An Introduction to the Algebra behind Elliptic Curves

Skyler Marks[*]

November 20, 2024

Abstract

The digital world is kept secure by cryptography. The idea behind most modern cryptographic systems is that if I have a public number (usually called a public key) and a secret number (usually called a private key) I can perform some operation with them to generate a third number (which is also public). This number is easy to generate by combining a public and private key, but hard to generate any other way; this allows us to verify the authenticity of a private key very easily. One such operation involves finding collinear points on an elliptic curve, giving rise to elliptic curve cryptography. This talk introduces elliptic curves over finite fields, explains how we can use their algebraic geometry to define an appropriate operation, and touches on why this operation is appropriate for cryptography. Although this talk is considerably mathematical, no background beyond calculus and basic (high school) algebra is necessary (although some linear algebra and a basic grasp of set theory will help).

A Warning: Throughout these notes, I've been somewhat loose with notation, using conventions that are standard in algebra but may seem confusing to the layperson. This is because these notes are meant to accompany a lecture, and I have the opportunity to explain as I go in that setting. If you have any questions, please consult a textbook (Dummit & Foote is an excellent resource for algebra), or email me at `skyler@bu.edu`

1 Intuition

Elliptic curve cryptography is fairly simple on the intuitive level; elliptic curves have the interesting property that each (non-vertical) line intersects that curve in three points. This allows us to define a operation, kind of like multiplication, by finding successive intersections.

Definition 1.1.

Definition 1.2. *An **elliptic curve** is the set of pairs of 'numbers' (for an appropriate definition of 'numbers', as we will describe) (x, y) satisfying the equation:*

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

Generally, in math, this definition is a little more complicated, because mathematicians deal with more exotic spaces than just the complex plane. However, for the sake of simplicity, we'll just work in within the complex numbers. An abstract equation is a little tricky to work with, so let's graph a prototypical elliptic curve:

[*]Boston University

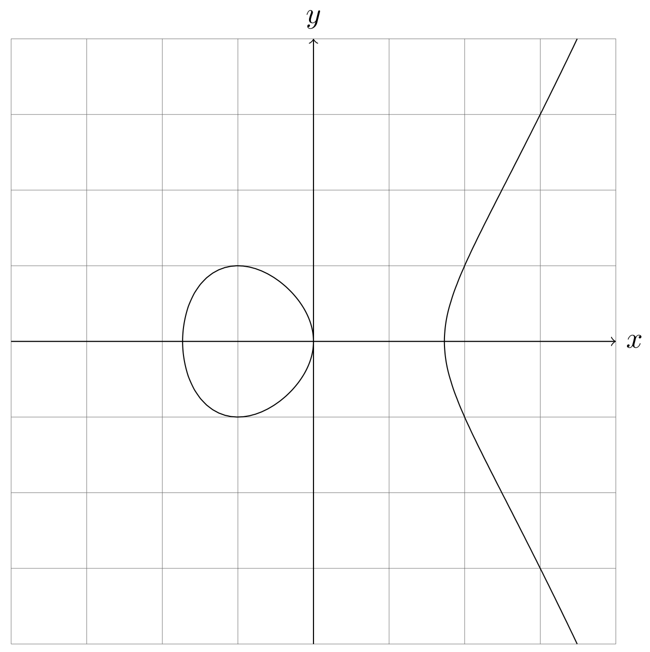

Figure 1: The graph of the equation $y^2 = x^3 - 3x$.

