\ln\ln n$ then the running time $O\big(e(n)b(n)\ln n\big)$ is between

$$O\big(\ln^3 n\big)$$

and

$$O\big(\ln^3 n \ln\ln^{2-\ln 2} n \ln\ln\ln n\big).$$

As $G(b(n)) \asymp \ln\ln n$ in both cases, $I(b(n)) \asymp \ln n/\ln\ln n$. Then the total running time is

$$O\big(\ln^4 n \ln\ln^{1-\ln 2} n \ln\ln\ln n\big).$$

(b): Batch trial division. If $b(n)$ is between $\asymp \ln n$ and $\asymp \ln^3(n)\ln\ln^{1-\ln 2} n$, then the running time $O\left(m\big(e(n)\ln n\big)\left(\dfrac{b(n)}{e(n)\ln n} + \ln\big(e(n)\ln n\big)\right)\right)$ is between

$$O\big(\ln^2 n \ln\ln^2 n \ln\ln\ln n\big)$$

and

$$O\big(\ln^3 n \ln\ln^{2-\ln 2} n \ln\ln\ln n\big).$$

As $G(b(n))$ and $I(b(n))$ are the same as in (a), the total running time will be also the same.

(c/d): Pollard ρ and $p-1$. If $b(n)$ is between $\asymp \ln n$ and $\asymp \ln^2 n \ln\ln^{2-2\ln 2} n$, then the running time $O\big(e(n)\sqrt{b(n)}m(\ln n)\big)$ is between

$$O\big(\ln^{2.5} n \ln\ln n \ln\ln\ln n\big)$$

and

$$O\big(\ln^3 n \ln\ln^{2-\ln 2} n \ln\ln\ln n\big).$$

As $G(b(n))$ and $I(b(n))$ are the same as in (a) and (b), the total running time will be also the same.

(e): ECM. If

$$b(n) \asymp (\ln n)^{\ln\ln n/\ln\ln\ln^2 n}$$

then the running time is

$$O\big(e(n)\ln b(n) m(\ln n) L_{b(n)}(\sqrt{2})\big) =$$

$$= O\big(\ln^{2+2/\sqrt{\ln\ln\ln n}} n \ln\ln^3 n/\ln\ln\ln n\big),$$

as

$$L_{b(n)}(\sqrt{2}) = e^{\sqrt{2\ln b(n)\ln\ln b(n)}} =$$

$$= e^{\sqrt{4(\ln\ln\ln n - \ln\ln\ln\ln n)\ln\ln^2 n/\ln\ln\ln^2 n}} <$$

$$< e^{\sqrt{4\ln\ln^2 n/\ln\ln\ln n}} =$$

$$= e^{2\ln\ln n/\sqrt{\ln\ln\ln n}} = \ln^{2/\sqrt{\ln\ln\ln n}} n.$$

As $G\big(b(n)\big) \asymp \ln\ln^2 n/\ln\ln\ln^2 n$, $I\big(b(n)\big) \asymp \ln n \ln\ln\ln^2 n/\ln\ln^2 n$. Then the total running time is

$$O\big(\ln^4 n \ln\ln\ln^3 n/\ln\ln^{\ln 2} n\big).$$

Conclusion. We can see that, in (e), the running time is $o(\ln^4 n)$, but in the other cases the running time of one step is too big. We could compensate it with a bigger $G\big(b(n)\big)$, but in this case the time of the factoring would increase more. Thus we have to change the other parameters, $d(n)$ and $s(n)$ to achieve $o(\ln^4 n)$.

'Strong factorization plus small discriminant' heuristic. As it is clear from the conclusion above, we have to decrease $d(n)$ and $s(n)$ in order to achieve $o(\ln^4 n)$ for (a), (b) and (c/d) too but we have to be careful and keep the expected value of new children, $\lambda = e^{\gamma}\frac{\ln b(n)}{\ln n}e(n)$ above 1. Let us take $d(n) \asymp \ln^2 n/\ln\ln^2 n$, $h(D) = O(\ln n/\ln\ln n)$, $h^*(D) = O\big(\ln n/\ln\ln^{1+\ln 2} n\big)$. Again, the time of the most critical steps D_3, D_5 and F_2 is the following:

D_3:

$$O\big(d(n)m(\ln n)\ln\ln n/\ln^{\ln 2} d(n)\big) = O\big(\ln^3 n \ln\ln\ln n/\ln\ln^{\ln 2} n\big);$$

D_5:

$$O\big(e(n)m(\ln n)\ln n\big) = O\big(\ln^3 n \ln\ln\ln n\big);$$

F_2a:

$$O\big(m(h(D)\ln n)\ln n\big) = O\big(\ln^3 n \ln\ln\ln n\big);$$

F_2b:

$$O\big(m(h^*(D)\ln n)\ln n\big) = O\big(\ln^3 n \ln\ln\ln n/\ln\ln^{\ln 2} n\big).$$

D_4: Like in the previous strategy, in the different cases of factoring we choose $b(n)$ in such a way that the running time of the factoring won't exceed the most critical step, in this case D_5, so we can take the running time of D_5 as the running time of one step. We will determine again here also the total running time of the algorithm for each factoring method.

