Speeding up the computations on an elliptic curve using addition-subtraction chains

F. Morain, J. Olivos

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 983

Programme 1

SPEEDING UP THE COMPUTATIONS ON AN ELLIPTIC CURVE USING ADDITION-SUBTRACTION CHAINS

François MORAIN
Jorge OLIVOS

Mars 1989

RR-0983

SPEEDING UP THE COMPUTATIONS ON AN ELLIPTIC CURVE USING ADDITION-SUBTRACTION CHAINS

Francois MORAIN * † Jorge OLIVOS ‡

Abstract. We show how to compute x^k using multiplications and only one division. We use this method in the context of elliptic curves for which a law exists with the property that division has the same cost as multiplication. The best algorithm is 11.11% faster than the ordinary binary algorithm and speeds up accordingly the factorization and primality testing algorithms using elliptic curves.

ACCELERATION DES CALCULS SUR UNE COURBE ELLIPTIQUE A L'AIDE DE CHAINES D'ADDITION-SOUSTRACTION

Résumé. Nous montrons comment calculer x^k en utilisant des multiplications et exactement une division. Nous utilisons cette méthode dans le cadre de calculs sur les courbes elliptiques, sur lesquelles on peut mettre une loi de groupe abélien pour laquelle la division a le même coût que la multiplication. Le meilleur algorithme est 11% plus rapide que la méthode binaire usuelle et cela permet d'accélérer les algorithmes de factorisation et de primalité en conséquence.

* Projet ALGO, Institut National de Recherche en Informatique et en Automatique, Domaine de Voluceau, B. P. 105, 78153 LE CHESNAY CEDEX (France).

† On leave from the French Department of Defense, Délégation Générale pour l'Armement.

‡ Departamento de Ciencias de la Computación, Universidad de Chile, Casilla 2777. Santiago. Chile. Proyecto Fondecyt 348/87.

SPEEDING UP THE COMPUTATIONS ON AN ELLIPTIC CURVE USING ADDITION-SUBTRACTION CHAINS

François Morain *†
Jorge Olivos ‡

January 30, 1989

Abstract

We show how to compute x^k using multiplications and only one division. We use this method in the context of elliptic curves for which a law exists with the property that division has the same cost as multiplication. The best algorithm is 11.11% faster than the ordinary binary algorithm and speeds up accordingly the factorization and primality testing algorithms using elliptic curves.

1. Introduction. Recent algorithms used in primality testing and integer factorization make use of elliptic curves defined over finite fields or Artinian rings (Cf. Section 2). An abelian law (noted +) can be defined on these curves that allows classical algorithms built up over $\mathbf{Z}/N\mathbf{Z}$ to be transposed, such as the $p-1$ factorization algorithm of Pollard [26,4,17,19],

*Institut National de Recherche en Informatique et en Automatique (INRIA), Domaine de Voluceau, B. P. 105 78153 LE CHESNAY CEDEX (France).

†On leave from the French Department of Defense, Délégation Générale pour l'Armement.

‡Departamento de Ciencias de la Computación, Universidad de Chile, Casilla 2777. Santiago. Chile. Proyecto Fondecyt 348/87.

the Fermat-like primality testing algorithms [1,12,18,23] and the public key cryptosystems based on RSA [27,14,16]. The basic operation performed on an elliptic curve is the computation of the analogue of $x^k \bmod N$ in the classical case where we work over $\mathbf{Z}/N\mathbf{Z}$. This operation, noted additively, is described in Section 2. Whatever the model of computation may be, the number of elementary operations (understood as being the modular product of two large integers) is very high. Thus, reducing the number of such operations is a primary goal when implementing these algorithms.

One can look first for expressions of the law that minimizes the number of operations [8]. Another idea is to reduce the number of multiplications needed to reach x^k. One way of handling this problem is to introduce the concept of *addition chain* [15] for exponents. An addition-chain for the exponent k is given as a $(r+1)$-tuple $(k_0, \ldots, k_r)$ such that:

$$1 = k_0,\ k_1,\ \ldots,\ k_r = k,\ k_i > 0, \tag{1}$$

$$\text{and } \forall\, i \in 1..r,\ \exists\ a(i),\ b(i) \leq i,\ k_i = k_{a(i)} + k_{b(i)}. \tag{2}$$

Thus, if we have an addition chain for k with length r, we canb compute x^k with r multiplications.