For my illustrations, I'll work with this curve - mostly because it looks nice. Some properties we can observe immediately include the symmetry over the x axis - this will be extremely useful for our algorithm. First, we pick a base point B on the curve - this will function like a seed for our algorithm. We'll also pick another point K. The main operation of the algorithm is to take a point K and connect it by a line with B. This line will then intersect the elliptic curve in a third point C:

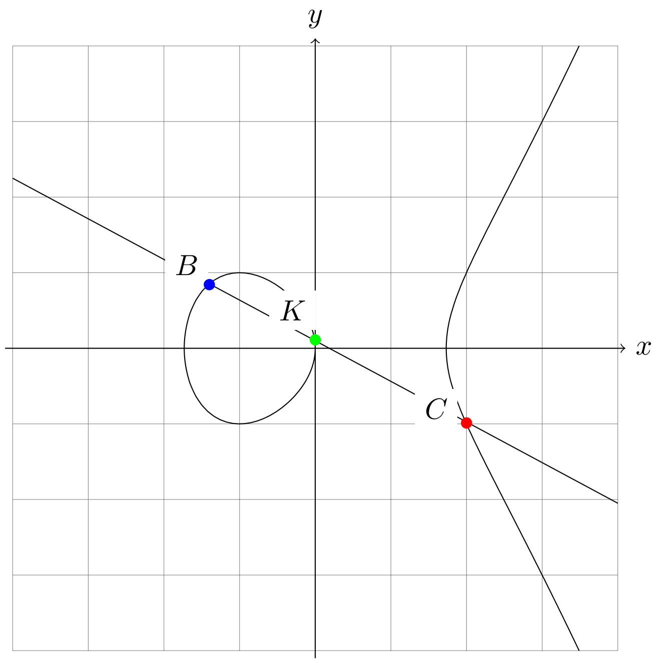

Figure 2: The first iteration of our process.

Now that we've got the gist of this operation down, we'll define our algorithm. The idea is we pick a private key n, which is just an integer (we'll see later on that there are some interesting ways to make the points B, C, and K integers as well). Leveraging the symmetry of the elliptic curve, we now reflect the point C over the X axis to obtain a new point on the elliptic curve K_1, and run the whole game again with K_1 to obtain a new point C_1:

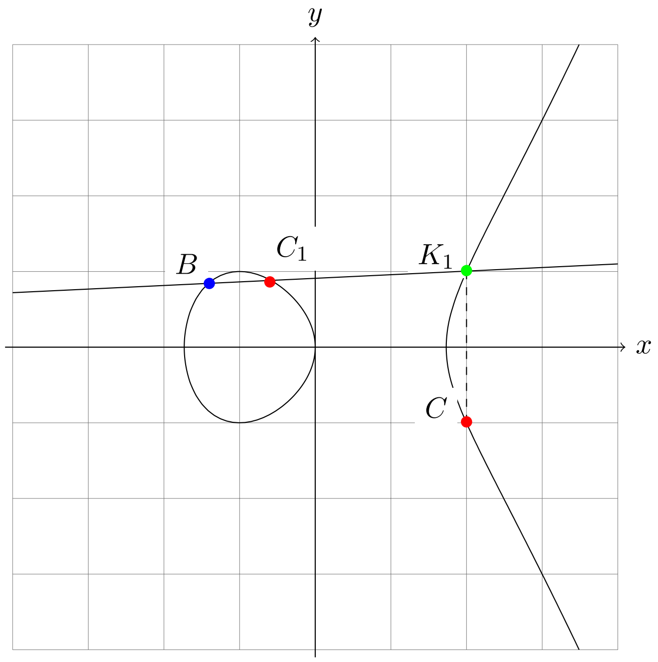

Figure 3: The second iteration of our process.

We continue this n times, reflecting C_{i-1} to K_i and finding a point C_i collinear with both K_i and B, to obtain a final point C_n, which will be our public key. It should be intuitively clear that, while it's easy to perform this process a given number of times, it's 'quite difficult' to work out how many times this process was performed in order to attain a particular result. Solving this problem (computing how many times this operation is performed) is called computing the **elliptic curve discrete logarithm** (a special case of the **discrete logarithm problem**). An active field of mathematical research is working out how difficult this problem is for elliptic curves; so far, no subexponential time has been found (see [1]) and the time complexity is greater than that of modular exponentiation.

