Fully deterministic ECM

Iram Chelli

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Fully Deterministic ECM

Iram Chelli

N° 7040

Septembre 2009

ISSN 0249-6399 ISRN INRIA/RR--7040--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

Fully Deterministic ECM

Iram Chelli

Thème : Algorithmique, calcul certifié et cryptographie
Équipe-Projet CACAO

Rapport de recherche n° 7040 — Septembre 2009 — 26 pages

Abstract: We present a FDECM algorithm allowing to remove - if they exist - all prime factors less than 2^{32} from a composite input number n. Trying to remove those factors naively either by trial-division or by multiplying together all primes less than 2^{32}, then doing a GCD with this product both prove extremely slow and are unpractical. We will show in this article that with FDECM it costs about a hundred well-chosen elliptic curves, which can be very fast in an optimized ECM implementation with optimized B_1 and B_2 smoothness bounds. The speed varies with the size of the input number n. Special attention has also been paid so that our FDECM be the most implementation-independent possible by choosing a widespread elliptic-curve parametrization and carefully checking all results for smoothness with Magma. Finally, we have considered possible optimizations to FDECM first by using a rational family of parameters for ECM and then by determining when it is best to switch from ECM to GCD depending on the size of the input number n. To the best of our knowledge, this is the first detailed description of a fully deterministic ECM algorithm.

Key-words: factorization, deterministic, elliptic curve, number field sieve.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

ECM complètement déterministe

Résumé : Nous présentons un algorithme FDECM permettant d'extraire - si ils existent - tous les facteurs premiers inférieurs ou égaux à 2^{32} d'un nombre composé donné en entrée n. La méthode naïve par "trial-division" ou par produit suivi de pgcd sont toutes deux extrêmement lentes et inadaptées dans la pratique. Nous montrons dans cet article qu'avec FDECM cela coûte moins de 100 courbes elliptiques bien choisies, ce qui peut être très rapide avec une implantation d'ECM et des bornes B_1, B_2 bien optimisées. Le temps d'exécution dépend de la taille du nombre n en entrée. Nous avons pris soin de rendre notre algorithme le plus indépendant possible d'une implémentation particulière par le choix de la paramétrisation des courbes utilisées et la vérification systématique avec Magma. Finalement, nous envisageons differentes possibilités d'optimisation à FDECM en utilisant une famille rationnelle de paramètres pour ECM et en déterminant à quel moment le passage au GCD devient moins coûteux qu'ECM et ce pour différentes tailles du nombre n en entrée. C'est, à notre connaissance, la première description détaillée d'un algorithme d'ECM complètement déterministe.

Mots-clés : factorisation, déterministe, courbe elliptique, crible algébrique.

Contents

1 Introduction **4**

2 Elliptic curves on a finite field **4**
2.1 Definition 4
2.2 Curve equations 5
2.2.1 Montgomery's form 5
2.3 Group structure 5
2.4 Cardinality of the group $E(\mathbb{F}_q)$ 5
2.5 Point addition laws 5
2.5.1 Weierstrass form 5
2.5.2 Montgomery's form 6
2.6 Computation of kP 6
2.6.1 Double and add 7
2.6.2 Lucas chains 7

3 The ECM method **8**
3.1 Smoothness criteria 8
3.2 ECM algorithm 8
3.2.1 Stage 1 9
3.2.2 Stage 2 9
3.3 Brent-Suyama's parametrization 9

4 Building σ-chains that yield all primes up to a given bound B **10**
4.1 Using ECM with prime input numbers 10
4.2 ECM Testing implementation 10

5 Choosing the best parameters for ECM **10**
5.1 The influence of B_1, B_2 bounds 10
5.1.1 Most efficient B_1, B_2 bounds on a sample of primes of given bitlength 10

6 Primes found with unsmooth curve order **11**
6.1 ECM implementation Optimizations and non totally deterministic behavior 11
6.2 Testing found primes for smoothness of curve order 11
6.3 Considering starting point order instead of only curve order . . . 12

7 Taking advantage of known curve order divisors **12**
7.1 Curves with torsion subgroup over $\mathbb{Q}$ of order 12 or 16 and known initial point 12
7.1.1 Reduction of curves with known torsion subgroups over $\mathbb{F}_p$ 12

8 Extension to higher powers **12**
8.1 Using optimal B_1, B_2 bounds for each subset of primes 13
8.2 Implementation-independent and optimized σ-chains with Magma 13
8.3 Building σ-chains for sets of non-consecutive primes 14
8.3.1 Implementation-independent σ chain for the RSA200 subset 16

9 Optimizations for DECM **16**
9.1 Using Rational values for σ 16
9.2 Switching from ECM to gcd or trial division and when 16

10 Appendix: **19**

1 Introduction

The Elliptic Curve Method (ECM) is currently the best-known general-purpose factorization algorithm for finding "small" prime factors in numbers having more than 200 digits (here "small" means up to 67 digits, which is the current ECM record [3]). It has a wide range of applications, in particular it is widely used in the current implementation of the Number Field Sieve (NFS). However ECM is a randomized algorithm: its success for a given number and given parameters depends on the choice of an initial seed. More precisely when running one single ECM curve it is a Monte Carlo algorithm: the running time is deterministic, but the success probability is random (0 or 1). If running several curves until a factor is found, we get a Las Vegas algorithm.
In some applications, one would like to be able to remove all prime factors up to a given bound M, from a given input number n, which might have thousands of digits, after a small number of operations depending only on n. This is what we call a *deterministic* factoring algorithm.
Our goal is to provide a fully deterministic ECM. A bound M being given, after some preprocessing work involving only M, an algorithm is issued, which given an integer n, outputs all prime factors of n less than M (and possibly larger prime factors). Of course the goal is to minimize the number of arithmetic operations (modular additions, multiplications and gcds) involving n, either in the worst-case, or in the average (considering all primes less than M as equiprobable).

2 Elliptic curves on a finite field

2.1 Definition

The equation of an elliptic curve defined over the field of real numbers can be written in the simple form of Weierstrass:

$$y^2 = x^3 + Ax + B$$

where the coefficients A and B are taken in the base field.
The discriminant of the curve $\Delta = -16(4A^3 + 27B^2)$ needs be non-zero, the preceding equation then defines a non-singular curve. The points of the curve are all those whose coordinates verify the equation, plus a point at infinity. This point at infinity is essential for it will be the neutral element, the "zero" for the addition law of points of the curve. Intuitively, it can be imagined as the point at the intersection of all vertical lines.