Many results are known [3,10,15,25,28] and some good algorithms exist, among which the *binary algorithm*, recalled in Section 3, and some of its variations (see [15] and the implementation of the 2^m method in [9]). The same authors have also studied in brief the so called *addition-subtraction chains*, defined as in (1) but with:

$$\forall\, i \in 1..r,\ \exists\ a(i),\ b(i) \leq i,\ k_i = \pm k_{a(i)} \pm k_{b(i)}. \tag{3}$$

This idea corresponds to the evaluation of x^k by multiplications and divisions. In the case of integers, division is a costly operation and this idea does not seem to have been implemented. The situation dramatically changes when we try to compute the law on an elliptic curve, because in this case, division is replaced by addition of the inverse and that inverse is available at no cost (Cf. Section 2). We are to describe two algorithms that use addition-subtraction chains to compute x^k. The expected gain is about 8.33% for the first one and 11.11% for the second one.

In Section 2, we list some results on elliptic curves. Section 3 describes the binary algorithm. The new algorithms are presented in Section 4 and

5. We analyze the cost of these algorithms in Section 6. In Section 7., we compare the implementation of our algorithms to that of the 2^m method.

2. The law on an elliptic curve. Let $\mathbf{K}$ be a field of characteristic prime to 6. An elliptic curve E over $\mathbf{K}$ is a non singular algebraic projective curve of genus 1. It can be shown [6,29] that E is isomorphic to a curve of equation:

$$y^2z = x^3 + axz^2 + bz^3, \tag{4}$$

with a and b in $\mathbf{K}$. We note $E(\mathbf{K})$ the set of points of coordinates $(x : y : z)$ which satisfy (4) with $z = 1$, together with the point at infinity: $O_E = (0 : 1 : 0)$.

We may define on $E(\mathbf{K})$ an abelian law, we shall introduce in the case where $\mathbf{K} = \mathbf{R}$. This law, noted additively, plays the role of multiplication in our earlier discussion of addition chains. Addition chains are thus used to compute kM on a curve.

In $\mathbf{R}$, let us consider the curve of equation:

$$y^2 = x^3 + ax + b. \tag{5}$$

We assume the quantity $\Delta = 4a^3 + 27b^2$ to be negative (say). Such a curve is represented in figure 1 below.

Let M_1 and M_2 be two points on the curve. We want to associate with them a third point M_3 lying on the curve that we call the *sum* of M_1 and M_2. Let $\mathcal{D}$ be the line M_1M_2 (if $M_1 = M_2$, $\mathcal{D}$ is the tangent line). One can see that $\mathcal{D}$ intersects E at P. The point $M_3 = M_1 + M_2$ is then taken to be the reflexion of P along the x-axis. The neutral element of this law is O_E. The inverse of a point M of coordinates $(x : y : 1)$ is $-M$ whose coordinates are $(x : -y : 1)$. Thus taking the inverse of a point is essentially free.

It is easy to make this process effective. We only list the results. The coordinates of $M_3 = (x_3 : y_3 : 1)$ are:

$$x_3 = \lambda^2 - x_1 - x_2, \tag{6}$$

$$y_3 = \lambda(x_3 - x_1) - y_1, \tag{7}$$

where:

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1} & \text{if } x_1 \neq x_2 \\ (3x_1^2 + a)(y_1 + y_2)^{-1} & \text{otherwise.} \end{cases}$$