This intuitive explanation is fairly complete, but lacks some nuance. In particular, this talk will address two main follow up questions to this naive treatment; firstly, we'll discuss a geometric formalism that allow us to deal with difficult edge cases regarding this operation, and secondly we'll use some abstract algebra to make computations with these curves feasible.

2 A whirlwind tour of Abstract Algebra

2.1 Set Theory

Set theory is the language of mathematics; we'll need some of it to formulate most of the notions we'll be discussing.

Definition 2.1. *For the purposes of this talk, a* ***set*** *will be a collection of objects. For more information, consider looking up the Wikipedia page for ZFC (Zermelo - Frankel -*

Choice set theory, the foundations for most modern math). An element of a set is one of the objects in the set. The notation $a \in A$ means that the object a is in the set A.

Definition 2.2. *Let A and B be sets. The union of A and B, denoted $A \cup B$, is the set of all elements which are in A or in B (inclusive or). The intersection of A and B, denoted $A \cap B$ is the set of all elements which are in A and in B. The difference $A - B$ or $A \setminus B$ is the set of all elements in A which are not in B.*

Definition 2.3. *A **function** from a set A to a set B is a rule that associates with every element of A an element of B. A function is **injective** if $f(a) = f(b)$ implies $a = b$, and **surjective** if for each b in B there is an a in A with $f(a) = b$.*

Definition 2.4. *The **Cartesian product** of two sets A and B, written $A \times B$, is the set of all pairs (a, b) with $a \in A$ and $b \in B$.*

Definition 2.5. *An **equivalence relation** on a set S is a subset R of $S \times S$ satisfying the following properties:*

- *Reflexivity: For every a in S, (a, a) is in R.*
- *Symmetry: If (a, b) in R, then (b, a) is in R.*
- *Transitivity: If (a, b) is in R and (b, c) is in R, then (b, a) is in R.*

We write $a \sim b$ to indicate that (a, b) is in the set R.

Lemma 2.1. *Let S be a set and $\sim$ be an equivalence relation on S. Let $[a]$ denote the set of all $b \in S$ satisfying $a \sim b$. Every $[a]$ is either equal or disjoint to every other $[b]$, and every element of S is in some $[a]$.*

Proof. By reflexivity, $a \in [a]$, so every element $a \in S$ is in $[a]$. If there is some c in both $[a]$ and $[b]$, then $a \sim c$ and $c \sim b$. But then because $d \sim b$ for each $d \in [b]$, we know that $a \sim d$ for each $d \in [b]$ by transitivity, so $[a]$ contains $[b]$. Similarly, $[b]$ contains $[a]$. $\square$

2.2 Groups

Somewhat central to elliptic curve cryptography (and most discrete logarithm style problems) is the notion of a group:

Definition 2.6. *A **group** is a set G equipped with a binary operation, that is, a function $f : G \times G \to G$. We'll often write the group operation using infix notation using an operator like $\bullet$; $(a \bullet b)$, for example, denotes $f(a, b)$. This binary operation satisfies the following properties:*

- *Associativity: $a \bullet (b \bullet c) = (a \bullet b) \bullet c$.*
- *Identity: There is an element e of the set G such that for each g in the set g, $e \bullet g = g \bullet e = g$.*
- *Inverses: For every g in the set G, there is an element g^{-1} in G satisfying $gg^{-1} = g^{-1}g = e$.*

Definition 2.7. *A group is **abelian** or **commutative** if $a \bullet b = b \bullet a$ for each $a, b \in G$.*

Example 2.1. The symmetries of a triangle are a group.

Example 2.2. The integers (under addition) form a group

Example 2.3. The integers modulo n form a group under addition.

Example 2.4. The integers modulo a *prime* p, if you take away 0, form a group under multiplication.

Proof. Associativity and identity follow from the fact that $a = b \implies a \equiv b \mod p$ (mod is a function); it suffices to show there are inverses. Let g be an element of G, the set of integers modulo a prime p. Recall that the GCD of two integers is a sum with integer coefficients of those integers (this can be proved using the Euclidean division algorithm). The GCD of g and p is 1, so there are integers a, b such that $ag + bp = 1$. Then $ag \equiv 1 \mod p$, so a is an inverse for g mod p (because modulo is multiplicative). $\square$

Exercise 2.1. Write (in pseudocode) an algorithm to find the GCD of two integers a and b using repeated division / finding the remainder. Prove that this algorithm terminates, and use it to show that the GCD of two integers can be written in the form $a'a + b'b$ where a', b' are also integers.

Exercise 2.2. If n is an non-prime integer, what elements in the set of integers mod n have multiplicative inverses?

Lemma 2.2 (Shoe-Socks Lemma)**.** *In any group,* $(ab)^{-1} = b^{-1}a^{-1}$.

Proof. Note that $b^{-1}a^{-1}ab = b^{-1}eb = e$. $\square$

Especially important to us are cyclic subgroups of a group.

Definition 2.8. *A* ***subgroup*** *H of a group G is a subset of G that satisfies the group axioms for the same operation as G.*

Definition 2.9. *A* ***cyclic subgroup generated by*** g *for some g in a group G is the set of all 'powers' of g, that is the set of all elements of the form $g \cdot g \cdot \ldots$ or $g^{-1} \cdot g^{-1} \cdot \ldots$, together with the identity.*

Exercise 2.3. Check that a cyclic subgroup is a subgroup!

2.3 Rings and Fields

Definition 2.10. *A* ***ring*** *is a set R with two binary operations called multiplication and addition, satisfying the following properties:*

- *Both operations are associative*
- *The set R is a group under multiplication with identity 0*
- *There is a multiplicative identity* 1
- *Multiplication distributes over addition; i.e., for every $a, b, c \in R$*

$$a(b + c) = ab + ac \text{ and } (b + c)a = ba + ca$$

A ring is called ***commutative*** *if $ab = ba$ for all a and b in the ring.*

Lemma 2.3 (Nifty Facts About Rings)**.** *For any a in R, we have that* $0a = 0$*; furthermore, the additive group of a ring R is abelian.*

Proof. Suppose $a \in R$. Then $a(0+1) = a$. Distributing and subtracting a on both sides yields $a0 = 0$. If $0 = 1$, this implies $a = 0$ for every a. If not, then for $a, b \in R$ we have $-1(a+b) = (-1a + -1b) = -a - b$. But by Shoe-Socks, $-1(a+b) = -b - a$, so $-a - b = -b - a$. Multiplying both sides by -1 and distributing gives $a + b = b + a$. □

Example 2.5. The integers are a ring.

Example 2.6. The integers mod n are a ring.

Example 2.7. Polynomials in n variables with real, integer, or complex coefficients (actually, in any ring) form a ring under the multiplication and addition formulas we're familiar with.

Definition 2.11. *A **subring** of a ring R is a subgroup of the additive group of R that is closed under multiplication. Subrings do not need to include 1.*

Definition 2.12. *An **ideal** of a ring R is a subring of R that is closed under multiplication by every element in the ring. If $\{a_k\}_{k \in K}$ is a collection of elements of R, the **ideal generated by** $\{a_k\}_{k \in K}$ or $(\{a_k\}_{k \in K})$ is the set of all sums of R-multiples of elements of the set $\{a_k\}_{k \in K}$.*

Definition 2.13. *An ideal I of R is **prime** if, for any elements a and b in R, $ab \in I$ implies either a or b (or both) in I. Equivalently, $R - I$ is multiplicatively closed.*

Definition 2.14. *An ideal $\mathfrak{m}$ of R is maximal if there is no ideal I satisfying $\mathfrak{m} \subsetneq I \subsetneq R$.*

Example 2.8. The ideals of $\mathbb{Z}$ are of the form $n\mathbb{Z}$. The primes are of the form $p\mathbb{Z}$ for p prime.

The ideal structure of polynomials is extremely complex and the study of them comprises a large part of the mathematical discipline known as *commutative algebra*.

Definition 2.15. *A **ascending chain** of ideals is a set of ideals $\mathfrak{a}_1, ..., \mathfrak{a}_n$ of a ring R satisfying $\mathfrak{a}_1 \subsetneq \mathfrak{a}_2 \subsetneq \mathfrak{a}_3 \subsetneq ... \subsetneq R$.*

Definition 2.16. *A ring is **Noetherian** if every ascending chain of ideals terminates.*

Exercise 2.4 (Hilbert's Basis Theorem)**.** Show that if R is a Noetherian ring, then $R[x]$ is likewise a Noetherian ring.

Corollary 2.1. *If R is a Noetherian ring, then $R[x_1, ..., x_n]$ is Noetherian.*

Exercise 2.5. The ideals of every Noetherian ring are finitely generated.

Definition 2.17. *The **codimension** of a Noetherian ring is the length of the longest ascending chain of ideals.*

Definition 2.18. *A **field** is a commutative ring F where the set $F - \{0\}$ is a group under the ring multiplication.*

Example 2.9. The rational numbers $\mathbb{Q}$, the set of ratios $\frac{p}{q}$ for p, q integers, form a field under the standard 'fraction multiplication'.

Example 2.10. The real numbers $\mathbb{R}$ and the complex numbers $\mathbb{Q}$ are fields.

Example 2.11. The integers modulo a prime p are a field.

Proof. Examples 2.3 and 2.2. □

Definition 2.19. *A* ***vector space*** *over a field k is an abelian group V together with an operation $\cdot \colon k \times V \to V$ called scalar multiplication that is distributive and satisfies $0 \cdot v = \vec{0}$ (where $\vec{0}$ is the identity of the group) and $1 \cdot v = v$.*

Example 2.12. The Cartesian product $k \times k \times k ... \times k$ is a vector space under componentwise addition and the scalar multiplication law:

$$x \cdot (a_1, a_2, ..., a_n) = (xa_1, xa_2, ..., xa_n)$$

We call this construction **affine n-space over k**

3 Zero Sets and Projective Space

We want to discuss the zero set of a particular polynomial, in a very general sense. To do this, we introduce some terminology:

Definition 3.1. *Fix a field k. The* ***zero set*** *of a polynomial $P(x_1, ..., x_n)$ is the set of all $(x_1, ..., x_n)$ in affine n-space such that $P(x_1, ..., x_n) = 0$.*

Definition 3.2. ***Projective n-space*** *over a field k is the set of equivalence classes of the set $k \times k \times .... \times k - (0, ..., 0)$ (multiplied $n+1$ times) by the equivalence relation $a \sim b$ if and only if $(a_1, ..., a_{n+1}) = \lambda(b_1, ..., b_{n+1})$*

Projective space is called 'projective' because it can be characterized as the set of all lines through the origin (or the additive identity of the vector space), and the notion of identifying a point with the line through it and the origin yields a formalization of the notion of projection.

Exercise 3.1. For those of you with a linear algebra or multivariate calculus background, convince yourself of the above statement.

The naming convention for projective n-space (as a quotient of affine $n+1$ space) may seem strange, but it comes from the fact that the equivalence relation 'cuts down' the dimension by 1. In particular, projective n-space 'looks locally' like affine n space; we can write down a function f the set of all points (equivalence classes) in projective n-space where the kth variable is nonzero to affine n space by the rule:

$$[(x_1, ..., x_n)] \mapsto \left(\frac{x_1}{x_k}, ..., \hat{x}_k, ..., \frac{x_n}{x_k}\right)$$

This map is 'continuous' in a way that we don't have time to make precise, but effectively it identifies affine space with projective space in a way that preserves geometric properties. This also motivates the idea that projective space 'adds points at infinity' - we can think of the points as infinity as the limits as that one coordinate goes to zero (after fixing a preferred coordinate at infinity).

We ultimately want to work with polynomial equations (specifically elliptic curves) defined over projective space. Naively, we could try to just plug the variables for projective space into a polynomial; however, that doesn't yield a well defined function. If P is a polynomial in n variables, in general we have:

$$\lambda P(x_1, ..., x_n) \neq P(\lambda x_1, ..., \lambda x_n)$$

In fact, equality only holds when P is linear. To fix this, we introduce:

Definition 3.3. *The **degree** of a term in a polynomial is the sum of the powers of the indeterminate variables in that term. A **homogeneous polynomial** in n variables is a polynomial who's terms all have the same degree.*

Example 3.1. The polynomials $f(x) = x^2$, $g(x,y) = x^3 + 3xy^2 + 6y^3$, and $h(x,y,z) = x^3 + y^2x + x^2y + y^3 + z^3$ are all homogeneous polynomials.

The main reason for this construction is that

Lemma 3.1. *The 'zero' of a homogeneous polynomial is a well defined notion in projective space.*

Proof. Let $[x] = [(x_1, ..., x_{n+1})]$ be an equivalence class in projective n-space over k and let P be a homogeneous polynomial of degree r. Then if $(y_1, ..., y_{n+1})$ is in $[x]$, we must have that $(y_1, ..., y_n) = (\lambda x_1, ..., \lambda x_n)$, and so $P(y_1, ..., y_n) = \lambda^r P(x_1, ..., x_n)$, which is zero if and only if $P(x_1, ..., x_n)$ is zero. $\square$

We can thus talk meaningfully about the zero set of a homogeneous polynomial in projective space. The use for this is that projective space effectively adds 'points at infinity' to our space; where, in the past, we would have some lines going off to infinity, now we can keep track of what happens at those points.

We can take a curve defined by a polynomial $P(x,y)$ with maximum degree n in affine 2-space (such as our chosen elliptic curve) and embed it into projective space by considering the zero set of the homogeneous polynomial $z^n P(x/z, y/z)$. A quick check demonstrates that all the zs in the denominator cancel, and the result is a homogeneous polynomial of degree n. Moreover, letting $z = 1$ recovers our original polynomial; as such, the 'affine zeros' will remain the same (we'll just add some zeros 'at infinity', or when $z = 0$). This notion becomes slightly more complex in more dimensions.

Definition 3.4. *A **projective variety** is the zero set of some collection of homogeneous polynomials in projective space.*

Definition 3.5. *A **projective variety of codimension one** is the zero set of one homogeneous polynomial in projective space.*

Definition 3.6. *A **divisor** is a formal sum with integer coefficients of codimension 1 subvarieties.*

We can intersect divisors of $\mathbb{P}^n$ with a variety V in $\mathbb{P}^n$ to obtain new divisors on that variety V.

Theorem 3.1. *The only rational functions defined on all of $\mathbb{P}^n$ are linear.*

Proof. Only linear functions preserve scalar multiples. $\square$

More notes on this part coming soon!

4 Putting it All Together

Now that we've developed some machinery, we get to play! Here's the gist of what we do.

1. Take an **elliptic curve** defined by a polynomial equation $P(x,y)$ over a **finite field** k (for computability).

2. Pick a **base point** for our elliptic curve.

3. Embed this curve into **projective space** using the homogeneous polynomials associated to $P(x, y)$.

4. We now have a **group** based on our geometric operation outlined in the introduction: Find collinear points, reflect across the y-axis. The identity of this group is the point at infinity, and the inverse of a point $[(x, y, z)]$ is $[(x, -y, z)]$ (this should be clear from geometry). Associativity for this group law, as well as closure (that adding two points on the curve gives you another point on the curve) are complicated and technical results that I don't completely understand at this point, so I'll spare you the details for now - maybe next time!