2.2 Curve equations

2.2.1 Montgomery's form

Elliptic curves in Weierstrass form are not very efficient in terms of computational cost. The Montgomery form is used instead [7]. An Elliptic curve E in Montgomery's form has equation:

$$by^2 = x^3 + a\,x^2 + x.$$

The condition that the curve be nonsingular is that $\delta = 4/b^6 - a^2/b^6 \neq 0$ with $b \neq 0$. It then suffices that $a^2 \neq 4$ and $b \neq 0$. In homogenous form we have the equation

$$by^2z = x^3 + a\,x^2z + xz^2.$$

Points on the curve are represented in projective coordinates $(x : y : z)$ so as to avoid inversions which are computationally costly and we disregard the y-coordinate simply noting $P = (x :: z)$ which simply means that we identify a point with its opposite. A point $(x : y : z)$ in projective coordinates can be converted to $(x/z, y/z)$ in affine coordinates whenever $z \neq 0$ and a point (x, y) in affine coordinates is simply the point $(x : y : 1)$ in projective coordinates. The triplets $(x : y : z)$ with $z = 0$ do not correspond to any affine solutions and are the point at infinity of the curve $(0 : 1 : 0)$. Montgomery's form can be obtained from Weierstrass form by the change of variables $X = (3x+a)/3b, Y = y/b, A = (3 - a^2)/3b^2, B = (2a^3/9 - a)/3b^2$.

2.3 Group structure

The set $E(K)$ of points of an elliptic curve E has an abelian group structure for a law $\oplus$ for which point O at infinity is the neutral element, and such that the inverse of a point $P = (x, y)$ of E is $-P = (x, -y) \in E$. Only associativity is not immediate to prove. This group law can be observed geometrically, which allows for graphical construction of the sum of two points on an elliptic curve.

2.4 Cardinality of the group $E(\mathbb{F}_q)$

Let N be the number of rational points on an elliptic curve E on a q-element finite field, we have

$$|N - (q + 1)| \leq 2\sqrt{q}.$$

This is an useful result for computation of the number of rational points on an elliptic curve is hard when q is a large integer, this theorem by Hasse gives an interesting approximation of the group order of the elliptic curve.

2.5 Point addition laws

2.5.1 Weierstrass form

In order to work on elliptic curves, we need to be able to compute point additions. We will only consider finite fields of characteristic different of 2 and 3. Let E be an elliptic curve in Weierstrass form on a finite field of characteristic

different of 2 and 3 and let $P = (x_p, y_p)$ and $Q = (x_q, y_q)$ be two points on that curve and $R = (x_r, y_r)$ the point such that $P + Q = R$.
Let $\alpha = \dfrac{y_q - y_p}{x_q - x_p}$ and $\beta = \dfrac{3x_p^2 + a}{2y_p}$.

First case: Distinct and non-opposite points If $x_p \neq x_q$ that is, P et Q are different and not the inverse of one another, then we have

$$\begin{cases} x_r = \alpha^2 - x_p - x_q \\ y_r = -y_p + \alpha(x_p - x_r). \end{cases}$$

Second case: Opposite points If $x_p = x_q$ and $y_p \neq y_q$ then $R = O$.
In that case we necessarily have $y_p = -y_q$ because of the curve equation. Points $P = (x_p, y_p)$ and $Q = (x_q, -y_p)$ are symmetric with respect to the x-axis. The point R then is point at infinity.

Third case: Point Doubling with nonzero y coordinate If $x_p = x_q$ and $y_p = y_q \neq 0$ that is $P = Q$ with $y_p \neq 0$, then $R = P + P = 2P$ and

$$\begin{cases} x_r = \beta^2 - 2x_p \\ y_r = -y_p + \beta(x_p - x_r). \end{cases}$$

Fourth case: Point Doubling with zero y coordinate If $x_p = x_q$ and $y_p = y_q = 0$ that is $P = Q$ with $x_p = 0$, then $R = O$.
The point is on the x-axis. The tangent to the curve at this point is vertical. The point $P + P = 2P$ is thus point at infinity.

2.5.2 Montgomery's form

The addition and doubling laws are as follows: Let $P = (x_P :: z_P)$ and $Q = (x_Q :: z_Q)$ be two distinct points and let $P - Q = (x_{P-Q} :: z_{P-Q})$ be their difference; then their sum $P + Q$ is computed by the following formula,

$$x_{P+Q} = 4z_{P-Q} * (x_P x_Q - z_P z_Q)^2, \qquad z_{P+Q} = 4x_{P-Q} * (x_P z_Q - z_P x_Q)^2.$$

The doubling is computed as follows:

$$x_{2P} = (x_P^2 - z_P^2)^2, \qquad z_{2P} = 4x_P z_P[(x_P - z_P)^2 + 4d\, x_P z_P].$$

where $d = (a + 2)/4$ and a is the coefficient in the elliptic curve equation in Montgomery's form.

2.6 Computation of kP

The ECM algorithm needs computation of point kP from a given integer k and point P. We make use of the forementioned point-addition formulas.

2.6.1 Double and add

Except for the doubling which is simply adding a point to itself, we have no "multiplication by an integer" formula. Point nP is computed by means of repeated addings and doublings from the binary expansion of integer n. This is called the "double-and-add" algorithm.

Algorithm DOUBLE AND ADD (Left to Right):

INPUT:	Point P from $E(\mathbb{F}_p)$ and $n \geqslant 1$
OUTPUT:	Point $R = nP$
1.	Set $Q = P$
2.	loop for $i = (\lceil \log_2(n) \rceil - 2)$ down to 0
	set $Q = 2Q$
	if $\mathrm{Bit}_i(n) = 1$ set $Q = Q + P$
3.	return Q (which equals nP)

Example: Compute addition and doubling formula for $k = 100 = (1100100)_2$

$$100P = 2(2(P + 2(2(2(P + 2P)))))$$

Computing $100P$ takes 6 doublings (length minus one of the binary expansion of 100) and 2 additions (number minus one of ones in the expansion).

2.6.2 Lucas chains

Lucas chains are a special case of addition chains in which the sum of two terms can appear if and only if their difference also appears. This condition is needed when the point addition laws in Montgomery homogeneous coordinates are to be used. It is still possible to "double" a point (i.e., computing $\pm 2P$) and if the difference $\pm(P - Q)$ between two points is known then we can compute their sum $\pm(P + Q)$: this operation is called pseudo-addition. To compute the multiple of a point we need to find chains of doubling and pseudo-additions which is a special case of addition chains called "Lucas Chains" [6]. For instance

$$1 \to 2 \to 3 \to 4 \to 7 \to 10 \to 17$$

is a Lucas chain for 17. One way to find such chains is to note that if we know $[n]P$ and $[n+1]P$ then we can compute $[2n]P$, $[2n+1]P$ and $[2n+2]P$. Hence we have binary chains: at each step we choose the point to double according to the binary expansion of the multiplier. For instance the following chain is the binary chain for 17:

$$1 \to 2 \to 3 \to 4 \to 5 \to 8 \to 9 \to 17.$$

This example shows that binary chains are not necessarily the shortest chains of doubling and pseudo-addition. Montgomery's PRAC is an heuristic algorithm designed to find short Lucas chains [6]. Description of the algorithm is beyond the scope of this work, but the two following examples show more cases where we have a Lucas chain that is shorter than its binary counterpart:

$$k = 9 \qquad 1 \to 2 \to 3 \to 4 \to 5 \to 9$$

$$1 \to 2 \to 3 \to 6 \to 9$$

$k = 13$ $\quad 1 \to 2 \to 3 \to 4 \to 6 \to 7 \to 13$

$\quad 1 \to 2 \to 3 \to 5 \to 8 \to 13.$

3 The ECM method

ECM is a generalization of Pollard's $p-1$ algorithm: instead of working in $\mathbb{F}_p^*$, we work in the group of points of an elliptic curve.
The number to be factored being n, we work over $\mathbb{Z}/n\mathbb{Z}$ as if it were a field. The only operation that might fail is ring inversion which is calculated using the Euclidean algorithm. If an inversion fails, then n is not coprime with that number and we find a factor of n by computing their gcd.