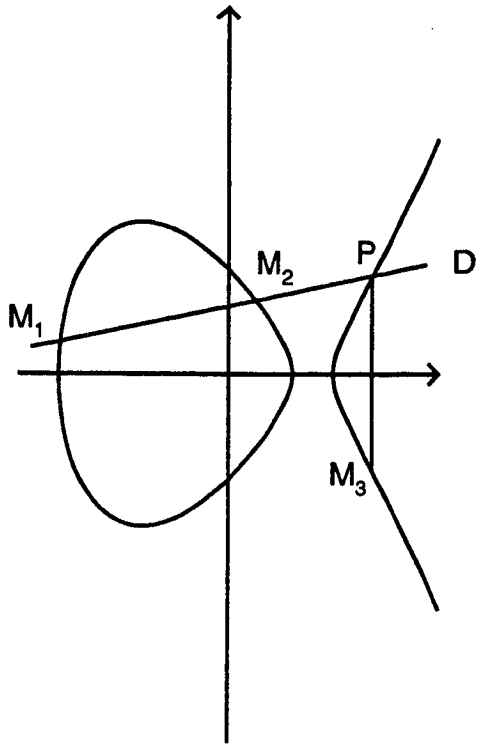

Figure 1: An elliptic curve over $\mathbf{R}$.

It should be noted that adding two different points on the curve requires three modular multiplications and one gcd, and that doubling a point on the curve costs one more modular multiplication. We will come back to this problem later.

If $\mathbf{K}$ is not a field, but just an Artinian ring, one can also define a law on $E(\mathbf{K})$ (Cf. [2]). The reason why elliptic curves are such attractive is that two different curves provide two different laws, contrary to the case where we work in $\mathbf{Z}/N\mathbf{Z}$, where we only have multiplication. In the case of integer factorization, different curves define distcinct factorization algorithms.

All the algorithms that use elliptic curves over finite fields require the computation of the point kM on E, where k is a large integer and M a point on the curve. These computations are to be made as fast as possible, so this motivates the study that follows.

3. The binary algorithm and some generalizations. We just describe the algorithm. For validity and explanation, see [15].
procedure BINEXP(P, M, k)
(* $P := kM$, $k = \sum_{i=0}^{i=n} k_i 2^i$, $2^n \leq k < 2^{n+1}$ *)

- $Q := M$;
- **if** $k_0 = 1$ **then** $P := M$ **else** $P := O_E$;
- **for** $i = 1 \,..\, n$
  $Q := 2Q$;
  if $k_i = 1$ **then** $P := P + Q$;
- **end.**

Following [15], we introduce:

$$\lambda(k) = \lfloor \log_2 k \rfloor, \tag{8}$$

$$\nu(k) = \text{Card}\,\{i \mid 0 \leq i \leq \lambda(k),\ k_i = 1\}. \tag{9}$$

Let C_{2P} be the cost of a "doubling" (i. e. evaluating $P + P$) and C_{P+Q} the cost of an "addition" (i. e. evaluating $P + Q$ with $P \neq Q$), supposed independent of P and Q. Then the cost of BINEXP is:

$$C_{\text{BINEXP}}(k) = \lambda(k)\, C_{2P} + (\nu(k) - k_0)\, C_{P+Q}. \tag{10}$$

The 2^m-ary algorithm (see [15]) consists in working in base 2^m, and not only in base 2. The cost of this algorithm is roughly (see [9]):

$$C_{m-\text{ary}}(k) = 2^{m-1} + \lambda(k)\, C_{2P} + (\nu_m(k) - k_0)\, C_{P+Q}, \tag{11}$$

where $\nu_m(k)$ computes the number of non-zero digits of k in base 2^m.

4. The first algorithm. We first explain the idea. Suppose we want to compute $29M$ on a curve. The binary representation of $k = 29$ is 11101. Now, we can write $k = 32 - 3 = 100000 - 11$ and compute in sequel $32M$, $3M$ and subtract these two points. In other words, *we replace a block of more than two 1's by a blocks of 0's and a subtraction.*

We now describe the first algorithm used to compute kM using two basic operations $+$ and $-$. The idea is to look for two integers k_- and k_+

such that $k = k_+ - k_-$ for which the evaluation of k_+M and k_-M require less operations than that of kM. We represent the algorithm by the automaton of figure 2. It is easy to deduce from this a procedure that computes kM recursively.

