Distributed computation of the number of points on an elliptic curve over a finite prime field

Johannes Buchmann, Volker Müller, Victor Shoup

SFB 124–TP D5

Report 03/95

27th April 1995

Johannes Buchmann, Volker Müller, Victor Shoup
Fachbereich Informatik
Universität des Saarlandes
Postfach 15 11 50
D–66041 Saarbrücken
Germany
email: `buchmann@cs.uni-sb.de`, `vmueller@cs.uni-sb.de`, `shoup@cs.uni-sb.de`

1 Introduction

In this paper we study the problem of counting the number of points on an elliptic curve over a finite prime field. This problem is not only very interesting for number theorists but has recently gained a lot of attention among cryptographers. The use of elliptic curves in public key cryptography was suggested by Koblitz [5] and Miller [7]. The security of their elliptic curve cryptosystems is based on the intractability of the problem of computing discrete logarithms in the elliptic curve group. The best algorithms known for solving this problem for arbitrary elliptic curves are the exponential square root attacks [9] which have running time proportional to the largest prime factor dividing the group order. Consequently, in order to guarantee the security of the system it is necessary to find this group order and its prime factorization. Although Schoof [11] proved that the cardinality of an elliptic curve group over a finite field can be computed in polynomial time, his algorithm is extremely inefficient in practice.

Recently, there has been a lot of progress concerning the problem of computing this group order $\#E(\mathbb{F}_p)$. Atkin [2] and Elkies [4] have developed new efficient algorithms. Those algorithms have been partially improved and implemented in Paris (see [3]) and Saarbrücken (see [6]). In both implementations the algorithm is distributed over a network of workstations by means of the system LiPS [10] which supports such distributions. The current record is the computation of the group order $\#E(\mathbb{F}_p)$, where p is a 375-digit prime (see [6]). That computation took approximately 1765 MIPS days. In this paper we briefly describe the state of the art of counting points on elliptic curve over finite prime fields. We explain the main computational problems and their solution by means of distributed and parallel computation.

2 The problem

We describe the problem explicitely. Let p be a prime number, $p > 3$. An *elliptic curve* over the prime field $\mathbb{F}_p$ of characteristic p is a pair $E = (a, b) \in \mathbb{F}_p^2$ with $4a^3 + 27b^2 \neq 0$. For example, for $p = 13$ the pair $E = (2, 3)$ is such a curve. The set $E(\mathbb{F}_p)$ of points on E is the set of all solutions $(x, y) \in \mathbb{F}_p^2$ of the equation

$$y^2 \quad = \quad x^3 + a\,x + b \tag{1}$$

together with an additional point $\mathcal{O}$ "at infinity" obtained by considering the projective closure of (1). The set $E(\mathbb{F}_p)$ has a group structure with the point $\mathcal{O}$ acting as the identity element. The problem is to find the cardinality $\#E(\mathbb{F}_p)$ of this group, i.e. the number of solutions of (1). It is known that

$$p + 1 - 2\sqrt{p} \quad \leq \quad \#E(\mathbb{F}_p) \quad \leq \quad p + 1 + 2\sqrt{p}.$$

There is an obvious method for finding $\#E(\mathbb{F}_p)$: for any pair $(x, y) \in \mathbb{F}_p^2$ check whether (x, y) is a solution of (1). Clearly, this method requires more than p^2

arithmetic operations and is, therefore, infeasible for large primes p. In our example, however, this method yields

$$\#E(\mathbb{F}_{13}) \quad = \quad 18.$$

3 The algorithm

We present a short overview of the algorithm and then describe the parts in detail (for a more exact description of the algorithm see [8]).

In a *precomputation*, the algorithm of Atkin and Elkies (AE) determines for the first few prime numbers l a polynomial $G_l(X,Y) \in \mathbb{Z}[X,Y]$ which is of degree $l+1$ in X.