5. Pick a **private key** (some integer n)

6. Now add the base point *to itself* n times - where n is your private key. The first of these additions is a little strange - adding a point to itself. The idea here is that (if we think in the real numbers for a moment) you take the *tangent* to this point (the limit as another point gets closer and closer to the base point). This can be made precise for a finite field using algebraic geometry, but the formalism requires far too much machinery for the scope of this talk. Suffice it to say, we obtain a point which is not the base point, and we continue with the operation, finding collinear points. There are explicit ways to compute this; see [2]. We omit these for brevity. This generates your **public key**.

7. If one was to simply add a point to itself over and over again, the time complexity would be prohibitive; we can, instead, repeatedly double the point and add these doubles to obtain the required number of repeated additions.

In order to have a large enough keyspace to make this system worthwhile, people selecting these curves and base points try to chose a base point which maximizes the size of the **cyclic group** generated by that base point. If the cyclic group is too small, it would be relatively easy to determine how many times you multiplied the base point together to get your public key. However, by picking a base point (and elliptic curve) which together lead to a large enough cyclic group, the system is made secure.

In addition to giving us our identity, it's my understanding that embedding into projective space somehow eliminates an inversion operation of a finite field element in the group law, thereby making the point addition far faster (finding inverses in a finite field is slow).

5 Gaps in this Picture

Although our image is much more rigorous now, we're still missing three main ideas, namely

- How do we compute the group law?
- Closure
- Computing $2P$ - adding a point to itself.

For more on these take a look at [2]

6 Computations with TinyEC

I wrote a quick script that I'll try to run for you all in python. Credit to `https://github.com/nakov/Practical-Cryptography-for-Developers-Book` for some of this code. The key here is recommended in `https://doi.org/10.6028/NIST.FIPS.186-4`, and I believe still extremely secure (I chose the highest level of security).

```
#!/usr/bin/python

from tinyec import registry
import random

curve = registry.get_curve('secp521r1') # This is the largest prime field key
                                        # recommended by the NSA as of
                                        # recentlyish.
print("There are", curve.field.h, "cyclic groups associated with this field.")
print("""The order of the cyclic group generated on this curve by this
         base point is:,""", curve.field.n)
privKey = random.randint(0, curve.field.n) # random isn't secure but it's fine.
pubKey = privKey * curve.g
print("My public key is:", pubKey)
print("I check that my private key works, and obtain:", privKey * curve.g)
print("I can't read that. Is it equal:", privKey * curve.g == pubKey)
print("My private key is (sshhh, don't tell):", privKey)
```

As a bonus... What does an elliptic curve over a finite field look like? For this we go back to our friend $y^2 = x^3 - 3x$, who looked so nice over the reals. Turns out when you plot this curve over a finite field, it looks a little... Disjoint. Here's some code:

```
import numpy
import matplotlib.pyplot as plt

p = 257 # To irritate the programmers
grid = numpy.zeros((p, p))
for i in range(0, p):
    for j in range(0, p):
        if (i**2)%p == (j**3-3*j)%p:
            grid[i][j] = 1
for j in range(0, p//2):
    grid[p, j] = 1

plt.imshow(grid, interpolation="nearest")
plt.show()
```

And the final product:

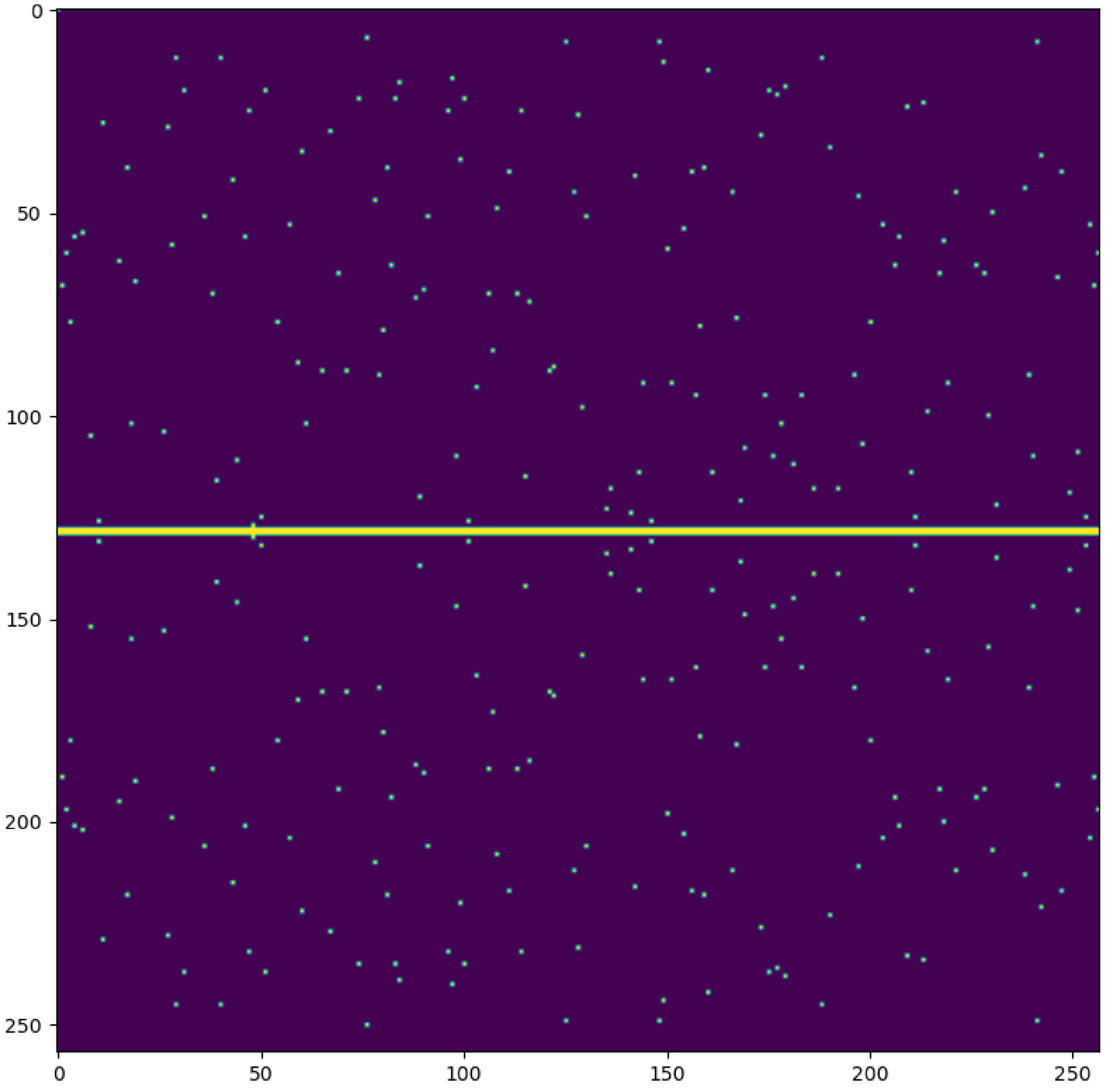

Figure 4: The elliptic curve $y^2 = x^3 - 3x$ over the integers mod 257. Note the symmetry across the line in the middle (which is actually the 0 axis), and think back to our group operation, reflecting across the zero-axis...

References

[1] Steven D. Galbraith and Pierrick Gaudry. “Recent progress on the elliptic curve discrete logarithm problem”. In: *Designs, Codes and Cryptography* 78 (Nov. 2015), pp. 51–72. DOI: `10.1007/s10623-015-0146-7`. URL: `https://eprint.iacr.org/2015/1022.pdf` (visited on 11/30/2020).

[2] Ben Lynn. *Elliptic Curves - Elliptic Curves*. crypto.stanford.edu. URL: `https://crypto.stanford.edu/pbc/notes/elliptic/`.