(a): Trial division. If $b(n)$ is between $\asymp \ln n$ and $\asymp \ln n \ln\ln n \ln\ln\ln n$, then the running time $O\big(e(n)b(n)\ln n\big)$ is between

$$O\big(\ln^3 n/\ln\ln n\big)$$

and

$$O\big(\ln^3 n \ln\ln\ln n\big).$$

As $G(b(n)) \asymp \ln\ln n$ in both cases, $I(b(n)) \asymp \ln n/\ln\ln n$. Then the total running time is

$$O\big(\ln^4 n \ln\ln\ln n/\ln\ln n\big).$$

(b): Batch trial division. If $b(n)$ is between $\asymp \ln n$ and $\asymp \ln^3 n/\ln\ln n$, then the running time $O\left(m\big(e(n)\ln n\big)\left(\dfrac{b(n)}{e(n)\ln n} + \ln\big(e(n)\ln n\big)\right)\right)$ is between

$$O\big(\ln^2 n \ln\ln n \ln\ln\ln n\big)$$

and

$$O\big(\ln^3 n \ln\ln\ln n\big).$$

As $G(b(n))$ and $I(b(n))$ are the same as in (a), the total running time will be also the same.

(c/d): Pollard ρ and $p-1$. If $b(n)$ is between $\asymp \ln n$ and $\asymp \ln^2 n$, then the running time $O\big(e(n)\sqrt{b(n)}m(\ln n)\big)$ is between

$$O\big(\ln^{2.5} n \ln\ln\ln n\big)$$

and

$$O\big(\ln^3 n \ln\ln\ln n\big).$$

As $G(b(n))$ and $I(b(n))$ are the same as in (a) and (b), the total running time will be also the same.

(e): ECM. If

$$b(n) \asymp (\ln n)^{\ln\ln n/\ln\ln\ln^2 n}$$

the same as in the previous strategy then the running time will be the same too, $o(\ln^3 n)$. As $G(b(n)) \asymp \ln\ln^2 n/\ln\ln\ln^2 n$, $I(b(n)) \asymp \ln n\ln\ln\ln^2 n/\ln\ln^2 n$. Then the total running time is

$$O(\ln^4 n\ln\ln\ln^3 n/\ln\ln^2 n).$$

Conclusion. As we can see in this strategy we could decrease the running time to $o(\ln^4 n)$ for each cases by decreasing $d(n)$ and $s(n)$, and λ is still above 1 as $e(n) \asymp s(n)$ thus

$$\lambda = e^\gamma \frac{\ln b(n)}{\ln n} e(n) \asymp e^\gamma$$

in (a), (b), (c/d) and

$$\lambda = e^\gamma \frac{\ln b(n)}{\ln n} e(n) \asymp e^\gamma \ln\ln n/\ln\ln\ln^2 n$$

in (e).

References

[1] **Atkin, A.O.L. and F. Morain,** Elliptic curves and primality proving, *Math. of Computation*, **61** (1993), 29–68.

[2] **Bosma, W., J. Cannon and C. Playoust,** The Magma algebra system. I. The user language, *J. Symbolic Comput.*, **24(3-4)** (1997), 235–265.

[3] **Bosma, W. and A.K. Lenstra,** An implementation of the elliptic curve integer factorization method, in: Bosma, W., van der Poorten, A., (Eds.), *Computational Algebra and Number Theory*, Mathematics and its Applications **325**, Kluwer, 119–136.

[4] **Crandall, R. and C. Pomerance,** *A Computational Perspective*, Springer Verlag, 2002.

[5] H. Cohen, *A course in computational algebraic number theory*, Springer Verlag, 1993.

[6] **Damm, F., F.P. Heider and G. Wambach,** MIMD-Factorisation on Hypercubes, in: De Santis, A., (Ed.), *Advances in Cryptology — EUROCRYPT'94*, Springer-Verlag, LNCS **950**, 1994, 400–409.

[7] **Deuring, M.,** Die Typen der Multiplikatorenringe elliptischer Funktionenkörper, *Abh. Math. Sem. Univ. Hamburg*, **14** (1941), 197–272.