If a particular finite prime field $\mathbb{F}_p$ and an elliptic curve $E = (a,b) \in \mathbb{F}_p^2$ are given, AE uses the polynomials $G_l(X,Y)$ to find $\#E(\mathbb{F}_p)$. It first determines for "sufficiently many" primes l the polynomial $G_{l,E}(X)$ which is obtained from $G_l(X,Y)$ by replacing the coefficients by their residue classes mod p and by substituting for Y the value $j(E) \in \mathbb{F}_p$ which is

$$j(E) \quad = \quad 1728\,\frac{4a^3}{4a^3+27b^2}.$$

Next it computes the degrees of the irreducible factors of $G_{l,E}(X)$. The sequence of those degrees is called the *decomposition type* of $G_{l,E}(X)$. From the decomposition type of $G_{l,E}(X)$ AE deduces possible values for the group order $\#E(\mathbb{F}_p) \bmod l$. Once there is information about $\#E(\mathbb{F}_p) \bmod l$ for sufficiently many prime numbers l, that information is used to find a multiple m of the order of a random point P on E in the interval $[p+1-2\sqrt{p},\, p+1+2\sqrt{p}]$. Typically, we have $m = \#E(\mathbb{F}_p)$, which can be checked by a verification procedure.

4 The precomputation

In the precomputation step we compute the polynomials $G_l(X,Y) \in \mathbb{Z}[X,Y]$. We will now describe how this is done. Let $j(\tau)$ be the Klein modular function (see [1]). That function is meromorphic and admits a Fourier expansion which can be explicitely determined. The first few terms of that expansion are

$$j(\tau) \quad = \quad e^{-2\pi i\tau} + 744 + 196\,884\; e^{2\pi i\tau} + 21\,493\,760\; e^{4\pi i\tau} + 864\,299\,970\; e^{6\pi i\tau} + \;\ldots\,.$$

The function $j(\tau)$ is trancendental over $\mathbb{C}$. Thus, substituting Y with $j(l\tau)$, we can view $G_l(X,Y)$ as a univariate polynomial in $\mathbb{Z}[j(l\tau)][X]$. Set $P_l(X) = G_l(X, j(l\tau))$. The coefficients of $P_l(X)$ belong to $\mathbb{Z}[j(l\tau)]$. They have, therefore, a Fourier expansion. On the other hand, the zeros of $P_l(X)$ are explicitely known. If

$$f_l(\tau) \quad = \quad \left(\frac{\eta(\tau)}{\eta(l\tau)}\right)^{2s},$$

where $\eta(\tau)$ is the Dedekind η-function and s is minimal such that $s(l-1)$ is divisible by 12, then those zeros are

$$z_k(\tau) \;=\; f_l\left(\tau + \frac{k}{l}\right), \quad 0 \leq k < l \qquad \text{and} \qquad z_l(\tau) \;=\; \frac{l^s}{f_l(l\tau)}.$$

The coefficients of $P_l(X)$ can be determined via Newton's formulas (see [14]) from the power sums

$$s_n(\tau) \quad := \quad \sum_{k=0}^{l} z_k^n, \qquad 1 \leq n \leq l+1.$$

From Fourier series expansions for $\eta(\tau)$ and $\eta(l\tau)$ it is possible to deduce Fourier series expansions for the coefficients of $P_l(X)$. On the other hand, since $P_l(X) = G_l(X, j(l\tau))$, we can also write down the coefficients of $P_l(X)$ using the Fourier expansion of $j(l\tau)$. Comparing coefficients, we find $G_l(X,Y)$.

For example, for $l = 3$ we have

$$f_3(\tau) \;=\; e^{-2\pi i\tau} - 12 + 54\, e^{2\pi i\tau} - 76\, e^{4\pi i\tau} - 243\, e^{6\pi i\tau} + 1188\, e^{8\pi i\tau} - 1384\, e^{10\pi i\tau} + \ldots\,.$$

Using the power sums

$$s_1(\tau) \;=\; -36, \qquad s_2(\tau) \;=\; 756$$

and

$$\begin{aligned} s_3(\tau) \quad &= \quad 3\,e^{-6\pi i\tau} \;-\; 17532 \;+\; 590652\,e^{6\pi i\tau} \;+\; 64481280\,e^{12\pi i\tau} \;+\; \ldots, \\ s_4(\tau) \quad &= \quad -144\,e^{-6\pi i\tau} \;+\; 424548 \;-\; 28351296\,e^{6\pi i\tau} \;-\; 3095101440\,e^{12\pi i\tau} \;+ \ldots, \end{aligned}$$

we can compute $G_3(X,Y) \in \mathbb{Z}[X,Y]$ as

$$G_3(X,Y) \;=\; X^4 + 36 \cdot X^3 + 270 \cdot X^2 + 756 \cdot X - X \cdot Y + 729.$$

