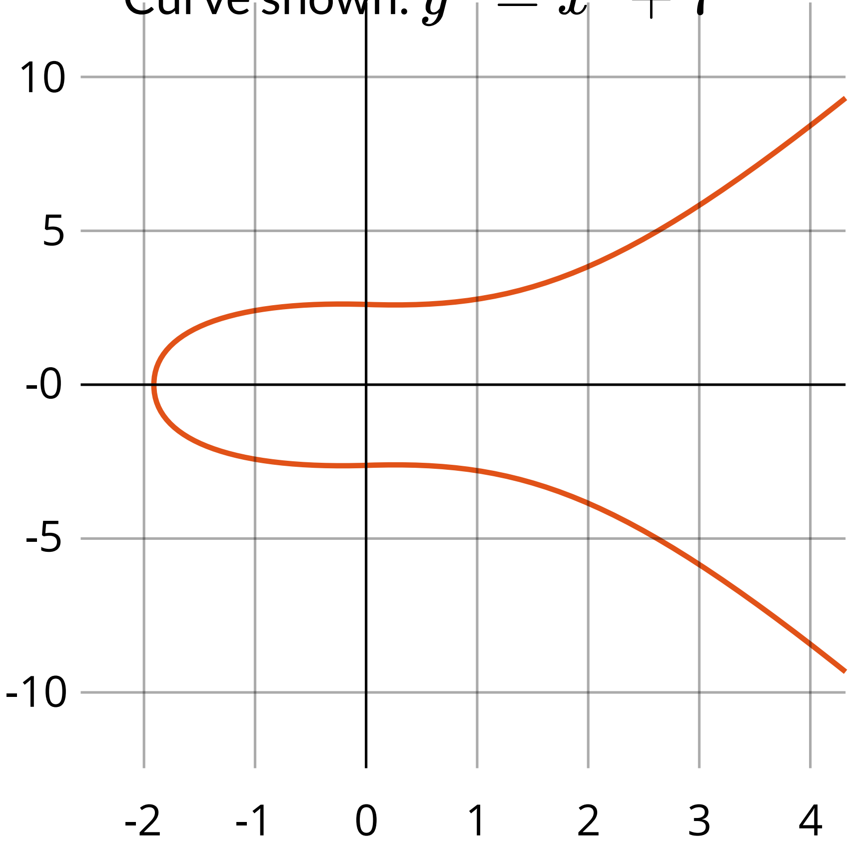
Ellipic Curves over Real Numbers

Curve shown: $y^2 = x^3 + 7$



Elliptic curve general form:

$$y^2 = ax^3 + bx + c$$

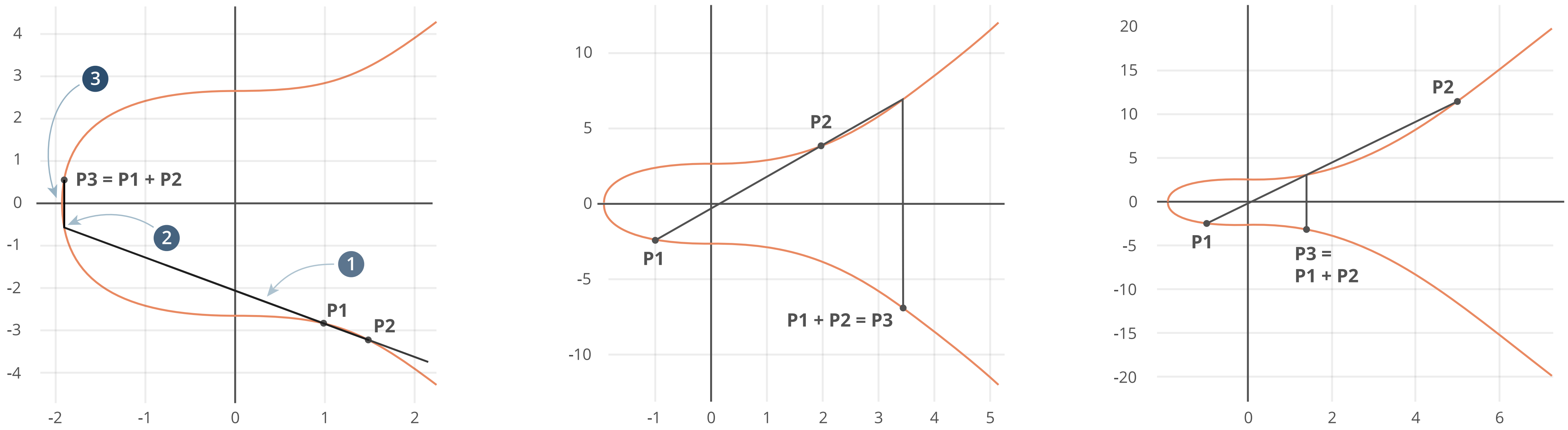
The secp256k1 curve form:

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

Elliptic curve points:

`EC-Point P(x,y)` on the elliptic curve fulfills the curve equation.

EC-Point Addition over Real Numbers



1) Form a line with P1 & P2

2) Intersect resulting line with EC

3) Reflect intersection point across X-axis for P3

EC-Point Addition (Computation)

For an elliptic curve of form:

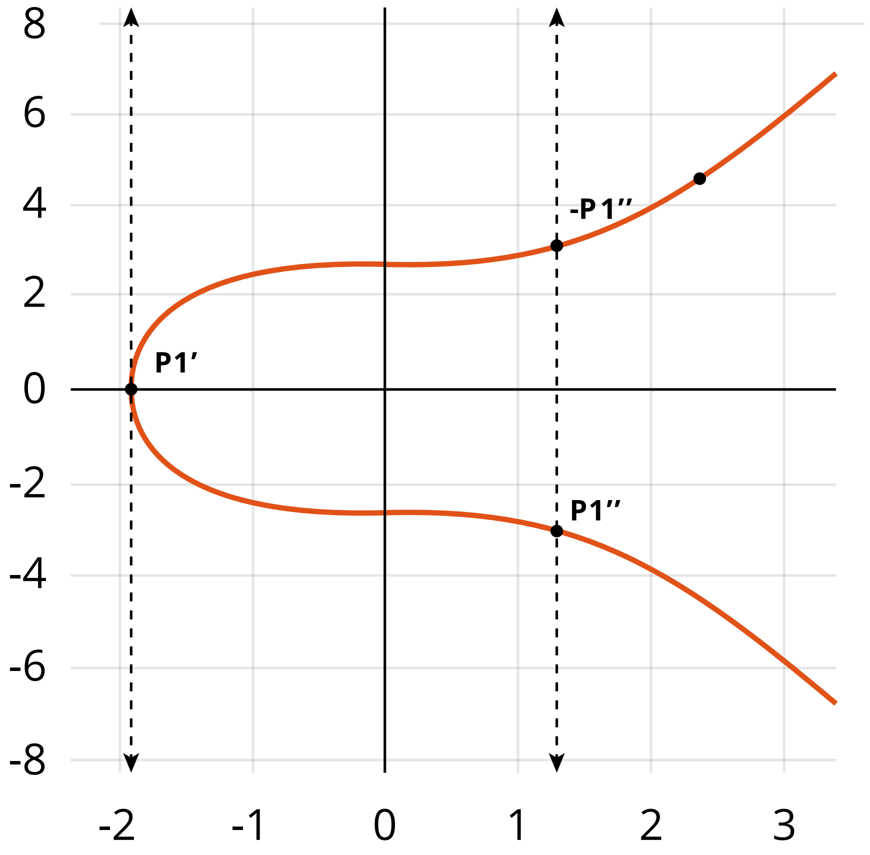
- $y^2 = ax^3 + bx + c$

Computation of $\boldsymbol{P_3 = P_1 + P_2}$

- $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2)$
  - $s = \frac{y_2 - y_1}{x_2 - x_1}$ for $x_1 \neq x_2$
  - $x_3 = s^2 - x_1 - x_2$
  - $y_3 = s(x_1 - x_3) - y_1$

The equations shown describe EC-point addition where $x_1 \neq x_2$.

EC Point Addition with **Infinity**



The Point at Infinity:

The (Inf/Inf) point is defined as a point which is infinitely far away in the direction of the y-axis.