3.1 Smoothness criteria

Let n be a positive integer, with prime factorization: $n = \prod_{i=1}^{m} p_i^{\alpha_i}$ and let B be a positive integer.

n is said to be **B-smooth** if all of its prime factors p_i verify $p_i \leqslant B$.
It is said to be **B-powersmooth** if for all i we have, $p_i^{\alpha_i} \leqslant B$.

For example $72900000000 = 2^8 3^6 5^8$ is 5-smooth since 5 is its largest prime factor, and is 5^8 - powersmooth.

An integer n will be said to be **(B_1, B_2)-smooth** if it is B_1-powersmooth for all but it's largest prime factor p_m, and we have $\alpha_m = 1$ and $p_m \leqslant B_2$.

3.2 ECM algorithm

The ECM method starts by the choice of a random nonsingular elliptic curve E over $\mathbb{Z}/n\mathbb{Z}$, n being the number to be factored, and a point P on it. We seek a positive integer k such that $[k]P = \mathcal{O}$ modulo an (unknown) prime divisor p of n but not modulo n. Then, in the computation of $Q = [k]P \bmod n$ the attempted inversion of an element not coprime to n reveals the factor p by simple computation of gcd(x_Q,n). For this to happen k needs to be chosen a multiple of $g_p = \#\mathcal{E}(\mathbb{F}_p)$ which is also unknown. Notice that it is possible that k also happens to be a multiple of the group order g_q for another prime factor q, especially if k is chosen too high or when group orders g_p and g_q are close. In that case the GCD will be a composite factor pq of n and in the worst case the GCD will be the input number n itself. The ECM method has two different stages: Stage 1 will compute $Q = [k]P$ and be successful if the order g of the curve is B_1-powersmooth (i.e., multiple of prime powers each less than B_1) for some bound B_1. Stage 2 is an improvement of the initial ECM algorithm to catch primes p for which $\#\mathcal{E}(\mathbb{F}_p)$ is close to being B_1-powersmooth, that is, the product of a B_1-powersmooth number by just one prime cofactor exceeding B_1 and less than a B_2 bound, that is, (B_1,B_2)-smooth. If this doesn't yield a factor, we can still try another elliptic curve which order might this time be B_1-smooth, a possibility not available in Pollard $p-1$ for which the group order

is fixed.
ECM is a probabilistic algorithm with heuristic expected running time to find a factor p of a number n

$$O(L(p)^{\sqrt{2}+o(1)}M(\log(n)))$$

where $L(p) = e^{\sqrt{\log(p)\log(\log(p))}}$ and $M(\log(n))$ is the complexity of multiplications modulo n. The complexity of ECM is dominated by the size of the smallest factor p of n rather than the size of the number n to be factored. However, ECM does not always find the smallest factor.

3.2.1 Stage 1

Stage 1 will yield a prime factor if the order g of the elliptic curve E is B_1-powersmooth. We compute $Q = [k]P$ where $k = \prod_{\pi \leqslant B_1} \pi^{[\log(B_1)/\log(\pi)]} = \mathrm{lcm}\,(1, 2, \ldots, B_1)$ so that all B_1-powersmooth numbers divide k. The main cost of stage 1 is the elliptic curve arithmetic.

3.2.2 Stage 2

When the order g is not B_1-smooth because of a single prime factor q above B_1, a bound $B_2 > B_1$ is set so that we have a chance that this prime factor q is below $B2$ and thus g is (B_1, B_2)-smooth. Prime q will then be found in stage 2.

3.3 Brent-Suyama's parametrization

This elliptic curve parametrization uses a single integer $\sigma > 5$, it is simple and of widespread use, this choice will therefore allow for easy reproduction of the results obtained, which is of crucial importance when performing deterministic ECM. A random integer $\sigma > 5$ is chosen. Here we will limit ourselves to a random 64-bit value. From this parameter we then compute:
$u = \sigma^2 - 5$, $v = 4\sigma$,
$x_0 = u^3 \pmod n$, $z_0 = v^3 \pmod n$,
$a = (v-u)^3(3u+v)/(4u^3v) - 2 \pmod n$, $b = u/z_0$ and,
$y_0 = (\sigma^2-1)(\sigma^2-25)(\sigma^4-25)$.
Let p be a prime factor of n, Hasse's theorem states that the order g of an elliptic curve E over $\mathbb{F}_p$ satisfies

$$|g - (p+1)| < 2\sqrt{p}.$$

When curve coefficients a,b vary, g essentially behaves as a random integer in the interval $[p+1-2\sqrt{p},\ p+1+2\sqrt{p}]$, with some additional conditions imposed by the type of curve chosen. Suyama's and Montgomery's form parametrizations both ensure 12 divides g over $\mathbb{F}_q$ [9]. This is of interest since this increases the probability that g is indeed smooth, $g = 12 * g'$and thus affects the probability of success of ECM for fixed given parameters. Furthermore, the known factor 12 can be further taken advantage of if we relax the smoothness criterion for stage 1 by increasing ν_2, the maximum power of 2 such that $2^{\nu_2} \leqslant B_1$ by 2, and ν_3, the maximum power of 3 such that $3^{\nu_3} \leqslant B_1$ by one.

4 Building σ-chains that yield all primes up to a given bound B

4.1 Using ECM with prime input numbers

Normally, ECM is run on a *composite* input number n that we are trying to factor. Here what we want to determine is whether a given prime p will be found by ECM when it is used on an input number n such that $p|n$, but the cofactor is undetermined. We thus run ECM in an unusual way, where the input number p is prime, and ECM returns p if $g = \#\mathcal{E}(\mathbb{F}_p)$ is $(B1, B2)$-smooth and fails if not. This contrasts with the usual ECM usage where having the input number returned is not satisfactory as this means no factor has been found.

We will begin by running tests on the primes up to $B = 2^{32}$. Our goal is first to find a set of elliptic curves or σ-chain that will ensure all of those primes are found. We will then try to optimize both the average cost of finding a prime and the worst case cost. To achieve this we first construct a precomputed table of all primes up to 2^{32}.

