2015

A Cryptographic Attack: Finding the Discrete Logarithm on Elliptic Curves of Trace One

Tatiana Bradley
Scripps College

A Cryptographic Attack: Finding the Discrete Logarithm On Elliptic Curves of Trace One

Tatiana Bradley

Lenny Fukshansky, Advisor
Christopher Towse, Reader

Submitted to Scripps College in Partial Fulfillment
of the Degree of Bachelor of Arts

March 13, 2015

Department of Mathematics

Abstract

The crux of elliptic curve cryptography, a popular mechanism for securing data, is an asymmetric problem. The elliptic curve discrete logarithm problem, as it is called, is hoped to be generally hard in one direction but not the other, and it is this asymmetry that makes it secure.

This paper describes the mathematics (and some of the computer science) necessary to understand and compute an attack on the elliptic curve discrete logarithm problem that works in a special case. The algorithm, propsed by Nigel Smart, renders the elliptic curve discrete logarithm problem easy in both directions for elliptic curves of so-called "trace one." The implication is that these curves can never be used securely for cryptographic purposes. In addition, it calls for further investigation into whether or not the problem is hard in general.

Contents

Acknowledgments

Firstly, I would like to give a huge thank you to my advisors Lenny Fukshansky and Chris Towse.

Professor Towse sparked my initial interest in cryptography, which led not only to this thesis but my decision to pursue Computer Security in graduate school next year. In addition, his help in the initial stages were instrumental to formulating the scope and structure of this paper, and developing the breadth of knowledge necessary to write it.

Professor Fukshansky graciously agreed to dive into an unfamiliar area, and did a phenomenomal job helping me understand the last tricky points I needed to bring everything together. Also, his coffee-making skills are second to none.

Finally, thank you to my best friend Rose DuCharme, for keeping me from hyperventilating, and my parents, for supporting and encouraging me in my roundabout journey towards mathematics.

Chapter 1

Introduction: Cryptography, Elliptic Curves, and an Asymmetric Problem

1.1 A Quick Introduction to Public Key Cryptography

Cryptography concerns itself with transforming data (a plaintext) into a so-called ciphertext that cannot be transformed back into its original state without access to secret information.

What distinguishes a public key encryption from any other form is that the operations used to transform the plaintext to the ciphertext are completely different from those that transform the ciphertext into the plaintext. The main advantange of this asymmetry is that it allows for agents to communicate secretly without ever needing to exchange private data, as the information needed to send a message can be published without revealing the method needed to decipher it.

The crux of any public key cryptosystem is the transformation, which is almost always a mathematical operation, that takes a plaintext to a ciphertext but does not work (easily) in reverse. Such a function must be like a trapdoor: it must be easy to get into, but hard to get out of.

One such a function, the elliptic curve discrete logarithm problem, was initally proposed independently by Neal Koblitz and Victor S. Miller in

1985 for use in cryptography, and is in wide use today. Unfortunately, as we will see, there are some cases in which this function is not a trapdoor at all.

This paper concerns itself with the algorithm proposed by Nigel Smart that quickly solves the ECDLP problem for a certain type of elliptic curve, with a special property called "trace one". We give an overview of the mathematics needed to understand the method, and show how a computer may easily perform it. Note that in practice, this sort of curve is easily avoided, and this method does not mean that elliptic curve cryptography is useless in general. We will not describe any actual cryptosystems, but those interested in this may see [HMV04], and excellent reference on the specific mechanisms for encrypting and decrypting data using elliptic curves.

1.2 Elliptic curves

Before we can understand the elliptic curve discrete logarithm problem, we must have a sense of what an elliptic curve is. It is a set of points that satisfy a certain equation, that has algebraic (and geometric) structure.

The equation we must satisfy is as follows:

Definition 1.1. Let K be a field. We define a **Weierstrass equation**:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in K$ and E is non-singular, meaning it has no multiple roots.

Definition 1.2. For K a field, the **K-rational points** on a Weierstrass equation E over K are the pairs $(x, y) \in K \times K$ that satisfy E, together with $\mathcal{O}$, the point at infinity (described in the Section 1.3 and again in Section 2.1). The set of K-rational points and is denoted $E(K)$, and describes an elliptic curve.

An elliptic curve can be defined over any field K, but we make most use of the cases when K is a finite field $\mathbb{F}_p$ where p is prime, or K is some

subset of the p-adics $\mathbb{Q}_p$ (which are introduced in Chapter 2).

Definition 1.3. Recall that the **characteristic** of a field K, denoted char(K) is the order of the smallest prime subfield of K, or 0 if K has no smallest prime subfield (for example, $\mathbb{R}$ or $\mathbb{Q}$).

Thus fields $\mathbb{F}_p$ where p is prime have characteristic p. When $\text{char}(K) \neq 2$ or 3, we can change variables to arrive at a **simplified Weierstrass equation**:

$$E : y^2 = x^3 + ax + b,$$

where $a, b \in K$. The simplified equation will suffice for our purposes. E is non-singular if the discriminant, $\Delta = -16(4a^3 + 27b^2)$, is not zero.

Example 1.4. Let $\tilde{E}$ be defined over $\mathbb{F}_{19}$ by the equation:

$$\tilde{E} : y^2 = x^3 + x + 4.$$

(Note that our convention, which is not necessarily standard, is to use $\tilde{E}$ to denote an equation defined over a finite field).

This is a valid Weierstrass equation because the discriminant is non-zero:

$$\Delta = -16(4 \cdot 1^3 + 27 \cdot 4^2) = -6979 \neq 0.$$

The 19 $\mathbb{F}_{19}$-rational points that satisfy $\tilde{E}$ are:

$\mathcal{O}$	$(0, 2)$	$(0, 17)$	$(1, 5)$	$(1, 14)$
$(5, 1)$	$(5, 18)$	$(6, 6)$	$(6, 13)$	$(8, 7)$
$(8, 12)$	$(9, 1)$	$(9, 18)$	$(10, 8)$	$(10, 11)$
$(11, 4)$	$(11, 15)$	$(14, 8)$	$(14, 11)$.	

These points form the set $\tilde{E}(\mathbb{F}_p)$, which, as we will see shortly, is a group. Note that the number of points on an elliptic curve is not always the same as the characteristic of its underlying field, and that this curve has the special property of having "trace one."

Definition 1.5. Let E be an elliptic curve over a finite field $\mathbb{F}_p$. The **trace** t

is defined by:

$$\#E(\mathbb{F}_p) = p + 1 - t,$$

where $\#E(\mathbb{F}_p)$ is the number of elements in $E(\mathbb{F}_p)$.

Remark. The trace is equal to one if and only if $E(\mathbb{F}_p)$ (including $\mathcal{O}$) has exactly p elements. This is the case in which we are able to solve the ECDLP quickly, as we will see in Chapter 4.

1.3 The group law

The set of points on an elliptic curve forms an abelian group under a special operation, which we call "addition". We first describe this addition geometrically, and then define it algebraically in a more formal way.

The geometric group law

Since the geometric group law only makes sense over a subset of $\mathbb{R}$, and not over the fields we will deal with, we give only an informal overview of it. To add two points P and Q on an elliptic curve $E(K)$:

1. Draw a straight line between P and Q, and find the third point of intersection R, which must exist as a point in $E(K)$.

2. Reflect R about the x-axis. This point, denoted $-R$, is equal to $P + Q$, is also in $E(K)$.

To add a point P to itself, follow the same method except begin by drawing a tangent line through P.

Note that in certain cases, the line drawn will be vertical. When this happens, the third point of intersection is called $\mathcal{O}$, the "point at infinity." This has the result that a point added to its negation is $\mathcal{O}$, which means that $\mathcal{O}$ is a natural identity element.