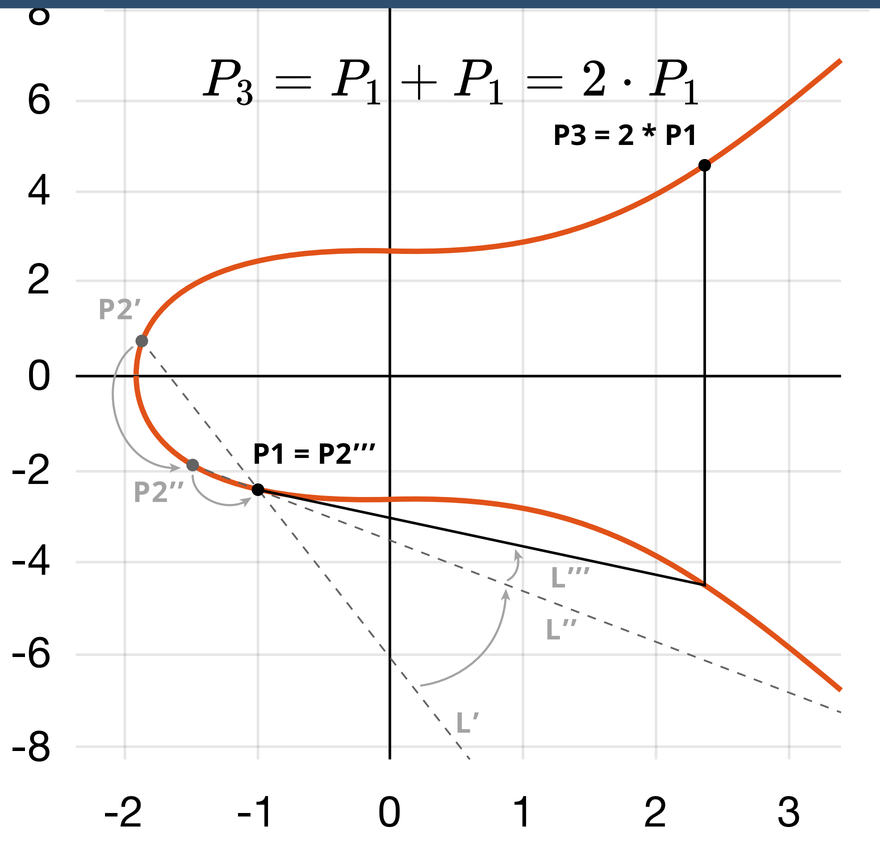
Therefore, we can add a point P1 to the infinity point simply by connecting a vertical line through P1.

$$P_1(x_1, y_1) + (Inf/Inf) = P_1(x_1, y_1)$$

$$P_1(x_1, y_1) + P_2(x_1, -y_1) = (Inf/Inf)$$

The infinity point is the group identity element

Scalar x EC Point



Scalar multiplication of an EC point P

- `s · P` equals adding `P` to itself `s` times.

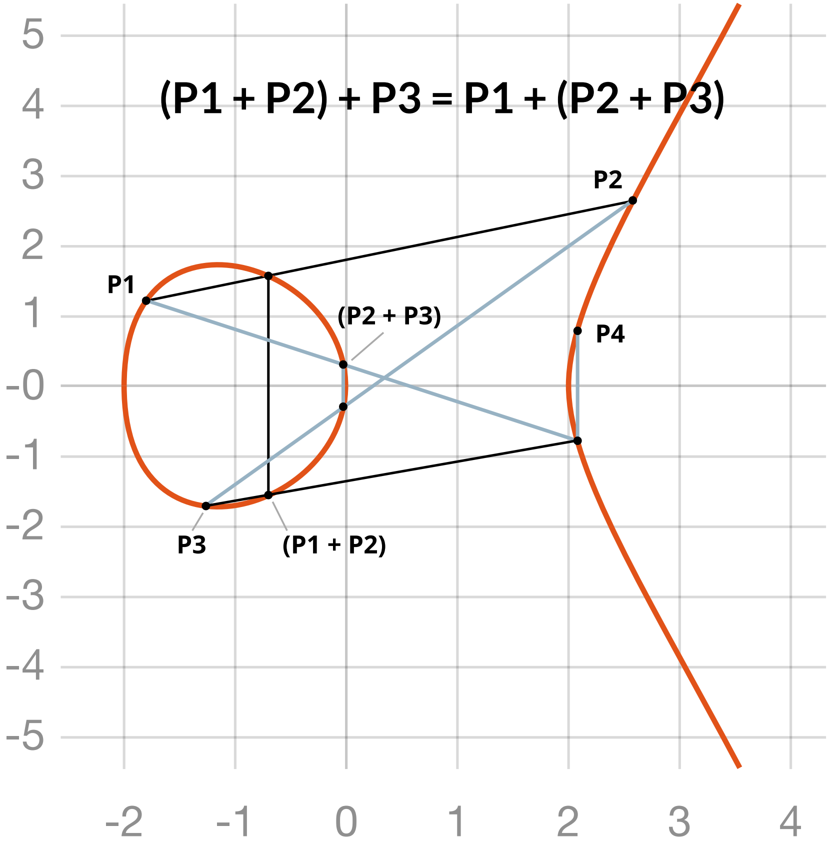
Point(x,y) + Point(x,y)