4.2 ECM Testing implementation

To determine which primes are found by a given curve with given parameters, we use a testing program "ecm_check" based on GMP-ECM, a free and well optimized ECM implementation, which outputs primes not found by ECM for given B_1, B_2, and σ. We will later discuss wether this methodology can be relied upon in the context of the present work, which aims at performing *deterministic* ECM.

5 Choosing the best parameters for ECM

5.1 The influence of B_1, B_2 bounds

The choice of B_1 and B_2 bounds is of great importance since it impacts both the running time of ECM and the number of primes found by the algorithm. When chosen too high ECM gets slower, and when chosen too low fewer primes are found. The parameters thus need to be fine-tuned for optimal results. Of course we want to minimize the running time while simultaneously maximizing the number of primes found. We determine the best B_1, B_2 values that minimize the time over found primes ratio or time per prime hit.

5.1.1 Most efficient B_1, B_2 bounds on a sample of primes of given bitlength

We have determined experimentally which are the most efficient B_1, B_2 bounds on a sample of primes of given bitlength. We have tested the million primes just below 2^α for $\alpha = 27..32$.

The best time per prime is achieved for the following B_1, B_2 bounds for the million primes below:

$\mathbf{2^{27}}$: $\boldsymbol{B_1 = 140}$, $\boldsymbol{B_2 = 7600}$, time(sec.): 29.59 hits: 247667 Ratio: $\mathbf{119\mu s}$.
$\mathbf{2^{28}}$: $\boldsymbol{B_1 = 140}$, $\boldsymbol{B_2 = 9200}$, time(sec.): 31.44 hits: 208562 Ratio: $\mathbf{151\mu s}$.
$\mathbf{2^{29}}$: $\boldsymbol{B_1 = 220}$, $\boldsymbol{B_2 = 10400}$, time(sec.): 40.6 hits: 211617 Ratio: $\mathbf{192\mu s}$.
$\mathbf{2^{30}}$: $\boldsymbol{B_1 = 220}$, $\boldsymbol{B_2 = 8800}$, time(sec.): 38.71 hits: 163260 Ratio: $\mathbf{237\mu s}$.
$\mathbf{2^{31}}$: $\boldsymbol{B_1 = 260}$, $\boldsymbol{B_2 = 11600}$, time(sec.):48.63 hits: 161424 Ratio: $\mathbf{301\mu s}$.
$\mathbf{2^{32}}$: $\boldsymbol{B_1 = 260}$, $\boldsymbol{B_2 = 11600}$, time(sec.):48.83 hits: 130144 Ratio: $\mathbf{375\mu s}$.

6 Primes found with unsmooth curve order

6.1 ECM implementation Optimizations and non totally deterministic behavior

We now have to take into consideration that the ECM factorisation program "ecm_check" may find primes p for which $E(p)$ is not B_1-B_2 smooth. This is due to the fact that when adding and doubling points on the curve it may happen that the addition of two points that we think are distinct fails because they are actually the same point and thus the doubling formula should have been used. This happens because in this ECM implementation points are not systematically tested to be different for the sake of speed improvement. Although this is generally not a problem when factoring with ECM because it finds extra primes to those for which the curve or the point order is smooth, it has to be avoided for a deterministic algorithm because additions and doublings are optimized and computed from a Lucas-chain optimization algorithm. As we have seen before, several Lucas-chains exist and we have absolutely no guarantee that two different implementations will compute the same chain nor find the same extra primes. We thus check that the orders of the curves modulo the primes found are indeed smooth, and haven't been found because of implementation-specific mechanisms. For $\sigma = 11$ we obtain 4691719 out of 7603553 primes for which the curve order is not smooth for bounds B1=315 and B2=5355 and shouldn't be hit by the curve when running ECM, while using the ecm_check program with the same curve and parameters 4574155 primes are reported not found. That is 117564 additional primes found by the ECM program and that are very implementation-dependent. There exists a possibility though that these primes get found by subsequent curves but this has to be carefully checked.
Actually, prime $p = 95062837$ doesn't get found by any of the subsequent curves of the second optimized σ-chain until second to last element and one can verify that $g = \#E(\mathbb{F}_p)$ is never $(B1, B2)$-smooth for any of the preceding sigmas and $(B1, B2)$ bounds. In addition, another case can be encountered where a prime is found while the order of the curve is not smooth modulo that prime. This can happen during initialization of the Elliptic curve or the starting point on it. This is the case for $\sigma = 11$ and $p = 31$ for example. However in this case the prime will be considered found by ECM.

6.2 Testing found primes for smoothness of curve order

In order to have the most implementation-independent results on ECM, we will check the σ-chains we obtained with ecm-check with Magma [2] considering smoothness of the curve order $g = \#E(\mathbb{F}_p)$ for every prime found p to make

sure it will indeed be found by any relatively "standard" ECM implementation since this would ensure it was really found by the ECM mechanism itself and not by any optimization or implementation side-effect.

6.3 Considering starting point order instead of only curve order

Still, we can do slightly better by considering the starting point order instead of the curve order, since this is a divisor of the curve order it has significantly higher probability to be smooth with the same B_1, B_2 parameters. We are allowed to do this without sacrificing to generality because the starting point P is fully determined by the choice of σ. Noting g the order of the starting point P, the multiplier, noted e, is precomputed from the chosen B_1 value of stage one. We then compute the order of the point yielded at the end of stage one $e\,P$, which is $g' = g/(g, e)$. If $g' = 1$ then stage one was successful. Otherwise if g' is prime and $B1 < g' \leqslant B2$ then it will be found in stage two. This algorithm has been implemented as a Magma script which outputs primes for which the starting point order does not verify any of the above propreties. The output file it yields is composed of the primes not found by a standard implementation of ECM.

7 Taking advantage of known curve order divisors

7.1 Curves with torsion subgroup over $\mathbb{Q}$ of order 12 or 16 and known initial point

Montgomery [8] showed how to select a curve whose torsion subgroup over $\mathbb{Q}$ has order 12 or 16 and with known initial point. Furthermore, Mazur [5] showed that this is the largest possible torsion subgroup for and elliptic curve over $\mathbb{Q}$.

7.1.1 Reduction of curves with known torsion subgroups over $\mathbb{F}_p$

If a curve E has torsion subgroup of order 12 (resp. 16) over $\mathbb{Q}$ then its reduction E_p mod p also has order divisible by 12 (resp. 16) unless p divides the discriminant of the curve i.e., E_p is singular mod p. Therefore for all but finitely many primes p the reduced curve E_p will have a known divisor. ECM will work if $\#E_p/12$ (resp $\#E_p/16$) is sufficiently smooth. The higher the known divisor, the more it is likely to be smooth. Thus torsion 16 curves are considered "better" for ECM.