[8] **Farkas, G. and G. Kallós,** Prime Numbers in Generalized Pascal Triangles, *Acta Tech. Jaur.*, **1,1** (2008), 109–118.

[9] **Farkas, G., G. Kallós and Gy. Kiss,** Large Primes in Generalized Pascal Triangles, *Acta Univ. Sapientiae, Inf.*, **3,2** (2011), 158–171.

[10] **Gowers, T.,**, *The Princeton Companion to Mathematics*, Princeton University Press, Princeton and Oxford, 2008.

[11] **Indlekofer, K.-H. and A. Járai,** Largest known twin primes, *Math. Comp.*, (1996).

[12] **Indlekofer, K.-H. and A. Járai,** Largest known twin primes and Sophie Germain primes, *Math. Comp.*, (1996).

[13] **Járai, A. and Gy. Kiss,** Finding suitable paths for the elliptic curve primality proving algorithm, *Acta Univ. Sapientiae, Inf.*, **5,1** (2013), 35–52.

[14] **Kac, M.,** *Statistical Independence in Probability, Analysis and Number Theory*, Wiley, New York, 1959.

[15] **Kirrinnis, P.,** *Zur Berechnung von Partialbruchzerlegungen und Kurvenintegralen rationaler Funktionen, PhD. Thesis*, Rheinischen Friedrich-Wilhelms Universität zu Bonn, 1993.

[16] **Kirrinnis, P.,** Partial Fraction Decomposition in $\mathbb{C}(z)$ and Simultaneous Newton Iteration for Factorization in $\mathbb{C}[z]$, *J.Complexity*, **14** (1998), 378–444.

[17] **Knuth, D.E.,** *The Art of Computer Programming, Vol. 1–3.*, Addison-Wesley, 1968.

[18] **Knuth, D.E.,** *The Art of Computer Programming, Vol. 1–3. Second Edition*, Addison-Wesley, 1981.

[19] **Koppen, V.,** *Probabilistic Model of Traversing Trees – A Simplified View on Elliptic Curve Primality Proving*, Bachelor Thesis, Radboud University Nijmegen, 2014.

[20] **Lenstra, A. K. and H.W. Lenstra Jr.,** Algorithms in Number Theory, in: van Leeuwen, J., (Ed.), *Algorithms in Complexity, Vol. A*, Elsevier, 1990, 673–716.

[21] **Morain, F.,** Distributed primality proving and the primality of $(2^{3539} + +1)/3$, in: Damgård, I.B., (Ed.), *Advances in Cryptology – EUROCRYPT'90*, 1991, 110–123.

[22] **Morain, F.,** Implementing The Asymptotically Fast Version Of The Elliptic Curve Primality Proving Algorithm, *Mathematics of Computation*, **76** (2007), 493–505.

[23] **Pohst, M.E.,** *Computational Algebraic Number Theory*, Birkhäser, 1993.

[24] **Ribenboim, P.,** *The Book of Prime Number Records*, Springer-Verlag, 1989.

[25] **Riesel, H.,** *Prime Numbers and Computer Methods for Factorisation*, Birkhäuser, 1985.

[26] **Schoof, R.,** Counting points on elliptic curves over finite fields, *J. Théor. Nombres, Bordeux*, **7** (1995), 219–264.

[27] **Schoof, R.,** Non-singular plane cubic curves over finite fields, *J. Combinatorial Theory A.*, **46, 2** (1987), 183–211.

[28] **Schönhage, A.,** *The Fundamental Theorem of Algebra in Terms of Computational Complexity – Preliminary Report*, Univ. Tübingen, 1982, 1–74.

[29] **Schönhage, A.,** Equation Solving in Terms of Computational Complexity, in: *Proceedings of the International Congress of Mathematicians*, Berkeley, California, USA, 1986, 131–154.

[30] **Schönhage, A.,** Numerik analytischer Funktionen und Komplexität, *Jber. d. Dt. Math. -Verein*, **92** (1990), 1–20.

[31] **Schönhage, A., A.F.W. Grotefeld and E. Vetter,** *Fast Algorithms: A Multitape Turing Machine Implementation*, B.I.Wissenschaftverlag, Mannheim, 1994.

Wieb Bosma, Eric Cator
Department of Algebra and Topology, Radboud University
Nijmegen
The Netherlands
bosma@math.ru.nl
e.cator@science.ru.nl

Antal Járai and Gyöngyvér Kiss
Department of Computer Algebra
Eötvös Loránd University
Budapest
Hungary
ajarai@moon.inf.elte.hu
kissgyongyver@gmail.com