The algebraic group law

Let K be a field with $char(K) \neq 2$ or 3, and let $E : y^2 = x^3 + ax + b$ with $a, b \in K$. We define the operation $+$ (called point addition) on $E(K)$, the set of K-rational points of E (including $\mathcal{O}$).

If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are in $E(K)$ then:

1. (Identity.) $\mathcal{O} + P = P$ and $P + \mathcal{O} = P$.
2. (Additive inverses.) $-P = (x_P, -y_P)$ is the the additive inverse of P and $P + (-P) = \mathcal{O}$.
3. (Point Doubling.) If P is not its own inverse, then $P + P = 2P = (x_{2P}, y_{2P})$ where
$$x_{2P} = N^2 - 2x_P,$$
$$y_{2P} = N(x_P - x_{2P}) - y_P,$$
$$\text{where } N = \frac{3x_P^2 + a}{2y_P}.$$
Of course, if $P = -P$, then $2P = \mathcal{O}$.
4. (Point Addition.) If $P \neq Q$ and $P \neq -Q$ then $P + Q = R = (x_R, y_R)$ where
$$x_R = M^2 - x_P - x_Q,$$
$$y_R = M(x_P - x_R) - y_P,$$
$$\text{where } M = \frac{y_Q - y_P}{x_Q - x_P}.$$

Note that the operations used to define the computation of x and y coordinates are field operations. So, to do point operations, we must know the arithmetic of the underlying field. In the case of $K = \mathbb{F}_p$, we use modular arithmetic.

Notation. Let P be a point on an elliptic curve. We use the notation $[n]P$ to denote scalar multiplication of P by a non-zero positive integer n. In other

words,

$$[n]P = P + P + ... + P \text{ (n times)}.$$

1.4 Asymptotic analysis and Big-O Notation

We take a moment here to introduce a well-known way of describing the performance of an algorithm, called "big-O" notation.

Suppose we have some function `f(n)` and we wish to know how long it takes to run. We could let a compute `f(n)` for many different values of n on a computer, measure the time it takes to run each, and try to extrapolate a pattern from how long these computations take. (This process is called benchmarking). One of the problems with this is that computer-specific (or random) factors can change the results, which makes it harder to compare algorithms without running many tests in a controlled environment. Another is that not every possible value of n can be checked, so a pattern that appears may be misleading. One solution is to use asymptitic analysis (also called complexity analysis) to abstract away these annoying details.

Informally, the way to do this is to count up all the "work" a function does in terms of a measure of problem size (usually denoted n), and find an expression for this work. The standard unit of work is any operation or combination of operations that can be done in so-called "constant time" by a computer, i.e., and operation where the number of steps needed is not related to the problem size. Some examples of constant time operations are integer arithmetic and comparison. Once we have an expression for the work, we extract the dominant term (i.e., the term that dominates as n tends to infinity), removing any constants, and call that the asymptotic running time.

Example 1.6. Let `matrix_sum(n)` be a function that computes the sum of all elements of an $n \times n$ matrix by iterating over each element and adding it to a running sum.

What is the work involved? We need to initialize a sum variable to 0, which takes one unit of work, and visit each one of th n^2 cells, which takes

n^2 work. As we visit each cell, we must add it to the total sum, for an extra n^2 work. So an expression for the work is $W = 1 + 2n^2$. The dominant term is n^2 (removing constants), so the asymptotic running time of our function is $O(n^2)$.

Here we state the formal definition of big-O for completeness. Note that this can be extended naturally to include any number of size parameters, but we will not need any more than one in this paper.

Definition 1.7. Let $f(n)$ and $g(n)$ be functions whose domain is $\mathbb{N}$. If there exist constants c and N such that $n > N$ implies $f(n) < c(g(n))$, we say that $f \in O(g)$. In other words, f is bounded above by g.

In our example above, $f(n)$ was $W = 1 + 2n^2$ and $g(n)$ was n^2.

Remark. As was hinted in the definition above, big-O is merely an upper bound on a function. So, for example, we could truthfully say that a constant time operation like integer addition was in $O(n^2)$, because the work involved is certainly bounded $above$ by n^2. However, it would be more informative to say that integer addition was in $O(1)$. There is another symbol Θ, which is used to indicate a function that is both an upper and lower bound on another function. However, we still use big-O notation in this paper because it is much more often seen in the literature, and we do not give unneccessarily loose upper bounds.

1.5 Discrete Logarithm Problems

Elliptic curves are useful for public-key cryptography because the group operations give us an (approximately) one-way function, which we discuss here.

1.5.1 The Finite Field Discrete Logarithm Problem

To motivate the name "discrete logarithm", we present here the classic discrete logarithm problem.

Suppose we have a finite field $\mathbb{F}_p$ where p is prime, and some integer a that is in the field. In addition, let $g \in \mathbb{F}_p$ be a primitive root modulo p. (Recall that a primitive root is an element of $\mathbb{F}_p$ which generates $\mathbb{F}_p^*$). Then there exists some field element n such that

$$a \equiv g^n \mod p.$$

The discrete logarithm problem is to find the exponent n. (This turns out to be hard, and is the basis for some cryptosystems). This setup looks fairly close to what we think of as a logarithm, but it is "discrete" because n must be one of the distinct field elements.

1.5.2 The Elliptic Curve Discrete Logarithm Problem

Definition 1.8 (Elliptic Curve Discrete Logarithm Problem)**.** Let E be an elliptic curve over a finite field $\mathbb{F}_p$, and let P be a point on E. Suppose we have a point Q on E that is some scalar multiple of P, i.e.,

$$[n]P = Q, \quad n \in \mathbb{N}.$$

The **elliptic curve discrete logarithm problem** (ECDLP) is to determine the natural number n, given E, P and Q.

Note that this positions multiplication by $[n]$ as analogous to exponeniation, and point addition as analogous to multiplication.

There is no known algorithm to solve this problem in the general case in better than $O(\sqrt{n})$ time. When key sizes are enormous, this operation becomes prohibitively expensive.

On the other hand, the problem of determining $[n]P$ given n and P is not hard. One simple (to describe) way to do this is by using successive squaring.

Definition 1.9 (Successive Squaring for Elliptic Curves)**.** Suppose we have an elliptic curve $E(K)$, a point $P \in E(K)$ and a nonnegative integer n. We can compute $[n]P$ recursively by calling SuccessiveSquare(n, P).

This algorithm leverages the fact that point doubling is just as fast as adding, but can give us large multiples of P quickly by doubling repeatedly.

This is a modification of an algorithm that computes large powers modulo p by repeated squaring.

```
SuccessiveSquare(m, Q)
INPUT:
     m - Non-negative integer
     Q - point on elliptic curve
OUTPUT:
    mQ - point that is Q + Q + ... + Q m times

If m = 0, return the point at infinity.
If m = 1, return Q.
If m is even, return [m/2](2Q) by calling
     SuccessiveSquare(m/2, 2Q).
If m is odd, return [(m-1)/2](2Q) by calling
     SuccessiveSquare((m-1)/2, 2Q) and add Q.
```

Note that "/" denotes integer division. This asymptotic running time of this algorithm is $O(\log n)$ group operations, a considerable improvement on brute force computation of $P + P + P + ...$ which takes $O(n)$ group operations. We are saying "group operations", because computer hardware cannot necessarily do arithmetic on very large numbers in constant time, so we want to leave open the possibility that doing a group operation is not negligible.

Example 1.10. Let us revist the curve we saw in Example 1.4. As we know, the point $(5, 1)$ is in the group $\tilde{E}(\mathbb{F}_p)$ (with $\tilde{E}$ defined as before). Denote this point $\tilde{P}$. Suppose we want to compute $[15]\tilde{P}$ via successive squaring.