- Consider Line L'(P1,P2).
- Converges to tangent of P1, as P1 = P2.
  - One intuitive reason for restrictions on a, b to avoid singular points

Distributivity:

- `(a + b)·C = a·C + b·C`
- (Can be proved algebraically)

Commutativity/Associativity of Point Addition



EC point addition commutivity

- P1 + P2 = P2 + P1
- (Same intersection point)

EC point addition associativity

- Order of addition doesn't change result.
- Associativity proof is rather involved.
  - `(P1 + P2) + P3 = P1 + (P2 + P3)`
  - (Associativity can be observed in example)
- Rather fragile property, reflection necessary!

Why reflect? Associativity

Without flipping:

- $A + B = C$

Implies (why?):

- $B + C = A$
- $C + A = B$
- $=> A = 0$:(

Stirring the pot

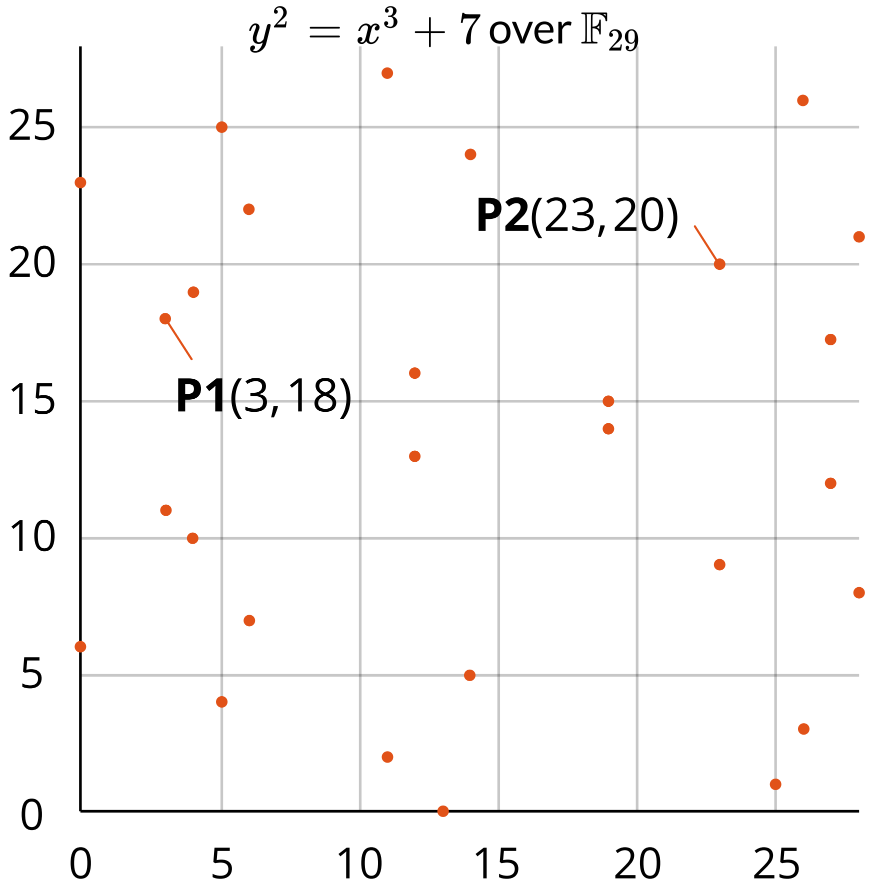
So far we established the group structure of elliptic curves

- Point addition
- Identity element
- etc.

