Counting the number of points on elliptic curves over finite fields of characteristic greater than three*

Frank Lehmann, Markus Maurer
Volker Müller, Victor Shoup

FB 14 Informatik
Universität des Saarlandes
Postfach 1150
D-66041 Saarbrücken
Germany
email: vmueller@cs.uni-sb.de

Abstract

We present a variant of an algorithm of Oliver Atkin for counting the number of points on an elliptic curve over a finite field. We describe an implementation of this algorithm for prime fields. We report on the use of this implementation to count the the number points on a curve over $\mathbf{F}_p$, where p is a 210-digit prime.

1 Introduction

In this paper we study the problem of counting the number of points on an elliptic curve over a finite 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 [Ko86] and Miller [Mi86]. 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 [Od86] 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 [Scho85] 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.

We present a short description of an variant of an algorithm of Oliver Atkin for counting points on elliptic curves over finite prime fields [At91]. We also describe an implementation of this algorithm and report on its effectiveness. With this implementation we were able to compute the group order of a curve defined over $\mathbf{F}_p$, where p is a 210-digit prime. The total time used was

*Appeared in *Proc. 1st Algorithmic Number Theory Symposium* (**ANTS**), pp. 60–70, 1994.

approximately 186 MIPS-days (this time does not include the time for precomputations that are independent of the input).

Background on the theory of elliptic curves can be found in [Si86]; a detailed description of the algorithm (including proofs of the theorems used) is given in [Mü94]. The implementation makes use of a library of routines for efficient computations with large polynomials over finite fields described in [Sho94].

2 Elliptic Curves and the Frobenius Endomorphism

Let $\mathbf{F}_q$ be the finite field with q elements, and let p be its characteristic. We shall assume that $p > 3$. Let E be an elliptic curve defined over $\mathbf{F}_q$ by the equation

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

where a and b are in $\mathbf{F}_q$, and the discriminant $4a^3 + 27b^2$ is nonzero.

Let $\overline{\mathbf{F}_q}$ be the algebraic closure of $\mathbf{F}_q$. For a field K, $\mathbf{F}_q \subset K \subset \overline{\mathbf{F}_q}$, the set $E(K)$ of K-rational points consists of the affine solutions $(x, y) \in K^2$ of (1), together with the point $\mathcal{O}$ "at infinity" obtained by considering the projective closure of (1). The set $E(K)$ has a group structure given by the well-known "tangent and chord method," with $\mathcal{O}$ acting as the identity element. The sum of two given points can be computed by simple formulas (see, e.g., [BuMü91]).

We will describe an algorithm that takes as input elements a and b defining an elliptic curve E as in (1), and produces as output the order of the group $E(\mathbf{F}_q)$. An important tool for solving this problem is the Frobenius Endomorphism for E.

Definition 1 *The* **Frobenius Endomorphism** *for E is the map*

$$\begin{array}{rccc} \Phi_E : & E(\overline{\mathbf{F}_q}) & \longrightarrow & E(\overline{\mathbf{F}_q}) \\ & (x, y) & \longmapsto & (x^q, y^q) \end{array}$$

The connection between this map and the problem of computing the order $\#E(\mathbf{F}_q)$ of $E(\mathbf{F}_q)$ follows from following well-known theorem of Hasse.

Theorem 1 (Hasse) *Let $c = q + 1 - \#E(\mathbf{F}_q)$. Then $|c| \leq 2\sqrt{q}$, and moreover the Frobenius Endomorphism Φ_E of E satisfies the equation $f(\Phi_E) = 0$ in the endomorphism ring of E, where*

$$f(X) = X^2 - cX + q \in \mathbf{Z}[X].$$

The value c in this theorem is the trace of Φ_E. The algorithm of Atkin consists of two steps: (1) for small primes l, compute a small set of possible values for c mod l; (2) use the Chinese remainder theorem to get a set of possible values for c, and then determine c from among these possible values.

3 Computing possible values for $c \bmod l$

In this section we describe an algorithm for computing a small set of possible values for c mod l. Schoof showed in [Scho85] how to compute $c \bmod l$ exactly with the help of $(l^2-1)/2$-degree polynomials. We will make use of a polynomial of degree $l+1$, but in general we are not able to compute $c \bmod l$ exactly.