Since $m = 15$ is odd, we break the problem into

$$[7]([2]\tilde{P}) + \tilde{P}.$$

Now, because 7 is odd, we break the problem further into

$$[3]([2]([2]\tilde{P})) + \tilde{P} + \tilde{P},$$

which becomes (since 3 is odd) :

$$[2]([2]([2]\tilde{P}) + \tilde{P} + \tilde{P} + \tilde{P}.$$

Notice that we have reduced a problem that would have taken 15 steps to just 6 steps: 3 doubles and 3 additions. The final answer is $(8, 1)$, which is, as we expected, a member of the group.

Those familiar with complexity analysis may be surprised that the running time $O(\sqrt{n})$ to compute the discrete log is considered slow enough to be viable for cryptography. The key idea is not that $\sqrt{n}$ is slow in itself, but that it is asymptotically $much\ slower$ than the $\log n$ time needed to perform a multiplication. Those interested in using elliptic curves in cryptography just need to make n very large.

To get a feel for the magnitude of the difference in speed, suppose n were 2^{128}, and we had a computer that could do 10^9 operations per second. It would take just $\log_2 2^{128} = 128$ operations to calculate $[n]P$, which would take .13 microseconds. On the other hand, our computer would have to perform $\sqrt{2^{128}} = 1.84 \times 10^{19}$ operations to retrieve n from $[n]P$, which would take just under $585\ years$.

Chapter 2

Some Geometry and Algebra

2.1 Projective Coordinates

We have so far discussed elliptic curves as solutions to an equation in two variables x and y. We now introduce an alternate coordinate system, called projective coordinates, that has some nice properties.

Definition 2.1 (Projective Space)**.** Let K be a field, and let V be any subspace of K^n where n is an integer greater than zero. For any elements x, y in V, we define an equivalence relation $\sim$ by

$$x \sim y \text{ if } x = \alpha y \text{ for some nonzero } \alpha \text{ in } K.$$

Projective space over K is $\mathbb{P}$ is:

$$\mathbb{P}(V) = V/\sim .$$

Hence the point $[x]$ in $\mathbb{P}(V)$ is the set of all vectors αx in V where α is a nonzero element of K.

Using the affine coordinate system and the simplified Weierstrass equation we introduced previously, we considered zeros of the function

$$f(x, y) = y^2 - x^3 - ax - b$$

to be points on an elliptic curve. In other words, we wanted pairs $(x, y) \in K^2$ such that $f(x, y) = 0$.

Let us introduce a new variable Z to homogenize the equation as follows:

$$F(X, Y, Z) = Y^2Z - X^3 - aXZ^2 - bZ^3.$$

(A homogeneous equation is one in which all terms have the same degree).

Points $(X : Y : Z)$ with $X, Y, Z \in K$ that satisfy this equation are called projective points, and the set of all such points form an alternate way of describing the K-rational points on E. We may easily convert an affine point (x, y) to a projective one by letting $Z = 1$ and keeping x and y, i.e., $(x : y : 1)$. In addition, by our eqivalence relation, if we divide each coordinate of a projective point by the same non-zero constant in K, we end up with the same point we started with.

One interesting thing is that we now have an algebraic representation of the point at infinity $\mathcal{O}$. It is $(0 : 1 : 0)$, the only projective point with Z-coordinate equal to 0.

2.2 Introduction to the p-adics

We now introduce the field $\mathbb{Q}_p$, whose elements are called p-adics. For a prime number p, $\mathbb{Q}_p$ is a completion of the rationals with respect to the p-adic absolute value, which we define shortly.

Before giving any formal definitions, we note that a p-adic number looks something like this:

$$a_k p^k + a_{k+1} p^{k+1} + ... + a_0 + a_1 p + a_2 p^2 + ...,$$

where p is a fixed prime, and $a_k, ..., a_0, ...$ are integers between 0 and $p - 1$. Note that there are always a finite number of negative powers of p, and a possibly infinite number of positive powers. We will see that the large powers of p are actually "small" with respect to the p-adic absolute value.

Definition 2.2. For a rational number a and a prime number p, separate out all factors of p from a and write:

$$a = p^r \frac{m}{n}$$

where r, m and n are integers, and p does not divide m or n. The exponent r is called the **p-adic ordinal** of a, denoted $\text{ord}_p(a)$.

In other words, the ordinal is the highest power of p dividing a. This may be negative, positive, or 0.

Note that for any rational number $a = \frac{x}{y}$, we have $\text{ord}_p(a) = \text{ord}_p(y) - \text{ord}_p(x)$.

Definition 2.3. For a prime p, we define a function $|.|_p : \mathbb{Q} \to \mathbb{Q}_{\geq 0}$ where for $a \in \mathbb{Q}$:

$$|a|_p = \begin{cases} p^{-\text{ord}_p(a)} & a \neq 0 \\ 0 & a = 0. \end{cases}$$

The function $|.|_p$ is called the **p-adic absolute value**.

Proposition 2.4. *The p-adic absolute value is a norm on $\mathbb{Q}$, and induces a metric*

$$d_p(a,\ b) = |a - b|_p$$

for $a, b \in \mathbb{Q}$.

Proof. Let a, b be elements of $\mathbb{Q}$. We want to show that $|a-b|_p$ is a metric on $\mathbb{Q}$. Three properties must hold:

1. (Non-negativity.) $d_p(a, b) \geq 0$, with equality only when $a = b$.
2. (Commutativity.) $d_p(a, b) = d_p(b, a)$.
3. (Triangle Inequality.) $d_p(a, b) \leq d_p(a, c) + d_p(c, b)$ for any c in $\mathbb{Q}$.

1. When $a = b$, we know $|a - b|_p = |0|_p$, which is defined to be 0. When $a \neq b$, then $|a - b|_p$ is $p^{-\text{ord}_p(a-b)}$, where $\text{ord}_p(a - b)$ is an integer. Raising a positive number to an integer power always results in a positive and non-zero value.

2. We must justify that $|a-b|_p = |b-a|_p$. This is clearly true in the case where $a = b$. When $a \neq b$, this is equivalent to the claim that $\text{ord}_p(a-b) = \text{ord}_p(b-a)$. Since $a - b = -(b-a)$, the two numbers differ only in sign. The ordinal of a number and its negative are equal, as they are each divisible by the same highest power of p.

3. We here show a stronger property, that $|a-b|_p \leq \max |a|_p, |b|_p$. Thus we will show that the p-adic norm induces an $ultrametric$ on $\mathbb{Q}$. First observe that the property clearly holds when a, b, or $(a-b)$ is equal to zero. So, assume that these three values are non-zero. Then we can write $a = \frac{m}{n}$ and $b = \frac{r}{s}$ for integers m, n, r, s in lowest terms. Thus $a - b = \frac{ms-rn}{ns}$. Therefore $\text{ord}_p(a-b) = \text{ord}_p(ms-rn) - \text{ord}_p(ns)$. Now each of ms and rn have some ordinal, which is the highest power of p that divides them. We can take the smallest of these out of $ms - rn$, meaning that $\text{ord}_p(ms - rn) \geq \min(\text{ord}_p(ms), \text{ord}_p(rn))$. Thus

$$\text{ord}_p(a-b) \geq \min(\text{ord}_p(ms), \text{ord}_p(rn)) - \text{ord}_p(ns),$$

and recombining these terms (which still preserves the minimum, as we are subtracting a constant from both terms) we have that:

$$\text{ord}_p(a-b) \geq \min(\text{ord}_p(ms/ns), \text{ord}_p(rn/ns))$$

$$= \min(\text{ord}_p(m/n), \text{ord}_p(r/s)) = \min(\text{ord}_p(a), \text{ord}_p(b))$$

From this fact it is easy to deduce that $|a-b|_p \leq \max |a|_p, |b|_p$.