ECs are defined over an underlying field

- $y^2 = ax^3 + bx + c$
- So far it was always $\mathbb{Q}$, but any field does the trick
  - The magic of abstract algebra
- $\Rightarrow$Final step: ECs over finite fields

Elliptic Curves over $\mathbb{F}_p$



Point on elliptic curve over $\mathbb{F}_p$

- Point coordinates fulfill following equation:
- $y^2 = ax^3 + bx^2 + c \ (mod\ p)$

Example EC points:

- $y^2 = x^3 + 7$ over $\mathbb{F}_{29}$
- $P_1\ (3, 18)$:
  - $18^2 = 3^3 + 7\ (mod\ 29) = 5$ ✓
- $P_2\ (23, 20)$:
  - $20^2 = 23^3 + 7\ (mod\ 29) = 23$ ✓
- Note: Elliptic curve plots are a set of non-continuous points since finite fields themselves are non-continuous.

EC-Point Addition over $\mathbb{F}_p$

EC-Point Addition over Reals $\mathbb{R}$:

Addition where x1 != x2:

$s = \frac{y_2 - y_1}{x_2 - x_1}$

$x_3 = s^2 - x_1 - x_2$

$y_3 = s\,(x_1 - x_3) - y_1$

Addition where P1 = P2:

$s = \frac{(3x_1^2 + a)}{2y_1}$

$x_3 = s^2 - 2x_1$

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

Addition with infinity:

$(x_1, y_1) + (\infty, \infty) = (x_1, -y_1)$

EC-Point Addition over Prime $\mathbb{F}_p$:

← EC-point addition equations over $\mathbb{R}$ apply to EC point addition over prime $\mathbb{F}_p$.

Example:

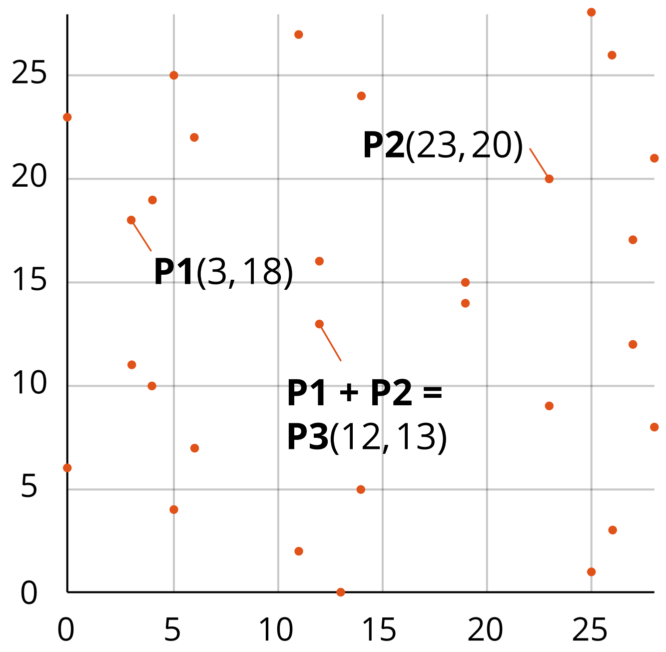
$$P_3\,(x_3, y_3) = P_1\,(3, 18) + P_2\,(23, 20)$$