procedure ADDSUBCHAIN-A(P, M, k)

$Q := M$;
$P := O_E$; { the result is contained in P }
TREAT0(k);
end.

procedure TREAT0(k) { invariant: $R = P + kQ$ }

if $k = 0$ **then return**(P);
  else if k is even
    then $Q := 2Q$;
      TREAT0($\lfloor k/2 \rfloor$);
    else TREAT1($\lfloor k/2 \rfloor$);
end.

procedure TREAT1(k) { invariant: $R = P + (2k + 1)Q$ }

if $k = 0$ **then return**($P + Q$);
  else if k is even
    then $P := P + Q$;
      $Q := 4Q$;
      TREAT0($\lfloor k/2 \rfloor$);
    else $P := P - Q$;
      $Q := 4Q$;
      TREAT11($\lfloor k/2 \rfloor$);
end.

procedure TREAT11(k) { invariant: $R = P + (k + 1)Q$ }

```
if k = 0 then return(P + Q);
        else if k is even
                then P := P + Q;
                     Q := 2Q;
                     TREAT0(⌊k/2⌋);
                else Q := 2Q;
                     TREAT11(⌊k/2⌋);
end.
```

Example. Using the binary method, $k = 1101001110111$ is calculated by 12 doublings and 8 additions: 20 operations. Using ADDSUBCHAIN-A, we compute:

k	1101001110111
k_-	100000010001
k_+	10001010001000

we count 13 doublings, 5 additions and 1 subtraction: 19 operations.
The following is an optimized iterative form of this algorithm.

procedure ADDSUBCHAIN(M, k);

$\{k = k_n \ldots k_0\}$

- $b := 0$, $P := O_E$, $Q := M$;

- **for** $i := 0..n$
  if $k_i = 0$
  then if $b \neq 0$
  then $P := P + Q$;
  if $b = 1$ **then** $Q := 2Q$;
  $b := 0$;
  $Q := 2Q$;
  else if $b = 0$
  then $b := 1$;
  else if $b = 1$
  then $P := P - Q$;
  $Q := 2Q$;
  $b := 2$;
  $Q := 2Q$;

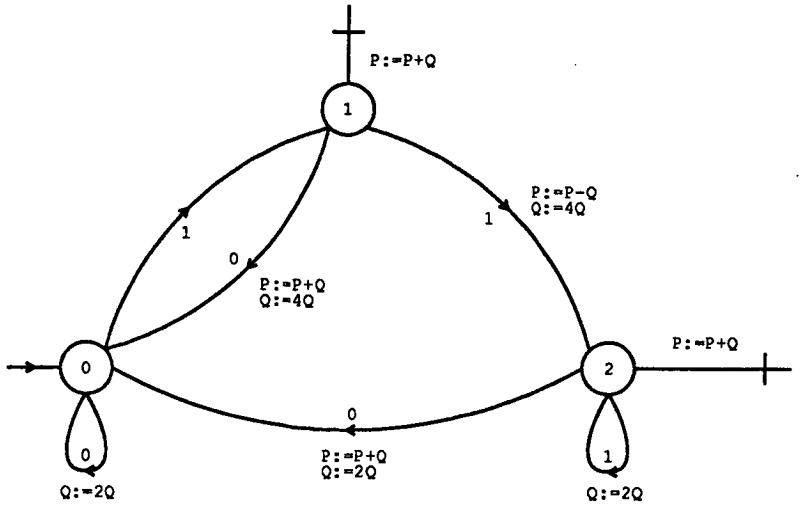

Figure 1: Finite Automaton, version A.