The takeaway is that $\mathbb{Q}_p$ is a field, and is the completion of the rationals with respect to the p-adic absolute value. The proof of this fact is quite involved, and can be found in [Gou93].

2.3 Working with the p-adics

Definition 2.5. A p-adic number a is called a **p-adic integer** if $ord_p(a) \geq 0$. The set of all p-adic integers is a group, and denoted $\mathbb{Z}_p$ (not to be confused

with the cyclic group $\mathbb{Z}/p\mathbb{Z}$).

Remark. A p-adic integer is always of the form

$$a_0 + a_1 p + a_2 p^2 + ...,$$

i.e., all powers of p are non-negative.

Notation. The set $p^n\mathbb{Z}_p$ where $n \geq 0$ is a subgroup of $\mathbb{Z}_p$ and has elements of the form $a_0 p^n + a_1 p^{n+1} + ...$ and $\text{ord}_p(x) \geq n$ for all $x \in p^n\mathbb{Z}_p$.

Properties of ordinals and absolute values

These facts have been shown in the proof of Proposition 2.4, but we state them here for easy reference. Let $a, b \in \mathbb{Q}_p$. Then:

$$\text{ord}_p(a+b) \leq \min\{\text{ord}_p(a), \text{ord}_p(b)\} \quad \textbf{and} \quad \text{ord}_p(ab) = \text{ord}_p(a) + \text{ord}_p(b).$$

$$|a+b|_p \geq \min\{|a|_p, |b|_p\} \quad \textbf{and} \quad |ab|_p = |a|_p + |b|_p.$$

Convergence

Convergence is easier to show in $\mathbb{Q}_p$ than in $\mathbb{R}$ because there is no conditional convergence. Thus it is sufficient to show that the terms of a sum tend to zero with respect to the p-adic absolute value in order to show that the sum converges in $\mathbb{Q}_p$.

Characteristic of the p-adics

The field of p-adics $\mathbb{Q}_p$ has characteristic 0, not p. This is due to the fact that $\mathbb{Q}$, which has characteristic 0, is a subfield of $\mathbb{Q}_p$.

Arithmetic

Since the p-adics are a field, we hope to be able to do addition and multiplication, as well as take additive and multiplicative inverses. All of these are possible, but they are not easy for a human to do by hand.

To give a sense of how arithmetic works, and motivate the next chapter, we give an example of point doubling over the p-adics.

Example 2.6. Recall $\tilde{E}$ from Example 1.4. Let

$$E : y^2 = x^3 + x + 4,$$

just like $\tilde{E}$, but view the paramters $a = 1$ and $b = 4$ as elements of $\mathbb{Q}_{19}$ rather than $\mathbb{F}_{19}$.

It turns out that $P = (5 + O(19^3), 1 + 13 \cdot 19 + 10 \cdot 19^2 + O(19^3))$ is a point satisfying E. How would we compute $[2]P$ using the point doubling formula? We know from Section 1.3 that if $p = (x, y)$, then $[2]P = (x_2, y_2)$ where

$$x_2 = N^3 - 2x,$$

$$y_2 = N(x - x_2) - y,$$

$$\text{where } N = \frac{3x^2 + a}{2y}.$$

Let us compute N first. The numerator is

$$N_{num} = 3(5 + O(19^3)) + 1 = 16 + O(19^3),$$

and the denominator is

$$N_{denom} = 2(1 + 13 \cdot 19 + 10 \cdot 19^2 + O(19^3)) = 2 + 7 \cdot 19 + 2 \cdot 19^2 + O(19^3).$$

Then

$$N = \frac{N_{num}}{N_{denom}} = 8 + 10 \cdot 19 + 12 \cdot 19^2 + O(19^3).$$

Now we may compute the coordinates:

$$x_2 = N^3 - 2x = 8 + 10 \cdot 19 + 12 \cdot 19^2 + O(19^3) = 8 + 8 \cdot 19 + 18 \cdot 19^2 + O(19^3),$$

$$y_2 = N(x - x_2) = 13 + 5 \cdot 9 + 9 \cdot 19^2 + O(19^3).$$

Note that we are using big-O not to say anything about running time but

just using the definition to indicate that that the rest of the terms are bounded above by a certain function (with respect to the p-adic absolute value).

For more information on the p-adics at an introductory level, see [Gou93].

Chapter 3

Elliptic Curves over the p-adics

Since $\mathbb{Q}_p$ is a field with characteristic 0, we can talk about elliptic curves over the p-adics, and all of the theory we built up in the previous sections applies. In particular, we may use the simplified Weierstrass equation that applies to fields that do not have characteristic 2 or 3.

From here on we use $\tilde{E}(\mathbb{F}_p)$ to mean the group of points satisfying a Weierstrass equation $\tilde{E}$ defined over a prime field $\mathbb{F}_p$, and $E(\mathbb{Q}_p)$ to denote the group of points satisfying the same equation over the p-adics (for the same value of p).

Recall that the goal of this paper is to show how the ECDLP (see Section 1.5) may be computed for elliptic curves of trace one. In this chapter, we build up most of the theory needed to understand this computation. This understanding of the theory owes much to [Mon02] and [Sil86].

We wish to establish a map from $\tilde{E}(\mathbb{F}_p)$ to $\mathbb{Z}/p\mathbb{Z}$ that transforms some $Q \in \tilde{E}(\mathbb{F}_p)$ to $n \in \mathbb{Z}/p\mathbb{Z}$ such that $[n]P = Q$ for a specified $P \in \tilde{E}(\mathbb{F}_p)$. It turns out this is feasible when $\#\tilde{E}(\mathbb{F}_p) = p$. We do this by transforming the points, via a series of maps, into $\mathbb{Z}/p\mathbb{Z}$.

3.1 Computing lifts and reducing modulo p

The first map is from $\tilde{E}(\mathbb{F}_p)$, our original curve, to a related curve over the p-adics, $E(\mathbb{Q}_p)$. To move from $\tilde{E}(\mathbb{F}_p)$ to $E(\mathbb{Q}_p)$, we compute what is called

a lift. A point $P \in E(\mathbb{Q}_p)$ is called a lift of a point $\tilde{P} \in \tilde{E}(\mathbb{F}_p)$ if it reduces mod p to $\tilde{P}$.

Reducing modulo p from $E(\mathbb{Q}_p)$ to $\tilde{E}(\mathbb{F}_p)$

We first must describe formally what it means to reduce a point modulo p. This will be a map from $E(\mathbb{Q}_p)$ to $\tilde{E}(\mathbb{F}_p)$. Because we are allowed to take out non-zero multiples in projective coordinates, any point P in $E(\mathbb{Q}_p)$ can be written in as $(X : Y : Z)$, where X, Y and Z are p-adic integers and at least one is not divisble by p. We define a map $r_p : \mathbb{Z}_p \to \mathbb{Z}/p\mathbb{Z}$ that reduces a p-adic integer modulo p in the expected way, by extracting the coefficient on the p^0 term. Then, to reduce a point mod p, we simply reduce each of its coordinates mod p: thus $\tilde{P} = (r_p(X) : r_p(Y) : r_p(Z))$.

Lifting from $\tilde{E}(\mathbb{F}_p)$ to $E(\mathbb{Q}_p)$

Computing a lift is more involved. This operation is not unique, as there may be multiple lifts of a given point in $\tilde{E}(\mathbb{F}_p)$, but to compute the discrete log, we only need one. As it happens, any choice of lift will give the same (correct) value of the discrete logarithm, provided that it does not cause an error such as divison by 0. We describe one way to compute lifts here, which fails with probability $1/p$, which is tiny in any real application. If it does fail, we may take an alternate lift.