8 Extension to higher powers

Using the same strategy as before, we extend this work to primes up to 2^{32}. We do this by intervals of same bitlength which are more convenient to work with. Excluding the first one, the size of each interval is approximately the double of the preceding. With a fixed cost-per-prime, the ECM running time then also would approximately double for each interval.

We count 98182656 primes in the interval $2^{31} - 2^{32}$, 50697537 primes in the interval $2^{30} - 2^{31}$, 26207278 primes in the interval $2^{29} - 2^{30}$, 13561907 primes in the interval $2^{28} - 2^{29}$, 7027290 primes in the interval $2^{27} - 2^{28}$, and 7603553 primes below 2^{27}.

But according to the figures shown in the previous section, the cost-per-prime is not fixed as the ECM running time increases with the size of the input factor and size of $B_1 - B_2$ smoothness bounds, and we observe an increase of between 24 and 28 percent for optimal $B_1 - B_2$ for each subsequent interval. When extending to higher powers, processing primes in the next higher binary interval induces an increase in time by a 2.5 multiplicative factor on average compared with the previous, which is significatively more than simply doubling for an interval twice as large.

8.1 Using optimal B_1, B_2 bounds for each subset of primes

We now compute σ-chains with the optimal B_1, B_2 bounds we have determined in section 5.1.1.

- Primes p up to 2^{27}: $B_1 = 140$, $B_2 = 7600$:

  We have built a 50-element σ-chain we have built that allows to find all primes up to 2^{27}.

- Primes p, $2^{27} < p < 2^{28}$: $B_1 = 140$, $B_2 = 9200$:

  We have built a 61-element σ-chain we have built that allows to find all primes p, $2^{27} < p < 2^{28}$.

- Primes p, $2^{28} < p < 2^{29}$: $B_1 = 220$, $B_2 = 10400$:

  We have built a 62-element σ-chain we have built that allows to find all primes p, $2^{28} < p < 2^{29}$.

- Primes p, $2^{29} < p < 2^{30}$: $B_1 = 220$, $B_2 = 8800$:

  We have built a 86-element σ-chain we have built that allows to find all primes p, $2^{29} < p < 2^{30}$.

- Primes p, $2^{30} < p < 2^{31}$: $B_1 = 260$, $B_2 = 11600$:

  We have built a 92-element σ-chain we have built that allows to find all primes p, $2^{30} < p < 2^{31}$.

- Primes p, $2^{31} < p < 2^{32}$: $B_1 = 260$, $B_2 = 11600$:

  Table 5 exhibits a 120-element σ-chain we have built that allows to find all primes p, $2^{31} < p < 2^{32}$.

8.2 Implementation-independent and optimized σ-chains with Magma

By implementation-independent we mean a σ-chain that has been verified to find all target primes relying solely on the fixed smoothness criterion and fixed B_1,B_2 bounds. It is thus fully optimization and implementation-independent. In addition, we have also built each chain as an extension of the preceding using

non-decreasing B_1,B_2 bounds so that for example a chain which finds all primes p such that $2^{31} < p < 2^{32}$ actually finds all primes $p < 2^{32}$, which is the goal which we had set to achieve, finding all primes up to a given bound M. Chains have been optimized using many different tools: shell scripts, unix tools such as wc, grep, cut; Magma scripts and ecm_check. We begin by $\sigma = 11$, then the rational σs. Then a script randomly generates σs, we run a loop and we keep generating σs until a desired threshold is attained then we save the σ value. The output file (primes not found) is then taken as input and we continue optimizing next σ value until all input primes have been found.

- Primes p up to 2^{27}: $B_1 = 140$, $B_2 = 7600$

  We have built an implementation-independent 52-element σ-chain which allows to find all primes up to 2^{27}. This is two additional curves compared to previously found chain.

- Primes p, $2^{27} < p < 2^{28}$: $B_1 = 140$, $B_2 = 9200$

  We have built an implementation-independent 67-element σ-chain which allows to find all primes $p < 2^{28}$. This is six additional curves compared to previously found chain. Results shown are for primes $2^{27} < p < 2^{28}$.

- Primes $2^{28} < p < 2^{29}$: $B_1 = 220$, $B_2 = 10400$

  We have built an implementation-independent 68-element σ-chain which allows to find all primes $p < 2^{29}$. This is six additional curves compared to previously found chain. Results shown are for primes $2^{28} < p < 2^{29}$.

- Primes $2^{29} < p < 2^{30}$: $B_1 = 220$, $B_2 = 10400$

  We have built an implementation-independent 90-element σ-chain which allows to find all primes $p < 2^{30}$. This is four additional curves compared to previously found chain. Results shown are for primes $2^{29} < p < 2^{30}$.

- Primes $2^{30} < p < 2^{31}$: $B_1 = 260$, $B_2 = 11600$

  We have built an implementation-independent 96-element σ-chain which allows to find all primes $p < 2^{31}$. This is four additional curves compared to previously found chain. Results shown are for primes $2^{30} < p < 2^{31}$.

- Primes $2^{31} < p < 2^{32}$: $B_1 = 260$, $B_2 = 11600$

  Table 6 exhibits an implementation-independent 124-element σ-chain we have built which allows to find all primes $p < 2^{32}$. This is four additional curves compared to previously found (unverified) chain. The figures shown "primes not hit" are for primes p, $2^{31} < p < 2^{32}$.

- Primes $p < 2^{32}$

  Table 7 summarizes the results found in each interval for the 39 first σ values. This table gives a synoptic view of the results achieved.

8.3 Building σ-chains for sets of non-consecutive primes

We now have a set of curves or σ-chains that guarantees to find all primes up to 2^{32}. We now consider the case where the primes we are looking for are not consecutive. For example in NFS, given an algebraic polynomial $f(x)$, the

primes appearing in the factorization of $f(a/b)$ are those for which $f(x)$ has a root mod p, which is a subset of all possible primes. It is thus possible to adapt the algorithm for a particular polynomial $f(x)$. Here we are going to focus on the polynomial that was used in the factoring of RSA200 by GNFS. The polynomial is : $f(x) = X5 * x^5 + X4 * x^4 + \ldots + X0$, where the coefficients are :
$X5 = 374029011720$,
$X4 = 2711065637795630118$,
$X3 = 194000719431775138658927 14$,
$X2 = -338034706092024130946804623 60399$,
$X1 = -120887311888241287002580512992469303610$,
$X0 = 3876720300079932118978295952993877119517 0960$.