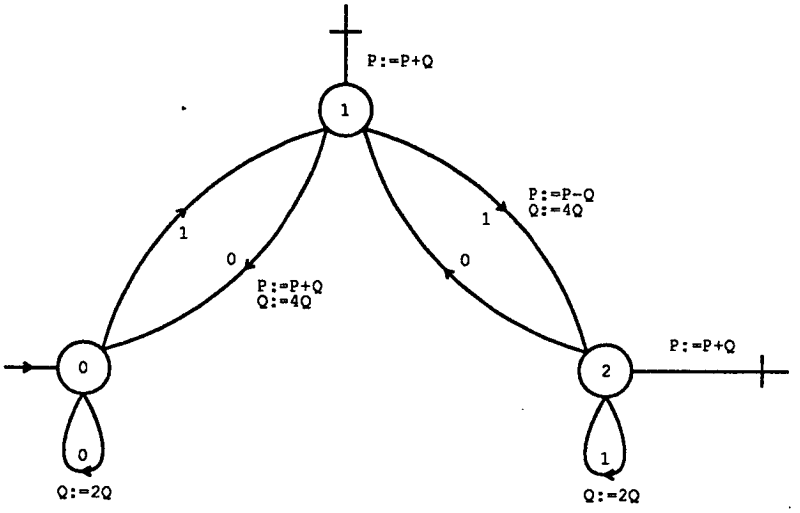

Figure 2: Finite Automaton, version B.

- $P := P + Q$;
- **end.**

5. A second algorithm. The idea is to distinguish between two patterns in the binary expansion of k, namely 1101 and 1100. This process is represented by a suitable modification of automaton A to produce automaton B. We can now use procedure ADDSUBCHAIN-B, that is the same as ADDSUBCHAIN-A after only modifying procedure TREAT11.

procedure TREAT11(k) { invariant: $R = P + (k+1)Q$ }

if $k = 0$ **then return**$(P + Q)$;
else if k is even
then TREAT1($\lfloor k/2 \rfloor$);
else $Q := 2Q$;
TREAT11($\lfloor k/2 \rfloor$);
end.

Example. With the same value of k, we find:

k	1101001110111
k_-	100000001001
k_+	10001010000000

we only require 13 doublings, 4 additions, and 1 subtraction: 18 operations.

6. Analysis of the algorithms. We recall that the expected cost of the binary method is $3/2n + O(1)$ and that of the 2^m method is $n(1 + (1 - 2^{-m})/m) + O(1)$.

In order to analyse the algorithm (both versions), we will count the number of operations required to calculate k, assuming that $C_{2P} = C_{P+Q} = 1$ for the sake of simplicity (however this is false in the case of elliptic curves – see Section 2.). As before, we introduce two auxiliary integers, k_- and k_+, such that $k + k_- = k_+$.

We will use an approach based on grammatical specification (see e.g. [13]). The associated bit strings belong to the language $L = 1\{0,1\}^*$. The

corresponding production rules, then, are the following:

$$\begin{array}{lcl} A & \rightarrow & 0T_0 + 1T_1 + x \\ T_0 & \rightarrow & 0T_0 + 1T_1 + x \\ T_1 & \rightarrow & 0T_0 + 1T_{11} + x \\ T_{11} & \rightarrow & 0T_0 + 1T_{11} + x \end{array}$$

where the symbol x represents the last 1 in the string of bits (i.e., the most significant bit). To obtain the generating function

$$A(z,u) = \sum_{n,m} a_{n,m} z^n u^m$$

where $a_{n,m}$ ($[z^n u^m]A(z,u)$) is the number of strings with $(n+1)$ bits (there are only 2^n since the last is always a 1), each string costing m operations. The corresponding system of equations is:

$$\begin{array}{lcl} A & = & uT_0 + u^2T_1 + 1 \\ T_0 & = & zuT_0 + zu^2T_1 + zu \\ T_1 & = & zuT_0 + zu^2T_{11} + zu^2 \\ T_{11} & = & zuT_0 + zuT_{11} + zu^2 \end{array}$$

We have used the fact that reading a bit always costs one doubling, and if the bit is a 1, one more addition. The symbol x has thus been replaced by imposing the boundary condition specified by each production rule.

