Lecture 6: Elliptic Curves

4/15/21

Plan

"let G be a group of prime order"

- Brief review of groups
- The original way of getting prime order groups
- The modern way - Elliptic Curves
  - → What are they?
  - → How to get a group?
  - → How to do crypto with them?

Logistics: - HW2 is out
- HW1 feedback

Brief review: groups

A set S and a mapping $\mu : S \times S \to S$

what does it take to be a (commutative) group?

Closure: for all $a, b \in G$, $a \cdot b \in G$

Associativity: for all $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

Identity: there exists $e \in G$ s.t. for all $a \in G$, $e \cdot a = a \cdot e = a$

Inverses: for each $a \in G$, there exists $b \in G$ (denoted as a^{-1}) s.t. $a \cdot b = b \cdot a = e$

Commutativity: for all $a, b \in G$, $a \cdot b = b \cdot a$

Some groups we've seen: $(\mathbb{Z}_p, +)$, $(\mathbb{Z}_p^*, \cdot)$

Note: we often refer to groups abstractly as $(G, \cdot)$ so regardless of what the actual group elements and operation are, we write them using multiplicative notation.

"Let G be a group of prime order"

We use this to describe many assumptions, e.g. DLog, CDH, DDH

But these assumptions are <u>not</u> true for just any group!

Example: A group where DLog is easy: $(\mathbb{Z}_p, +)$

Q: how to find DLog b/w $g, h \in \mathbb{Z}_p$, i.e. x s.t. $g \cdot x = h$

A: $x = h \cdot g^{-1}$

(In our usual notation, we denote the group operation with "$*$", so DLog is expressed as $g^x = h$. When the operation is "$+$", DLog is expressed as $g \cdot x = h$.)

So what group do we use?

The old way: start with $(\mathbb{Z}_p^*, \times)$ ← Multiplicative group mod p

issue: $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ has only $p-1$ elements, so it is not of prime order.

resolution: Work over a prime order subgroup of $\mathbb{Z}_p^*$.
use safe primes of the form $p = 2q+1$ where q is also prime.
This guarantees that there is a subgroup of order q.

Bigger issue: discrete log over $\mathbb{Z}_p^*$ is a little too easy

Subexponential time alg for dlog in $\mathbb{F}_p^*$ takes time $2^{\tilde{O}(\sqrt[3]{\log p})}$

For $\lambda = 128$ bits of security, NIST recommends $|p| = 3072$ bits!

big & slow

So how do we do it today?

Detour: hobbies in antiquity → finding points on curves

Pythagoras: find $(x,y) \in \mathbb{Q} \times \mathbb{Q}$ s.t. $x^2+y^2=1$

Fermat: find $(x,y,z) \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ s.t. $x^3+y^3=z^3$

Fermat's last theorem says there is no such x, y, z.

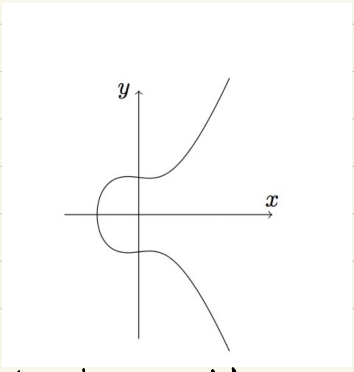
Diophantus: $(x,y) \in \mathbb{Q} \times \mathbb{Q}$ s.t. $y^2=x^3-x+9$

---

history lesson: Diophantus was really into finding rational solutions to bivariate equations and wrote many books on it. Centuries later, Fermat wrote his famous last theorem in the margins of one of these books. A few more centuries later, in the 1990s, the theorem was proven! A fictionalized version of the events surrounding the discovery of the proof is documented in the off-Broadway musical Fermat's Last Tango.

Elliptic curves are also part of the story of proving Fermat's last theorem!

What does Diophantus' weird curve look like?



(not drawn to scale)

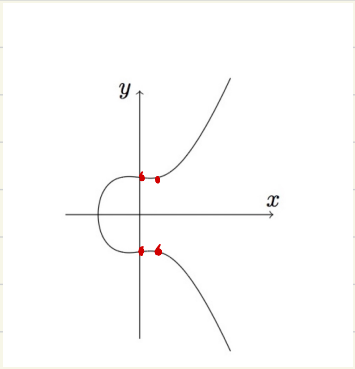
This is called an Elliptic Curve and generally has the form

$$y^2 = x^3 + Ax + B \quad \text{s.t. } 4a^3+27b^2 \neq 0$$

And these are the starting point for the groups we use in crypto today.

Building intuition for Elliptic Curves

Let's look at Diophantus' curve again: $y^2 = x^3 - x + 9$



Some easy rational points:

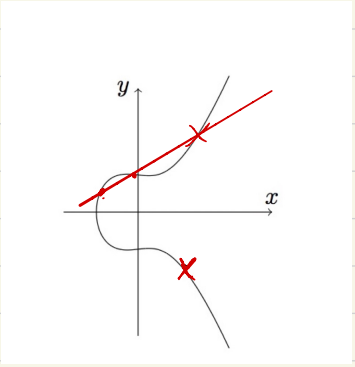
$x = 0,\ y = 3, -3$

$x = 1,\ y = 3, -3$

$x = 2,\ y = \sqrt{15}$ Not rational!

How to get more rational points?

1) if (x, y) is on curve, so is $(x, -y)$



2) Chord method: given 2 points on curve, drawing a line through them gives a third point on the curve

3) tangent method: given 1 point on curve, its tangent line crosses 1 other point on the curve.

So we can get new points on curve by drawing lines through known points and by flipping?

Do the points on the curve form a group with the "draw line + flip" operation?

It turns out they almost form a group.

Let $\tilde{E}(\mathbb{Q})$ represent the set of rational points on the curve.

"Draw line + flip" is a mapping $\boxplus : \tilde{E}(\mathbb{Q}) \times \tilde{E}(\mathbb{Q}) \to \tilde{E}(\mathbb{Q})$

But what is $(x,y) \boxplus (x,-y)$?

We get around this by adding a special point at infinity, denoted by O, and set the output of $\boxplus$ to be O whenever the slope of the line that would be drawn is undefined.

Also define $O \boxplus O = O$, $(x,y) \boxplus O = O \boxplus (x,y) = (x,y)$

Let $E(\mathbb{Q})$ represent $\tilde{E}(\mathbb{Q}) \cup \{O\}$. Is this a group?

what does it take to be a (commutative) group?

Closure: for all $a,b \in G$, $a \cdot b \in G$ ✓

Associativity: for all $a,b,c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ ✓

Identity: there exists $e \in G$ s.t. for all $a \in G$, $e \cdot a = a \cdot e = a$ ✓

Inverses: for each $a \in G$, there exists $b \in G$ (denoted as a^{-1}) s.t. $a \cdot b = b \cdot a = e$ ✓

Commutativity: for all $a,b \in G$, $a \cdot b = b \cdot a$ ✓

It's a group!!

Elliptic Curves over finite fields

We described $E(\mathbb{Q})$, the set of rational points on the curve

But for crypto, we use $E(\mathbb{F}_p)$, the set of points in $(\mathbb{F}_p \times \mathbb{F}_p)$ on the curve.

All the intuition above for $\mathbb{Q}$ transfers to $\mathbb{F}_p$

Some facts:

Hasse's Thm: $|E(\mathbb{F}_p)| = p+1+t$ where $|t| \leq 2\sqrt{p}$

Schoof's algorithm efficiently computes $|E(\mathbb{F}_p)|$

Of course we can't just use any curve or any choice of $\mathbb{F}_p$, we need to pick ones where discrete log is hard and $|E(\mathbb{F}_p)| = q$ for a prime q. More on this later.

↑ Why not p? Turns out discrete log is always easy when $|E(\mathbb{F}_p)| = p$.

Once we have such a group, we can use it for crypto!

Notation: for $X \in E(\mathbb{F}_p)$, $\alpha \in \mathbb{N}$, $\alpha X = \underbrace{X \boxplus X \boxplus \cdots \boxplus X}_{\alpha \text{ times}}$

Elliptic Curves → group abstraction

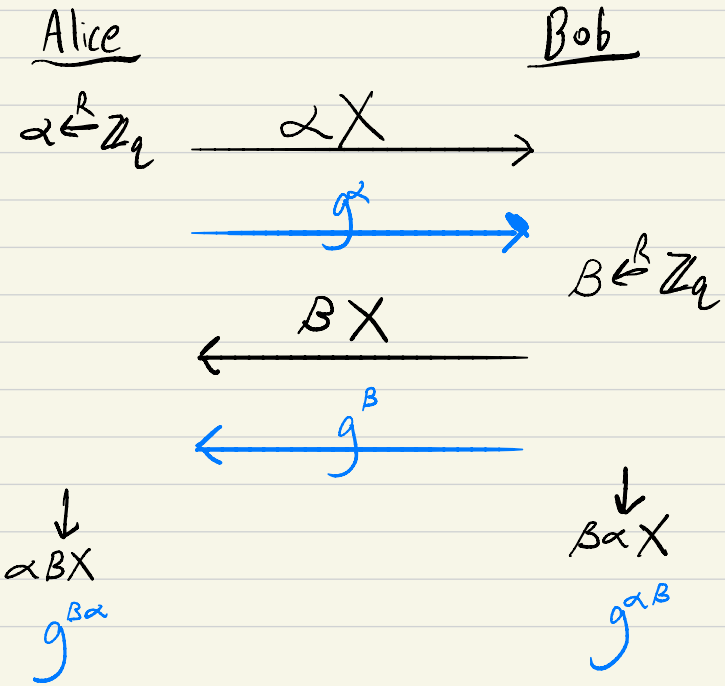
When we say we have elt g of a group G of prime order q

We mean...

We have a prime p, parameters $A, B \in \mathbb{F}_p$ s.t. $y^2 = x^3 + Ax + B$ is an elliptic curve, and

We have a point $X \in \mathbb{F}_p \times \mathbb{F}_p$ on $E(\mathbb{F}_p)$ that has order q.

Example: Diffie-Hellman Key exchange



(equivalent operation using group abstraction shown in blue)

What does it mean for discrete log to be hard in $E(\mathbb{F}_p)$?

Discrete log problem: Given $X, \alpha X \in E(\mathbb{F}_p)$, find α.

Using the group abstraction: Given g, $g^\alpha \in G$, find α.

For most elliptic curves, best known algorithm for discrete log runs in time $\Omega(\sqrt{q})$.

This means if we want 128 bits of security, we need $|q| \approx 256$ bits. q is usually close to p, so p will also be 256 bits. This means an element $X \in \mathbb{F}_p \times \mathbb{F}_p$ takes $256+256 = 512$ bits to represent. As a further optimization, we can just send the x-coordinate of a point.

Takeaway: for $\lambda = 128$ bits of security using elliptic curves, a group element can require as little as 256 bits = 32 Bytes!

12× smaller than the old way and faster too!

Elliptic Curves in the real world

Picking a curve where discrete log is hard is a difficult task w/many pitfalls, so we always use one of a few standard, well-studied curves.

NIST curve P256 (AKA secp256r1)

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

$$y^2 = x^3 - 3x + b \qquad b = \text{(see book)}$$

→ p designed for efficient computation mod p
→ b chosen by running a seed s through a public, deterministic alg.
→ no idea how s was chosen :/

P256 is used for $\lambda = 128$ bit security.
Can use secp521r1 for $\lambda = 256$ bit security if needed.

Curve25519

$$p = 2^{255} - 19 \leftarrow \text{largest prime} < 2^{256}$$

$$y^2 = x^3 + 486662x^2 + x$$

↑ This is not the standard form we saw before.
This is called a montgomery curve,
See Boneh-Shoup 15.2-15.3 for details

# of points on this curve is composite: $8q$. Need to work in order q subgp.

Wrapping up

- Most of the public key crypto used today uses elliptic curves
- This is the tip of the iceberg. Elliptic curves have many more nuances than covered here and also many more applications, eg.,
  - → Pairings (topic for next class)
  - → hashing to curves (how to go from $\{0,1\}^*$ to $E(\mathbb{F}_p)$)
  - → isogenies (applications to post-quantum crypto)
  - → and more...
- Most crypto libraries don't expose their elliptic curve groups directly as an API, but there are some special purpose elliptic curve libraries.
  - → Reminder: best to stick to reliable implementations of standard curves.
- We will typically continue to use the group abstraction, but it's good to know what's underneath in real implementations.