This polynomial does have a root for a subset of 128740271 elements of the set of 203280221 primes less than 2^{32}. We will refer to these subsets as p232 and RSA200 from now on. Although the RSA200 subset has a significantly smaller cardinal being about 63,3 percent of the size of p232, this does not lead to a shorter chain than the one built previously for p232 as all 124 sigma values need be taken to find all primes of RSA200. This is not really a suprise since Table 8 shows that each curve keeps finding about the same percentage of primes with RSA200 as it did with with p232. Thus the total number of curves to cover both sets remains almost unchanged. If the percentage of primes not found is noted p, then we need about k curves to find N primes:

$$N\,p^k = 1$$
$$N = (1/p)^k$$
$$\log(N) = k\,\log(1/p)$$
$$k = \log(N)/\log(1/p)$$

With this formula the expected number of curves to find 98182656 primes would be:

$$\log(98182656)/\log(100/87) \approx 122.$$

which is very close to the figure of 124 we achieved [Table 6].

So if we have a strictly smaller set of cardinal aN with $a < 1$ we get:

$$\log(aN)/\log(1/p) = \log(N)/\log(1/p) + \log(a)/\log(1/p).$$

The theorical gain is thus $\log(a)/\log(1/p)$:

$$\log(63.3/100)/\log(100/87) \approx -3.23.$$

So we can expect a theoretical improvement of about 3 curves for the RSA200 subset. Possible improvements on the σ-chain will be discussed in the next section.

8.3.1 Implementation-independent σ chain for the RSA200 subset

Table 8 exhibits an implementation-independent 124-element σ-chain we have built which allows to find all primes in the RSA200 set such that $p < 2^{32}$. The figures shown "primes not hit" are for primes p, $2^{31} < p < 2^{32}$. As indicated above, this chain is the same we had built for Table 6 and all of its elements need to be used.

9 Optimizations for DECM

9.1 Using Rational values for σ

As we have already seen, $\sigma = 11$ appears to perform noticeably better than random curves for finding primes. Other curves of the infinite family of $\sigma = 11$ perform similarly with an above-average performance. They have rational sigmas. We have gathered 18 curves with a relatively small denominator [1]. Another infinite family of curves, those for $\sigma = 9/4$ appear to show the same type of behaviour. The main difference being that the first family appears to favour primes p congruent to 1 mod 4 whereas the second favours primes in the other congruence class, 3 mod 4. We have gathered 7 curves of the second family with not too large denominator. Adding the value 11 to this chain, this results in an optimized 26-element rational chain which gives the best results that we have achieved so far (Table 9). The chain is the following:

11, 11/13, 15/47, 2171/719, 239/241, 149763/17279, 46079/40321,
6733681/9843601, 105169691/3623761, 91420289327/29689850877,
1067200567947371/146190423103201, 1369773196877505024 3/43620080634846719,
2016229864862551391632811/63340029429527474146223 9,
4421288286863274768090606374 87/68777227360771197231889013763,
29354031349/30361489933, 120433561856161/94676700609121,
2019133253589073919/390740626527360 7681,
2056082418138188582375 99/22218381066763702498704 1,
1614144431848627784468501107 31/11649720428741196519803318017 3,

9/4, 121/169, 25921/144, 5248681/4020025, 6197310729/2620825636,
9088783887121/19343046113329, 451586581827690241/10030226005832256.

9.2 Switching from ECM to gcd or trial division and when

As can be seen in the previous tables, the last third of the curves invariantly accounts for a very small proportion ,i.e., less than 1/1000 th, of the primes found. It thus would be less costly to find these by taking a gcd of the number we want to factor with the product of those primes or simply trial-divide the input number by the remaining primes after ECM has been performed. Still it has to be determined which is faster depending on the size of the input number to factor and when it is best to switch from ECM to gcd or trial division, that is, the threshold after which running an ECM curve has a greater cost per prime than performing a gcd or trial division. We will time this with three differently sized input numbers: a 100-digit prime, a 1000-digit prime and finally a 10000-digit prime noted N100, N1000, and N10000 respectively. We have implemented the GCD and trial division in C using the GMP library. The huge

product of several millions of primes needed for the GCD first done naively was excessively slow and thus has been modified to be computed by means of a product tree. Timings for ECM have been measured with GMP-ECM [4] and not with ecm_check since the latter is limited to two machine words for input. As GMP-ECM was designed and optimized for a different usage with substantially larger B_1,B_2 bounds to find factors well larger than 2^{32}, it induces some overhead when used with such unusually small B_1,B_2. We can thus expect ECM to be made significantly faster than the results shown with some special-purpose ECM implementation or if extending ecm_check to support several more machine words input. The timings for trial division are shown in Table 1. The cost per prime for GCDs is invariably inferior to that of trial division. We will thus switch to GCD after ECM. Performing a GCD presents the minor drawback of potentially returning a composite number product of several of the prime divisors of input number n. This is not a concern since in this work our goal is simply to remove factors from the input number up to a bound N and not returning a factorization. Of course the following results are very platform and implementation dependent and have to be fine-tuned for each user's particular situation. The testing platform is an Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz.

input number size in digits	time per division 10^{-8} seconds
100	3.98
1000	27.7
10000	2730

Table 1: Costs per prime of trial division for different input number sizes

σ-chain			Time			
curve	not found	found by curve	ECM 10^{-3}s.	per prime 10^{-8}s.	GCD 10^{-2}s.	per prime 10^{-8}s.
26	2956228	482448	4.135	0.86	6.59	2.23
27	2541845	414383	"	1.00	5.65	2.22
28	2186403	355442	"	1.16	4.93	2.25
29	1882720	303683	"	1.36	4.19	2.22
30	1622204	260516	"	1.59	3.59	2.21
31	1397929	224275	"	1.84	3.22	2.30
32	1204092	193837	"	2.13	2.66	2.21
33	1038338	165754	"	2.49	2.3	2.21
34	895161	143177	"	2.89	1.94	2.17
35	772361	122800	"	3.37	1.69	2.19
36	666561	105800	"	3.91	1.41	2.11
37	575642	90919	"	4.55	1.2	2.08
38	500127	75515	"	5.47	1.03	2.06
39	432032	68095	"	6.07	0.86	1.99

Table 2: Compared costs-per-prime of ECM and GCD with implementation-independent σ-chain: 100 digit prime input, $B_1 = 260$, $B_2 = 11600$

Table 2 shows that for a **100 digit** input number, curve 33 is more expensive than gcd we stop at **curve 32** and then switch to gcd. The precomputed product of primes not found output by curve 32 is saved to file in GMP RAW format, since this will be used when computing their gcd with any given 100 digit input number. The file requires 4.6MB of disk space.

σ-chain			Time			
curve	not found	found by curve	ECM 10^{-2}s.	per prime 10^{-8}s.	GCD 10^{-2}s.	per prime 10^{-8}s.
17	11829522	2018899	7.51	3.72	81.8	6.91
18	10113388	1716134	"	4.38	70.1	6.93
19	8652081	1461307	"	5.14	60.1	6.95
20	7406529	1245552	"	6.03	50.1	6.76
21	6344287	1062242	"	7.07	41.1	6.48
22	5439359	904928	"	8.30	35.5	6.53

Table 3: Compared costs-per-prime of ECM and GCD with implementation-independent σ-chain: 1000 digit prime input, $B_1 = 260$, $B_2 = 11600$

Table 3 shows that for a **1000 digit** input number, curve 21 is more expensive than gcd we stop at **curve 20** and then switch to gcd. The precomputed product of primes not found output by curve 21 is saved in GMP RAW format,

since this will be used when computing their gcd with any given 1000 digit input number. The file requires 24MB of disk space.

