An Introduction to Elliptic Curve Cryptography

Andrej Ljubić

Teddy Rocks Maths Essay Competition

1 A Brief Introduction to Cryptography

Since the dawn of time, humans have needed to communicate in secret. Generals needed to command armies, bosses needed to send private emails and shoppers needed to submit their credit card details without them being made public. **Cryptography** is the field of mathematics that concerns itself with keeping important communications secret from those who should not have access.

In general, there are two types of cryptography - **symmetric** and **asymmetric**. Symmetric cryptography can be thought of like a bike lock - when you want to lock your bike, you use your **private key**. Nobody can access or unlock the bike unless they have your key. Symmetric cryptography works on the same principle - if Alice wants to send a message to Bob, she uses her **private key** to **encrypt** her message, producing what we call **ciphertext**. Once Bob receives the ciphertext, he uses the **same key** to **decrypt** the message by performing the inverse operation. We call this *symmetric* because there is such a symmetry between the encryption and decryption - both operations are the same, just performed in a reverse order.

Asymmetric cryptography is more like a **ballot box** in a voting system. Absolutely anyone can cast their vote and place it inside the box - but once it's in the box, there's no getting it back. In order to retrieve the vote, the owner has to use their personal **private key** to open up the box and extract all the votes. In asymmetric cryptography, we do a similar thing. Alice creates two keys - a **public key**, which everyone has access to, and a **private key**, which only she has. Anybody who wants to send Alice any secret information uses the **public key** to encrypt their data - hundreds of people could do this. But the encryption method is designed in such a way that the operation is *irreversible* given only the public key - in order to decrypt the data, Alice has to use her **private key**. This is called *asymmetric* because the encryption and decryption operations rely on different keys.

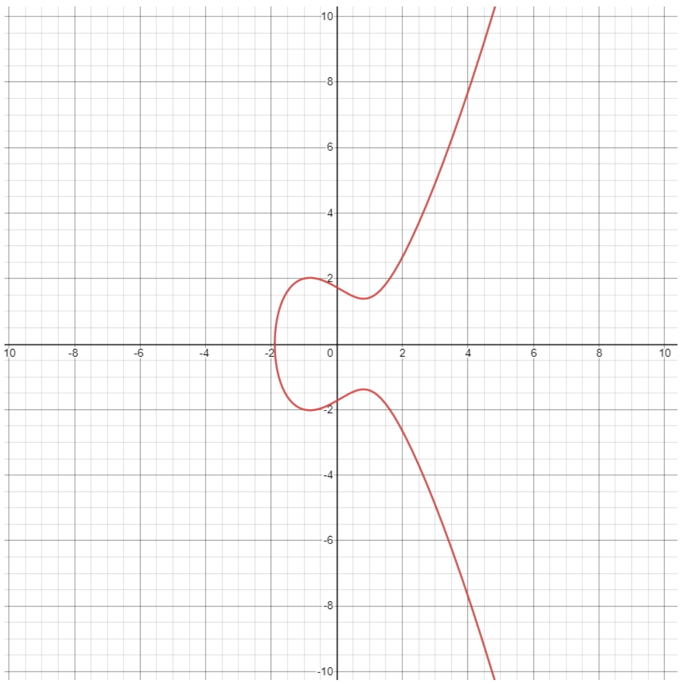
When implemented, symmetric cryptography is much faster. So why do we use asymmetric? Well, in practise, symmetric requires both the sender and the receiver to have access to the **same key**, which cannot be accessed by anybody else. In the era of computers and the internet, millions of users are accessing thousands of websites every second - and there is no secure way to share the key across public channels, for what is to stop people from eavesdropping? This is why we utilise asymmetric cryptography to negotiate a **shared secret**, a value known only to the two involved in the exchange. This shared secret is then used as the key for the faster symmetric protocol in future communications. Elliptic Curve Cryptography (often abbreviated ECC) is one such asymmetric protocol used for generating this shared secret.