Computing $G_l(X,Y)$ means performing additions, subtractions, multiplications and divisions of truncated Fourier series expansions. A large prime l, for which we have computed $G_l(X,Y)$, is $l = 829$. In that computation, we had to use 44766 terms of all occurring Fourier series. The coefficients of $G_{829}(X,Y)$ have approximately 640 decimal digits. To avoid computing with multi-precision integers, we use Chinese remaindering, i.e. we determine $G_l(X,Y)$ modulo many 32-bit primes, the so called Chinese primes. We then use the FFT-algorithm to do the multiplication of the truncated Fourier series. This requires a special choice of the Chinese primes. The computation of $G_l(X,Y)$ modulo the various Chinese primes is distributed over a network of workstations using the distributed system LiPS [10]. For computing $G_{829}(X,Y)$, 86 Chinese primes were necessary. The computation of $G_{829}(X,Y)$ modulo each of those primes took approximately 9 hours on a SPARC ELC workstation. Distributed over a network of 36 SPARC ELC workstations, the real time for computing $G_{829}(X,Y)$ was 28 hours; the total running time was approximately 827 hours (689 MIPS days).

5 Computing the group order modulo a prime number l

We describe, how to obtain information about $\#E(\mathbb{F}_p) \bmod l$ for a prime p, an elliptic curve E over $\mathbb{F}_p$ and a prime l. So far the largest p, for which such a computation has been carried out, is $p = 10^{374} + 169$; the elliptic curve was $E = (9051969, 11081969)$ (see [6]). We will illustrate the description by giving numerical data of this computation.

Instead of determining $\#E(\mathbb{F}_p) \bmod l$ directly, the algorithm exhibits information about $c = p + 1 - \#(\mathbb{F}_p)$. It is known that $|c| \leq 2\sqrt{p}$ (see [13]). First the j-invariant of E is computed which is

$$j(E) \quad = \quad 1728\,\frac{4a^3}{4a^3 + 27b^2}.$$

Then we calculate the polynomial $G_{l,E}(X) \in \mathbb{F}_p[X]$ which is obtained by substituting in $G_l(X,Y)$ the variable Y with $j(E)$ and reducing the coefficients by their residue classes modulo p. In order to obtain information about $c \bmod l$ we now exhibit the degrees of the irreducible factors of $G_{l,E}(X)$ in $\mathbb{F}_p[X]$. It can be shown that there are only the following possibilities:

1. $G_{l,E}(X)$ has a linear factor in $\mathbb{F}_p[X]$. Then $c \bmod l$ can be computed exactly using a method of Elkies [4].

2. $G_{l,E}(X)$ factors into a product of irreducible polynomials in $\mathbb{F}_p[X]$ which are all of the same degree $d > 1$. Using all elements of order d in the finite field $\mathbb{F}_{l^2}^*$, we can compute a list of $\varphi(d)$ possible values for $c \bmod l$, where $\varphi(\,)$ is the Euler totient function.

Here is a list of decomposition types of polynomials $G_{l,E}(X)$ and the corresponding number $k(l)$ of possibilities for $c \bmod l$ for our example. We also list the computation times.

l	decomp. type	$k(l)$	comp. time
5	(3, 3)	2	2 min 30 s
7	(8)	4	3 min 50 s
11	(1, 1, 10)	1	13 min 32 s
13	(1, 1, 12)	1	11 min 38 s
17	(9, 9)	6	20 min 10 s
211	(212)	104	5 h 16 min
223	(1, 1, 222)	1	7 h 43 min
227	(1, 1, 226)	1	11 h 45 min
229	(230)	88	5 h 2 min
233	(234)	72	5 h 22 min
401	(1, 1, 400)	1	17 h 53 min
409	(1, 1, 408)	1	22 h 7 min
419	$(140, \dots, 140)$	48	10 h 12 min
421	(422)	210	10 h 52 min
431	(1, 1, 215, 215)	1	26 h 40 min
607	(608)	288	17 h 25 min
617	$(103, \dots, 103)$	102	15 h 49 min
619	$(4, \dots, 4)$	2	13 h 43 min
631	(1, 1, 630)	1	34 h 42 min
641	$(1, 1, 128, \dots, 128)$	1	32 h 6 min