Let $\tilde{P}$ be a point satisfying:

$$\tilde{E} : y^2 = x^3 + ax + b$$

where $a, b \in \mathbb{F}_p$ and $p \neq 2, 3$. In affine coordinates, we may write $\tilde{P} = (\tilde{x}, \tilde{y})$ for $\tilde{x}, \tilde{y} \in \mathbb{F}_p$. We wish to find p-adic integers x and y such that $P = (x, y) \in E(\mathbb{Q}_p)$, where E is $\tilde{E}$ viewed over the p-adics.

Let us rewrite the Weierstrass equation as a function f in two variables x and y:

$$f(x, y) = y^2 - x^3 - ax - b.$$

Then x and y satisfy E if and only if $f(x, y) = 0$.

Now, the easiest way to choose x and y would be to simply let $x = \tilde{x}$ and $y = \tilde{y}$. Unfortunately, this choice would cause our algorithm to break down in later stages. However, we may choose $x = \tilde{x}$. The harder one is y.

Since y is a p-adic integer, it has an expansion of the form

$$y = h_0 + h_1 p^1 + h_2 p^2 + ...,$$

where the h_i are between 0 and $p - 1$.

Because we require $f(x, y) = 0$, it must also be that $f(x, y) \equiv 0 \bmod p^i$ for any $i > 0$ (and indeed any modulus). Therefore $f(x, y) \equiv 0 \bmod p$. Writing this explicitly, we have that

$$(h_0 + h_1 p + h_2 p^2 + O(p^3))^2 - \tilde{x}^3 - a\tilde{x} - b \equiv 0 \mod p.$$

Reducing gives:

$$(h_0)^2 - \tilde{x}^3 - a\tilde{x} - b \equiv 0 \mod p,$$

which implies that we may set $h_0 = \tilde{y}$.

Now by the same token (and replacing h_0 with $\tilde{y}$):

$$(\tilde{y} + h_1 p + h_2 p^2 + O(p^3))^2 - \tilde{x}^3 - a\tilde{x} - b \equiv 0 \mod p^2,$$

which reduces to

$$(\tilde{y} + h_1 p)^2 - \tilde{x}^3 - a\tilde{x} - b \equiv 0 \mod p^2,$$

and with some algebraic manipulation gives us that

$$(\tilde{y} - \tilde{x}^3 - a\tilde{x} - b) + 2\tilde{y}h_1 p \equiv 0 \mod p^2.$$

We would like to solve for $h_1 \bmod p$. Notice that the first part of the left hand side of the above congruence is precisely $f(\tilde{x}, \tilde{y})$, which we know to be congruent to 0 mod p, and thus divisible by p. Therefore we can rewrite the expression as

$$\frac{f(\tilde{x}, \tilde{y})}{p} + 2\tilde{y}h_1 \equiv 0 \mod p,$$

which can be rearranged to obtain a value for h_1:

$$h_1 \equiv -\frac{f(\tilde{x}, \tilde{y})}{2p\tilde{y}} \mod p.$$

We may proceed in a similar way to compute any other value of h_i. We here present without detailed explanation the formula for h_2, which we will need in our computation of the discrete logarithm:

$$h_2 \equiv -\frac{f(\tilde{x}, \tilde{y} + h_1 p)}{2p^2(\tilde{y} + h_1 p)} \mod p.$$

3.2 The formal group $\hat{E}(p\mathbb{Z}_p)$

The next step is to map from $E(\mathbb{Q}_p)$ to $\hat{E}(p\mathbb{Z}_p)$, a formal group with elements in $p\mathbb{Z}_p$.

Before we describe the formal group law on $\hat{E}(p\mathbb{Z}_p)$, we need to build up some theory. Again, consider a curve E defined over $E(\mathbb{Q}_p)$ with equation:

$$E : y^2 = x^3 + ax + b.$$

Notice that x and y are algebraically dependent, since they are related by E. Thus we we may parameterize the equation.

To do this, we introduce a new parameter z and use a change of variables from the xy plane to the zw plane, where:

$$z = -\frac{x}{y}, \quad w = -\frac{1}{y}$$

(Note that the new parameter z is distinct from the parameter Z we used in projective coordinates.) Then, noting that $y = -\frac{1}{w}$ and $x = \frac{z}{w}$, we can rewrite E as:

$$E : \frac{1}{w^2} = \frac{z^3}{w^3} + a\frac{z}{w} + b,$$

and multiplying by w^3 gives us:

$$w = z^3 + azw^2 + bw^3.$$

This equation is still in terms of w, however, and we would like it to rely only on z. Recursively substituting the right-hand side into any occurences of w on the right gives us a power series in z.

To see how this works, here is the first substitution:

$$w = z^3 + az(z^3 + azw^2 + bw^3)^2 + b(z^3 + azw^2 + bw^3) = z^3 + az^7 + bz^3 + ...$$

From this power series $w(z)$ we arrive at expressions (called Laurent series because they have negative powers) for x and y in terms of z.

$$x(z) = \frac{z}{w} = \frac{1}{z^2} - az^2 + ...,$$

$$y(z) = -\frac{1}{z^3} + az + ...$$

We want $x(z)$ and $y(z)$ to converge in $\mathbb{Q}_p$. We claim that this is true when z is an element of $p\mathbb{Z}_p$ and the coefficients a and b of E are p-adic integers. If our claim is correct, we will have established an injective map from $p\mathbb{Z}_p$ to $E(\mathbb{Q}_p)$, which takes a parameter z in $\mathbb{Z}_p$, and finds a point $P = (x(z), y(z))$ that is on $E(\mathbb{Q}_p)$. Note that we may also map a point in $E(\mathbb{Q}_p)$ back to $p\mathbb{Z}_p$ by our definition of z, namely that $z = -x(z)/y(z)$.

Proof. We now show that $x(z)$ and $y(z)$ converge in $\mathbb{Q}_p$ if $z \in p\mathbb{Z}_p$ and $a, b \in \mathbb{Z}_p$. Notice that

$$|x(z)|_p \leq \max\{|z^{-2}|_p, |-az^2|_p, ...\} = \max\{|z|_p^{-2}, |a|_p|z|_p^2, ...\}$$

We know $|z|_p \leq p^{-1}$ and $|a|_p, |b|_p \leq 1$ by the conditions we put on them. Thus $|z|_p^{-2} \geq p^2$ while $|a|_p|z|_p^2 \leq p^{-2}$.

Thus there is a finite upper bound on the absolute value of $x(z)$, so it converges. A similar argument tells us the same for $y(z)$.

Definition 3.1. For E an elliptic curve defined over $\mathbb{Q}_p$, we define the group $\hat{E}(p\mathbb{Z}_p)$ as the set $p\mathbb{Z}_p$ with addition law

$$z_1 \oplus z_2 = F(z_1, z_2)$$

where $F(z_1, z_2)$ is a formal power series that depends on the parameters a, b from the Weierstrass equation, which is why the group is called $\hat{E}$. We do not actually need an explicit expression for F to do our computation. To see how it would be derived, see [Sil86].

3.3 The groups $E_n(\mathbb{Q}_p)$

We now take a brief interlude from our series of maps to introduce an important class of subgroups of $E(\mathbb{Q}_p)$, and some of their properties.