$$s = \frac{20-18}{23-3} = 2 \cdot 20^{29-2}\ (mod\ 29) = 3$$

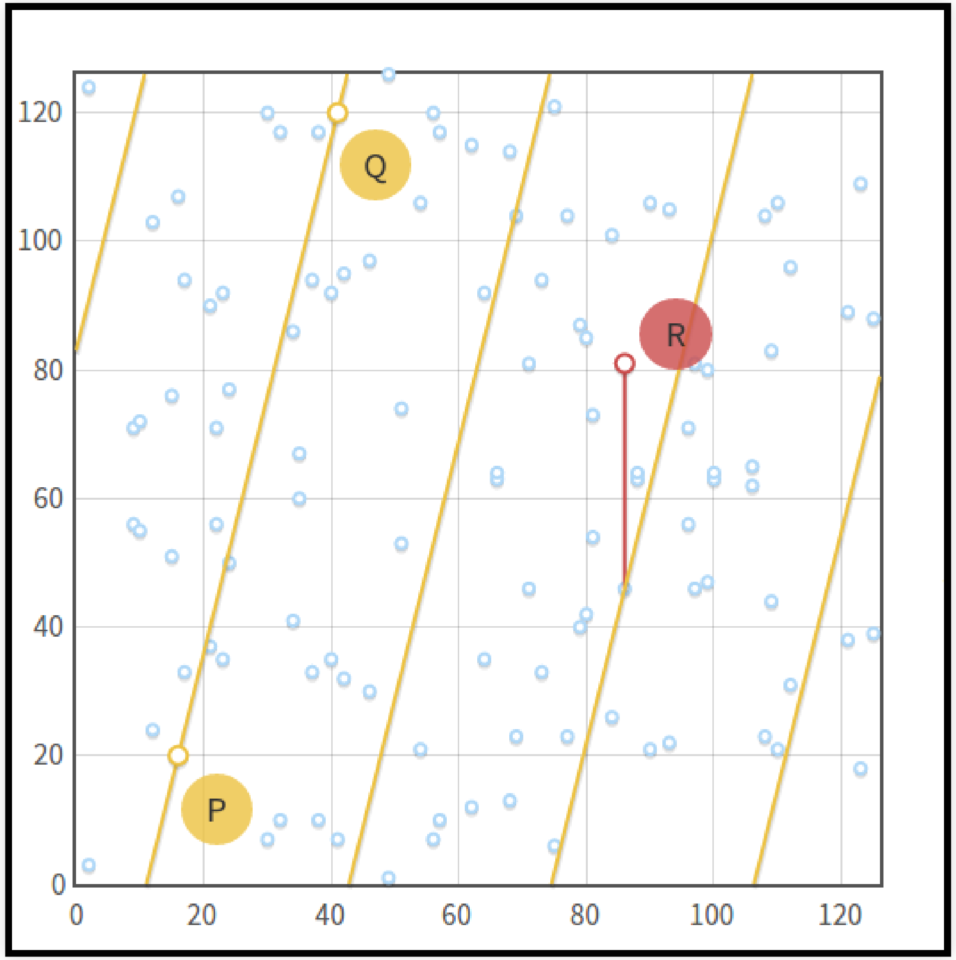
$$x_3 = 3^2 - 23 - 3\ (mod\ 29) = 12$$

$$y_3 = 3 \cdot (3 - 12) - 18\ (mod\ 29) = 13$$

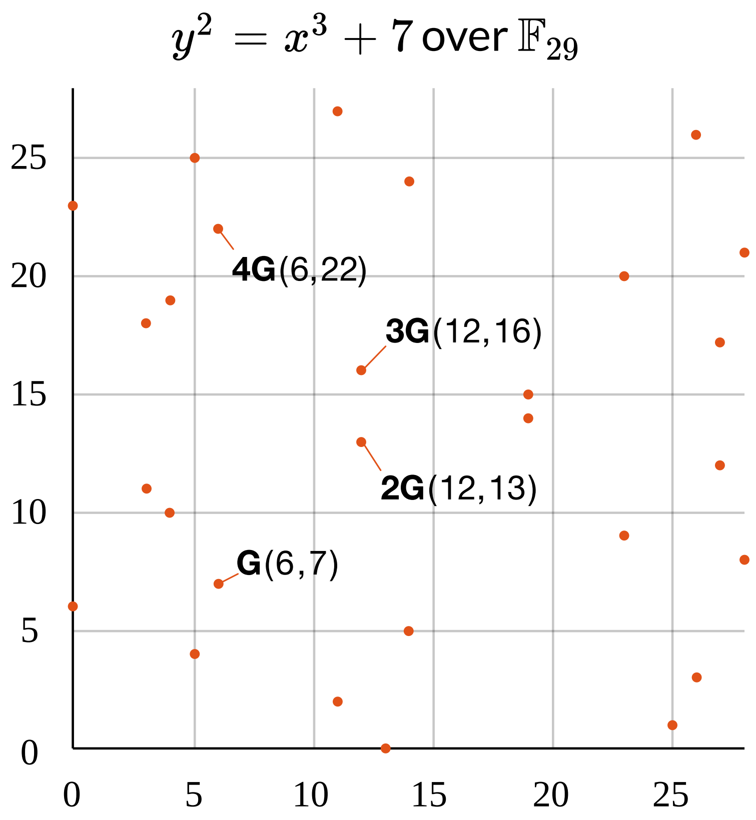
$y^2 = x^3 + 7$ over $\mathbb{F}_{29}$



Geometric intuition for EC-Point Addition over $\mathbb{F}_p$



Elliptic Curve Point Groups (I)



Symmetry: Each point `P(x,y)` has an inverse `P(x,-y)`

- $P(x, y) + P(x, -y) = (Inf/Inf)$
- Same relationship as over $\mathbb{R}$
- Blackboard: Prove that both points are on the curve

Points form a cyclic point group

- $1 \circ G$
- $2 \circ G$
- $\ldots$
- $|\mathbb{G}| \circ G = (Inf, Inf)$
- (Proof of cyclic behaviour omitted)

Elliptic Curve Point Groups (II)

1) Closed group operation (Point addition)

- EC point addition over $\mathbb{F}_p$
- EC point addition is Associative & Commutative

2) Generator group element (G)

- $G + G + G \dots G + G + G = s \circ G = P \in \mathbb{G}$
- Finite number $|\mathbb{G}|$ of elements in cyclical group $\mathbb{G}$

3) Neutral group element (inf/inf)

- $P + (inf, inf) = P$

4) Group element inverse

- $P(x, y) + P(x, -y) = (inf, inf)$

Disrete Log over Elliptic Curves

Discrete Log Problem:

$P = G + G + G \ldots G + G + G = k \circ G$

- Given P, solve for k
- Number of Points $|\mathbb{G}|$ in Group: $\approx p$ (Schoof's Algorithm)
- EC multiplication is more like a black-box-operation than modulo-exponentiation over $\mathbb{F}_p$