A major portion of the computing time is spent on the determination of the decomposition type of $G_l(X)$. We first compute $X^p \bmod G_{l,E}(X)$. Then we compute $\gcd(X^p - X, G_{l,E}(X))$. If this gcd is non-trivial, then we can compute a root of $G_{l,E}(X)$ modulo p, and from this we compute the value of $c \bmod l$ exactly using the Elkies algorithm. Otherwise we search for the smallest d dividing the degree $l+1$ of the polynomial $G_{l,E}(X)$, such that $X^{p^d} \equiv X \bmod G_{l,E}(X)$. The computation of $X^{p^d} \bmod G_{l,E}(X)$ is done with a repeated modular composition algorithm (see [12]), which uses the following fact: let $X^{p^k} \equiv g(X) \bmod G_{l,E}(X)$. Then we have for all $1 \le s \le k$ the following formula for computing $X^{p^{k+s}} \bmod G_{l,E}(X)$:

$$X^{p^{k+s}} \equiv g\left(X^{p^s}\right) \bmod G_{l,E}(X).$$

To carry out these computations, we need to perform polynomial arithmetic modulo $G_{l,E}(X)$. Multiplication of polynomials is done using a combination of Chinese remaindering and the FFT. Small primes r are chosen so that $r-1$ is divisible by a high power of two, and the product of these primes is a bit bigger than p^2. To multiply two polynomials over $\mathbb{F}_p$, the coefficients (represented as nonnegative integers less than p) are reduced modulo the small primes; then we compute the product polynomial modulo each small prime via the FFT; finally, we apply the Chinese remainder algorithm to each coefficient, and reduce modulo p.

In practice, this runs much faster than the classical "school" method for the size of polynomials we are considering (the cross-over point being less than degree 50), and is critical in obtaining reasonable running times.

Division by $G_{l,E}(X)$ with remainder is done using a standard reduction to polynomial multiplication; however, as $G_{l,E}(X)$ remains fixed for many divisions, it pays to perform some precomputation on $G_{l,E}(X)$. With this precomputation, one squaring modulo $G_{l,E}(X)$ costs about 1.5 times the cost of simply multiplying two degree l polynomials. Details on these algorithms can be found in [12].

To compute the group order $\#E(\mathbb{F}_p)$, we have to carry out this computation for many primes l. In our example we had to use all primes $l \leq 839$. Again the computation is distributed over a network of workstations with LiPS.

6 Combining possible values

Suppose that p, E and c are as in the previous section. We will describe how we actually compute the order of the group $E(\mathbb{F}_p)$ after knowing possible values for $c \bmod l_i$ for primes $l_1, \ldots, l_r$ with $\prod_{i=1}^{r} l_i > 4\sqrt{p}$. Let m_1 be the product of all prime numbers l_i for which we know $c \bmod l_i$ exactly. By Chinese remaindering we find a number $c_1 \in \{0, \ldots, m_1 - 1\}$ with $c \equiv c_1 \bmod m_1$. The remaining primes are divided into two sets L_2 and L_3. From the possible values for c modulo the elements of L_2 we determine by Chinese remaindering the set C_2 of all possible values of c modulo the product m_2 of the primes in L_2. The modulus m_3 and the set C_3 are obtained from L_3 in an analogous way. L_2 and L_3 are chosen such that C_2 and C_3 are approximately of equal cardinality. Now we know that

- $c \equiv c_1 \bmod m_1$,
- $c \equiv c_2 \bmod m_2$ for some $c_2 \in C_2$,
- $c \equiv c_3 \bmod m_3$ for some $c_3 \in C_3$.

To find the correct values for c_2 and c_3, we use Atkin's variant of Shank's "Baby-step/Giantstep" method. It is possible to write

$$c \quad = \quad c_1 + m_1 \cdot (m_2\, r_3 + m_3\, r_2)$$

with integers $|r_2| \leq m_2/2$ and $|r_3| \leq m_3$ satisfying

$$r_2 \equiv c_2\,(m_1\, m_3)^{-1} - c_1 \bmod m_2 \qquad \text{and} \qquad r_3 \equiv c_3\,(m_1\, m_2)^{-1} - c_1 \bmod m_2. \tag{2}$$

By (2), we can compute a candidate for r_2 for each element in C_2 and a candidate for r_3 for each element of C_3. The correct values for r_2 and r_3 are determined using Lagrange's theorem which implies that

$$(p + 1 - c) \cdot Q \quad = \quad \mathcal{O}$$

for any point Q on E. We choose a random point Q on E and check whether

$$(q + 1 - c_1) \cdot Q - m_1\, m_3\, r_2 \cdot Q \quad = \quad m_1\, m_2\, r_3 \cdot Q \tag{3}$$

is satisfied. This check is done by computing and storing the left hand side of (3) for all candidates for r_2. Those points are then ordered according to the x-coordinate to allow binary search. Then we compute the right hand side of (3) for all candidates r_3 and compare it to the stored points.