Definition 3.2. Let $E(\mathbb{Q}_p)$ be an elliptic curve. The group $E_1(\mathbb{Q}_p)$ is defined to be:

$$E_1(\mathbb{Q}_p) = \{P \in E(\mathbb{Q}_p) \mid \tilde{P} = \mathcal{O}\}.$$

In words, E_1 is the set of points on E that reduce modulo p to $\mathcal{O}$. This leads naturally to the following proposition:

Proposition 3.3.

$$E(\mathbb{Q}_p)/E_1(\mathbb{Q}_p) \simeq E(\mathbb{F}_p).$$

Proof. Define a map $r : E(\mathbb{Q}_p) \to E(F_p)$ where $r(P) = \tilde{P}$. By definition, the kernel of this map is $E_1(\mathbb{Q}_p)$. The result follows from the First Isomorphism Theorem.

Definition 3.4. The subgroup $E_n(\mathbb{Q}_p)$ (for $n \in N$) of $E(\mathbb{Q}_p)$ is defined:

$$E_n(\mathbb{Q}_p) = \{P \in E(\mathbb{Q}_p) \mid \mathrm{ord}_p(x_P) \leq -2n\} \cup \{\mathcal{O}\},$$

where x_P is the x-coordinate of P.

Remark. The group $E_n(\mathbb{Q}_p)$ is a subgroup of $E_k(\mathbb{Q}_p)$ for all $k \leq n$.

For the case that $n = 1$,

$$E_1(\mathbb{Q}_p) = \{P \in E(\mathbb{Q}_p) \mid \text{ord}_p(x_P) \leq -2\} \cup \{\mathcal{O}\},$$

which turns out (as we would hope) to be equivalent to Definition 3.2. This may seem mysterious, but before we justify it let us determine why it is that if $\text{ord}_p(x)$ is negative, it must be even. Assume that a and b are p-adic integers, i.e., that $\text{ord}_p(a) \geq 0$ and $\text{ord}_p(a) \geq 0$. We know x and y must satisfy

$$y^2 = x^3 + ax + b,$$

so by properties of p-adic ordinals must also satisfy

$$2\text{ord}_p(y) \geq \min\{3\text{ord}_p(x), \text{ord}_p(a) + \text{ord}_p(x), \text{ord}_p(b)\}.$$

with equality if there are no ties for the minimum. Because $\text{ord}_p(x) < 0$ and $\text{ord}_p(a), \text{ord}_p(b) \geq 0$, the minimum must be $3\text{ord}_p(x)$, with no ties. Thus

$$2\text{ord}_p(y) = 3\text{ord}_p(x),$$

which with some manipulation implies that

$$\text{ord}_p(y) = -3n,\ \text{ord}_p(x) = -2n$$

for some natural number n.

So why is it that points of the form $P = (p^{-2n} + ..., p^{-3n})$ reduce modulo p to $\mathcal{O}$? Write

$$P = (c_1p^{-2n} + ... : d_1p^{-3n} + .. : 1)$$

in projective coordinates, and then take out a factor of p^{-3n}. Then

$$P = (c_1p^n + ... : d_1 + ... : p^{3n}).$$

Reducing mod p gives $\tilde{P} = (0 : d_1 : 0)$, which is equivalent to $\mathcal{O}$, as we may take out the non-zero scalar d_1.

3.4 Mapping between $\hat{E}(p\mathbb{Z}_p)$ and $E_1(\mathbb{Q}_p)$

We have already established that we may map between $\hat{E}(p\mathbb{Z}_p)$ and $E(\mathbb{Q}_p)$ using the Laurent series for z, $x(z)$ and $y(z)$ we presented in the previous section. We state this formally now with a slight modification, and give the stronger claim that it is a group isomorphism.

Proposition 3.5. *The map*

$$\nu_p : \hat{E}(p\mathbb{Z}_p) \to E_1(\mathbb{Q}_p)$$

$$\nu_p(z) = (x(z), y(z))$$

is a group isomorphism.

Remark. We are actually more interested in the inverse map:

$$\nu_p^{-1} : E_1(\mathbb{Q}_p) \to \hat{E}(p\mathbb{Z}_p)$$

$$\nu_p^{-1}(x, y) = -\frac{x}{y},$$

as it will allow us to continue solving the discrete logarithm.

We will not prove this proposition, but point out some interesting things about it. Firstly, we may apply the map ν_p^{-1} to any group $E_n(\mathbb{Q}_p)$, and the co-domain will be $\hat{E}(p^n\mathbb{Z}_p)$. But, if we wish to end up in $\hat{E}(p\mathbb{Z}_p)$, which we do, we must begin in $E_1(\mathbb{Q}_p)$. We can see why this is by looking at the smallest terms in the Laurent series $x(z)$ and $y(z)$, which are $\frac{1}{z^2}$ and $-\frac{1}{z^3}$. We know that $z \in p\mathbb{Z}_p$, meaning it is divisible by p. Thus the $x(z)$ term must be divisible by p^{-2}, and thus $\text{ord}_p(x) \leq -2$, which means that the point $(x(z), y(z))$ is, by our second definition of $E_1(\mathbb{Q}_p)$, a member of that group.

Thus we cannot use ν_p^{-1} directly to map our points in $E(\mathbb{Q}_p)$ into $\hat{E}(p\mathbb{Z}_p)$, so we need a way to map $E(\mathbb{Q}_p) \to E_1(\mathbb{Q}_p)$. This is possible when our initial curve $\tilde{E}(\mathbb{F}_p)$ has trace one, and we will describe how to do it in Chapter 4.

Of course $\hat{E}(p\mathbb{Z}_p)$ is hardly the group we want to end up in, so we must find more maps to get us to $\mathbb{Z}/p\mathbb{Z}$.

3.5 The formal logarithm

In this section we will introduce a map called the formal logarithm, denoted $\log_F$ that takes us from the formal group $\hat{E}(p\mathbb{Z}_p)$ to plain $p\mathbb{Z}_p$ with the usual addition law. This is the second-to-last step in our compuation of the discrete logarithm.

We first introduce a certain differential form called an invariant differential.

Definition 3.6. An invariant differential on a formal group defined over a ring is

$$\omega(T) = P(T)dT,$$

where $P(T)$ is a power series in T that satisfies

$$w \circ F(T, S) = w(T),$$

where $F(T, S)$ is the formal group law.

Note that such an invariant differential will always exist.

Of course, we are dealing with the formal group $\hat{E}(p\mathbb{Z}_p)$ under the group law F that we introduced in Definition 3.1. Let $\omega(T)$ be a power series $c_0 + c_1T + c_2T^2$ with the c_i chosen such that $\omega(T)$ is an invariant differential on $\hat{E}(p\mathbb{Z}_p)$. Define

$$\log_F : \hat{E}(p\mathbb{Z}_p) \to p\mathbb{Z}_p$$

$$\log_F(T) = \int \omega(T).$$

Proposition 3.7.

$$\log_F(F(T, S)) = \log_F(T) + \log_F(F(T, S)).$$

In other words, $\log_F$ *is a group homomorphism.*

Proof. By our definition of $\log_F$, we know $\log_F(F(T,S)) = \int \omega(F(T,S))$, which by the fact that ω is an invariant differential, is equal to $\int \omega(T)$. Taking the integral of $\omega(T)$, we get $\log_F(T) + g(S)$, where g is some function that does not depend on T but may depend on S.

To determine the value of $g(S)$, let $T = 0$. Then $F(T,S) = S$, because F is a group law and T is set to be the identity. Thus $\log_F(F(T,S)) = \log_F(S) = \log_F(0) + g(S) = 0 + g(S)$. Therefore $g(S) = log(S)$, and combining our results we arrive at:

$$\log_F(F(T,S)) = \log_F(T) + \log_F(S).$$

What this really ends up looking like is:

$$\log_F(T) = T + \frac{c_1}{2}T^2 + \frac{c_2}{3}T^3 + ...,$$

where $c_1, c_2... \in p\mathbb{Z}_p$.