- **Only general discrete log solutions are known for Elliptic Curves** $\mathcal{O}(\sqrt{|\mathbb{G}|})$
  - 160bit (group order) / 80bit (security)
  - 256bit (group order) / 128bit (security)
- **In Comparison: Index-calculus(DH, DSA, Elgamal), Factorization(RSA)**
  - 1040bit (group order) / 80bit (security)
  - 3072bit (group order) / 128bit (security)
- **This is a conjecture! ($\mathcal{P} \stackrel{?}{=} \mathcal{NP}$)**

Double and Add

Computing $P = k \circ G$ from known k must be efficient

$\mathbf{26 \circ G = 11010 \circ G}$

- Bitscan from left to right.
- Value of Bit0 is 1: G
- Value of Bit1 is 1: $2 \circ G + G = 3 \circ G$
- Value of Bit2 is 0: $2 \circ 3G = 6 \circ G$
- Value of Bit3 is 1: $2 \circ 6G + G = 13 \circ G$
- Value of Bit4 is 0: $2 \circ 13G = 26 \circ G$

25 Group operations reduced to 6

$\mathcal{O}(\log n)$ Complexity

Bitcoin Private & Public Keys

The secp256k1 EC point group:

$y^2 = x^3 + 7$ over $\mathbb{F}_p$ where $p = 2^{256} - 2^{32} - 977$ (How big is 2^{256}?)

$G = ($ `79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798,`
`483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8` $)$

$|G| =$ `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141`

$p =$ `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F`

Bitcoin private & public keys:

$$\boldsymbol{P = k \circ G}$$

(Private key scalar k is chosen, secp256k1-point P is the public key)

The secp256k1 EC-Point Group is used to generate Bitcoin private/public keys.