$A(z,u)$ has been obtained using MAPLE ([7]). Since we are interested in the expected cost, we calculate

$$a(z) = \left(\frac{\partial A}{\partial u}\right)\Big|_{u=1} = \frac{z(6-6z-6z^2)}{(1-2z)^2}$$

from this we deduce

$$\frac{1}{2^n}[z^n]a(z) = [z^n]a(z/2) = \frac{11}{8}n + \frac{7}{4}.$$

This should be compared to the cost of the binary algorithm which is $3/2n + O(1)$. The relative savings in the number of additions is 25%, and an overall savings of 8.33% if all operations are considered.

In a similar manner, for version (B), the production rules are:

$$\begin{array}{lcl} B & \to & 0T_0 + 1T_1 + x \\ T_0 & \to & 0T_0 + 1T_1 + x \\ T_1 & \to & 0T_0 + 1T_{11} + x \\ T_{11} & \to & 0T_{110} + 1T_{11} + x \\ T_{110} & \to & 0T_0 + 1T_{11} + x \end{array}$$

The corresponding equations are:

$$\begin{array}{lcl} B & = & uT_0 + u^2T_1 + 1 \\ T_0 & = & zuT_0 + zu^2T_1 + zu \\ T_1 & = & zuT_0 + zu^2T_{11} + zu^2 \\ T_{11} & = & zuT_{110} + zuT_{11} + zu^2 \\ T_{110} & = & zuT_0 + zu^2T_{11} + zu \end{array}$$

From which is found:

$$b(z) = \left(\frac{\partial B}{\partial u}\right)\Big|_{u=1} = \frac{z(6z^3 - 7z^2 - 6z + 6)}{(1-2z)^2(z^2-1)}.$$

The expected cost is:

$$[z^n]b(z/2) = \frac{4}{3}n + \frac{35}{18} + O(2^{-n}).$$

With version B we achieve a savings of 33% with respect to additions, and 11.11% if all operations are considered.

We must note that version B of the algorithm was discovered with the help of $\Lambda\gamma\Omega$ [11], a powerful system designed to perform automatic analysis of a broad class of algorithms, developed at INRIA. The analyses were verified by this system.

We now give the results for elliptic curves when we have: $C_{2P} = k + 4$ and $C_{P+Q} = k + 3$, where k is the cost of a gcd over the integers, the unit being the time of a modular multiplication modulo N. For example, in [4], the author takes $k = 30$. P. Zimmermann kindly computed the costs of the algorithms given this assumption, using $\Lambda\gamma\Omega$. Here are the results:

Binary algorithm	$(3k+11)/2\, n + O(1)$
2^m-ary algorithm	$(k+4+(k+3)(1-2^{-m})/m))\, n + O(1)$
Algorithm A	$(11k+41)/8\, n + O(1)$
Algorithm B	$(4k+15)/3\, n + O(1)$

Our algorithms require no extra-storage, contrary to the 2^m method, which needs to store $2 \times 2^{m-1}$ n-bits integers (if we work over $\mathbf{Z}/N\mathbf{Z}$ with N an n-bit integer).

7. Implementation and conclusions. The first author has used the second algorithm in his implementation of the so called Atkin's test ([23]). The overall gain is about 3% in time, for 100-digit numbers and 2.7% for 300-digit numbers.

It should be possible to combine the idea of addition-subtraction chains with that of the 2^m-ary algorithm. As for now, it is not clear how we could do that, the problem being that our algorithm must keep track of what happened a few bits before.

Acknowledgments. The second author would like to express his gratitude to INRIA for an invited visit during which his work on the subject was done. Thanks are due to P. Flajolet. This work has also benefitted from financial assistance from the French-Chilean cooperation program whose help is gratefully acknowledged.

Both authors would like to acknowledge the help of P. Zimmermann with the $\Lambda\gamma\Omega$ system.

References

[1] A. O. L. ATKIN *In preparation.*

[2] W. BOSMA. Primality testing using elliptic curves. Report 85-12, Math. Instituut, Universeit van Amsterdam.

[3] A. BRAUER. On addition chains. *Bull. Amer. Math. Soc.*, **45**, 1939, p.736-739.

[4] R. P. BRENT. Some integer factorization algorithms using elliptic curves. Research Report CMA-R32-85, The Australian National University, Canberra, 1985.

