Schoof Algortithm and the CM method. Lecture 7.

Elisa Lorenzo García

21th March 2016

In today's lecture, we study Schoof algorithm to compute the number of rational points of an elliptic curve $E/\mathbb{F}_q$ over a finite field. The main reference is Schoof algorithm, [3], but we also follow Chenal's master thesis [1, Section 6], that is available on line at

http://algant.eu/documents/theses/chenal.pdf

and that we reproduce here. In the second part of today's lecture, we study the complex multiplication method, that given an integer N and a prime number p, produces, in case that it existed, an elliptic curve $E/\mathbb{F}_p$ such that $\#E(\mathbb{F}_p) = N$. We follow Holm's thesis, [2, Chapter 4], that we reproduce here and that can be found in

http://bjarkiholm.com/publications/Holm_2005_partiii-essay.pdf.

1 Exercises

Exercise 1.1. *Use Schoof's algorithm to compute the number of points of the elliptic curve over* $\mathbb{F}_{19}$, $E : y^2 = x^3 + 2x + 1$.

Exercise 1.2. *Use the CM method to compute an elliptic curve over* $\mathbb{F}_{17}$ *with* $\#E(\mathbb{F}_{17}) = 12$. *Hint: work with* $D = -2$ *and compute the Hilbert polynomial by evaluating the* $j-$*invariant.*

Exercise 1.3. *Use the CM method to compute an elliptic curve over* $\mathbb{F}_7$ *with* $\#E(\mathbb{F}_7) = 8$. *Hint:* $H_{-23}(x) = X^3 + 3491750X^2 - 5151296875X + 12771880859375$.

References

[1] M. Chenal, *Applications of Complex Multiplication of Elliptic Curves*, Master thesis at Università degli Studi di Padova and Université Bordeaux 1, 2012.

[2] B. Holm, *Constructing Elliptic Curves with a Given Number of Points*, thesis at University of Cambridge, 2005.

[3] R. Schoof, *Counting points on elliptic curves over finite fields*, Journal de Théorie des Nombres de Bordeaux 7, 219-254, 1995.

CHAPTER 6

Schoof's Algorithm

Abstract

In this chapter we will present Schoof's deterministic algorithm to compute the number of $\mathbb{F}_q$-points of an elliptic curve that is defined over a finite field $\mathbb{F}_q$ and which is given by a Weierstrass equation. The algorithm takes - as we shall see - $O(\log^{8+o(1)} q)$ elementary operations (bit operations). If we use fast exponentiation arithmetic, the total cost will be reduced to $O(\log^{5+o(1)} q)$.

6.1 Motivation

Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$. For many applications, it is important to have an efficient way to compute the number of points in $E(\mathbb{F}_p)$, where p has usually many decimal digits. For instance, in elliptic curve cryptography it is important to know the number of points to judge the difficulty of solving the discrete logarithm problem in the group of points on an elliptic curve.

So it is a major problem to study an efficient way of computing $\#E(\mathbb{F}_p)$ for large primes. Before 1985, approaches to counting points on elliptic curves such as the naive one (which we will consider in a moment) and baby-step giant-step algorithms were not practical and had an exponential running time. In 1985, René Schoof from university of Amsterdam published, in his paper [8], an efficient way to determine $\#E(\mathbb{F}_p)$. It was a theoretical breakthrough, as it was the first deterministic polynomial time algorithm for counting points on elliptic curves.

In the 1990's, Elkies and Atkies improved Schoof's algorithm by studying elliptic curves over $\mathbb{C}$. The so called Schoof-Elkies-Atkin (SEA) algorithm can deal the problem of computing $\#E(\mathbb{F}_p)$, where p has hundreds of digits, see [9].

6.2 The Setup and the Naive Method

Let $p \neq 2, 3$ be a prime and let E be an elliptic curve over $\mathbb{F}_p$ given by a Weierstrass equation

$$y^2 = x^3 + Ax + B \tag{6.1}$$

for some $A, B \in \mathbb{F}_p$. The curve is not singular, so $4A^3 + 27B^2 \not\equiv 0 \pmod{p}$. We know that the set $E(\mathbb{F}_p)$ of rational points of E consists of the solutions $(a, b) \in \mathbb{F}_p^2$ satisfying the curve equation and the point at infinity O. Using the group law on elliptic curves restricted to this set we know that this set $E(\mathbb{F}_p)$ forms an abelian group, with O acting as the zero element. The number of points in $E(\mathbb{F}_p)$ with given X-coordinate $x \in \mathbb{F}_p$ is 0, 1 or 2.

Let $\left(\frac{\cdot}{p}\right)$ denote the usual quadratic symbol[(1)]. There are

$$1 + \left(\frac{x^3 + Ax + B}{p}\right)$$

rational points on E with X-coordinate equal to x. For a discussion involving the complexity of Legendre symbol, see Appendix A.1. Including the point at infinity, the set of rational points $E(\mathbb{F}_p)$ of E therefore has cardinality

$$\#E(\mathbb{F}_p) = 1 + \sum_{x \in \mathbb{F}_p} \left(1 + \left(\frac{x^3 + Ax + B}{p}\right)\right)$$

The 1 in this equation stands for the point at infinity $O = [0 : 1 : 0]$. If we group altogheter the ones in the previous equation, we get

$$\#E(\mathbb{F}_p) = 1 + \sum_{x \in \mathbb{F}_p} \left(1 + \left(\frac{x^3 + Ax + B}{p}\right)\right) = 1 + p + \sum_{x \in \mathbb{F}_p} \left(\frac{x^3 + Ax + B}{p}\right)$$

Remark 6.2.1. The same argument holds if we consider a finite field $\mathbb{F}_q$ with $q = p^r$, where $p \neq 2, 3$ is prime and $r \in \mathbb{N}$ (and we use Jacobi's symbol instead of Legendre's).

[(1)] Recall that an integer a is a quadratic residue modulo p if it is congruent to a perfect square modulo p and is a quadratic nonresidue modulo p otherwise. The Legendre symbol is defined as follows:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 \text{ if } a \text{ is a quadratic residue modulo } p \text{ and } a \not\equiv 0 \pmod{p} \\ -1 \text{ if } a \text{ is a quadratic non-residue modulo } p \\ 0 \text{ if } a \equiv 0 \pmod{p}. \end{cases}$$

Remark 6.2.2. We deduce a first trivial bound $\#E(\mathbb{F}_p) \leq 2p+1$ (see Remark 4.1.8). We can use this equation to compute the number of points in $E(\mathbb{F}_p)$: we then have to compute p Legendre symbols. Now, every Legendre symbol is computed using fast exponentiation, and therefore by the result in Appendix A.1, the total running time is $O(q\log^3 q)$ if we use the elementary multiplication algorithm.

So this naive counting algorithm is not polynomial time: the size of the result is the number of digits in $\#E(\mathbb{F}_q)$ and this is $\leq \log_{10}(3q+1)+1$. An efficient counting algorithm should run in time polynomial in $\log q$, and this is the case of Schoof's algorithm.

6.3 The Idea Behind the Algorithm

To set up the situation, let $E/\mathbb{F}_q$ be an elliptic curve defined over a finite field $\mathbb{F}_q$, where $q = p^r$ for $p \neq 2,3$ a prime and r an integer ≥ 1, and consider its Weierstrass equation

$$y^2 = f(x) := x^3 + Ax + B$$

with $A, B \in \mathbb{F}_q$. Hasse's theorem 4.1.10 says that

$$\#E(\mathbb{F}_q) = q + 1 - t \tag{6.2}$$

with

$$|t| \leq 2\sqrt{q} \tag{6.3}$$

We start by quickly describing the ideas behind Schoof's Algorithm, which computes $\#E(\mathbb{F}_q)$ in polynomial time, i.e., it computes $\#E(\mathbb{F}_q)$ in $O(\log^c q)$ steps, where c is fixed, independent of q. Its approach is to compute the cardinality $\#E(\mathbb{F}_q)$ by making use of Hasse's theorem on elliptic curves along with the Chinese remainder theorem and division polynomials.

Hasse's theorem simplifies our problem by narrowing down $\#E(\mathbb{F}_q)$ to a finite (even if large) set of possibilities. We want to compute the value t as seen in equation 6.2: it is enough to compute t modulo N where $N > 4\sqrt{q}$. While there is no efficient way to compute $t \pmod N$ directly for general N, it is possible to compute $t \pmod l$ for l a small prime, rather efficiently. We choose $S = \{l_1, l_2, ..., l_r\}$ to be a set of distinct primes such that $\prod l_i = N > 4\sqrt{q}$. Given $t \pmod{l_i}$ for all $l_i \in S$, the Chinese remainder theorem allows us to compute $t \pmod N$.

Now, in order to compute $t \pmod l$ for a prime $l \neq p$, we make use of the theory of the Frobenius endomorphism ϕ and division polynomials. Note that considering primes $l \neq p$ is no loss since we can always pick a

bigger prime to take its place to ensure the product is big enough. Let

$$\begin{aligned} \phi : E(\overline{\mathbb{F}}_q) &\to E(\overline{\mathbb{F}}_q) \\ (x,y) &\to (x^q, y^q) \end{aligned}$$

be the q-power Frobenius map, so Theorem 4.2.5(b) tells us that

$$\phi^2 - t\phi + q = 0 \tag{6.4}$$

in $\mathrm{End}(E)$. In particular, if $P \in E(\mathbb{F}_q)[l]$ (the group of l-torsion points, see Definition 2.4.3), then

$$\phi^2(P) - [t]\phi(P) + [q]P = O$$

so if we write $P = (x,y)$ (we assume that $P \neq O$), then

$$(x^{q^2}, y^{q^2}) - [t](x^q, y^q) + [q](x,y) = O$$

One could try to compute these points (x^{q^2}, y^{q^2}), (x^q, y^q) and $[q](x,y)$ as functions in the coordinate ring $\mathbb{F}_p[x,y]/(y^2 - f(x))$ of E and the search for a value of t which satisfies the equation. However, the degrees get very large and this approach is hopeless.

A key observation is that, since the point $P = (x,y)$ is chosen to have order l, we have $[q]P = [\overline{q}]P$, where $\overline{q}$ is the unique integer such that $q \equiv \overline{q} \pmod{l}$, $0 \leq \overline{q} < l$.

Now, note that $\phi(O) = O$ and that for any integer r we have $r\phi(P) = \phi(rP)$. Thus $\phi(P)$ will have the same order as P: since $P = (x,y) \in E[l]$, we have also $\phi(P) \in E[l]$ and so $[t](x^q, y^q) = [\bar{t}](x^q, y^q)$, where $t \equiv \bar{t} \pmod{l}$ with $0 \leq \bar{t} < l$. Hence we have reduced our problem to solving the equation

$$(x^{q^2}, y^{q^2}) + [\bar{q}](x,y) \equiv [\bar{t}](x^q, y^q) \tag{6.5}$$

Remark 6.3.1. We can compute the trace t of the Frobenius endomorphism modulo l by checking which of the relations

$$(\phi^2 - \bar{t}\phi + q)P = 0$$

hold on $E[l]$.

Remark 6.3.2. For ease of notation, when the context is clear we will often write mP instead of $[m]P$ for the multiplication-by-m map.

Remark 6.3.3. Of course, we don't know the value of $\bar{t}$, so for each integer n between 0 and l we compute $[n](x^q, y^q)$ for a point $(x,y) \in E[l]/\{O\}$ and check to see whether it satisfies

$$[n](x^q, y^q) = (x^{q^2}, y^{q^2}) + [q](x,y)$$

However, the individual points in $E[l]$ tend to be defined over large extension fields of $\mathbb{F}_q$, so we instead work with all of the l-torsion points simultaneously. To do this, we use the division polynomial (Definition 6.5.2)

$$\psi_l(x) \in \mathbb{F}_q[x]$$

whose roots are the x-coordinates of the nonzero l-torsion points of E. (For simplicity, we assume that $l \neq 2$.) This division polynomial has degree $\frac{1}{2}(l^2-1)$ and is easily computed using the recurrence described in Definition 6.5.2. We then perform all computations in the quotient ring

$$R_l = \mathbb{F}_q[x,y] / \left(\psi_l(x), y^2 - f(x)\right)$$

Thus anytime we have a nonlinear power of y, we replace y^2 with $f(x)$, and anytime we have a power x^d with $d \geq \frac{1}{2}(l^2-1)$, we divide by $\psi_l(x)$ and take the remainder. In this way we never have to work with polynomials of degree greater than $\frac{1}{2}(l^2-3)$.

Our goal is to compute the value of $t \pmod{l}$ for enough primes l to determine t. Hasse's theorem says that $|t| \leq 2\sqrt{q}$, so it suffices to use all primes $l \leq l_{\max}$ such that

$$\prod_{l \leq l_{\max}} l \geq 4\sqrt{q} \tag{6.6}$$

The preceding discussion is the idea behind the following algorithm which computes $\#E(\mathbb{F}_q)$.

Algorithm 4 Schoof's Algorithm

$A := 1$;
$l := 3$;
while $A < 4\sqrt{q}$ **do**
  for $n = 0, \ldots, l-1$ **do**
    if $(x^{q^2}, y^{q^2}) + [q](x,y) = [n](x^q, y^q)$ in the ring R_l **then**
      Break out the loop;
    end if
  end for
  $A = l \cdot A$;
  $n_l = n$;
  $l :=$ next largest prime;
end while
Use the Chinese remainder theorem to find an integer a satisfying $a \equiv n_l \pmod{l}$ for all the stored values of n_l;
return $\#E(\mathbb{F}_q) = q + 1 - a$

Even before going to the detailed description of Schoof's algorithm, we can discuss its complexity already.

6.4 The Complexity

We prove that the running time of Schoof's algorithm is $O(\log^8 q)$. We begin by verifying three claims, having in mind the pseudo-code algorithm 4.

(a) The largest prime l used by the algorithm satisfies $l \leq O(\log q)$.

By classical results in analytic number theory, we know that the prime number theorem implies the statement

$$\lim_{X\to\infty} \frac{1}{X} \sum_{l\leq X, l\text{prime}} \log l = 1$$

Hence $\prod_{l\leq X} l \approx e^X$, so in order to make the product larger than $4\sqrt{q}$, it suffices to take $X \approx \frac{1}{2}\log(16q)$.

(b) Multiplication in the ring R_l can be done in $O(l^4\log^2 q)$ bit operations.

Elements of the ring R_l are polynomials of degree $O(l^2)$. Multiplication of two such polynomials and reduction modulo $\psi_l(x)$ takes $O(l^4)$ elementary operations (additions and multiplications) in the field $\mathbb{F}_q$. Similarly, multiplication in $\mathbb{F}_q$ takes $O(\log^2 q)$ bit operations. So basic operations in R_l take $O(l^4\log^2 q)$ bit operations.

Remark 6.4.1. A bit operation is a basic computer operation on one or two bits. Examples of bit operations include addition, multiplication, and, or, xor, and complement. Observe that, if we use fast exponentiation methods, we can reduce multiplication in R_l to $O((l^2\log q)^{1+\epsilon})$ bit operations, at the cost of a larger big-O constant.

(c) It takes $O(\log q)$ ring operations in R_l to reduce $x^q, y^q, x^{q^2}, y^{q^2}$ in the ring R_l.

In general, the square-and-multiply algorithm (see [10, §XI.1]) allows us to compute powers x^n and y^n using $O(\log n)$ multiplications in R_l. We note that this computation is done only once, and then the points

$$(x^{q^2}, y^{q^2}) + [q \bmod l](x, y) \quad \text{and} \quad (x^q, y^q)$$

are computed and stored for use in step (4) of Schoof's algorithm.

We now use (a), (b), and (c) to estimate the running time of Schoof's algorithm. From (a), we need to use only primes l that are less than $O(\log q)$. There are $O(\log q/\log\log q)$ such primes, so that is how many times the A-loop, steps (2)-(9), is executed. Then, each time we go through the A-loop, the n loop, steps (3)-(5), is executed $l = O(\log q)$ times.

Further, since $l = O(\log q)$, claim (b) says that basic operations in R_l take $O(\log^6 q)$ bit operations. The value of $[n](x^q, y^q)$ in step (4) can be computed in $O(1)$ operations in R_l from the previous value $[n-1](x^q, y^q)$, or we can be inefficient and compute it in $O(\log n) = O(\log l) = O(\log \log q)$ R_l-operations using the double-and-add algorithm.

Hence the total number of bit operations required by Schoof's algorithm is

$$A \text{ loop} \cdot n \text{ loop} \cdot \text{bit operations per } R_l \text{ operation}$$

i.e.

$$O(\log q) \cdot O(\log q) \cdot O(\log^6 q) = O(\log^8 q)$$

bit operations.

This completes the proof that Schoof's algorithm computes $\#E(\mathbb{F}_q)$ in polynomial time.

Remark 6.4.2. If we use fast arithmetic methods, we have that the total cost of Schoof algorithm is actually

$$T = O(\log^{5+o(1)} q)$$

The memory space used by the algorithm is

$$M = \log^3 q$$

because we need to store the division polynomials ψ_l. They have degree $(l^2-1)/2$ and coefficients in $\mathbb{F}_q$.

6.5 The Division Polynomials

Remark 6.5.1. From now on, we will just consider the problem of computing $\#E(\mathbb{F}_q)$ for q prime. In fact we have seen in Theorem 4.2.5(a) how one can easily compute $\#E(\mathbb{F}_q)$, with $q = p^r$, once $\#E(\mathbb{F}_p)$ is known.

Schoof idea to compute easily t modulo primes l relies on the introduction of the so called *division polynomials* ψ_n for the curve E. By their very definition, these polynomials ψ_n cancel exactly on n-torsion points. Let $E[n] = \{P \in E(\overline{\mathbb{F}}_p) \mid nP = 0\}$. We want to define $\psi_n := \psi_n(x, y) \in \mathbb{F}_q[x, y]$ in such a way that

$$\psi_n(x, y) = 0 \Leftrightarrow (x, y) \in E[n]$$

Definition 6.5.2. These polynomials are defined recursively as follows for $n \in \mathbb{Z}_{\geq -1}$.

$$\begin{aligned}
\psi_{-1}(x,y) &= -1, \psi_0(x,y) = 0, \psi_1(x,y) = 1, \psi_2(x,y) = 2y, \\
\psi_3(x,y) &= 3x^4 + 6Ax^2 + 12Bx - A^2, \\
\psi_4(x,y) &= 4y(x^6 + 5Ax^4 + 20Bx^3 - 5A^2x^2 - 4ABx - 8B^2 - A^3), \\
\psi_{2n}(x,y) &= \psi_n(\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2)/2y \quad (n \in \mathbb{Z}_{\geq 1}), \\
\psi_{2n+1}(x,y) &= \psi_{n+2}\psi_n^3 - \psi_{n+1}^3\psi_{n-1} \qquad (n \in \mathbb{Z}_{\geq 1}).
\end{aligned}$$

For practical reasons, we now define the polynomials $f_n(x) \in \mathbb{F}_q[x]$ as follows. First we eliminate all y^2-terms from ψ_n: using the elliptic curve equation we can replace y^2 with $x^3 + Ax + B$. More generally we can replace y^{2k} with $(x^3 + Ax + B)^k$. This allows us to express the division polynomials as elements $\psi_n'(x,y)$ of $\mathbb{F}_q[x]$ or $y\mathbb{F}_q[x]$.

$$f_n(x) = \psi_n'(x,y) \quad \text{if } n \text{ is odd}, \tag{6.7}$$
$$f_n(x) = \psi_n'(x,y)/y \text{ if } n \text{ is even}. \tag{6.8}$$

These polynomials, by definition, also have the property that $f_n(x) = 0$ if and only if x is the x-coordinate of a point of order n. From the recursive formulas for ψ_n given above, one easily deduces that

$$\begin{aligned}
\deg f_n &= \frac{1}{2}(n^2 - 1) \text{ if } n \text{ is odd}, q \nmid n, \\
\deg f_n &= \frac{1}{2}(n^2 - 4) \text{ if } n \text{ is even}, q \nmid n
\end{aligned}$$

Proposition 6.5.3. *Let $P = (x,y) \in E(\overline{\mathbb{F}}_q)$ with $P \notin E[2]$ and let $n \in \mathbb{Z}_{\geq -1}$; then*

$$nP = 0 \Leftrightarrow f_n(x) = 0 \tag{6.9}$$

Proof. See Lang, *Elliptic curves: diophantine analysis*. □

Proposition 6.5.4. *Let $P = (x,y) \in E(\overline{\mathbb{F}}_q)$; let $n \in \mathbb{Z}_{\geq 1}$ with $nP \neq 0$; then*

$$nP = \left(x - \frac{\psi_{n-1}\psi_{n+1}}{\psi_n^2}, \frac{\psi_{n+1}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2}{4y\psi_n^3} \right) \tag{6.10}$$

(By ψ_k we mean $\psi_k(x,y)$).

Proof. See Lang, *Elliptic curves: diophantine analysis*. □

These explicit formulas will enable us to do the computations on l-torsion points of $E(\overline{\mathbb{F}}_q)$ that we need in our algorithm.

6.6 Algorithm Implementation

We are now ready to see the algorithm in detail. After computing a number L for which condition 6.6 holds and after making a list of the polynomials f_n (for $n \in [1, L]$), our second step is to compute $t \pmod{l}$ for different primes $l \leq L$. For the case of $l = 2$ we devise an *ad hoc* procedure, so we consider this case first. For $l > 2$ we will proceed differently, by mean of the Frobenius endomorphism and the division polynomials.

6.6.1 The case $l = 2$

Let $l = 2$. As usual, assume q is odd. Then we have

$$\#E(\mathbb{F}_q) = q + 1 - t \equiv t \bmod 2$$

Now, by Lagrange's and Sylow's theorems, $\#E(\mathbb{F}_q)$ is even if and only if there exists a subgroup of order 2. So in particular

$$t \equiv 0 \bmod 2 \Leftrightarrow \exists P \in E(\mathbb{F}_q) \text{ such that } 2P = O$$

Now, by definition of addition in the group, any element of order 2 must be of the form $P = (x_0, 0)$. Therefore, $t \equiv 0 \bmod 2$ if and only if the polynomial $x^3 + Ax + B$ has a root $x_0 \in \mathbb{F}_q$. Thanks to a basic result in algebra, this is true if and only if $\gcd(x^q - x, x^3 + Ax + B) \neq 1$. To sum up,

$$t \equiv \begin{cases} 1 (\bmod 2) & \text{if } \gcd(x^q - x, x^3 + Ax + B) = 1 \\ 0 (\bmod 2) & \text{if } \gcd(x^q - x, x^3 + Ax + B) \neq 1 \end{cases}$$

6.6.2 The case $l > 2$

We now proceed to see the part of the algorithm in which the computation modulo primes $l \neq 2$ is made explicit. Recall equation 6.5:

$$(x^{q^2}, y^{q^2}) + q(x, y) \equiv \tau(x^q, y^q)$$

where we called $\tau := \bar{t}$. By the Remark 6.3.1, we can compute $t \pmod{l}$ by checking which of the relations

$$\phi_l^2 + q = \tau \phi_l \quad (\tau \in \mathbb{Z}/l\mathbb{Z}) \tag{6.11}$$

holds on $E[l]$. These tests can be effected by computations with polynomials in $\mathbb{F}_q[X, Y]$: let l be a prime not equal to 2 or p and let $P = (x, y) \in E[l]$ not equal to 0. By Proposition 6.5.4 the relation 6.11 holds for (x, y) if and

only if

$$(x^{q^2}, y^{q^2}) + \left(x - \frac{\psi_{q-1}\psi_{q+1}}{\psi_q^2}, \frac{\psi_{q+2}\psi_{q-1}^2 - \psi_{q-2}\psi_{q+1}^2}{4y\psi_q^3} \right) =$$
$$= \begin{cases} 0 & \text{if } \tau \equiv 0 \ (\bmod\ l), \\ \left(x^q - \left(\frac{\psi_{\tau-1}\psi_{\tau+1}}{\psi_\tau^2} \right)^q, \left(\frac{\psi_{\tau+2}\psi_{\tau-1}^2 - \psi_{\tau-2}\psi_{\tau+1}^2}{4y\psi_\tau^3} \right)^q \right) & \text{otherwise} \end{cases} \tag{6.12}$$

(By ψ_k we denote $\psi_k(x, y)$ as before). By Proposition 6.5.3 the point $P = (x, y)$ is in $E[l]$ if and only if $\psi_l(x, y)$ or, equivalently, $f_l(x) = 0$. Using formula 6.1 and the addition formulas of Theorem 2.2.2, the relation 6.12 can be transformed into relations of the form

$$H_1(x) = 0 \text{ and } H_2(x) = 0$$

for some polynomials in $\mathbb{F}_q[X]$. This comes from the fact that $P = (x, y)$ satisfies 6.12 if and only if $-P = (x, -y)$ does. The final test boils down to testing whether

$$H_1 \equiv 0 \pmod{f_l} \text{ and } H_2 \equiv 0 \pmod{f_l} \tag{6.13}$$

in $\mathbb{F}_q[X]$.This test is done for every $\tau \in \mathbb{Z}/l\mathbb{Z}$, until a value of τ is encountered for which 6.6.2 holds; then we have that $t \equiv \tau (\bmod\ l)$. Note that testing 6.11 is equivalent to testing whether $\phi_l^2 + k = \tau\phi_l$, holds on $E[l]$, where $k \equiv q \pmod{l}$ and $1 \leq k < l$.

We will use now formula 6.12; since we use the addition formulas of Theorem 2.2.2 to evaluate 6.12, we distinguish the cases where the points are distinct or not: first test whether there is a nonzero point $P = (x, y)$ in $E[l]$ for which $\phi_l^2 P = \pm kP$ holds. Here $k \equiv q \pmod{l}$ and $1 \leq k < l$. So we must test Whether

$$x^{q^2} = x - \frac{\psi_{k-1}\psi_{k+1}}{\psi_k^2}(x, y)$$

holds or, using $f_m(X)$ rather than $\psi_m(X, Y)$

$$x^{q^2} = \begin{cases} x - \frac{f_{k-1}(x)f_{k+1}(x)}{f_k^2(x)(x^3+Ax+B)} & \text{if } k \text{ is even} \\ x - \frac{f_{k-1}(x)f_{k+1}(x)(x^3+Ax+B)}{f_k^2(x)} & \text{if } k \text{ is odd} \end{cases}$$

Note that the denominators in the above expressions do not vanish on $E[l]$. In order to simplify notation, put

$$g_k(x) := \begin{cases} (x^{q^2} - x)f_k^2(x)(x^3 + ax + b) + f_{k-1}(x)f_{k+1}(x) & k \text{ even} \\ (x^{q^2} - x)f_k^2(x) + f_{k-1}(x)f_{k+1}(x)(x^3 + ax + b) & k \text{ odd} \end{cases}$$

We find that

$$\phi_l^2 P = \pm kP \text{ if and only if } g_k(x) = 0$$

and we can test whether a point like P exists in $E[l]$ by computing $\gcd(g_k(x), f_l(x))$. We have two cases:

Case 1 If $\gcd(g_k(x), f_l(x)) \neq 1$, we have that a point P exists in $E[l]$ with $\phi_l^2 P = \pm qP$.

Case 2 If $\gcd(g_k(x), f_l(x)) = 1$, we have that $\tau \neq 0$ in 6.11, and we test equation 6.5 for various values of τ. In testing 6.11 for these values we can, when adding $\phi_l^2(x, y)$ and $q(x, y)$, apply the version of the addition formulas where the two points have distinct X-coordinates.

We now discuss the two cases in detail. *Case 1*. This is the case where for some nonzero $P \in E[l]$ we have that $\phi_l^2 P = \pm qP$. We'll see that, in this case, $t \in \{0, -2w, +2w\}$, where $w^2 \equiv q \pmod{l}$. If $\phi_l^2 P = -qP$, for some nonzero P, we have by 6.4 that $t\phi_l P = 0$, whence, since $\phi_l P \neq 0$, that $t \equiv 0 \pmod{l}$. If $\phi_l^2 P = qP$ for some nonzero P, we have by 6.4 that

$$(2q - t\phi_l)P = 0 \quad \text{and} \quad \phi_l P = \frac{2P}{t} P$$

(Note that $t \not\equiv 0 \pmod{l}$ since $l \neq 2, p$). From this, by squaring both sides, we deduce that $4q^2 = t^2\phi_l^2 = t^2 q$, i.e. $t^2 \equiv 4q \pmod{l}$. Therefore, q is a quadratic residue. Let $w \in \mathbb{Z}$ with $0 < w < l$ denote a square root of $q \pmod{l}$; this number may be computed by successively trying $1, 2, \ldots$. Once w is found, we have $4w^2 = t^2$, so that $t = \pm 2w$. Now

$$(\phi_l - w)(\phi_l + w) = \phi_l^2 - q = 0 \quad \text{so } \phi_l P = \pm wP$$

and therefore the eigenvalues of ϕ_l acting on $E[l]$ are w and $-w$. We can decide Case l by the following computations.

If $(\frac{q}{l}) = -1$ we clearly have that $t \equiv 0 \pmod{l}$; if not, we compute w, a square root of $q \pmod{l}$ with $0 < w < l$ and we test whether w or $-w$ is an eigenvalue of ϕ_l; if this is not the ease, we conclude that $t \equiv 0 \pmod{l}$ and if indeed a nonzero point P exists with $\phi_l P = \pm wP$, we test whether either $\phi_l P = wP$ or $\phi_l P = -wP$ holds. In the first case we have $t \equiv 2w \pmod{l}$; in the second case, $t \equiv -2w \pmod{l}$. Put

$$h_w(x) := \begin{cases} (x^q - x) f_w^2(x)(x^3 + ax + b) + f_{w-1}(x) f_{w+1}(x) & w \text{ even} \\ (x^q - x) f_w^2(x) + f_{w-1}(x) f_{w+1}(x)(x^3 + ax + b) & w \text{ odd} \end{cases}$$

Computing explicitly, with $w^2 \equiv q \pmod{l}$, we have that

- if $\gcd(h_w(x), f_l(x)) = 1$, we have $\phi_l^2 P = -qP$ so that $t \equiv 0 \pmod{l}$;

- if $\gcd(h_w(x), f_l(x)) \neq 1$, then $t \equiv \pm 2w \pmod{l}$ and we test the y-coordinate of $\phi_l P = \pm wP$ to determine the sign, like follows.

So suppose we know that $\phi_l P = \pm wP$: we need test the y-coordinate of $\phi_l P = wP$. Equation 6.10 gives for the y-coordinate,

$$y^q \equiv \frac{\psi_{w+2}\psi_{w-1}^2 - \psi_{w-2}\psi_{w+1}^2}{4y\psi_w^3} (\text{mod } \psi_l, q)$$

Put

$$r_w := \begin{cases} 4(y^2)^{(q+3)/2} f_w^3(x) - f_{w+2}(x) f_{w-1}^2 + f_{w-2}(x) f_{w+1}^2(x) & w \text{ even} \\ 4(y^2)^{(q-1)/2} f_w^3(x) - f_{w+2}(x) f_{w-1}^2 + f_{w-2}(x) f_{w+1}^2(x) & w \text{ odd} \end{cases}$$

where, as usual, $y^2 = x^3 + ax + b$. Notice that $r_w(x)$ is also a polynomial in x only since all exponents of y are even.

- If $\gcd(r_w(x), f_l(x)) = 1$ then there is no $P \in E[l]$ for which $\phi_l P = wP$, so $t \equiv -2w \pmod{l}$
- If $\gcd(r_w(x), f_l(x)) \neq 1$ such a point exists and $t \equiv 2w \pmod{l}$.

Case 2. This is the case where $\phi_l^2 P \neq \pm qP$ for any $P \in E[l]$. In this case we will test which of the relations 6.11 holds with $\tau \in \mathbb{Z}/l\mathbb{Z}^\times$. We have with $P = (x, y)$ and $k \equiv q \pmod{l}$ and $0 < k < l$, that

$$\phi_l^2 P + qP = \left(-x^{q^2} - x + \frac{\psi_{k-1}\psi_{k+1}}{\psi_k^2} + \lambda^2, y^{q^2} - \lambda \left(-2x^{q^2} - x + \frac{\psi_{k-1}\psi_{k+1}}{\psi_k^2} + \lambda^2 \right) \right),$$

where

$$\lambda = \frac{\psi_{k+2}\psi_{k-1}^2 - \psi_{k-2}\psi_{k+1}^2 - 4y^{q^2+1}\psi_k^3}{4\psi_k y((x - x^{q^2})\psi_k^2 - \psi_{k-1}\psi_{k+1})}$$

Note that the denominator of λ does not vanish on $E[l]$ since ψ_k has no zeros on $E[l]$ and since we are in Case 2. Let $\tau \in \mathbb{Z}$ with $0 < \tau < l$; we have

$$\tau\phi_l P = \left(x^q - \left(\frac{\psi_{\tau+1}\psi_{\tau-1}}{\psi_\tau^2} \right)^q, \left(\frac{\psi_{\tau+2}\psi_{\tau-1}^2 - \psi_{\tau-2}\psi_{\tau+1}^2}{4y\psi_\tau^3} \right)^q \right)$$

In a way analogous to the computations above one can test, by computations in $\mathbb{F}_q[X]$, which of the relations 6.11 holds by trying $\tau = 1, \ldots, l-1$. The computations involve evaluating polynomials modulo $f_l(X)$ and testing whether they are zero $f_l(X)$. We do not give all the details here, since they are quite long, but it is not difficult to fill in the details. Long story short, testing whether $\phi_l^2 + q = \tau\phi_l$, holds on $E[l]$ boils down to testing whether

$$((\psi_{k-1}\psi_{k+1} - \psi_k^2(X^{q^2} + X^q + X))\beta^2 + \psi_k^2\alpha^2)\psi_\tau^{2q} + \psi_{\tau-1}^q\psi_{\tau+1}^q\beta^2\psi_k^2$$

and

$$4Y^q\psi_\tau^{3q}(\alpha((2X^{q^2}+X)\beta^2\psi_k^2-\psi_{k-1}\psi_{k+1}\beta^2+\psi_k^2\alpha^2)-Y^{q^2}\beta^3\psi_k^2)+ \\ -\beta^3\psi_k^2(\psi_{\tau+2}\psi_{\tau-1}^2-\psi_{\tau-2}\psi_{\tau+1}^2)^q \tag{6.14}$$

are zero mod $f_l(x)$. Here

$$\alpha=\psi_{k+2}\psi_{k-1}^2-\psi_{k-2}\psi_{k+1}^2-4Y^{q^2+1}\psi_k^3$$

and

$$\beta=((X-X^{q^2})\psi_k^2-\psi_{k-1}\psi_{k+1})4Y\psi_k$$

By the expressions 6.14 we understand the polynomials in $\mathbb{F}_q$ one gets after eliminating Y using 6.14 and, if necessary, by dividing the expressions by Y. The result is a polynomial in $\mathbb{F}_q[X]$.

This completes the description of the second step of the algorithm.

6.7 Putting all Together

To sum up, the steps of the algorithm are

Step 1 Compute a number L for which condition 6.6 holds and of making a list of the polynomials f_n for $n=1,2,\ldots,L$.

Step 2 Computation of $t \pmod l$ for every prime $l \leq L$ not equal to p.

Step 3 Computation of t from the values of t mod l obtained using the Chinese Remainder Theorem and the estimate 6.3

Remark 6.7.1. Step 3 is straightforward. This completes the description of the algorithm.

6.8 Improvements

In the 1990s, Noam Elkies, followed by A. O. L. Atkin, devised improvements to Schoof's basic algorithm by restricting the set of primes $S=\{l_1,\ldots,l_s\}$ considered before to primes of a certain kind. These came to be called Elkies primes and Atkin primes respectively. A prime l is called an Elkies prime if the characteristic equation: $\phi^2-t\phi+q=0$ splits over $\mathbb{F}_l$, while an Atkin prime is a prime that is not an Elkies prime. Atkin showed how to combine information obtained from the Atkin primes with the information obtained from Elkies primes to produce an efficient algorithm, which came to be known as the Schoof-Elkies-Atkin (SEA) algorithm. The first problem to address is to determine whether a given prime is Elkies or Atkin. In order to do so, we make use of modular polynomials, which come

from the study of modular forms and an interpretation of elliptic curves over the complex numbers as lattices. Once we have determined which case we are in, instead of using division polynomials, we proceed by working modulo the modular polynomials f_l which have a lower degree than the corresponding division polynomial ψ_l (degree $O(l)$ rather than $O(l^2)$). This results in a further reduction in the running time, giving us an algorithm more efficient than Schoof's, with complexity $O(\log^6 q)$ for standard arithmetic and $O(\log^4 q)$ using fast exponentiation techniques.

4 Constructing Elliptic Curves

Now that we have presented the relevant theory of elliptic curves, j-functions and quadratic fields, it is time that we look at the task of constructing elliptic curves with a certain number of points over a prime field. In this section we will present a general method for such construction, which is based on the theory of complex multiplication and the class field theory of imaginary quadratic fields. We will refer to this simply as the *complex multiplication method*, or 'CM method' for short.

We begin this section by briefly describing a trivial "brute force" procedure for finding an elliptic curve with a given cardinality. Next we present the CM method and discuss computational complexity and other practical aspects. The main step of the CM method, constructing the class polynomial for discriminant D, will be covered in detail in §5.

4.1 Naive Method

A naive method to produce an elliptic curve with N rational points in $\mathbb{F}_p$ is to randomly pick an element $a \in \mathbb{F}_p \backslash \{\frac{-27}{4}\}$ and try whether the elliptic curve

$$E_a : y^2 = x^3 + ax - a$$

has N rational points. We observe that the point point $P = (1, 1)$ is on E_a for all a. By writing $N = p + 1 - t$, we check whether P is annihilated by either $N = p+1-t$ or $N' = p+1+t$. If it is, then we know that E_a has $p+1\pm t$ rational points. If P was annihilated by N then we are finished but if it was annihilated by N' then we take the quadratic "twist" of E_a by equation (17). The procedure Naive-Method(p, N) illustrates this method.

NAIVE-METHOD(p, N)

```
1  P ← (1, 1)
2  t ← p + 1 − N
3  i ← 0
4  S ← F_p \ {−27/4}
5  repeat
6      Pick a random a ∈ S
7      E_a ← y^2 = x^3 + ax − a
8      if P is annihilated by p + 1 − t then
9          return E_a
10     elseif P is annihilated by p + 1 + t then
11         return quadratic twist of E_a
12     end
13     S = S − {a}
14 until S = ∅
```

Although the distribution of group orders $|E_a(\mathbb{F}_p)|$ is not even among the elements $a \in \mathbb{F}_p$, we can expect to check approximately $O(\sqrt{p})$ curves on average before finding a right one. According to Bröker and Stevenhagen [4] the expected running time of the naive algorithm is $O(\sqrt{p}) \times$ (constructing curve + multiplying P + counting points) $= O(N^{1/2+\epsilon})$, for some small $\epsilon > 0$. When N is small it may be feasible to use the naive method. All the other alternatives we will discuss in this essay are only assymptotically faster in N and may very well be slower for small inputs. However for large N, say $N \gg 10^{10}$, the naive method becomes quite impractical.

4.2 Complex Multiplication Method

By the Deuring Lifting Theorem we can consider every elliptic curve over $\mathbb{F}_p$ as the reduction of some elliptic curve over a number field K with the same endomorphism ring. Our task is to construct a curve having exactly N rational points over $\mathbb{F}_p$.

Let K be the imaginary quadratic field of discriminant D where p splits as the product of two elements. If we look at elliptic curves over K with complex multiplication by the full ring of integers $\mathcal{O}_K$ in K then we are able to apply Theorem 3.9 which immediately gives us the desired cardinality over $\mathbb{F}_p$. To

find such a field K we choose a fundamental discriminant $D < 0$ such that $4p = t^2 + s^2|D|$ has a solution for $t = p + 1 - N$ and s any integer. Then, by Theorem 2.4, (p) splits into two distinct ideals in K. In other words, p is a norm in K and $p = \pi\overline{\pi}$, where π is an element of $\mathcal{O}_K$. If we next find the equation of an elliptic curve E over $\mathbb{C}$ with $\mathrm{End}_{\mathbb{C}}(E) \simeq \mathcal{O}_K$ then by Theorem 3.9 the reduction of E modulo p will give us a curve with $p + 1 - t = N$ rational points over $\mathbb{F}_p$. We know that $j(E)$ is an algebraic integer of degree equal to $h(D)$, the class number of K. By Theorem 2.4 we know that our choice of D implies that the minimal polynomial $H_D(X)$ of $j(E)$ splits completely into linear factors modulo p. This allows us to consider a root of $H_D(X) \equiv 0 \pmod{p}$ as the j-invariant of $\tilde{E}$, the reduction of E modulo p. We finally construct $\tilde{E}$ by the formulas given in §3.6 and that solves our task. The interaction of theory which underlies the CM method is further illustrated in the following diagram.

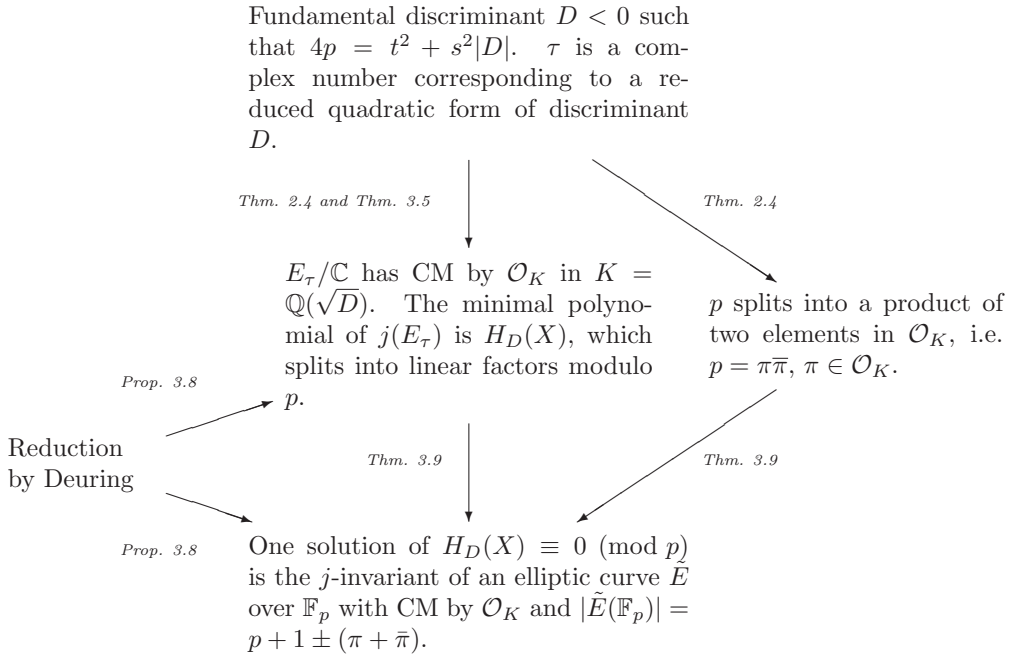

Figure 1: Basic theory behind the complex multiplication method

Let us assume that $D < -4$. Once we know the j-invariant in $\mathbb{F}_p$, the elliptic curve with $p + 1 - t$ points is constructed from one of the two equations (17). To

see whether we have the right curve we can pick a random rational point and see whether it is annihilated by N. If that is not the case then we have selected a curve with $p+1+t$ points and the opposite equation gives the right choice. The complex multiplication method is summarised by the following procedure.

Remark. The two special cases $D = -3$ and $D = -4$ can be handled in a manner similar to the general case. To select between the different isomorphism classes in each case we can use certain algorithms instead of multiplying random points as in the general case. We refer the reader to [2] for full details.

CM-Method(p, N)

1 $t \leftarrow p + 1 - N$
2 Find a fundamental discriminant D which satisifies $4p = t^2 + s^2|D|$ for $s \in \mathbb{Z}$.
3 Construct the Hilbert class polynomial $H_D(X)$.
4 Compute a solution j_0 of $H_D(X) \equiv 0 \pmod{p}$.
5 Construct the equation of an elliptic curve E over $\mathbb{F}_p$ of invariant j_0.
6 Find a random point P on E.
7 **if** $[N]P \neq O_E$ **then**
8 $\quad E \leftarrow$ quadratic "twist" of E
9 **end**
10 **return** E

4.3 Computational Aspects

Apart from the construction of the Hilbert class polynomial $H_D(X)$, which we will look at in detail in §5, the other hard step in this algorithm is computing a root of $H_D(X) \equiv 0 \pmod{p}$. Many standard factorisation algorithms are known for that purpose. See for example Cohen [5, §1.6.1] or Knuth [8, §4.6.2].

Finding a "good" discriminant D that is suitable for our prime p is quite simple, as we are expecting a certain cardinality N (the situation becomes more involved when we use this method for primality testing, as we will see in §6). We set $t = p + 1 - N$ and

$$D = \frac{t^2 - 4p}{s^2} = \frac{(p + 1 - N)^2 - 4p}{s^2},$$

and search for an s such that s^2 divides $(t^2 - 4p)$ and $D < 0$ is fundamental and as small as possible. The condition that D is small is important as we expect

the algorithm to run in time asymptotic to a power of $|D|$, as we will soon see[2]. Of course, a necessary condition for this algorithm to produce a solution is that the integer N is contained in the Hasse interval $\mathcal{H}_p$. In fact, since we restrict to the case of a prime field $\mathbb{F}_p$, all integers in $\mathcal{H}_p$ do occur as the group order $E(\mathbb{F}_p)$ of some elliptic curve E over $\mathbb{F}_p$ [4].

We should note one special case for the algorithm, namely when $N = p + 1$. Then by Theorem 3.7 we know that we can pick any supersingular curve over $\mathbb{F}_p$. There are many criteria for supersingular curves, see for example [16, §V, Theorem 4.1]. As an example, the elliptic curve $E/\mathbb{F}_p$ defined by $y^2 = x^3 + 1$ is supersingular if and only if $p \equiv 2 \pmod 3$.

Finally we remark on the computational complexity of the method. The two time consuming steps in the algorithm are the construction of the Hilbert class polynomial $H_D(X)$ and the computation of a root of $H_D(X)$ modulo p. Crucial to the complexity analysis is the estimate $\log(h(D)) \sim \log(\sqrt{d})$ [11, §XVI.4], where we write $d = |D|$. It follows that the approximation $h(D) \sim \sqrt{|D|}$ should not be too bad. We will see in the next section that the basic complex analytic method to construct $H_D(X)$ takes time $O(d^2(\log d)^2)$. By one approximation [12, §5.10] it takes time $O(d(\log p)^3)$ to calculate a solution to $H_D(X) \equiv 0 \pmod p$. Which one of these two steps will dominate the running time of the algorithm depends of course on the relative size of d and p. In general, we expect the $O(d^2(\log d)^2)$ term to prevail if we seek elliptic curves with a large number of rational points. The other steps in the algorithm count less towards the overall complexity. For example, there is an algorithm to compute $[m]P$, for an integer m and a point P on E, in time asymptotically $O(\log m)$ [14].

[2] In elliptic curve cryptography the discriminant has to be of certain minimal size to ensure security. According to [1] some cryptography standards recommend using elliptic curves with complex multiplication by an order of discriminant at least equal to 200.