2 Elliptic Curves

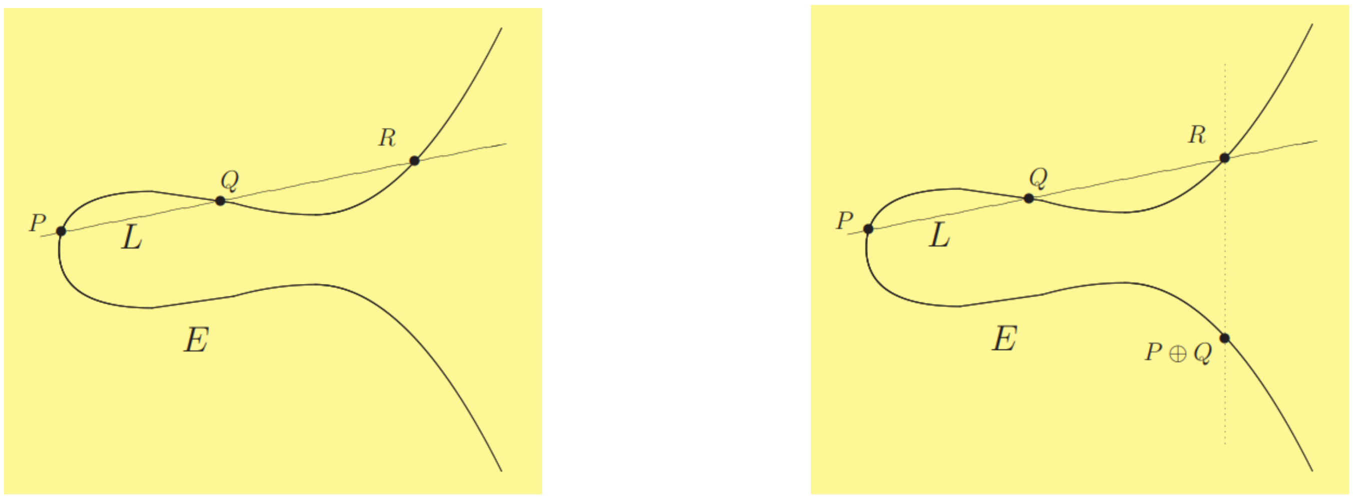
Confusingly, elliptic curves actually have nothing to do with the concept of an **ellipse** - they are completely distinct, but unfortunately named. Elliptic Curves are, quite simply, a special type of curve. Here we'll be talking about curves in **Weierstrass Form**, which looks like the following:

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

Where a and b are integers. A good example is the following, from the equation $y^2 = x^3 - 2x + 3$:



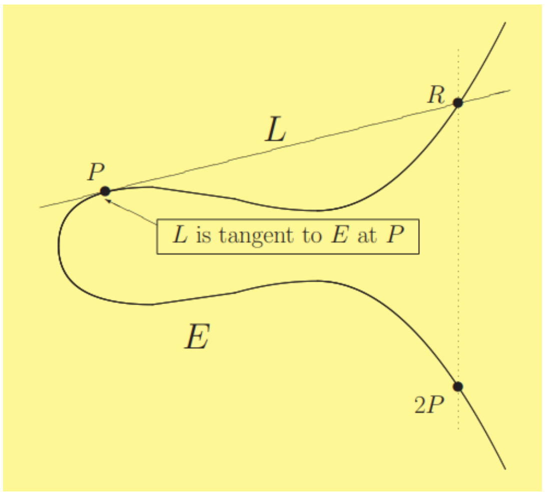
The **rational points** on an elliptic curve are all the points on the curve with rational x and y values (rational numbers are those that can be represented as fractions). The beauty of these curves is that the rational points form a **group**, meaning they are all linked by an operation. If we choose any two rational points, drawing a line between them intersects the curve at a third point R. If we reflect R in the x-axis, we get the point R' and we say this point R' is the result of the **point addition** of P and Q.



(a) Finding the third point R (b) Finding $P + Q$

Figure 1: Images courtesy of Joseph Silverman's 2006 presentation slides

We can also **double** a point, meaning we compute $P + P$. Remember that in this context (and from now on) the $+$ symbol is not regular addition but our new operation **point addition**, which given two points on the curve returns a third. How we do that may not be immediately obvious, for surely there are an infinite number of lines passing through P? What we do in this case is take the **tangent to the curve**. The tangent is a line which only touches the curve at one point - you can think of it as being exactly parallel to the specific point you are drawing it at. Once we find the point of intersection of the curve, we again reflect it in the x-axis to calculate $2P$.