σ-chain			Time			
curve	not found	found by curve	ECM s.	per prime 10^{-8}s.	GCD 10^{-2}s.	per prime 10^{-8}s.
7	59624575	10892870	2.94	26.99	30.99s	51.97
8	50478207	9146368	"	32.14	26.28s	52.06
9	42790288	7687919	"	38.24	22.29s	52.09
10	36317090	6473198	"	45.42	18.92s	52.10
11	30851865	5465225	"	53.79	16.11s	52.22
12	26236335	4615530	"	63.7	13.73s	52.33

Table 4: Compared costs-per-prime of ECM and GCD with implementation-independent σ-chain: 10000 digit prime input, $B_1 = 260$, $B_2 = 11600$

Table 4 shows that for a **10000 digit** input number, curve 11 is more expensive than gcd we stop at **curve 10** and then switch to gcd. The precomputed product of primes not found output by curve 10 is saved in GMP RAW format, since this will be used when computing their gcd with any given 10000 digit input number. The file requires 134MB of disk space.

As we have seen, switching to GCD induces an increase in speed at the cost of disk space usage.

We also notice that the bigger the input number is, the earlier the switch to GCD. As the size of the modulus increases the modular reduction of the big product modulo the input number becomes relatively faster.

Acknowledgment

The author would like to thank Paul Zimmermann and the CACAO team at LORIA.

10 Appendix:

Curves	σ	primes not hit		Curves	σ	primes not hit
0		98182656				
	11	83011498			11921	10513
	235	71107946			24645	9127
	245	60912918			20631	7935
	1451	52210045			29588	6867
	1828	44762224			12941	6049
	29761	38383519			25641	5245
	21483	32928029			17611	4579
	159667972	28253400			17100	3999
	48402876	24249517		70	198	3482
10	917203524	20821752			29835	3051
	125584170	17881015			13483	2642
	526285807	15359623			14279	2317
	543094040	13197315			9482	2027
	522902334	11343594			3182	1757
	721074508	9751533			5834	1518
	402384384	8384249			19304	1329
	252939494	7212598			7187	1158
	184500796	6206246			16395	1028
	537234902	5340990		80	1869	887
20	345473498	4596611			26533	780
	246433302	3958200			18104	692
	372510762	3409631			2164	591
	1317505440	2935911			17519	510
	3914670462	2529222			23437	449
	922832476	2180553			25161	389
	2629173557	1880343			9203	341
	1770419160	1621396			9857	296
	1459896001	1397917			12653	265
	5613440127	1206013		90	19444	234
30	885745614	1040535			1452	198
	1864284	898036			12738	167
	281	775065			31366	143
	20547	669574			26097	132
	17666	578561			601	106
	21064	499727			28186	95
	5009	431777			6270	81
	20790	373234			26243	72
	4159	322508			13029	66
	7324	278972		100	23371	62
40	12224	241109			26441	54
	29151	208586			10096	50
	14081	180533			4086	41
	12415	156213			7952	34
	22466	135128			19578	32
	24322	117076			7269	26
	9822	101489			13787	21
	4004	87833			6115	18
	10560	76122			14456	16
	15332	65959		110	7984	14
50	19346	57225			8222	13
	1709	49624			15518	9
	295	43058			10149	8
	23753	37426			6187	6
	5332	32517			2542	5
	22996	28104			2176	4
	24452	24469			21621	3
	19188	21264			27934	2
	20260	18355			17098	1
	10241	15966		120	23637	0
60	24363	13935				
	84	12072				

Table 5: Chain for primes p, $2^{31} < p < 2^{32}$: $B_1 = 260$, $B_2 = 11600$

Curves	σ	primes not hit in $[2^{31}, 2^{32}]$		Curves	σ	primes not hit
0		98182656				
1	11	83644710		63	99911372	13622
2	235	72151958		64	644445221387	11894
3	245	62234951		65	66626	10408
4	1451	53713090		66	9437864411915	9085
5	1828	46368992		67	8444365156	7960
6	29761	40035815		68	65221387	6974
7	21483	34579894		69	77445454444	6074
8	159667972	29873369		70	964	5322
9	48402876	25814273		71	455687	4675
10	917203524	22315831		72	2313212312	4054
11	125584170	19293173		73	111114	3517
12	526285807	16684630		74	8873	3076
13	543094040	14432970		75	61626	2700
14	522902334	12489035		76	194364411915	2354
15	721074508	10806446		77	184443656	2037
16	402384384	9353550		78	61221387	1769
17	252939494	8099640		79	77445122144	1568
18	184500796	7014741		80	9614	1370
19	537234902	6076313		81	45151687	1204
20	345473498	5264062		82	1112312	1054
21	246433302	4562058		83	11314	914
22	372510762	3955411		84	87873	797
23	1317505440	3428167		85	6162613	709
24	3914670462	2972661		86	19436441191513	628
25	922832476	2578975		87	184443656133	545
26	2629173557	2238615		88	6122138713	470
27	1770419160	1942669		89	7744512214431	403
28	1459896001	1685891		90	681134864864	360
29	5613440127	1464184		91	1000	311
30	885745614	1271857		92	939393	265
31	1864284	1105017		93	10000	237
32	65498453	959831		94	38789	205
33	98465438	834191		95	32198	183
34	6543541884	724708		96	989796	163
35	635438543438	629998		97	584668	145
36	99999999	547484		98	21235741	127
37	36	476104		99	418555746341	114
38	10	416192		100	68888844	100
39	13	361893		101	44489411135	89
40	27	314877		102	4984186	83
41	17	274146		103	4498649689	71
42	548673341	238799		104	454153333111	59
43	235546846	207949		105	864168789	53
44	99873354	181201		106	3231354	46
45	35488242578637	157664		107	8874	41
46	77887453	137452		108	9874135555	37
47	12312	119459		109	41633333338789685555	33
48	754834	104137		110	666666488843	28
49	25746	90935		111	6384778748777	25
50	63873	79242		112	648565213243	21
51	17119118	69141		113	53486	20
52	3323123	60252		114	543789	16
53	887883442	52552		115	2354311223	14
54	737172658	46045		116	26366666	13
55	9874655525	40178		117	51587	10
56	98765525	35008		118	34534859405	8
57	1888	30548		119	546	6
58	33233	26663		120	1748	4
59	223223	23250		121	1327	3
60	743	20346		122	1001	2
61	1748761	17805		123	10999	1
62	1348	15534		124	57166	0