Remark. In fact, $\log_F$ is a group isomorphism from $\hat{E}(p\mathbb{Z}_p)$ and $p\mathbb{Z}p$. Furthermore, for any n, $\log_F$ is restricted to be an isomorphism between $\hat{E}(p^n\mathbb{Z}p)$ and $p^n\mathbb{Z}p$. Since $E_n(\mathbb{Q}_p) \simeq \hat{E}(p^n\mathbb{Z}_p)$ (see discussion after Proposition 3.5), we now know that $E_n(\mathbb{Q}_p) \simeq p^n\mathbb{Z}_p$. Thus

$$E_n(\mathbb{Q}_p)/E_{n+1}(\mathbb{Q}_p) \simeq p^n\mathbb{Z}_p/p^{n+1}\mathbb{Z}_p \simeq \mathbb{F}_p^+, \tag{3.1}$$

where $\mathbb{F}_p^+$ is the additive group of $\mathbb{F}_p$.

We are now ready to do some synthesis and fill in the missing pieces. We do this in the next chapter, and show a small example of the full algorithm.

Chapter 4

Solving the Discrete Logarithm Problem over Curves of Trace One

In this section we wil summarize and demonstrate the fast algorithm initially proposed by Nigel Smart in [Sma99] for computing the discrete logarithm on curves of trace one, and discuss its running time and implications.

4.1 Summary of algorithm

We now summarize the method for finding the discrete log on a curve of trace one, and fill in any results not shown in the previous chapters. Recall our setup for the discrete logarithm problem. We have an elliptic curve $\tilde{E}/\mathbb{F}_p$ with $p \neq 2$ or 3 and

$$\tilde{E} : y^2 = x^3 + ax + b$$

where $a, b \in \mathbb{F}_p$. For two points $\tilde{P}, \tilde{Q} \in \tilde{E}(\mathbb{F}_p)$ that satisfy

$$[n]\tilde{P} = \tilde{Q}, \tag{4.1}$$

we wish to determine the natural number n. Notice that, because $\mathcal{O}$ is the identity,

$$\tilde{Q} - [n]\tilde{P} = \mathcal{O}.$$

The first step is to obtain lifts of $\tilde{P}$ and $\tilde{Q}$ by the method described in Section 3.1. We denote these lifts P and Q respectively, and they lie in $E(\mathbb{Q}_p)$, where E is the same as $\tilde{E}$, but the parameters a, b are interpreted as elements of $\mathbb{Q}_p$.

Now recall that the reduction map $\tilde{E}(\mathbb{F}_p) \to E(\mathbb{Q}_p)$ is a homomorphism. Thus the reduction mod p of $Q-[n]P$ is the same as the sum of the reduction of P (times $[n]$) and the reduction of Q. By equation (4.1), we deduce that

$$Q - [n]P = R,$$

where R is an element of $E_1(\mathbb{Q}_p)$ since it reduces mod p to $\mathcal{O}$.

Now we claim that multiplying a point in $E(\mathbb{Q}_p)$ by $[p]$ maps it into $E_1(\mathbb{Q}_p)$ and similarly, multiplying a point in $E_1(\mathbb{Q}_p)$ by $[p]$ maps it into $E_2(\mathbb{Q}_p)$. This is the step that requires our curve to be of trace one. Recall from Proposition 3.3 and (3.1) that:

$$E(\mathbb{Q}_p)/E_1(\mathbb{Q}_p) \simeq \tilde{E}(\mathbb{F}_p),$$

$$E_1(\mathbb{Q}_p)/E_2(\mathbb{Q}_p) \simeq \mathbb{F}_p^+.$$

Thus each of the quotient groups $E(\mathbb{Q}_p)/E_1(\mathbb{Q}_p)$ and $E_1(\mathbb{Q}_p)/E_2(\mathbb{Q}_p)$ have order p, since by assumption $\#\tilde{E}(\mathbb{F}_p) = p$.

The identity of $E(\mathbb{Q}_p)/E_1(\mathbb{Q}_p)$ is the coset $E_1(\mathbb{Q}_p)$. Consider the coset $S+E_1(\mathbb{Q}_p)$ where S is an element of $E(\mathbb{Q}_p)$. Since $S+E_1(\mathbb{Q}_p) \in E(\mathbb{Q}_p)/E_1(\mathbb{Q}_p)$, it must have order dividing p, which means that $[p](S + E_1(\mathbb{Q}_p)) = E_1(\mathbb{Q}_p)$. But it is also true that $[p](S + E_1(\mathbb{Q}_p)) = [p]S + E_1(\mathbb{Q}_p)$. Thus $[p]S \in E_1(\mathbb{Q}_p)$, which is what we wanted to show. The proof is similar to show that multiplication by $[p]$ maps $E_1(\mathbb{Q}_p)$ into $E_2(\mathbb{Q}_p)$.

We conclude that

$$[p]Q - [n]([p]P) = [p]R,$$

where $[p]R \in E_2(\mathbb{Q}_p)$ and $[p]Q, [p]P \in E_1(\mathbb{Q}_p)$.

Now we are in good shape, because we know how to work with $E_1(\mathbb{Q}_p)$. Recall that the map ν_p^{-1} takes $E_1(\mathbb{Q}_p)$ to $\hat{E}(p\mathbb{Z}_p)$, and the map $\log_F$ takes $\hat{E}(p\mathbb{Z}_p)$ to $p\mathbb{Z}_p$. We define the composition of these maps as follows:

$$\psi_p : E_1(\mathbb{Q}_p) \to p\mathbb{Z}_p$$

where $\psi_p(P) = \log_F \circ \nu_p^{-1}(P)$ for all $P \in E_1(\mathbb{Q}_p)$. It turns out we only need to compute $\psi_p \bmod p^2$, because we do not need any more terms to find n. Let $S = (x, y)$ be a point in $E_1(\mathbb{Q}_p)$. First observe that $\nu_p^{-1}(S) = -\frac{x}{y}$ is an element of $\hat{E}(p\mathbb{Z}_p)$, so it is divisible by p. We can now write $\psi_p(S) = \log_F(-\frac{x}{y})$. Now recall that

$$\log_F(T) = T + \frac{d_1}{2}T^2 + \frac{d_2}{3}T^3 + ...$$

But since our input to $\log_F$ is $-\frac{x}{y}$, which is divisble by p, all of the squared and higher terms are divisible by p^2. We are working modulo p^2 so we may ignore them. Thus

$$\psi_p(S) = \log_F(-\frac{x}{y}) \equiv -\frac{x}{y} \mod p^2. \tag{4.2}$$

Continuing to solve the discrete log, we take the $\psi_p([p]P)$ and $\psi_p([p]Q)$, to get values in $p\mathbb{Z}_p$. Notice that $\psi_p([p]R) = \psi_p([p]Q) - n(\psi_p([p]P))$ is an element of $p^2\mathbb{Z}_p$ because ψ_p maps elements of $E_2(\mathbb{Q}_p)$ to $p^2\mathbb{Z}_p$. So we have that

$$\psi_p([p]Q) - n(\psi_p([p]P)) \equiv 0 \mod p^2.$$

Now, because $\psi_p([p]P)$ and $\psi_p([p]Q)$ are elements of $p\mathbb{Z}_p$, their p-adic expansions are of the form:

$$\psi_p([p]P) = c_1 p + c_2 p^2 + O(p^3)$$

$$\psi_p([p]Q) = d_1 p + d_2 p^2 + O(p^3)$$

where the coefficients c_i, d_i are in $\mathbb{Z}/p\mathbb{Z}$.