We remark that the algorithm only computes a candidate for the group order which must then be proven correct. So far, we have encountered no case in which the algorithm failed to find the group order.

In our example, we found 70 primes for which $c \bmod l$ could be computed exactly. The values of c_1 and m_1 were

```
c_1 = - 9864783057708164595502611646928152198562353124209875699\
        3425959176113645649555164159267043456184583649770069315\
        0497598992251884698163262879949594064047103794475295456\
        48489625068
```

and

```
m_1 = 1798472129187453299822428344084849721539162013066088847852\
      7094357189089303747931555452324762152102065987796224816663\
      3898175467543430076017989011585982034614879939889617470098\
      578086  .
```

The sets L_2 and L_3 contained 3 primes. The moduli m_2 and m_3 were $m_2 = 43055$ and $m_3 = 24735859$. We found a set C_2 of 160 possible values for c_2 and a set C_3 of 96 possible values for c_3. The Babystep/Giantstep part took 20 minutes (17 MIPS minutes).

Using this algorithm, we were able to compute the group order in our example. For $p = 10^{374} + 169$ and $E = (9051969, 11081969)$ we computed the group order $\#E(\mathbb{F}_p)$ as

```
999999999999999999999999999999999999999999999999999999999999999\
999999999999999999999999999999999999999999999999999999999999999\
999999999999999999999999999999999999999999999999999999999999933\
484119377057740549595798019340277299296194413840780451243105834\
253911258272979527787642571590910992330571399128986816096949176\
53999177002350595375701482255870650298836577293810050217802.
```

The total running time for this computation was approximately 1765 MIPS days (not including the precomputation step; on a network of 50 SPARC ELC workstations the computation took one week of real time.

7 Further improvements

As we have seen, even when distributed on a network of workstations, the determination of $\#E(\mathbb{F}_p)$ for a 375-digit prime p required one week of computing time. For cryptographers who wish to check the cryptographic properties of an elliptic curve this is still too slow. The time critical parts of the computation are the Fourier series calculation and the polynomial computations in the main part of the algorithm. Both are done using FFT. Those FFT computations can be parallelized and we expect this parallelization to reduce the running time by a considerable factor.

References

[1] T. Apostol, *Modular Functions and Dirichlet Series in Number Theory,* Springer-Verlag, 1990

[2] A.O.L. Atkin, *The number of points on an elliptic curve modulo a prime I/II,* unpublished manuscripts

[3] J. M. Couveignes, F. Morain, *Schoof's algorithm and isogeny cycles,* Proceedings of ANTS I, 1994

[4] N. Elkies, *Explicit Isogenies,* Preprint 1991

[5] N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, **48** (1987), 203-209

[6] F. Lehmann, M. Maurer, V. Müller, V. Shoup, *Counting the Number of Points on Elliptic Curves over Finite Fields of Characteristic Greater than Three,* Proceedings of ANTS I, 1994

[7] V. Miller, *Uses of elliptic curves in cryptography*, Advances in Cryptology: Proceedings of Crypto '85, Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, 417-426

[8] V. Müller, *Die Berechnung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakteristik größer 3,* Thesis, University of Saarland, to be published

[9] A. Odlyzko, *Discrete logarithms and their cryptographic significance*, Advances in Cryptology: Proceedings of Eurocrypt '84, Lecture Notes in Computer Science, **209** (1985), Springer-Verlag, 224-314

[10] R. Roth, Th. Setz, LiPS*: a system for distributed processing on workstations*, University of Saarland, 1993

[11] R. Schoof, *Elliptic curves over finite fields and the computation of square roots mod p*, Mathematics of Computation, **44** (1985), 483-494

[12] V. Shoup, *A New Polynomial Factorization Algorithm and its Implementation*, Preprint, 1994

[13] J. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, 1985

[14] B. L. van der Waerden, *Algebra*, Springer-Verlag, 1971