Table 6: Implementation-independent chain for primes $p < 2^{32}$: $B_1 = 260$, $B_2 = 11600$

Curve	not found	found by curve
0	203280221	
1	167337256	35942965
2	140221231	27116025
3	117723721	22497510
4	99077329	18646392
5	83522874	15554455
6	70517445	13005429
7	59624575	10892870
8	50478207	9146368
9	42790288	7687919
10	36317090	6473198
11	30851865	5465225
12	26236335	4615530
13	22332868	3903467
14	19029404	3303464
15	16225960	2803444
16	13848421	2377539
17	11829522	2018899
18	10113388	1716134
19	8652081	1461307
20	7406529	1245552
21	6344287	1062242
22	5439359	904928
23	4664421	774938
24	4003012	661409
25	3438676	564336
26	2956228	482448
27	2541845	414383
28	2186403	355442
29	1882720	303683
30	1622204	260516
31	1397929	224275
32	1204092	193837
33	1038338	165754
34	895161	143177
35	772361	122800
36	666561	105800
37	575642	90919
38	500127	75515
39	432032	68095

Table 7: Synoptic view of the results achieved: 39 first σ's of implementation-independent chain for all primes $p < 2^{32}$ with optimal B_1, B_2

Curves	σ	primes not hit in $[2^{31}, 2^{32}]$		Curves	σ	primes not hit
0		62183060				
1	11	52974463		63	99911372	8572
2	235	45695382		64	644445221387	7496
3	245	39415885		65	66626	6568
4	1451	34016309		66	9437864411915	5732
5	1828	29364908		67	8444365156	5035
6	29761	25353426		68	65221387	4413
7	21483	21897962		69	77445454444	3858
8	159667972	18918373		70	964	3361
9	48402876	16347337		71	455687	2969
10	917203524	14132605		72	2313212312	2587
11	125584170	12217400		73	111114	2253
12	526285807	10564859		74	8873	1987
13	543094040	9139212		75	61626	1752
14	522902334	7907806		76	194364411915	1524
15	721074508	6842230		77	184443656	1323
16	402384384	5922639		78	61221387	1138
17	252939494	5128353		79	77445122144	1007
18	184500796	4441257		80	9614	884
19	537234902	3846893		81	45151687	779
20	345473498	3332246		82	1112312	677
21	246433302	2888399		83	11314	578
22	372510762	2503318		84	87873	502
23	1317505440	2169669		85	6162613	454
24	3914670462	1881013		86	19436441191513	405
25	922832476	1631907		87	184443656133	350
26	2629173557	1416312		88	6122138713	298
27	1770419160	1228926		89	7744512214431	247
28	1459896001	1066834		90	681134864864	218
29	5613440127	926454		91	1000	191
30	885745614	804827		92	939393	165
31	1864284	699137		93	10000	145
32	65498453	607362		94	38789	124
33	98465438	527862		95	32198	109
34	6543541884	458796		96	989796	97
35	635438543438	398549		97	584668	87
36	99999999	346238		98	21235741	73
37	36	301172		99	418555746341	63
38	10	263289		100	68888844	55
39	13	228882		101	44489411135	52
40	27	199026		102	4984186	48
41	17	173284		103	4498649689	43
42	548673341	150942		104	454153333111	33
43	235546846	131456		105	864168789	30
44	99873354	114397		106	3231354	28
45	35488242578637	99584		107	8874	26
46	77887453	86760		108	9874135555	23
47	12312	75319		109	41633333338789685555	20
48	754834	65611		110	666666488843	18
49	25746	57400		111	6384778748777	17
50	63873	50023		112	648565213243	15
51	17119118	43640		113	53486	14
52	3323123	38070		114	543789	12
53	887883442	33144		115	2354311223	11
54	737172658	29005		116	26366666	10
55	9874655525	25327		117	51587	8
56	98765525	22063		118	34534859405	6
57	1888	19213		119	546	5
58	33233	16752		120	1748	3
59	223223	14629		121	1327	3
60	743	12830		122	1001	2
61	1748761	11216		123	10999	1
62	1348	9786		124	57166	0

Table 8: Implementation-independent chain for RSA-200 subset of primes $p < 2^{32}$: $B_1 = 260$, $B_2 = 11600$

Curve	primes not hit
0	98182656
1	83644710
2	71282670
3	60767235
4	51816244
5	44200893
6	37710868
7	32185774
8	27477708
9	23466067
10	20046796
11	17129213
12	14639387
13	12517697
14	10705083
15	9160152
16	7839365
17	6710204
18	5746507
19	4921104
20	4217969
21	3616810
22	3102325
23	2662114
24	2284073
25	1959876
26	1682662

Table 9: Implementation-independent rational chain for primes $2^{31} < p < 2^{32}$: $B_1 = 260$, $B_2 = 11600$

Curve	primes not hit
0	62183060
1	52974463
2	45145099
3	38485915
4	32819160
5	27996696
6	23886905
7	20387690
8	17408478
9	14867396
10	12701039
11	10854726
12	9276736
13	7931528
14	6783399
15	5803755
16	4967298
17	4251394
18	3641127
19	3118514
20	2672925
21	2292148
22	1965941
23	1686929
24	1447810
25	1242548
26	1066917

Table 10: Implementation-independent rational chain for RSA-200 subset of primes $2^{31} < p < 2^{32}$: $B_1 = 260$, $B_2 = 11600$

References

[1] R. Barbulescu. Familles de courbes elliptiques adaptées à la factorisation des entiers. 2009. Master's thesis, LORIA.

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

[3] Paul Zimmermann et. al. The ECMNET Project. `http://www.loria.fr/~zimmerma/records/ecmnet.html`.

[4] Pierrick Gaudry, Brian Gladman, Jim Fougeron, Laurent Fousse, Alexander Kruppa, Dave Newman, Jason Papadopoulos, and Paul Zimmermann. *GMP-ECM*, 6.2.3 edition, 2009. `http://gforge.inria.fr/projects/ecm`.

[5] B. Mazur. Rational isogenies of prime degree. *Inventiones Math.*, 44:129–162, 1978.

[6] P. L. Montgomery. Evaluating recurrences of form $x_{m+n} = f(x_m, x_n, x_{m-n})$ via Lucas chains, 1983. Available at ftp.cwi.nl/pub/pmontgom/Lucas.ps.gz.

[7] P. L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Math. Comp.*, 48(177):243–264, Jan. 1987.

[8] P. L. Montgomery. *An FFT Extension of the Elliptic Curve Method of Factorization*. PhD thesis, University of California, Los Angeles, 1992. `ftp.cwi.nl:/pub/pmontgom/ucladissertation.psl.gz`.

[9] P. Zimmermann and B. Dodson. 20 years of ECM. In *ANTS VII*, volume 4076 of *Lecture Notes in Comput. Sci.*, pages 525–542. Springer-Verlag, 2006.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399