Let E be a fixed elliptic curve over $\mathbf{F}_q$ and l an odd prime, $l \neq p$. The **l-torsion group** $E[l]$ of E is the subgroup of points $P \in E(\overline{\mathbf{F}_q})$ such that $l \cdot P = \mathcal{O}$. It is well-known that

$$E[l] \cong \mathbf{Z}/l\mathbf{Z} \times \mathbf{Z}/l\mathbf{Z}.$$

It follows that there are exactly $l+1$ (cyclic) subgroups of $E(\overline{\mathbf{F}_q})$ of order l, which we denote by $C_1, \ldots, C_{l+1}$.

Since $\Phi_E(E[l]) \subset E[l]$, we can consider the restriction of the Frobenius Endomorphism to $E[l]$, yielding an automorphism on $E[l]$ whose characteristic polynomial is $\bar{f}(X) \in \mathbf{F}_l[X]$, obtained by reducing the coefficients of the polynomial $f(X)$ in Theorem 1 mod l.

We can compute information about c mod l by examining the behavior of the groups C_i under the action of the Frobenius Endomorphism and its powers. The following theorem relates this behavior to the splitting type of $\bar{f}(X)$.

Theorem 2 *For $1 \leq i \leq l+1$, let d_i be the least positive integer such that $\Phi_E^{d_i}(C_i) = C_i$.*

1. *If $\bar{f}(X) = (X-\alpha)^2$ with $\alpha \in \mathbf{F}_l$, then either $d_i = 1$ for all $1 \leq i \leq l+1$ or there exists exactly one j with $d_j = 1$ and $d_i = l$ for all $1 \leq i \leq l+1, i \neq j$.*

2. *If $\bar{f}(X) = (X-\alpha)\cdot(X-\beta)$ with $\alpha, \beta \in \mathbf{F}_l$ and $\alpha \neq \beta$, then there exist $i_1 \neq i_2$ with $d_{i_1} = d_{i_2} = 1$ and $d_i = d = ord(\alpha/\beta)$ for $1 \leq i \leq l+1, i \neq i_1, i_2$.*

3. *If $\bar{f}(X)$ is irreducible over $\mathbf{F}_l$, then $d_i = d$ for all $1 \leq i \leq l+1$, where $d = ord(\alpha^{l-1})$ and α is a root of $\bar{f}(X)$ in $\mathbf{F}_{l^2}$.*

Thus we know the splitting type of $f(X)$ mod l if we know the behavior of the groups C_i under powers of the Frobenius Endomorphism. Possible values for c mod l can then easily be determined: we obtain two possible values for c mod l in case 1 and $\varphi(d)$ possible values in the cases 2 and 3.

The next problem is the determination of the values d_i in Theorem 2. Let C be a finite subgroup of $E(\overline{\mathbf{F}_q})$. Then it is known that there exists an elliptic curve (unique up to isomorphism) which we denote by E/C and an isogeny $\psi : E(\overline{\mathbf{F}_q}) \to (E/C)(\overline{\mathbf{F}_q})$ such that the kernel of ψ is C. We write j/C to denote the j-invariant of E/C. The following theorem gives the connection between such j-invariants and the values d_i in Theorem 2.

Theorem 3 *Let E be a non supersingular elliptic curve over $\mathbf{F}_q$ such that there exists no $\mathbf{F}_q$-isogeny (i.e. an isogeny defined over $\mathbf{F}_q$) to an elliptic curve of j-invariant 0 or 1728. Then for $1 \leq i \leq l+1$ we have*

$$d_i \quad = \quad \min_{k \in \mathbf{N}} \left\{ j/C_i \in \mathbf{F}_{q^k} \right\}.$$

For supersingular elliptic curves the group order is given by a theorem of Waterhouse; the group order of elliptic curves which are $\mathbf{F}_q$-isogenous to curves with j-invariant 0 or 1728 can be computed by finding elements of given norm in some maximal order of an imaginary quadratic field. More details can be found in [Mü94]. Thus we can assume that the assumptions of Theorem 3 are fulfilled.

So to compute d_i, we have to find minimal extensions of $\mathbf{F}_q$ which contain the j-invariants j/C_i. This problem is solved by finding the splitting type of a special polynomial, as the following theorem shows.

Theorem 4 *Let E be an elliptic curve and $C_i, (1 \leq i \leq l+1)$ be all subgroups of $E(\overline{\mathbf{F}_q})$ of exact order l. There exists a* **modular polynomial** *$\Phi_l(X,Y) \in \mathbf{F}_p[X,Y]$ such that all roots of $\Phi_l(X, j(E)) \in \mathbf{F}_q[X]$ are given by the j-invariants j/C_i, $1 \leq i \leq l+1$.*

Note that by Theorem 4 the degree of the l-th modular polynomial is exactly $l+1$. Combining all Theorems of this section we obtain the following algorithm for computing some information about the trace c of the Frobenius Endomorphism of an elliptic curve E over $\mathbf{F}_q$ modulo some odd prime l:

1. compute the l-th modular polynomial $\Phi_l(X,Y) \in \mathbf{F}_p[X,Y]$.
2. substitute the j-invariant of E and obtain $\Phi_l(X, j(E)) \in \mathbf{F}_q[X]$.
3. compute the splitting type of this polynomial and obtain $\{d_i\}_{i=1,\ldots,l+1}$.
4. compute information about $c \bmod l$ with Theorem 2.

4 Computing modular polynomials

In the last section we described how to compute partial information about the order of $E(\mathbf{F}_q)$ modulo some odd prime l if we know the l-th modular polynomial for fields of characteristic p. There is also a notion of modular polynomials for elliptic curves defined over $\mathbf{C}$, the field of complex numbers (see [Si86]). We recall this briefly, and mention how it is related to the notion of modular polynomials over finite fields.

An elliptic curve defined over $\mathbf{C}$ is isomorphic to $\mathbf{C}/L$, where $L = \mathbf{Z} + \mathbf{Z}\,\tau$ $(\Im(\tau) > 0)$ is a two-dimensional lattice in $\mathbf{C}$. The j-invariant of an elliptic curve over $\mathbf{C}$ can be interpreted as the j-invariant $j(\tau)$ of the corresponding lattice L. The classical modular polynomial for the complex numbers is a polynomial $\tilde{\Phi}_l(X,Y) \in \mathbf{Z}[X,Y]$, such that the roots of $\tilde{\Phi}_l(X, j(\tau))$ are given by

$$j\left(\frac{\tau+n}{l}\right) \quad \text{for} \quad 0 \leq n < l \qquad \text{and} \qquad j\Big(l\,\tau\Big).$$

It is known that a modular polynomial for $\mathbf{F}_p$ is equal to the reduction modulo p of the corresponding modular polynomial for $\mathbf{C}$.

Actually, we have chosen to work with a different modular polynomial: one that has the same splitting type as the above modular polynomial, but is slightly easier to compute. We now describe this polynomial.

Let $q_\tau = \exp(2\pi i \tau)$ and $\eta(\tau) = q_\tau^{1/24} \cdot \prod_{n=1}^{\infty}(1 - q_\tau^n)$ be the Dedekind η-function. Let $s \in \mathbf{N}$ be minimal such that $v = (s\,(l-1))/12 \in \mathbf{N}$ and define

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

Using the function $f(\tau)$ we can prove the following theorem (see [Mü94]).

Theorem 5 *There exist coefficients $a_{r,k} \in \mathbf{Z}$ such that $\sum_{r=0}^{l+1}\sum_{k=0}^{v} a_{r,k} \cdot j(l\tau)^k \cdot f(\tau)^r = 0$. For an elliptic curve E over $\mathbf{F}_q$ the polynomial*

$$G_l(X) \quad = \quad \sum_{r=0}^{l+1}\sum_{k=0}^{v} a_{r,k} \cdot j(E)^k \cdot X^r \quad \in \ \mathbf{F}_q[X]$$

has the same splitting type as the l-th modular polynomial $\Phi_l(X, j(E))$ for $\mathbf{F}_q$.

Next we describe the ideas for computing the coefficients $a_{r,k}$ used in the definition of $G_l(X)$. We can show that the roots of the polynomial

$$H_l(X) \quad = \quad \sum_{r=0}^{l+1} \underbrace{\left(\sum_{k=0}^{v} a_{r,k} \cdot j(l\tau)^k\right)}_{=:k_r(\tau)} \cdot X^r$$

are given by

$$f\left(\tau + \frac{n}{l}\right), \quad 0 \le n < l \qquad \text{and} \qquad \frac{l^s}{f(l\tau)}.$$

Since we can expand the functions $f(\tau)$ and $f(l\tau)$ as power series in q_τ with some precision, we can compute the power series expansions of the power sums of all zeroes of $H_l(X)$. Note that the r-th power sum of the "first" l zeroes can be computed by changing the power series expansion of $f(\tau)^r$ appropriately. Then we can use Newton's identities to compute power series expansions of the coefficients $k_r(\tau)$ of the polynomial $H_l(X)$. Knowing the coefficients $k_r(\tau)$ we can compute the values of $a_{r,k}$ by comparing minimal powers in the power series expansions of $k_r(\tau)$ and powers of $j(l\tau)$.

The advantage of $G_l(X)$ in comparison to the original l-th modular polynomial is the smaller precision which we need in the computation of power series expansions. For $l \equiv 1 \bmod 12$ the precision of power series expansions we need for $G_l(X)$ is about $\frac{1}{12}$ of the precision of power series expansions for computing $\Phi_l(X)$.

The method explained above works for all odd prime numbers l, but for some l we have used yet other functions for computing an analogue to $G_l(X)$, but which are sometimes easier to compute.

5 Computing $c \bmod l$ exactly

One problem with our method as we have described it so far is that we obtain only partial information about $c \bmod l$. As we consider more and more primes l, the set of possible values for c grows exponentially in the number of primes.

Fortunately, for primes l such that $f(X)$ splits mod l, (which we expect to happen about half the time), we can compute the value of c mod l exactly. We now describe the method.

Assume that the modular equation $G_l(X)$ has at least one root in $\mathbf{F}_q$. By the theorems of section 3 there exists at least one group C_i which is invariant under the Frobenius Endomorphism. Consider the polynomial

$$f_l(X) \quad := \prod_{\pm P \in C_i - \{\mathcal{O}\}} \Big(X - x(P) \Big).$$

This polynomial has degree $(l-1)/2$, and since C_i is invariant under Φ_E, its coefficients lie in $\mathbf{F}_q$. We can efficiently compute $f_l(X)$ with the help of one root of the modular equation. This computation is somewhat complicated; a detailed description can be found in [Mü94].

Assume that we know $f_l(X)$. By the invariance of C_i under Φ_E there exists a number $1 \leq \alpha < l$ such that the Frobenius Endomorphism satisfies for all points $P \in C_i$

$$\Phi_E(P) \quad = \quad \alpha \cdot P.$$

Similar to Schoof's algorithm, we transform this equation in the Endomorphism ring of C_i into polynomial equations modulo $f_l(X)$. For $1 \leq j < l$ we have to check whether

$$x^q \cdot \psi_j^2(X) \quad \equiv \quad x \cdot \psi_j^2(X) - \psi_{j-1}(X) \cdot \psi_{j+1}(X) \mod f_l(X)$$

and

$$4\,(X^3 + aX + b)^{(q+1)/2} \psi_j^3(X) \quad \equiv \quad \psi_{j+2}(X) \cdot \psi_{j-1}^2(X) - \psi_{j-2}(X) \cdot \psi_{j+1}^2(X) \mod f_l(X)$$

holds, where $\psi_j(X)$ is the reduced j-th division polynomial (compare [BuMü91]). For $j = \alpha$ both equations are true and we have found α. From Theorem 2 we know that

$$X^2 - \bar{c}\,X + \bar{q} \quad = \quad (X - \alpha) \cdot (X - \beta)$$

holds in $\mathbf{F}_l[X]$ and since we know α and $\bar{q}$, we can compute β and $\bar{c}$ by comparing coefficients.

6 Combining possible values

We will describe how we actually compute the order of the group $E(\mathbf{F}_q)$ after knowing possible values for c mod l_i for odd primes $l_1, \ldots, l_r$ with $\prod_{i=1}^r l_i > 4\sqrt{q}$. Using Chinese remaindering we can compute moduli $m_i, i = 1, 2, 3$, a value c_3, and sets L_1, L_2 such that

- $c \equiv c_3 \bmod m_3$
- $c \equiv c_1 \bmod m_1$, where $c_1 \in L_1 = \{c_{1,1}, \ldots, c_{1,k_1}\}$.
- $c \equiv c_2 \bmod m_2$, where $c_2 \in L_2 = \{c_{2,1}, \ldots, c_{2,k_2}\}$.

Note that we do not know the values of c_1 and c_2. We use a so-called "baby step/giant step" strategy to compute these values. Write

$$c \quad = \quad c_3 + m_3 \cdot (m_1\, r_2 + m_2\, r_1),$$

where r_1, r_2 are numbers which satisfy

$$r_2 \equiv c_2\,(m_1\,m_3)^{-1} - c_3 \bmod m_2 \qquad \text{and} \qquad r_1 \equiv c_1\,(m_2\,m_3)^{-1} - c_3 \bmod m_1.$$

If we choose c_3 as the least positive residue mod m_3 and the number r_1 as the least absolute residue mod m_1, we can show that $|r_2| \leq m_2$.

By Lagrange's theorem, a random point $Q \in E(\mathbf{F}_q)$ satisfies $(q+1-c)\cdot Q = \mathcal{O}$ and—substituting the equation for c—we get

$$(q+1-c_3)\cdot Q - m_1\,m_3\,r_2 \cdot Q \quad = \quad m_2\,m_3\,r_1 \cdot Q. \tag{2}$$

Since we do not know c_1 and c_2, we compute corresponding numbers $r_{1,i}, r_{2,j}$ for all elements in the sets L_1 and L_2. Then we check (2) for all these possibilities $r_{1,i}$ and $r_{2,j}$ with the following algorithm:

1. for all $1 \leq j \leq k_2$
   (a) compute $H_j = (q+1-c_3)\cdot Q - m_1\,m_3\,r_{2,j} \cdot Q$
   (b) If $H_j = \mathcal{O}$ then return $q+1-c_3 - m_1\,m_3\,r_{2,j}$
   (c) store $(H_j, r_{2,j})$ in a table T.
2. for all $1 \leq i \leq k_1$
   (a) compute $K_i = m_2\,m_3\,r_{1,i} \cdot Q$
   (b) If (K_i, x) exists in T for some number x, return $q+1-c_3 - m_1\,m_3\,x - m_2\,m_3\,r_{1,i}$.

Obviously this algorithm doesn't guarantee that the result really is the correct order of the group $E(\mathbf{F}_q)$; it returns only a multiple of the order of the point Q which lies in the interval given by Hasse's theorem. In practice the result is equal to the desired group order, which can be probabilistically checked by using a few more random points. Moreover there is an algorithm which really proves the correctness of the group order (see [Mü94]).

7 Computational results and implementation details

We have recently implemented this algorithm and used it to compute the group order of a curve defined over $\mathbf{F}_p$, where p is a 210-digit prime. We chose $p = 10^{209} + 103$, and the curve

$$E : y^2 = x^3 + 9051969x + 11081969.$$

The group order was computed as

```
10000000000000000000000000000000000000000000000000000000000000000\
00000000000000000000000000000000000000004179063042315198589540022\
67099915195519032247333419733547622071791625087426579598414738335\
917295830214295.
```

This number happens to be particularly easy to factor and its factorization is

$$3^2 \cdot 5 \cdot 13 \cdot 19 \cdot 53623 \cdot 12342839 \cdot P,$$

where P is a prime number. Using this factorization, we were able to prove the correctness of the group order.

The total computation (not including the computation of modular polynomials, which is a precompuation that is independent of the input), took approximately 186 MIPS-days, of which 163 were used in computing the splitting type and 13 MIPS-days to perform the baby step/giant step computation.

In the computation, we used modular polynomials for l up to 503. Of the 95 odd primes up to 503, we found that for this curve, 39 were so-called Elkies primes, i.e., primes for which the polynomial $f(X)$ splits, and we can quickly compute $c \bmod l$ exactly.

We have computed modular polynomials for l up to 503, and have stored these on disk. The storage requirement is approximately 40 MBytes. Individual modular equations are relatively expensive to compute. For example, in our implementation, we used 155 MIPS-days to compute the modular polynomial for $l = 503$.

It is fairly easy to distribute much of the computation over a network of workstations. We have done so using LiPS [Li93]. This proved quite effective.

We now give some details of the implementation.

7.1 Computing Modular Polynomials

To compute the l-th modular polynomial, we compute it modulo several small primes, and then use Chinese remaindering to get the coefficients over $\mathbf{Z}$. These small primes are chosen so that each prime r fits in one computer word, and so that $r-1$ is divisible by a high power of 2. This allows us to multiply polynomials of very large degree modulo r very quickly using the FFT. Arithmetic modulo r is performed using floating point operations (which is both portable and fast).

We distributed the computation modulo different primes r over a network of workstations.

7.2 Computing the splitting type

Assuming we have precomputed all of the necessary modular polynomials, the most time consuming step is the computation of the splitting type of $G_l(X)$ modulo p, and to compute a root should it admit one.

These problems are special cases of the polynomial factorization problem, and the techniques used are closely related.

To compute the splitting type of the modular polynomial $G_l(X)$, we first compute $X^p \bmod G_l(X)$. Then we compute $\gcd(X^p - X, G_l(X))$. If this gcd is non-trivial, then we can compute a root of $G_l(X)$ modulo p, and from this we compute the value of $c \bmod l$ exactly. If this gcd is trivial, then we know that $G_l(X)$ factors as a product of distinct irreducibles, all of the same (unknown) degree d. We can still obtain partial information about $c \bmod l$ by computing this value d.

To carry out these computations, we need to perform polynomial arithmetic modulo $G_l(X)$. 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 $\mathbf{F}_p$, 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 $G_l(X)$ with remainder is done using a standard reduction to polynomial multiplication; however, as $G_l(X)$ remains fixed for many divisions, it pays to perform some precomputation on $G_l(X)$. With this precomputation, one squaring modulo $G_l(X)$ costs about 1.5 times the cost of simply multiplying two degree l polynomials.

Also, a new algorithm for computing the value d was employed which is quite efficient in practice—the number of polynomial multiplications is a small multiple of $l^{1/2}\log_2 l$.

We also remark that very good performance for these algorithms was obtained, despite the fact that the algorithms were implemented in C and are very portable. Details on these algorithms and their implementation can be found in [Sho94].

The computation for different primes l was distributed over a network of workstations.

7.3 Baby step/giant step computation

Several tricks were used to speed up the baby step/giant step computation in section 6.

1. We store only one computer word of the x-coordinate of points computed in the baby step part. After having done all baby steps we sort the table according to the "x-coordinate" (using quicksort). Thus we are able to use binary search for checking a match in the following giant step part. If we find a match, we recompute the original baby step point (using the second component of the table entry) and compare.

2. For doing fast steps we sort the numbers $r_{1,i}, r_{2,j}$ such that we can do one step by adding some small multiple of Q. These different multiples of Q are computed with the following trick: precompute the table

$$1\cdot Q,\ 2\cdot Q,\ldots, L\cdot Q,\ 2\,L\cdot Q,\ 4\,L\cdot Q,\ldots, L^2\cdot Q,\ 2\,L^2\cdot Q,\ 4\,L^2\cdot Q,\ldots,\ L^3\cdot Q$$

   for some constant $L \approx 200$. Since with $x\cdot Q$ we directly know the point $-x\cdot Q$, we can compute the point $x\cdot Q$ for all $|x| < L^3$ with at most three additions by expanding x as

$$x \quad = \quad x_1 + x_2\cdot(2\,L) + x_3\cdot(2\,L^2), \qquad |x_i| \le L.$$

3. The most time consuming part of point addition is the inversion of some field element. We are doing several steps in this part in parallel so that we can use a trick of Montgomery which reduces the number of inversions on the cost of a few multiplications (see [Mo87]).

References

[At91] A.O.L. Atkin: *The number of points on an elliptic curve modulo a prime,* unpublished manuscript

[BuMü91] J. Buchmann, V. Müller: *Computing the number of points on an elliptic curve over finite fields,* Proceedings of ISSAC 1991, 179-182

[Ko86] N. Koblitz: *Elliptic curve cryptosystems*, Mathematics of Computation, 48 (1987), 203-209

[Mi86] 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

[Mü94] V. Müller: *Die Berechnung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakteristik größer 3,* Thesis, to be published

[Mo87] P. L. Montgomery: *Speeding the Pollard and Elliptic Curve Methods for Factorization,* Mathematics of Computation, 48 (1987), 243 - 264

[Od86] 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

[Li93] R. Roth, Th. Setz: *LiPS: a system for distributed processing on workstations*, University of Saarland, 1993

[Scho85] R. Schoof: *Elliptic curves over finite fields and the computation of square roots mod* p, Mathematics of Computation, 44 (1985), 483-494

[Sho94] V. Shoup: *Practical computations with large polynomials over finite fields*, in preparation (1994).

[Si86] J. Silverman: *The arithmetic of elliptic curves*, Springer-Verlag, 1986