Why secp256k1

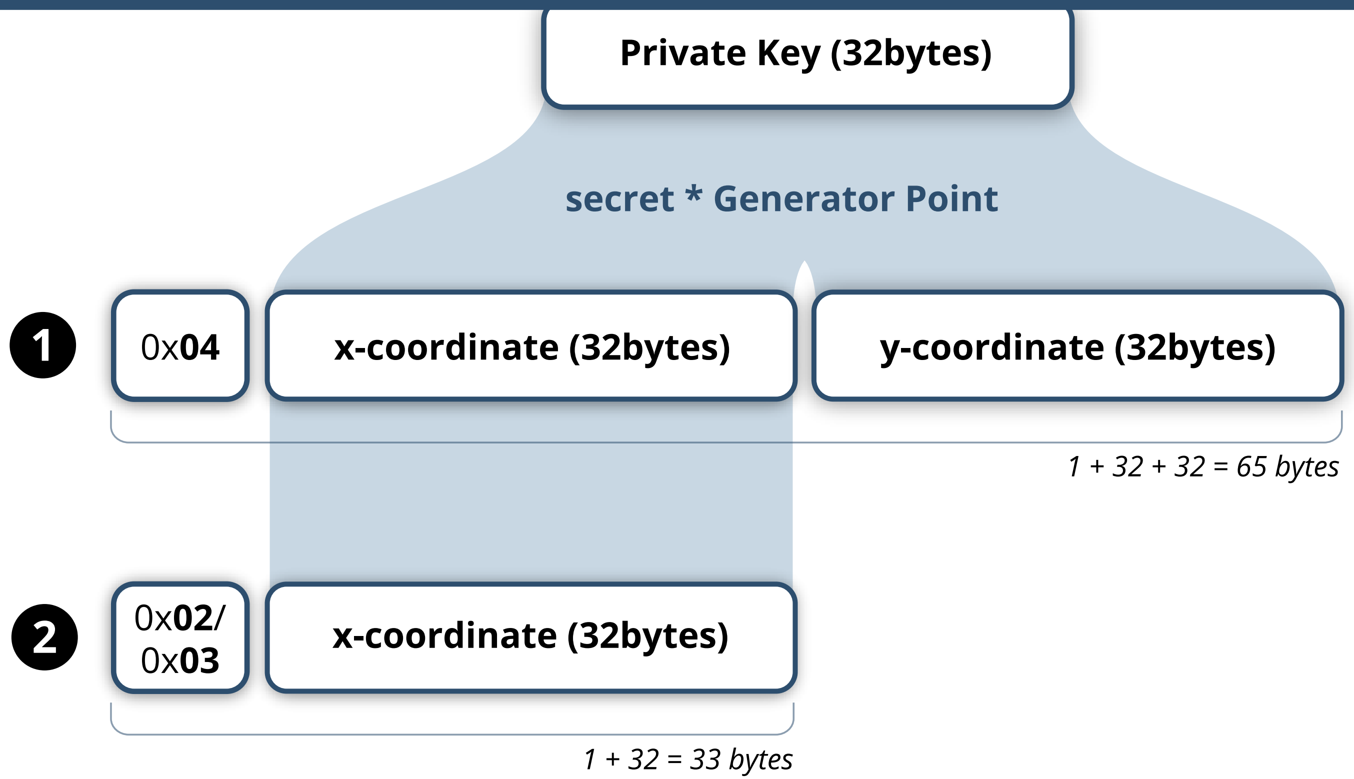
secp256k1 has some special properties that speed up some operations

- $a = 0$

Prime order also chosen in a somewhat predictable manner

However, secp256k1 was virtually unused before Bitcoin

Bitcoin Public Key Point **Serialisation**




```
0428026f91e1c97db3f6453262484ef5f69f71d89474f10926aae24d3c3eeb5f00 → x
  c41b6810b8b305a05de2b4448d7e2a079771d4c018b923a9ab860e4b0b4f86f6 → y

0228026f91e1c97db3f6453262484ef5f69f71d89474f10926aae24d3c3eeb5f00 → x
```

❶ Uncompressed Public Key Point

The uncompressed public key point directly represents both x and y-coordinates.

❷ Compressed Public Key Point

The compressed public key point implies its y-coordinate from its x coordinate.

A compressed public key begins with 0x02/0x03 to imply an even/odd y-coordinate.

Necessarily even/odd, because scalar field order is prime (odd). For given x, both +/-y values valid, which are complements in the odd field order.

Given x, +/-y can be found with the Tonelli–Shanks algorithm