(a) Finding $P + P = 2P$

Figure 2: Image courtesy of Joseph Silverman's 2006 presentation slides

Calculating the tangent involves some **implicit differentiation**, which is a technique that gives us an equation for calculating the gradient of the tangent at a specific point (x, y):

$$\begin{aligned} y^2 &= x^3 - 2x + 3 \\ 2y\frac{dy}{dx} &= 3x^2 - 2 \\ \frac{dy}{dx} &= \frac{3x^2 - 2}{2y} \end{aligned}$$

And substituting the point $P = (x, y)$ yields the gradient at the point P.

You may ask what happens if you attempt to calculate $P + (-P)$, i.e. the point on the curve with the same x-coordinate but the negative y-coordinate. Clearly, if we go through both points, there is no third intersection with the curve, and point addition fails. We say the result of this addition is $\mathcal{O}$, the **point at infinity**. If you want to add this point to any point X, we say that $X + \mathcal{O} = X$ - you can treat $\mathcal{O}$ as 0 in regular arithmetic, since it's what we call an **identity element** - adding to it has no effect.

You would be right to notice that the numbers could get **very** big **very** quickly, and square rooting these huge numbers would be extremely resource-intensive. Instead, when we use ECC practically, we choose a large prime number p. When we calculate points, we take the x and y coordinates and divide them by p, keeping the remainder - this is called a calculation *modulo p*. The group laws and the rules of point addition still hold (which is incredible!), but the numbers remain small and by keeping the numbers small the calculations become much more efficient.

3 Generating a Shared Secret

Now we have a decent geometric understanding of what adding and doubling mean in terms of rational points on the curve, we can discuss how this is used to generate a **shared secret**.

If Alice and Bob wish to generate a shared secret, they first choose an elliptic curve. The curve can be made public, but this doesn't compromise the security of the exchange. Think of the example of $y^2 = x^3 - 2x + 3$ from earlier. They next choose a rational point G, which we call the **generator** point. This point is also made public.

Now Alice and Bob generate a private number each, and we'll call these numbers a and b respectively. Alice then calculates the rational point A, which is done by adding the generator G to itself a times, i.e. $A = aG$. Bob does the same with *his* private number to calculate $B = bG$. Remember that even though we write bG as if it's multiplication, it's actually just b series of **point** additions (not regular addition!) of G to itself! The multiplication notation is just a handy shortcut. Once this is done, Alice sends A to Bob and Bob sends B back.

Let's say that $a = 1000$ and $b = 800$. Alice will receive $B = 800 \times G$ from Bob while Bob receives $A = 1000 \times G$ from Alice. The trick here is that Alice can now add B to G another a (1000) times, that is calculate $B{+}G{+}G{+}...{+}G$ until she calculates $S_a = B + 1000 \times G = 800 \times G + 1000 \times G = 1800 \times G$, the equivalent of performing a **point addition** of G to itself a total of 1800 times. Bob, in turn, adds A to G another b (800) times. Note that this means Bob computes $S_b = A + 800 \times G = 1000 \times G + 800 \times G + 1800G\times$. But that means $S_a = S_b$, as both are the result of adding G to itself a total of 1800 times!

The incredible thing here is that the value S is a **shared secret**, a value known only to Alice and Bob. The x-coordinate of S can be used as the private key for a symmetric protocol, with which Alice and Bob can communicate efficiently in secret.

4 The Security and Applications of ECC

At first glance, it's perhaps not evident as to why the shared secret is known only to Alice and Bob. After all, an attacker knows G and can intercept A and B. Since we know that $A = a \times G$, one could reason that since the only value in the equation we don't know is a we could rearrange it somehow and solve for a. This is known as the **Elliptic Curve Discrete Logarithm Problem**, or ECDLP, and is deceptively hard; there is no feasible way to solve this problem except in very specific circumstances where the curve parameters make the curve easier to attack (remember again that $\times$ denotes a sequence of **point additions**, not standard multiplication, so the rearrangement isn't quite that simple!). In reality, the curves used are standardised and chosen by professionals in the field, making it less and less likely for the security of the system to be compromised.

Elliptic curves have a wide variety of appliacations, most notably in the TLS cryptographic protocol - the protocol that underpins the entirety of the security of the web by providing encryption to HTTPS connections and, as a result, protecting your passwords and private information from being read. They are also used in **digital signatures**, schemes for proving the authenticity of digital messages and documents to assert that the contents have not been tampered with. Outside of cryptography, elliptic curves were used in Andrew Wiles' proof of *Fermat's Last Theorem*, which states that there are no integer solutions for n where integers $a, b, c > 0$ and $a^n + b^n = c^n$.