Thus

$$n \equiv \frac{\psi_p([p]Q)}{\psi_p([p]P)} \mod p.$$

To solve the p-adic division problem, we note that

$$n(c_1 p + c_2 p^2 + O(p^3)) = d_1 p + d_2 p^2 + O(p^3),$$

and thus $nc_1 p = d_1 p \mod p^2$. Therefore $n = \frac{c_1}{d_1} + O(p^2)$. So the final answer is:

$$n \equiv \frac{c_1}{d_1} \mod p,$$

and we have our discrete logarithm.

4.2 Example Computation

Here we show an example of computing the discrete logarithm of a curve of trace one over a small field. These computations were performed using a Python script, and made use of the Sage Math python modules, which have built-in support for p-adic and finite field arithmetic.

Recall our curve $\tilde{E}$ from Example 1.4 defined over $\mathbb{F}_{19}$ by:

$$\tilde{E} : y^2 = x^3 + x + 4,$$

which has trace one, as it has 19 elements.

Now let $\tilde{P} = (5, 1)$ and $\tilde{Q} = (8, 1)$. We showed in Example 1.10 that $[15]\tilde{P} = \tilde{Q}$, but suppose all we knew was that

$$[n]\tilde{P} = \tilde{Q}$$

for some natural number n. How would we solve the discrete log problem and determine n without using brute force?

Since $\tilde{E}(\mathbb{F}_{19})$ has trace one, we may apply the method we have just developed.

1. Computation of lifts. First, we take lifts of $\tilde{P}$ and $\tilde{Q}$ by the process described in Section 3.1.

$$P = (5 + O(19^3), 1 + 13 \cdot 19 + 10 \cdot 19^2 + O(19^3))$$

$$Q = (8 + O(19^3), 7 + 14 \cdot 19 + 3 \cdot 19^2 + O(19^3))$$

2. Multiplication by p. We now multiply by $[p]$, which we may do quickly using the successive squaring algorithm from Section 1.9, which makes use of the group law from Section 1.3, applied using p-adic arithmetic as described in Section 2.3.

$$[p]P = (9 \cdot 19^{-2} + 12 \cdot 19^{-1} + O(19^0),\ 8 \cdot 19^{-3} + 17 \cdot 19^{-}2 + O(19^{-1}))$$

$$[p]Q = (16 \cdot 19^{-2} + 12 \cdot 19^{-1} + O(19^0),\ 7 \cdot 19^{-3} + 18 \cdot 19^{-}2 + O(19^{-1}))$$

3. Computing ψ_p. We then compute ψ_p using Equation 4.2.

$$\psi_p(P) = 6 \cdot 19 + 2 \cdot 19^2 + O(19^3)$$

$$\psi_p(Q) = 14 \cdot 19 + 13 \cdot 19^2 + O(19^3)$$

4. Wrapping up. Finally, we extract the first coefficient from $\psi_p(P)$ and $\psi_p(Q)$ to obtain:

$$n = \frac{6}{14} = 15 \mod 19.$$

Thus $[15]P = Q$, as we expected.

4.3 Final notes

Note that in some cases, the method described above does not quite work. In the final step, we divide by d_1, the p coefficient of $\psi_p(Q)$, which must be non-zero. If it is zero, we must start over and compute a different lift of P and Q.

4.3.1 Running time

If we use the successive squaring technique to multiply by $[p]$, the running time of Smart's algorithm is $O(\log p)$ group operations where p is the prime modulus of the finite field $\mathbb{F}_p$. Note that most of the operations, such as computing lifts and taking Ψ involved take constant time (i.e., their running time does not depend on p). The most expensive operation is the multiplication by $[p]$, which as we have discussed, takes on the order of $\log p$ operations.

4.3.2 Implications

The take-away from this discussion is that the discrete logarithm on elliptic curves of trace one is not a one-way function. It is just as easy and fast (for a computer, at least) to compute the discrete log as it is to do the multiplication by $[n]$ in the first place. Therefore, curves used for cryptography must not have trace one.

Luckily, this is easy to avoid. There are methods to determine the number of points on an elliptic curve that do not require explicitly finding the points (see [BSS99] Chapter VI) and it is not particularly common for a curve to have trace one, as, by Hasse's Theorem, the group order satisfies

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

so there is quite a possible range of possible orders.

4.4 Python Script

We present here the Python script used to do the computations shown in the previous section, which can be used to reproduce the results shown, or to do experiments with any other elliptic curve.

Note that someone with just a basic knowledge of programming, and access to the neccessary algorithm specifications, could write code similar

to that below. Thus, even though the mathematics is advanced, the practical method truly is simple.

```
from sage.all import *
def discrete_log(P, Q, E, p):
    # Create fields
    F    = GF(p)
    Q_p = Qp(p, prec = 3, \
    type = 'capped-rel', print_mode = 'series')
    print P
    print Q
    # Compute lift
    liftP = lift(P, E, F, Q_p)
    liftQ = lift(Q, E, F, Q_p)
    print liftP
    print liftQ
    # Multiply by [p]
    multP = successiveSquare(p, liftP, E)
    multQ = successiveSquare(p, liftQ, E)
    print multP
    print multQ
    # Compute psi(P) and psi(Q)
    psiP = psi(multP)
    psiQ = psi(multQ)
    print psiP
    print psiQ
    # Compute final answer
    pval = F( psiP.add_bigoh(2) / p )
    qval = F( psiQ.add_bigoh(2) / p )
    print pval
    print qval
    log = qval / pval
    print log
def lift(P, E, F, Q):
    x = P[0]
    y = P[1]
    a = E[0]
```

```
    b = E[1]
    p = int( F.order() )
    newx = Q(x)
    h0 = y
    h1numer = ((y ** 2) - (x ** 3) - (a * x) - b) / p
    h1 = int( - F(h1numer) / F(2 * y) )
    h2numer = (( (y + h1*p) ** 2) - (x ** 3) - a * x - b) / (p ** 2)
    h2 = int( - F(h2numer) / F(2 * (y + h1*p)) )
    newy = Q(h0 + h1*p + h2*(p**2))
    return [newx, newy]
def psi(P):
    x = P[0]
    y = P[1]
    return - (x / y)
def successiveSquare(m, P, E):
    if m==0: return "INF"
    elif m==1: return P
    elif (m % 2 == 0):
      return successiveSquare(m/2, double(P, E), E)
    else:
      return add(P, successiveSquare(m/2, double(P, E), E), E)
def double(P,E):
    if P == "INF": return P
    elif P == neg(P): return "INF"
    x = P[0]
    y = P[1]
    a = E[0]
    N = ((3 * (x ** 2)) + a) / (2*y)
    newx = (N ** 2) - 2 * x
    newy = N * (x - newx) - y
    return [newx, newy]
def neg(P):
    return [P[0], -P[1]]
def add(P, Q, E):
    if P == "INF": return Q
    elif Q == "INF": return P
    elif P == Q: return double(P, E)
```

```
    elif P == neg(Q): return "INF"
    xp = P[0]
    yp = P[1]
    xq = Q[0]
    yq = Q[1]
    M = (yq - yp) / (xq - xp)
    xr = (M ** 2) - xp - xq
    yr = M*(xp-xr) - yp
    return [xr, yr]
```

Bibliography

[BSS99] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*, volume 265 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1999.

[Gou93] Fernando Q. Gouvêa. *p-adic numbers*. Universitext. Springer-Verlag, Berlin, 1993. An introduction.

[HMV04] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Professional Computing. Springer-Verlag, New York, 2004.

[Mon02] Jean Monnerat. Computation of the discrete logarithm on elliptic curves of trace one: Tutorial. Technical Report EPFL/IC/2002/49, Swiss Federal Institute of Technology, Lausanne, Switzerland, 2002.

[Sil86] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1986.

[Sma99] N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.