[5] J. BRILLHART, D. H. LEHMER, J. L. SELFRIDGE, B. TUCKERMAN. *Factorizations of* $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to high powers.* Contemporary Mathematics, **22**, AMS, 1983.

[6] J. W. S. CASSELS. Diophantine equations with special references to elliptic curves. *J. London Math. Soc.*, **41**, 1966, p.193-291.

[7] B. W. CHAR, K. O. GEDDES, G. H. GONNET, S. M. WATT. *MAPLE Reference Manual, Fourth Edition.* Symbolic Computation Group, Department of Computer Science, University of Waterloo, 1985.

[8] D. V. CHUDNOVSKY, G. V. CHUDNOVSKY. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Research report RC 11262, IBM, Yorktown Heights, 1985.

[9] H. COHEN, A. K. LENSTRA. Implementation of a new primality test. *Math. of Comp.*, **48**, 177, 1987, pp. 103-121.

[10] P. ERDŐS. Remarks on number theory III: on addition chains. *Acta Arithmetica*, **6**, 1960, p.77-81.

[11] P. FLAJOLET, B. SALVY, P. ZIMMERMANN. Lambda-Upsilon-Omega: An Assistant Algoritms Analyser. Technical Report, INRIA, 1988.

[12] S. GOLDWASSER, J. KILIAN. Almost all primes can be quickly certified. *Proc. 18th STOC*, Berkeley, 1986, p.316-329.

[13] D. H. GREENE. Labelled Formal Languages and Their Uses. Technical Report STAN-CS-83-982, Stanford University, 1983.

[14] B. S. KALISKI, JR. A pseudo-random bit generator based on elliptic logarithms. *Proc. Crypto 86*, p.13-1,13-21.

[15] D. E. KNUTH. *Seminumerical algorithms.* The Art of Computer Programming, T. II, Addison-Wesley.

[16] N. KOBLITZ. Elliptic curve cryptosystems. *Math. of Comp.*, **48**, 177, 1987, p.203-209.

[17] H. W. LENSTRA, JR. Factoring with elliptic curves. Report 86-18, Math. Inst., Univ. Amsterdam, 1986.

[18] H. W. LENSTRA, JR. Elliptic curves and number theoretic algorithms. Report 86-19, Math. Inst., Univ. Amsterdam, 1986.

[19] H. W. LENSTRA, JR. Factoring integers with elliptic curves. *Annals of Math.*, **126**, 1987, pp. 649-673.

[20] D. P. MCCARTHY. The optimal algorithm to evaluate x^n using elementary multiplication methods. *Math. of Comp.*, **31**, 137, 1977, p.251-256.

[21] D. P. MCCARTHY. Effect of improved multiplication efficiency on exponentiation algorithms derived from addition chains. *Math. of Comp.*, **46**, 174, 1986, p.603-608.

[22] P. L. MONTGOMERY. Modular multiplication without trial division. *Math. of Comp.*, **44**, 170, 1985, p.519-521.

[23] F. MORAIN. Implementation of the Atkin-Goldwasser-Kilian test. INRIA Research Report 911, 1988.

[24] F. MORAIN, J. OLIVOS. Un algoritmo de Evaluacion de Potencia utilizando Cadenas de Suma y Resta. *Proc. XIV Conference Latinoamericana de Informatica (CLEI ,Expodata)*, Buenos Aires, Spetember 1988.

[25] J. OLIVOS. On Vectorial Additions Chains. *J. of Algorithms*, **2**, 1981, p.13-21.

[26] J. M. POLLARD. Theorems on factorization and primality testing. *Proc. Cambridge Phil. Soc.*, **76**, 1974, p.521-528.

[27] R. L. RIVEST, A. SHAMIR, L. ADLEMAN. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, **21**, 2, 1978, p.120-126.

[28] A. SCHÖNHAGE. A lower bound for the length of addition chains. *Theor. Comput. Science*, **1**, 1, 1975, p.1-12.

[29] J. T. TATE. The arithmetic of elliptic curves. *Inventiones Math.*, **23**, 1974, p.179-206.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique