EXAMINING SCHOOF'S ALGORITHM FOR FINDING THE NUMBER OF POINTS ON AN ELLIPTIC CURVE

BENJAMIN POMERANZ

1. Background for the Algorithm

First, we will take a look at the algorithm without going in depth on the mathematics. This is a version of Schoof's algorithm that is not completely optimized, and further improvements on this algorithm were made in the 1990s as well. However, to understand its approach to finding the number of points on an elliptic curve, the original algorithm is what we will examine.

We will use ψ_l to refer to $\psi_l(x)$, the l-th division polynomial. See [2], chapter 3.2 for more on Division polynomials. They have the following important traits for Schoof's Algorithm: Division polynomials are polynomials $\in \mathbb{Z}[A, B, x, y]$ but they are a function of only x, A, B for odd l, and l is always odd in the case where we are computing ψ_l in this algorithm. Moreover, we plug in A, B from the elliptic curve, so when we use ψ_l we can assume it is a polynomial in $\mathbb{Z}[x]$. Amazingly, the roots of $\psi_l(x)$ over the field $\mathbb{F}_q$ are the x coordinates of all points in the torsion group $E[l]$ on the elliptic curve defined by A, B over that field.

We'll try not to confuse this with

$$\phi_q(x, y) = (x^q, y^q)$$

the Frobenius (hey! Frobenius!) Endomorphism. Importantly, the Frobenius Endomorphism does not raise a point to the q-th power, but rather acts on the x and y coordinates. It has the following property, proved in [2] as theorem 4.10:

$$\phi_q^2 - a\phi_q + q = 0$$

This is proved to be true uniquely for a defined as

$$a = \#(\mathbb{F}_q) - q - 1$$

Note also that we can use modular arithmetic and the Euclidean algorithm almost identically with polynomials of one variable as we do with integers. So, if $a \equiv x + 4 \mod x^2 + 6x + 3$, we have $a - (x + 4) = r(x^2 + 6x + 3)$ for some polynomial r. We will denote the point at infinity on the Elliptic Curve as O. We are computing

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

which amounts to computing a since we know q.

As is outlined on page 127 of [2] Washington's "Elliptic Curves: Number Theory and Cryptography," the algorithm goes as follows for an elliptic curve $y^2 = x^3 + Ax + b$, over a field $\mathbb{F}_q$. Note that all operations are assumed to occur in that field. Note that we can assume $q = p^n$, for a prime p, because it is the order of a finite field. Thus, the characteristic of $\mathbb{F}_q = p$.

2. Schoof's Algorithm

(1) Choose a set of primes S, where none of the primes are p, such that

$$\prod_{l \in S} l > 4\sqrt{q}$$

(2) If $l = 2$, we have $a \equiv 1 \mod (2)$ iff $\gcd(x^3 + Ax + B, x^q - x) = 1$

(3) For odd primes, we do the following to find $a \mod (l)$:

(a) $q_l := q \mod (l)$ s.t. $|q_l| < l/2$

(b) Compute the x coordinate of

$$((x^{q^2}, y^{q^2}) + q_l(x, y)) \mod \psi_l$$

and call it x'.

(c) For j in $1, 2, \ldots, (l-1)/2$:

(i) Calculate the x coordinate of $(x_j, y_j) := j(x, y)$

(ii) If $x' - x_j^q \equiv 0 \mod \psi_l$, go to (iii). If not, go to the next j. If no j's satisfy this, go to (d).

(iii) Compute y', y_j. If $(y' - y_j^q)/y \equiv 0 \mod \psi_l$, then $a \equiv j \mod l$. Else, $a \equiv -j \mod l$.

(d) Let $w^2 \equiv q \mod l$. If w does not exist (q is not a square in $\mathbb{F}_l$), then $a \equiv 0 \mod l$.

(e) Finally, if we have w, then check whether $\gcd(\text{numerator of } x^q - x_w, \psi_l)) = 1$. If so, $a \equiv 0 \mod l$. If not, calculate $\gcd(\text{numerator of } (y^q - y_w)/y, \psi_l))$. If this gcd is not 1, we have $a \equiv 2w \mod l$. Otherwise, $a \equiv -2w \mod l$.

(4) Now, use the Sunzi Remainder Theorem with all of these assorted congruencies to find $a \mod \prod_{l \in S} l$, and choose the unique value of a s.t. $|a| \leq \sqrt{2}q$.

Now, we just compute $q + 1 - a$ and there we have it. Easy! Well, maybe not. That was pretty confusing, and I'm unconvinced it even works. Why are we modding by some random polynomial? Where are all of these congruencies coming from? Let us go through each step of the algorithm, breaking down why this works, up to nodding at some theorems that you can find proofs for in [2] or online.

3. Wait, what? Why?!?

Firstly, lets address (1). What is the significance of $4\sqrt{q}$ as a modulus? Well, Helmut Hasse proved in 1933 that the number of points on an elliptic curve is within $2\sqrt{q}$ of $q+1$, so $|a| \leq 2\sqrt{q}$. However, because a could be positive or negative if we find $a \mod 2\sqrt{q}$, for example, then the true value of a could be $a\%2\sqrt{q}$ or $-(q - (a\%2\sqrt{q})$, which both have absolute values less than $2\sqrt{q}$. So, to uniquely determine a such that both the positive and negative possible a's cannot have values $\leq 2\sqrt{q}$, we need to find $a \mod$ a number exceeding $4\sqrt{q}$.

Next, point (2). This is actually just checking if the whole group of points on the elliptic curve has even order. If it does, then $q + 1 - a \equiv 0 \mod (2)$, and since $q + 1$ is even a must also be. If and only if there is a point of order 2 in a group, the group has even order, and on an elliptic curve a point of order 2 is just a root. So, to check for a root we check if $x^3 + Ax + b$ has a root in $\mathbb{F}_q$, and the elements of $\mathbb{F}_q$ are equivalent to the roots of $x^q - x \mod q$. Think about why this is true - it is a corollary of Fermat's Little Theorem. So, if our elliptic curve shares a root with $x^q - x$ over $\mathbb{F}_q$, it has a point $(e, 0)$ of order 2 and thus is a group of order 2 and thus $a \equiv 0 \mod 2$.

We can check for this by taking $\gcd(x^3 + Ax + B, x^q - x)$, as if they share a root they share some polynomial factor. So, if they don't and their gcd is 1, there are no roots on the curve, the order of the group is odd, and we know $a \equiv 1 \mod 2$. Otherwise it is 0.

Now, onto the general case of odd primes, which is a little trickier. We have a little whiles to go before we get to step (3), so bear with me. Section 3.2 in [2] outlines more on division polynomials and proves that for a point (x, y) on our elliptic curve, (x, y) has order $n \iff \psi_n(x) = 0$. This is one of the theorems that we won't prove here. Remember, as shown in theorem 4.10 in [2]:

$$\phi_q^2 - a\phi_q + q = 0$$

Note that this equation is referring to operations on a point (x, y), so q is applied to a point as a coefficient not added as an integer value.

Ok, now take a point, generically written as (x, y). Set

$$q_l = q \mod l$$

s.t.

$$|q_l| < l/2$$

Then, by 4.10, we have that

$$(1) \qquad (x^{q^2}, y^{q^2}) - a(x^q, y^q) + q(x) = 0 \Rightarrow$$

$$(2) \qquad (x^{q^2}, y^{q^2}) + q_l(x, y) = a(x^q, y^q)$$

Well, we have a few points of order l, and a coefficient q_l which is taken $\mod l$, and our point a as a coefficient for a point of order l, so this determines $a \mod l$. That is to say, if the equation holds when we plug in $a = j$ for some j, then $j \equiv a \mod l$.

Now, we can actually start working towards a value for $a \mod l$. Firstly, we don't want the two points on the LHS of (2) to be inverses or the same point, so we will treat that as a separate case: let's assume there exists a point (x, y) of order l such that

$$(x^{q^2}, y^{q^2}) \neq \pm q_l(x, y)$$

. We will deal with the case where there are no such points later. Then, we set

$$(x', y') := (x^{q^2}, y^{q^2}) + q_l(x, y) = a(x^q, y^q)$$

As a convention, we will say that for an integer j, $j(x, y) := (x_j, y_j)$. Now, we can find the values of (x', y'), using elliptic curve addition and multiplication, adding the points in the equation above which do not share an x value. Then, we compare this to different values of $a(x^q, y^q)$, to see when the equation holds. Lets call these values we are plugging in for a, j. Note that we are working with polynomials, and not any specific points (a, b), but we can of course find this new point (x', y') in terms of x and y, and we have y^2 in terms of x by the equation for the elliptic curve up to being positive or negative, so we can write x', y' as rational functions of x if y' has no odd y terms, and if it does we can just divide by y. When we find a j that solves this equation in x, we have $|a| \mod l$, and we use the y coordinate to find the sign.

So, this takes us to (c)-(ii). We have plugged in a value j for a in the equation, and are checking if the x_j^q value is the same as x'. However, recall that we are looking for this result for points of order l, or else it doesn't give us the information we want about a. So, how do we know that a

point is of order l? This takes us back to division polynomials - if $\psi_l(x) = 0$, we have (x, y) is of a point of degree l. So, if as we check in (c)-(ii),

$$x' - x_j^q \equiv 0 \mod \psi_l$$

this implies that

$$x' - x_j^q = n\psi_l$$

for some n, and thus if we plug in a a root of ψ_l, which is equivalent to a point of order l, we get $x' - x_j^q = 0$, and thus we have found the j value which completes our equation for a point (x, y) of order l. However, we still don't know if j is positive or negative as that only affects y_j^q, and we have been looking at the x coordinates. y' and y_q^j both are raised to an odd power, so we can divide by y and then write these values as a function of x. So, by a similar argument to the x coordinate, we expect

$$\frac{y' - y_j^q}{y} \equiv 0 \mod \psi(l)$$

so just plug in positive j, and if it satisfies that congruence, we have found our j. Otherwise, use negative j. Note, that is exactly what we do in step (c)-(iii). Now, we will move on from (c) to (d), the case where there is no j that satisfies our equation.

We now must be in the case we ignored earlier where

$$\phi_q^2(x, y) = (x^{q^2}, y^{q^2}) = \pm q(x, y)$$

for all $(x, y) \in E[l]$. Let's first assume

$$\phi_q^2(x, y) = (x^{q^2}, y^{q^2}) = q(x, y)$$

In this case, we have

$$a\phi_q(x, y) = \phi_q^2(x, y) + q(x, y) = 2q(x, y)$$

by rearranging (1), and

$$a^2 q(x, y) = a^2(\phi_q^2)(x, y) = (a\phi_q)^2(x, y) = (2q)^2(x, y)$$

This gives that

$$a^2 q = 4q^2 \mod l$$

which implies that $q = (a^{-1} \cdot 2 \cdot q)^2$ and we'll write $q = w^2$.

If q is not a square then we must be in the case where $\phi_q^2(x, y) = -q(x, y)$, or (3)(d) in the algorithm. In that case, we have $a(x^q, y^q) = -q(x, y) + q(x, y) \Rightarrow a \equiv 0 \mod l$.

Now, for the case were q is a square we have

$$(\phi_q - w)(\phi_q + w) = \phi_q^2 - q(x, y) = q(x, y) - q(x, y) = O$$

So, there exists a point $P \in E[l]$ s.t. $\phi_q P = \pm wP$, and this implies that $\phi_q P$ and wP have the same x coordinate for some point, which we check by asserting

$$\gcd(x^q - x_w, \psi_l) \neq 1$$

The reason we use gcd here as opposed to the earlier $\mod \psi_l$ is that we only need this to be true of a single point in P, and the relation does not necessarily hold for every point of order l simultaneously. So, we check whether the two polynomials share a root, rather than whether they share all roots.

If our assertion holds, and there is a point P for which $\phi_q P = \pm wP$, then we have

$$O = (\phi_q^2 - a\phi_q + q)P = (q \pm aw + q)P \Rightarrow 2q = 2w^2 = \pm aw \Rightarrow a = \pm 2w$$

Note that if you select $+$ or $-$ the math is the same throughout. So, in step $(3)(e)$ of the algorithm, we check the prior assertion, and if it holds we find whether we are in the positive or negative case using the y coordinate similarly to in step $(c)(iii)$. If the assertion doesn't hold, we must be outside of the case where $\phi_q^2 = q_l$, and instead are in the case where $\phi_q^2 = -q_l$ and thus $a \equiv 0 \mod l$.

Now, we just use the Sunzi Remainder Theorem to find $a \mod N$, where $N > \sqrt{4}q$, and pick a s.t. $|a| \leq \sqrt{2}q$, and we have arrived at a.

4. Code

I had hoped to go through a similar example to the book using Sympy, [1], a library to manipulate polynomials in Python. However, my attempt was unsuccessful for reasons I am unsure of. Even evaluating a congruence Washington gives in their example to test the algorithm did not return the value (0) which it should have. This is all included in the code file as SchoofExamples.py, and you can see more there. However, I instead enjoyed implementing a function to test whether an elliptic curve over a prime field has an even number of points, which is the first step of Schoof's algorithm. I wrote a method to test this for many points, and one to plot these as well in a 3-d graph (see evenCurves.py to play with all of these). For A and B going from 1 to 100 and q a prime less than 1000, the proportion of even elliptic curves is 68.45%!

```
x, y = sp.symbols('x y')

def is_even(A, B, q) -> bool:
    F = sp.FiniteField(q)
    y2 = sp.poly(x**3 + A*x + B, domain = F)
    field_poly = x**q - x
    gcd = sp.gcd(y2, field_poly, domain=F)
    if (gcd == 1):
        return False
    else:
        return True

def proportion_even(a_list, b_list, q_list):
    num_even = 0
    total = 0
    for A in a_list:
        for B in b_list:
            for q in q_list:
                total+=1
                if is_even(A, B, q):
                    num_even += 1
    return num_even / total
```

References

[1] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, Jan. 2017.

[2] L. C. Washington. *Elliptic Curves: Number Theory and Cryptography, Second Edition*. Chapman & Hall/CRC, 2 edition, 2008.