Bitcoin:
Programming the Future of Money

Topics in Computer Science - ITCS 4010/5010, Spring 2025

Dr. Christian Kümmerle

Lecture 11

Elliptic Curve-Based Digital Signature Schemes

Main Reference:
- "Programming Bitcoin: Learn How to Program Bitcoin from Scratch", Jimmy Song, 1st Edition, O'Reilly, 2019, Chapters 3
- "Mastering Bitcoin", Harding & Antonopoulos, Chapter 8 on "Digital Signatures"

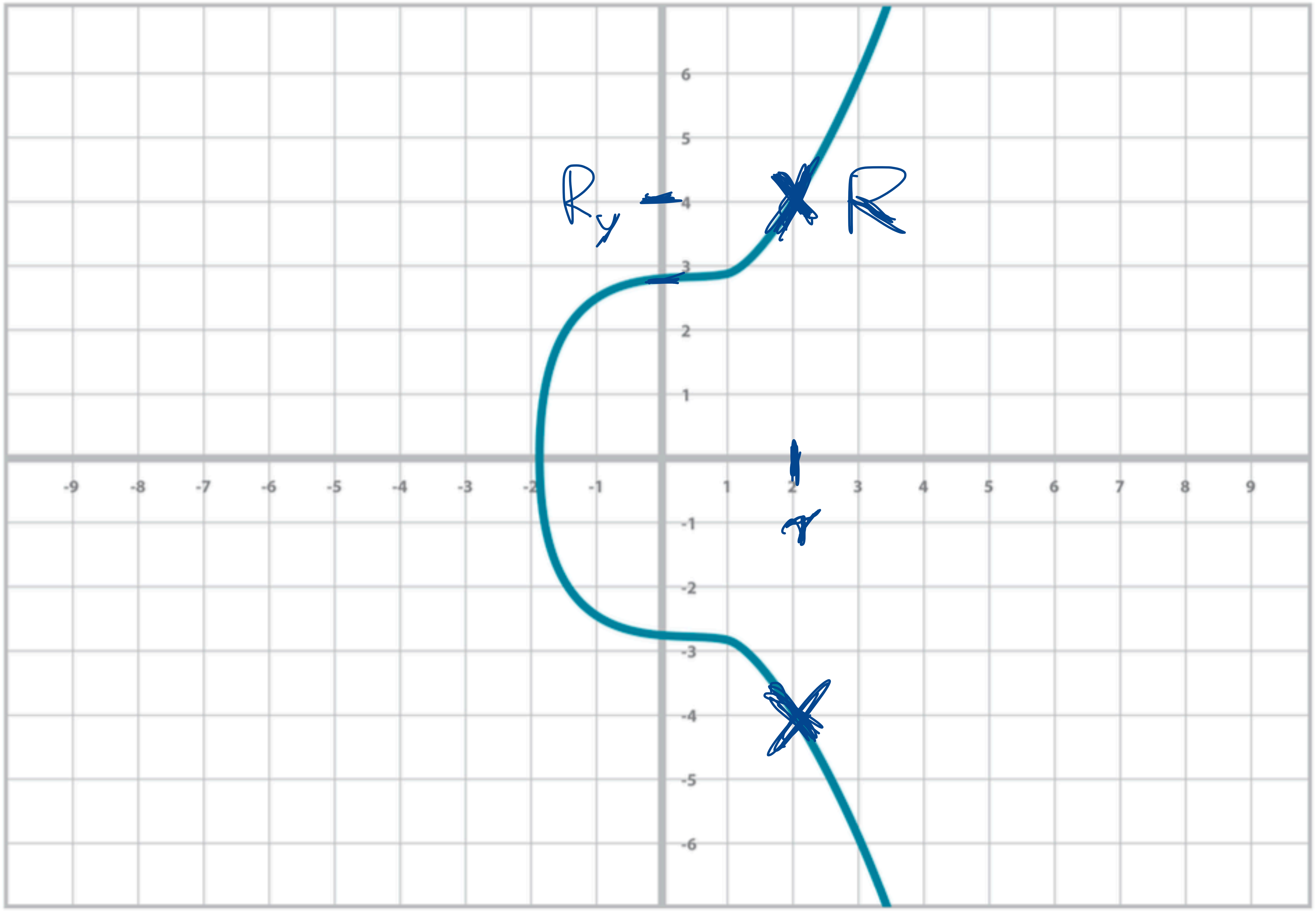
UNIVERSITY OF NORTH CAROLINA
CHARLOTTE

NEXT WEEK ON TUESDAY: MIDTERM

In-Class Midterm Exam on Tuesday, February 25

- Written exam
- 75 minutes duration (5:30-6:45 pm)
- You can bring one US letter-sized “cheat sheet” of paper (front and back), with handwritten notes on it.
- Non-programmable calculator is allowed
- No other tools are allowed
- Covers everything in this class so far
- Review suggestions: All lectures, all required readings, quizzes and homework

ELLIPTIC CURVE SECT256K1



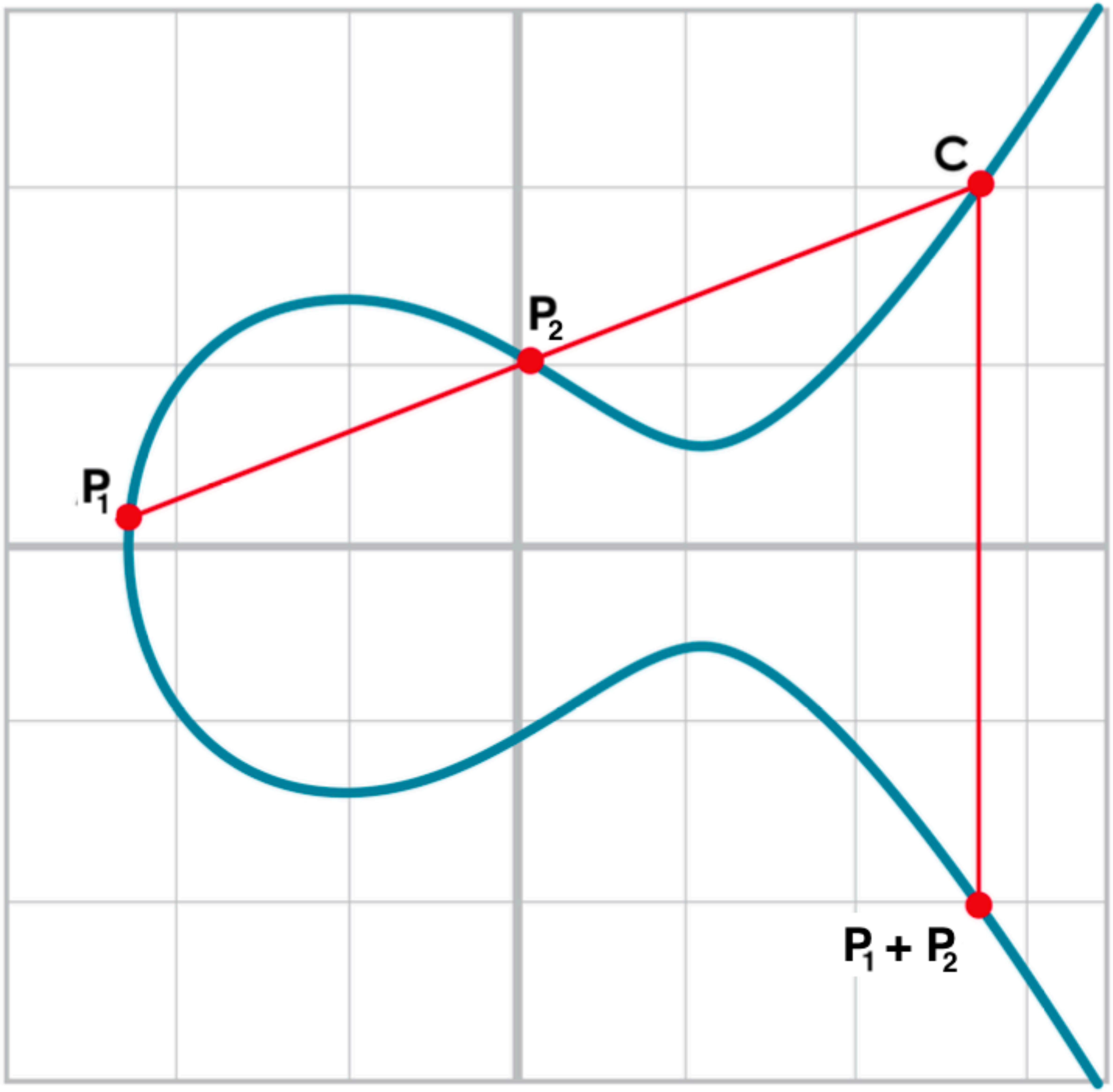
EC equation:

$$y^2 = x^3 + 7$$

$S_{a,b} = \{(x, y) : y^2 = x^3 + ax + b\}$
with $a = 0$ and $b = 7$.

POINT ADDITION ON ELLIPTIC CURVES

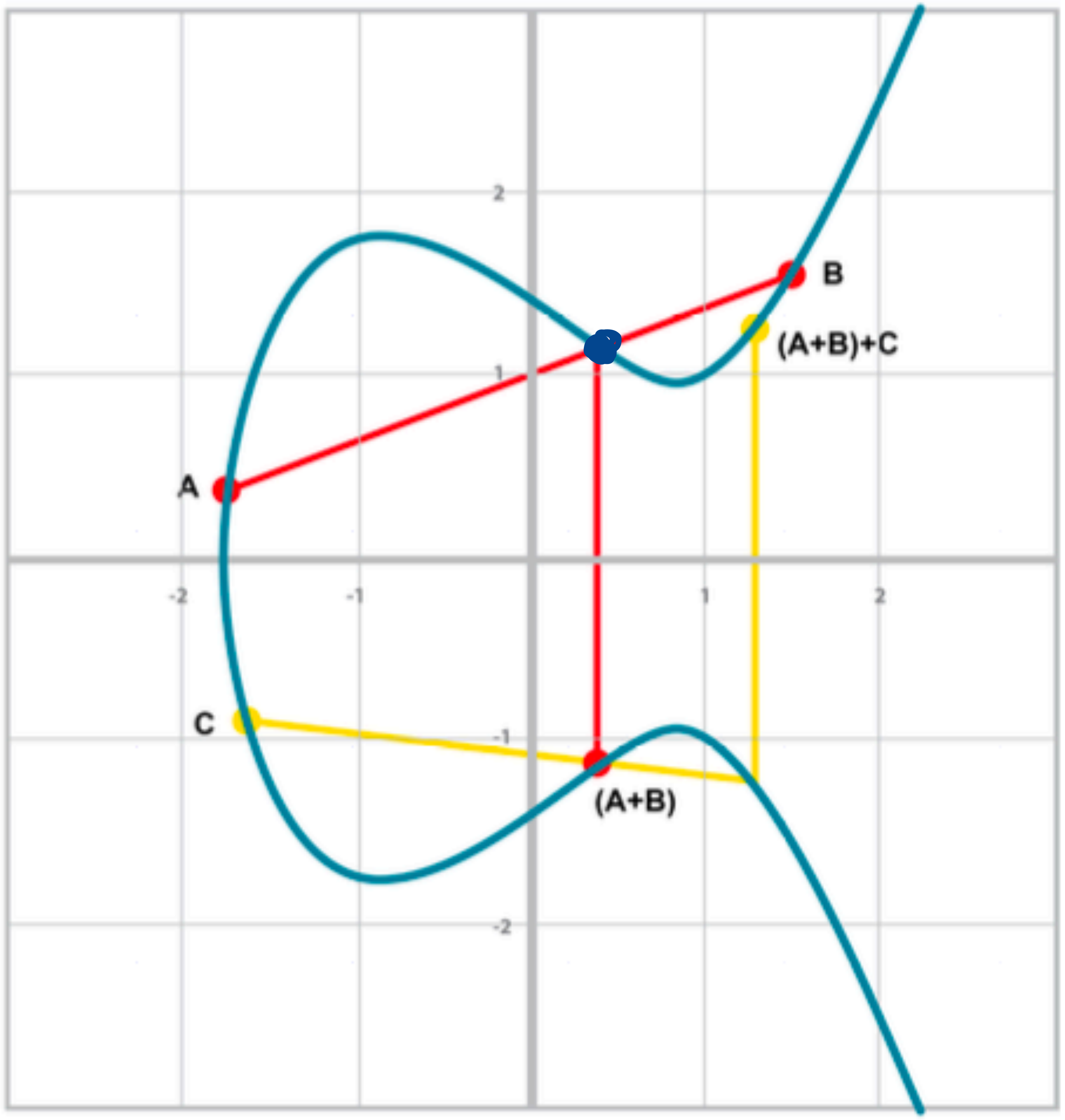
Answer:
We need associativity.



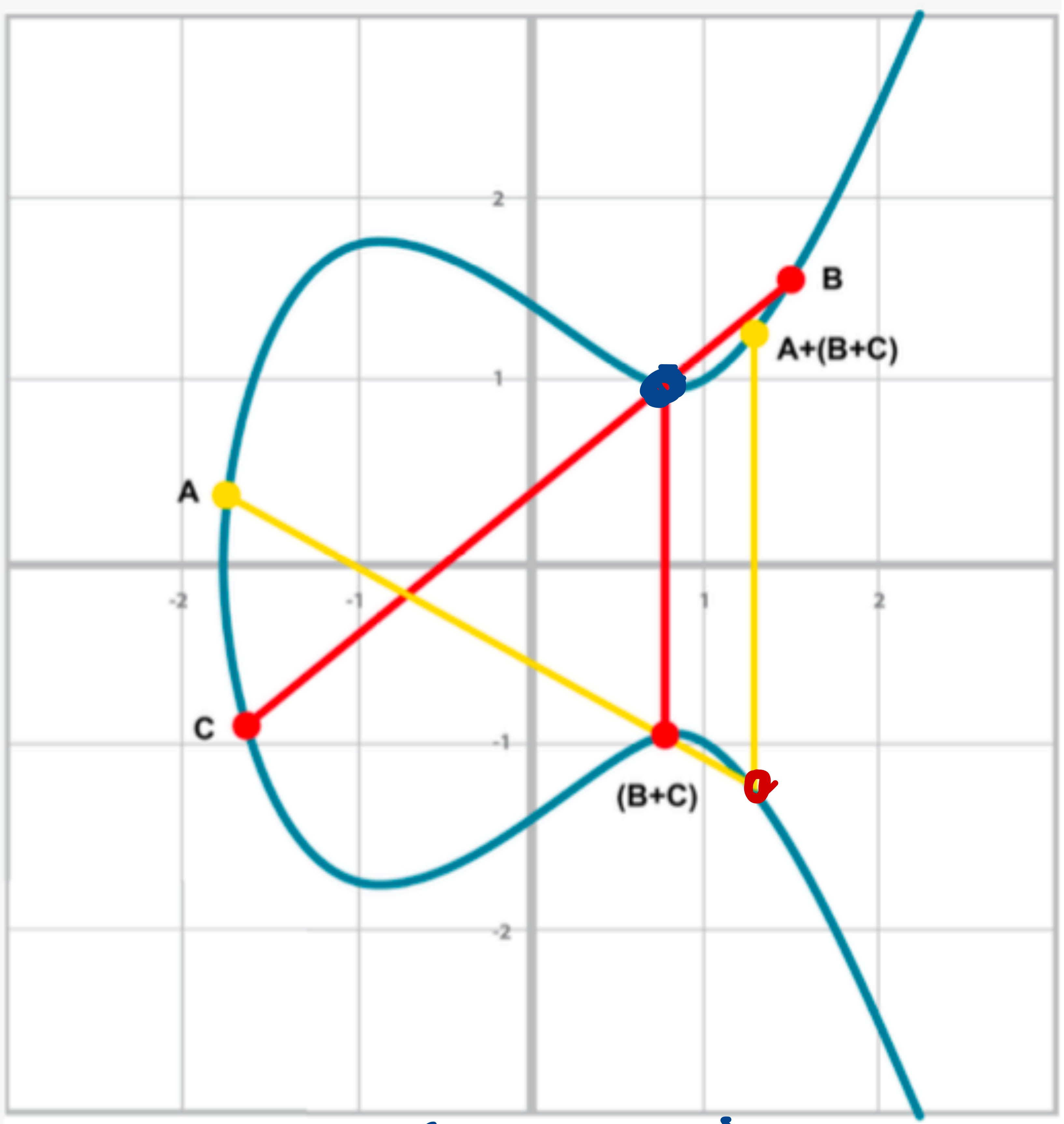
Q: Why do we need to "reflect" C at x-axis to define $P_1 + P_2$?

POINT ADDITION ON ELLIPTIC CURVES

We need: $(A+B)+C = A+(B+C)$



$(A+B)+C$



$A+(B+C)$

POINT ADDITION ON ELLIPTIC CURVES

If $A = (x_1, y_1) \in S_{a,b}$, $B = (x_2, y_2) \in S_{a,b}$ are two points on elliptic curve $S_{a,b} := \{(x,y) : y^2 = x^3 + ax + b\}$, the desirable properties of point addition:

(i) $(A+B)+C = A+(B+C)$ "associativity"

(ii) $A + B = B + A$ "commutativity"

(iii) $A + O = A$ for "point at infinity" O "additive identity"

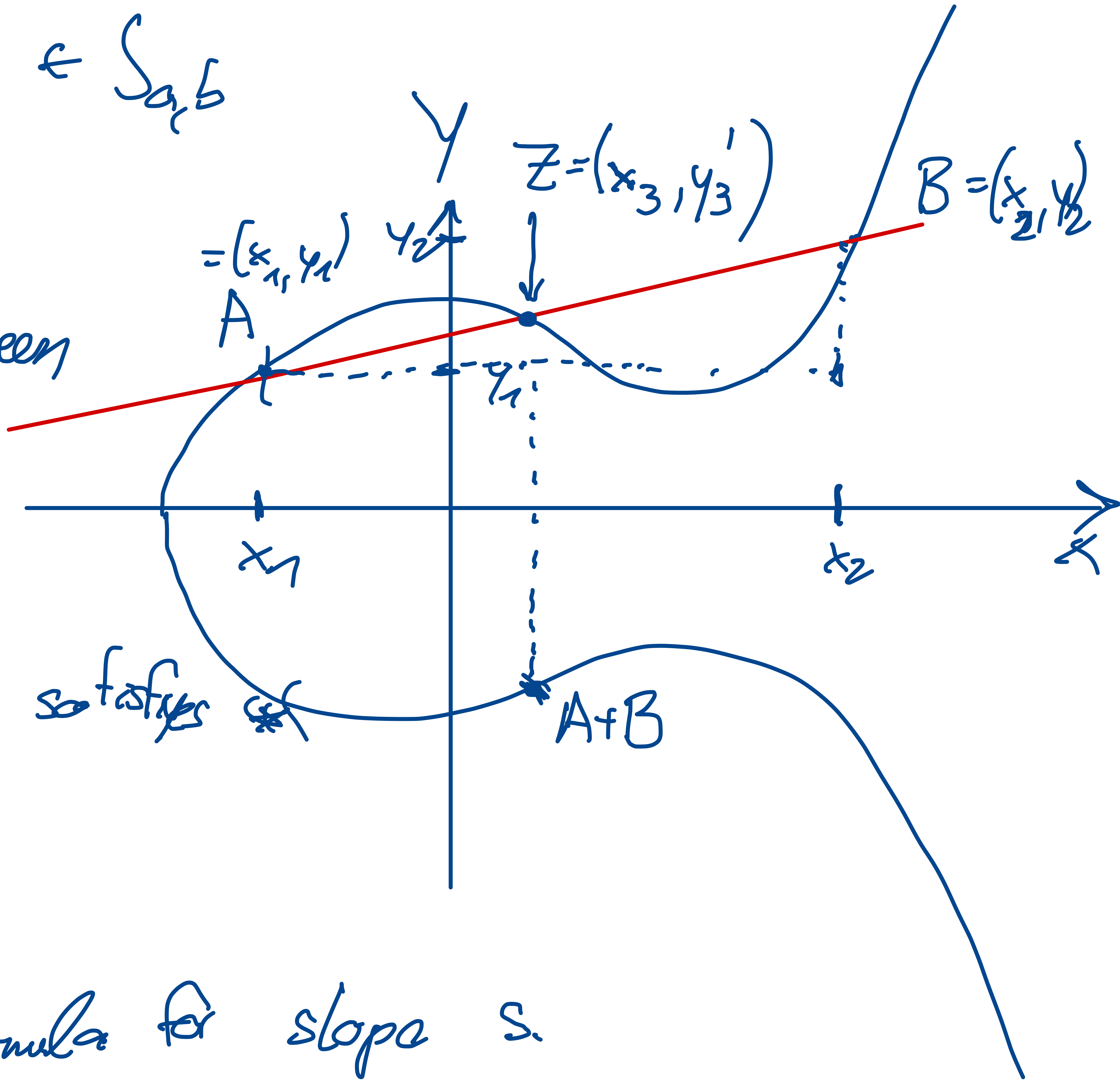
(iv) There exists an additive inverse $-A$ s.t. $A+(-A) = O$ for all $A \in S_{a,b}$.

$2A := A+A$

$3A = A+A+A$

...

$A = (x_1, y_1) \in S_{a,b}$, $B = (x_2, y_2) \in S_{a,b}$



Case 1: $x_1 \neq x_2$

Idea: Define line equation of line between A and B:

$$\boxed{y = s(x - x_1) + y_1} \quad (*)$$

- We observe: Inserting $x = x_1$ and $y = y_1$ satisfies $(*)$
- Insert $x = x_2$, $y = y_2$:

$$y_2 = s(x_2 - x_1) + y_1$$

$$\Rightarrow \quad s = \frac{y_2 - y_1}{x_2 - x_1} \quad \Rightarrow \text{Formula for slope } s.$$

POINT ADDITION ON ELLIPTIC CURVES

To find point Z, we need:

(I) $\boxed{y = s(x-x_1)+y_1}$ (I) Line equation with $s = \frac{y_2-y_1}{x_2-x_1}$

(II) $y^2 = x^3 + ax + b$ (elliptic curve equation)

$(a+b)^2 = a^2 + 2ab + b^2$

Squaring (I) and plugging into (II):

$[s(x-x_1)+y_1]^2 = x^3 + ax + b$

$\Leftrightarrow s^2(x-x_1)^2 + 2\cdot s(x-x_1)y_1 + y_1^2 = x^3 + ax + b$

$\Leftrightarrow x^3 - (s^2x^2 - 2s^2x_1x + s^2x_1^2) - 2sxy_1 + 2sx_1y_1 + ax + b - y_1^2 = 0$ (**)

POINT ADDITION ON ELLIPTIC CURVES

⇒ We can compute x_3 given

- x_1 (first coordinate of A)
- x_2 (first coordinate of B)
- s (defined as $s = \frac{y_2 - y_1}{x_2 - x_1}$).

y-coordinate y_3' of Z:

From line equation $y_3' = s(x_3 - x_1) + y_1$

We obtain $A + B = (x_3, y_3)$ by setting

$$y_3 = -y_3' = -s(x_3 - x_1) - y_1$$

Summary:

1) $s = \frac{y_2 - y_1}{x_2 - x_1}$
2) $x_3 = s^2 - x_1 - x_2$
3) $y_3 = s(x_1 - x_3) - y_1$

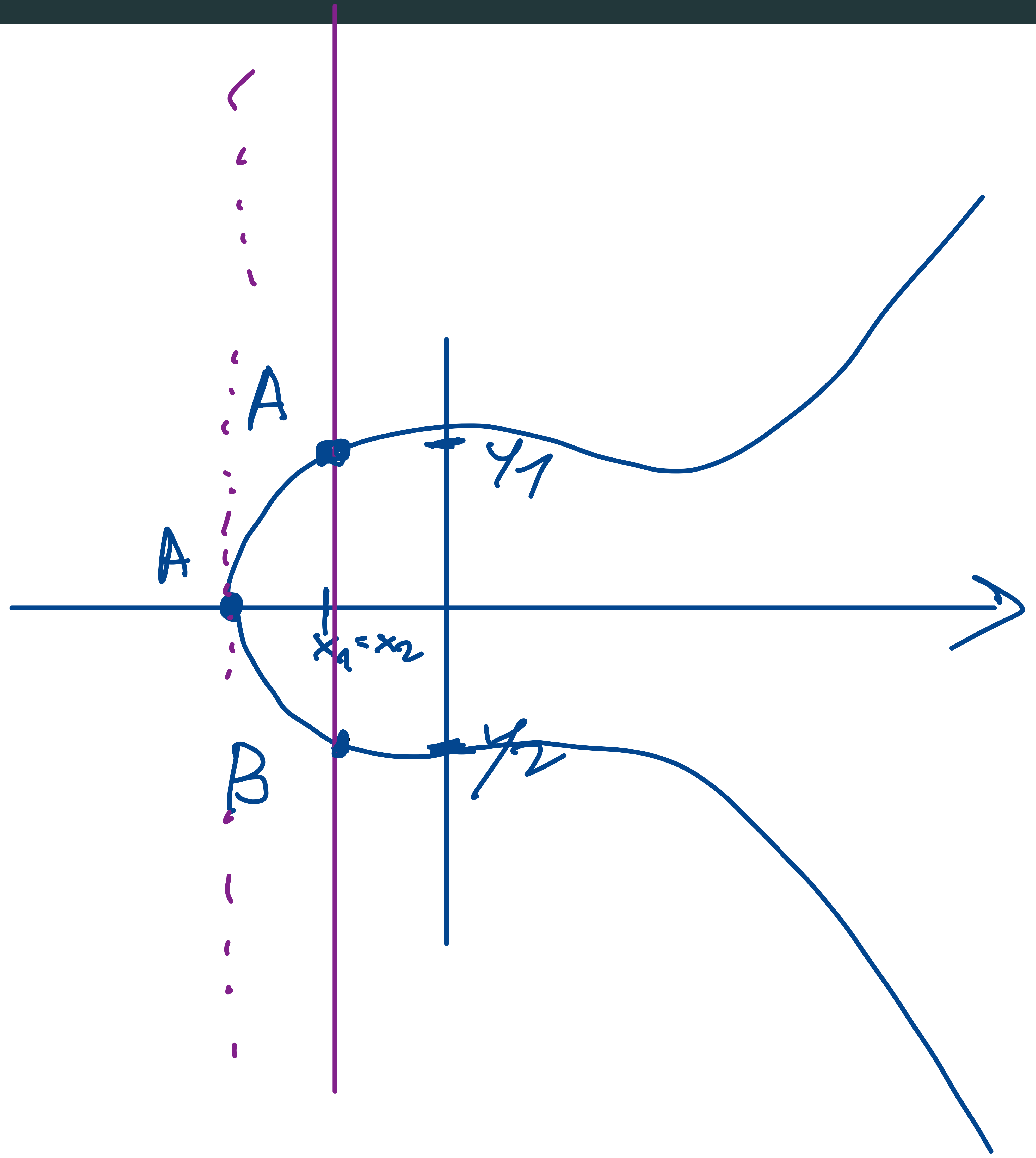
POINT ADDITION ON ELLIPTIC CURVES

$A = (x_1, y_1) \in S_{a,b}$, $B = (x_2, y_2) \in S_{a,b}$

Case 2: $x_1 = x_2$, and $y_2 = -y_1$

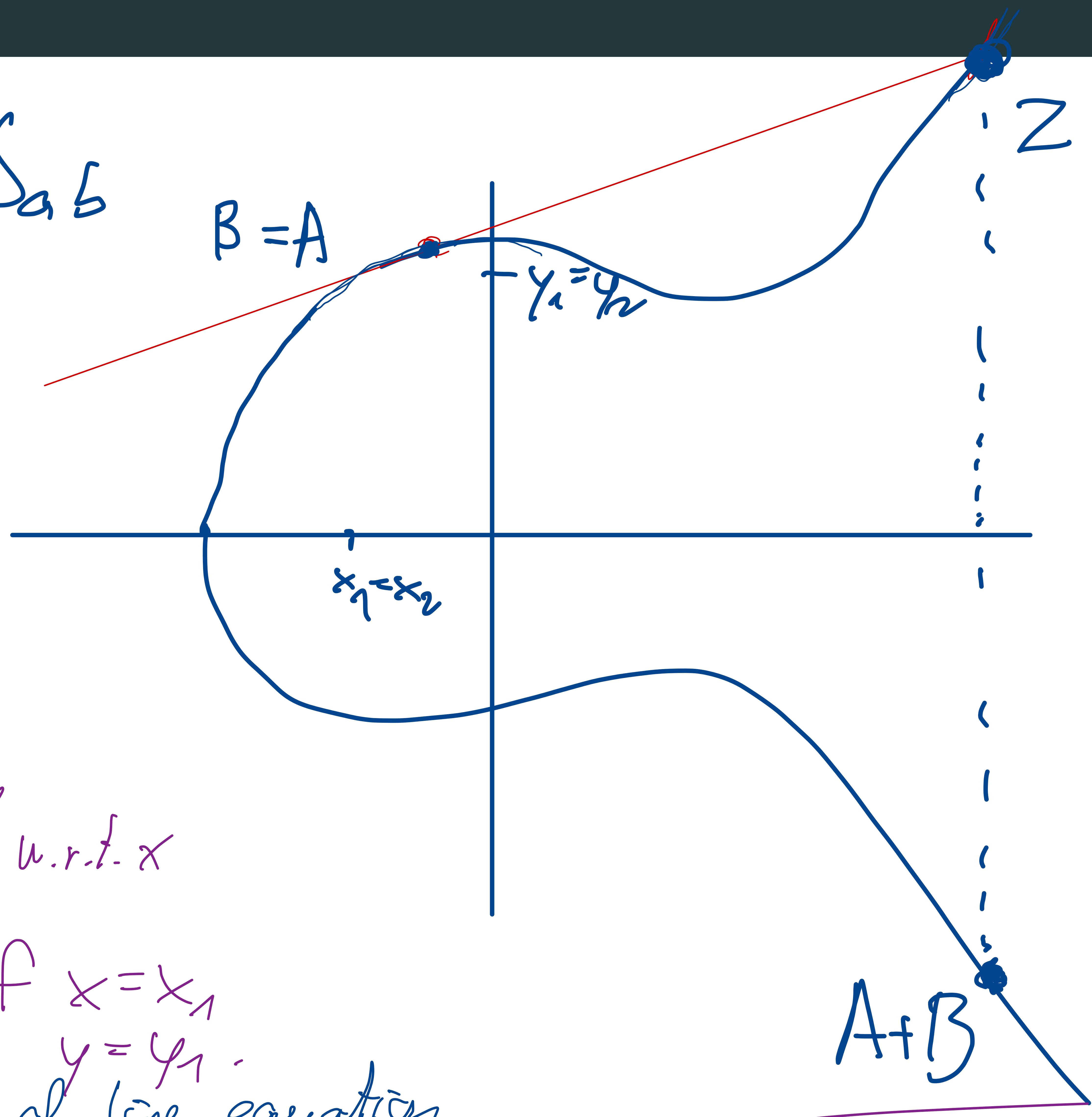
$\Rightarrow$ Case of straight line being vertical

$P := A + B = \mathcal{O}$, point at infinity



POINT ADDITION ON ELLIPTIC CURVES

$A = (x_1, y_1) \in S_{a,b}$, $B = (x_2, y_2) \in S_{a,b}$



Case 3: $x_1 = x_2,\ y_2 = y_1,\ y_1 \neq 0$

Idea: Consider $y^2(x) = x^3 + ax + b$

Take derivative w.r.t. x on both sides $\Rightarrow$ $2y \cdot y' = 3x^2 + a + 0$

derivative of $y = \sqrt{x^3 + ax + b}$ w.r.t. x

this is exactly slope s if $x = x_1$, $y = y_1$.

$\Rightarrow \quad 2y_1 \cdot s = 3x_1^2 + a$

$\Leftrightarrow \quad \boxed{s = \frac{3x_1^2 + a}{2y_1}}$

slope s of line equation

$y = s(x - x_1) + y_1$

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

POINT ADDITION ON ELLIPTIC CURVES

Summary of steps to compute:

As in case 1

1) $s = \dfrac{3x_1^2 + a}{2y_1}$

2) $x_3 = \left(s^2 - x_1 - x_2\right) \overset{x_2 = x_1}{=} s^2 - 2x_1$

3) $y_3 = s(x_1 - x_3) - y_1$

Case 3: $x_1 = x_2 \,, y_1 = y_2 \neq 0$

SCALAR MULTIPLICATION ON ELLIPTIC CURVES

We want to define "multiples" of elliptic curve points.

E.g. Given $A = (x_1, y_1) \in S_{a,b}$, compute $3 \cdot A \in S_{a,b}$

⚠ Different from defining $A \cdot B$, where $A, B \in S_{a,b}$.

pos.

Define : $k \cdot A = \underbrace{A + A + \dots + A}_{k \text{ times}}$ if k is integer

Observation: If we consider $S_{a,b}^{F_p} = \{(x,y) \in F_p \times F_p : y^2 = x^3 + ax + b\}$,

finite field of order p

then solving $\boxed{k \cdot A = B \text{ for } k}$ (given $A, B \in S_{a,b}^{F_p}$) is extremely hard if order p is very large.

Elliptic Curve-Based Digital Signature Schemes

We create now secure identity and signature schemes based on the finite field arithmetic of points $P = (x, y) \in S_{a,b}$ where $x, y \in \mathbb{F}_p$ for suitable a, b and very large p.

$$S_{a,b} = \{(x,y) \in \mathbb{F}_p \times \mathbb{F}_p : y^2 = x^3 + ax + b\}$$

We choose:

- Finite field $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$ with $\boxed{p = 2^{\overbrace{256}^{\approx 10^{77}}} - 2^{32} - 977}$ very large
- $a = 0$, $b = 7$ (thus, $S_{a,b} = S_{0,7} = \{(x,y) \in \mathbb{F}_p \times \mathbb{F}_p : y^2 = x^3 + 7\}$)
- Generator point $G = (G_x, G_y)$, with $G_x, G_y \in \mathbb{F}_p$ specific, well-chosen numbers
- $n := \underbrace{|\{f \cdot G : f \in \mathbb{N}\}|}$ this quantity is called (group) order of the (Abelian) group $\{f \cdot G : f \in \mathbb{N}\} \subset S_{a,b}$.

also very large but $n < p$

these numbers can be looked up

Notation:
If S is a set, $|S|$ counts how many distinct elements are in set S.

ELLIPTIC-CURVE BASED DIGITAL SIGNATURE SCHEMES

- Since $p < 2^{256}$, but $p \approx 2^{256}$ (same order of magnitude), it is convenient to represent $G_x, G_y \in \mathbb{F}_p$ as 256-bit integers.
- Note: $2^{256} > 10^{77}$

  Nr. of atoms in universe: $\approx 10^{80}$ (according to estimates)

  $\Rightarrow$ 2^{256} is HUGE!

DIGITAL SIGNATURE SCHEMES USED IN BITCOIN

- **Elliptic Curve Digital Signature Algorithm (ECDSA)**
  - Concept proposed by Neal Koblitz and Victor S. Miller in 1985
  - Standardized in 2000 by NIST
  - Used in Bitcoin since 2009, was freely available
  - Used by all address formats before Taproot upgrade

- **Schnorr Signatures:**
  - Proposed and **patented** by Claus-Peter Schnorr in 1990
  - Has certain advantages over ECDSA (will see later) and simpler
  - Patent expired in 2010, so not available at inception of Bitcoin
  - Implemented in address format introduced by 2021 Taproot upgrade

ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

ECDSAsign(e, k, m)

▷ $R = k \cdot G$ [$\in S_{0,7}$]

▷ Define r as x-coordinate of $R = (r, R_y)$

▷ Compute $z = \text{hash}(m)$

▷ Compute $s = (z + r \cdot e)/k$

return (r, s)

use SHA256(SHA256(·))

makes sure that z is 256-bit integer

[this is done in F_n]

where n is group order of generator group

ECDSAverify(P, m, r, s)

▷ Compute $z = \text{hash}(m)$

▷ Compute $u = z/s$ [in F_n]

▷ Compute $v = r/s$ [in F_n]

▷ testval $= u \cdot G + v \cdot P$

▷ If (x-coordinate of testval) $== r$
return True

Else
return False

Glossary:

▷ $e \in F_p$: private key

▷ $G \in S_{0,7}$: Generator point

▷ $P \in S_{0,7}$: Public key (satisfies $P = e \cdot G$)

▷ $k \in F_p$: random (private) nonce

▷ $r \in F_p$: public nonce (derived from private nonce)

▷ m : message to be signed

▷ s : signature

ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

Intuition: The formulas of ECDSAsign and ECDSAverify are derived from following 3 equations:

$$u \cdot G + v \cdot P = R \qquad \text{(I)} \quad \text{(from ECDSA verify)}$$

$$k \cdot G = R \qquad \text{(II)} \quad \text{(from -''- sign)}$$

$$e\,G = P \qquad \text{(III)} \quad \text{(equation relating public \& private key)}$$

(III) in (I) $\Rightarrow$

$$u \cdot G + v\,e \cdot G = k \cdot G$$

This equation is satisfied if scalars in front of generator point G are matching, i.e., if:

$$u + ve = k$$

in finite field $\mathbb{F}_n$

inserting (*) for k $\Leftarrow$

$$\underbrace{u + ve = \frac{z}{s} + \frac{r}{s} \cdot e}$$

this equality holds if $u = \frac{z}{s}$ and $v = r/s$ in $\mathbb{F}_n$

from step 4 of ECDSAsign:

$$s = (z + re)/k$$

$$\Leftrightarrow k \underset{(*)}{=} \frac{z + re}{s}$$

Quiz Time

Which TWO statements about cryptographic hash function used in the Bitcoin protocol are **WRONG**?

a) The input has to be of fixed length.

b) The output has to be of fixed length.

c) They are used to construct hash pointers.

d) In bitcoin mining, computing one hash function output requires specialized mining hardware (ASICs).

SOME QUIZ QUESTIONS

Which TWO statements about cryptographic hash function used in the Bitcoin protocol are **WRONG**?

a) **The input has to be of fixed length.**

b) The output has to be of fixed length.

c) They are used to construct hash pointers.

d) **In bitcoin mining, computing one hash function output requires specialized mining hardware (ASICs).**

Under a so-called ***gold standard***, what is the primary mechanism that maintains the value of a currency?

a) Governments set the exchange rate based on international trade balances.

b) The currency value is tied to a specific quantity of gold.

c) The value is determined solely by market forces between competing private banks, which may or may not back the currency by gold.

d) The currency consists only of gold or silver coins, and the value is guaranteed through the scarcity of these.

SOME QUIZ QUESTIONS

Under a so-called ***gold standard***, what is the primary mechanism that maintains the value of a currency?

a) Governments set the exchange rate based on international trade balances.

b) The currency value is tied to a specific quantity of gold.

c) The value is determined solely by market forces between competing private banks, which may or may not back the currency by gold.

d) The currency consists only of gold or silver coins, and the value is guaranteed through the scarcity of these.

SOME QUIZ QUESTIONS

What is the definition of a **collision-free** cryptographic hash function $h : D \to R$?

What is the definition of a **collision-free** cryptographic hash function $h : D \to R$?

It is computationally infeasible to find two different inputs $x, y \in D$, $x \neq y$ with same output $H(x) = H(y)$.