UnitelmaSapienza
Università degli Studi di Roma

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Elliptic Curve Cryptography

(for mobile)

F. Parisi Presicce

Digital Signature

- majority of public-key crypto use integer/polynomial arithmetic with very large numbers/polynomials, imposing a significant load in storing and processing keys and messages
- an alternative is to use **elliptic curves,** which offer same security with smaller key sizes
- an elliptic curve is defined by an equation in two variables x and y, with real coefficients, as the cubic elliptic curve

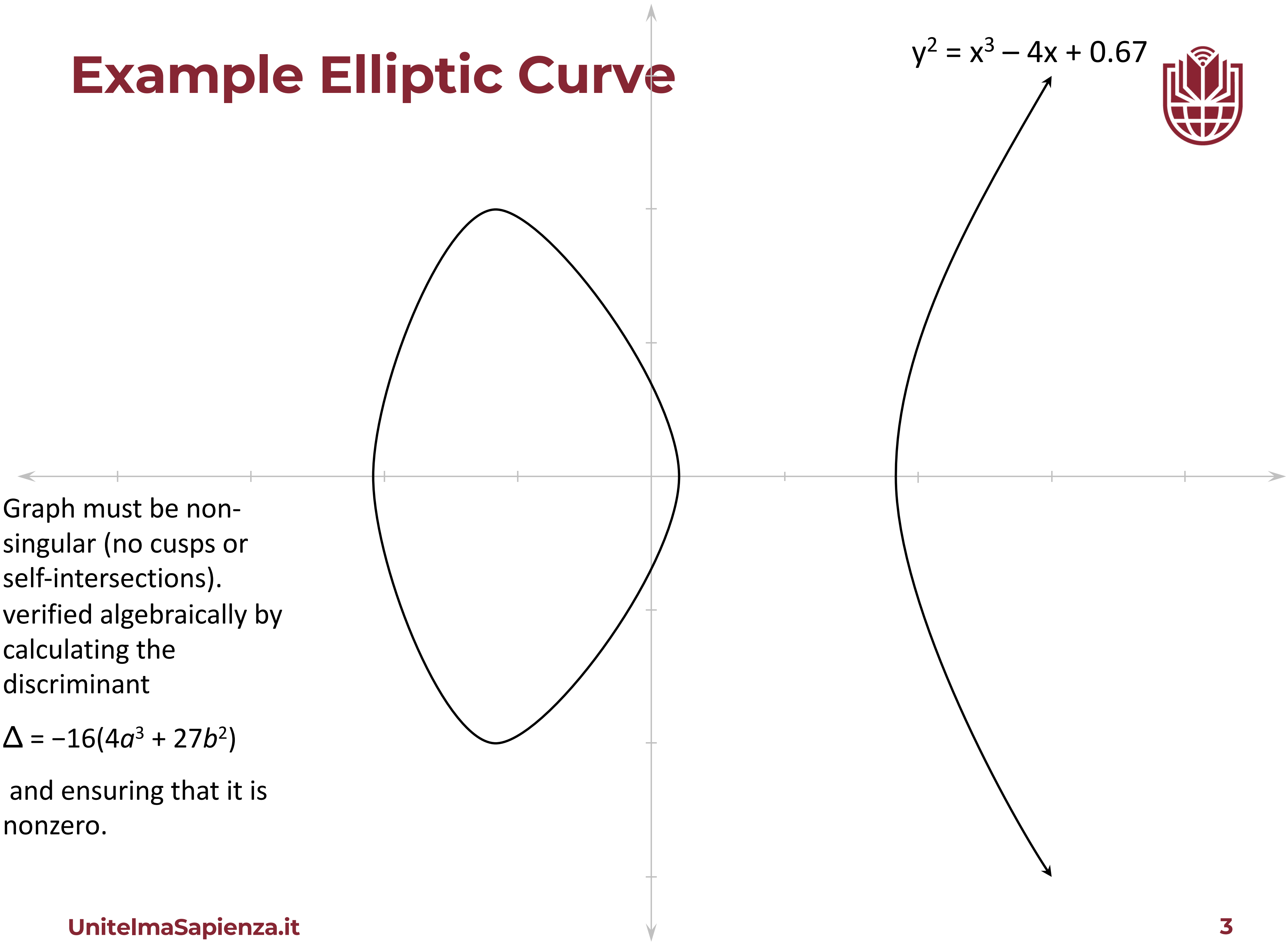
  $y^2 = x^3 + ax + b$

  - where x, y, a, b are all real numbers
  - also define a “zero point” O

More info

http://www.ruhr-uni-bochum.de/itsc/tanja/summerschool/slides.html

Example Elliptic Curve



Graph must be non-singular (no cusps or self-intersections). verified algebraically by calculating the discriminant

$\Delta = -16(4a^3 + 27b^2)$

and ensuring that it is nonzero.

Addition algebraically

Adding distinct points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ not negative of each other,

$P + Q = R$

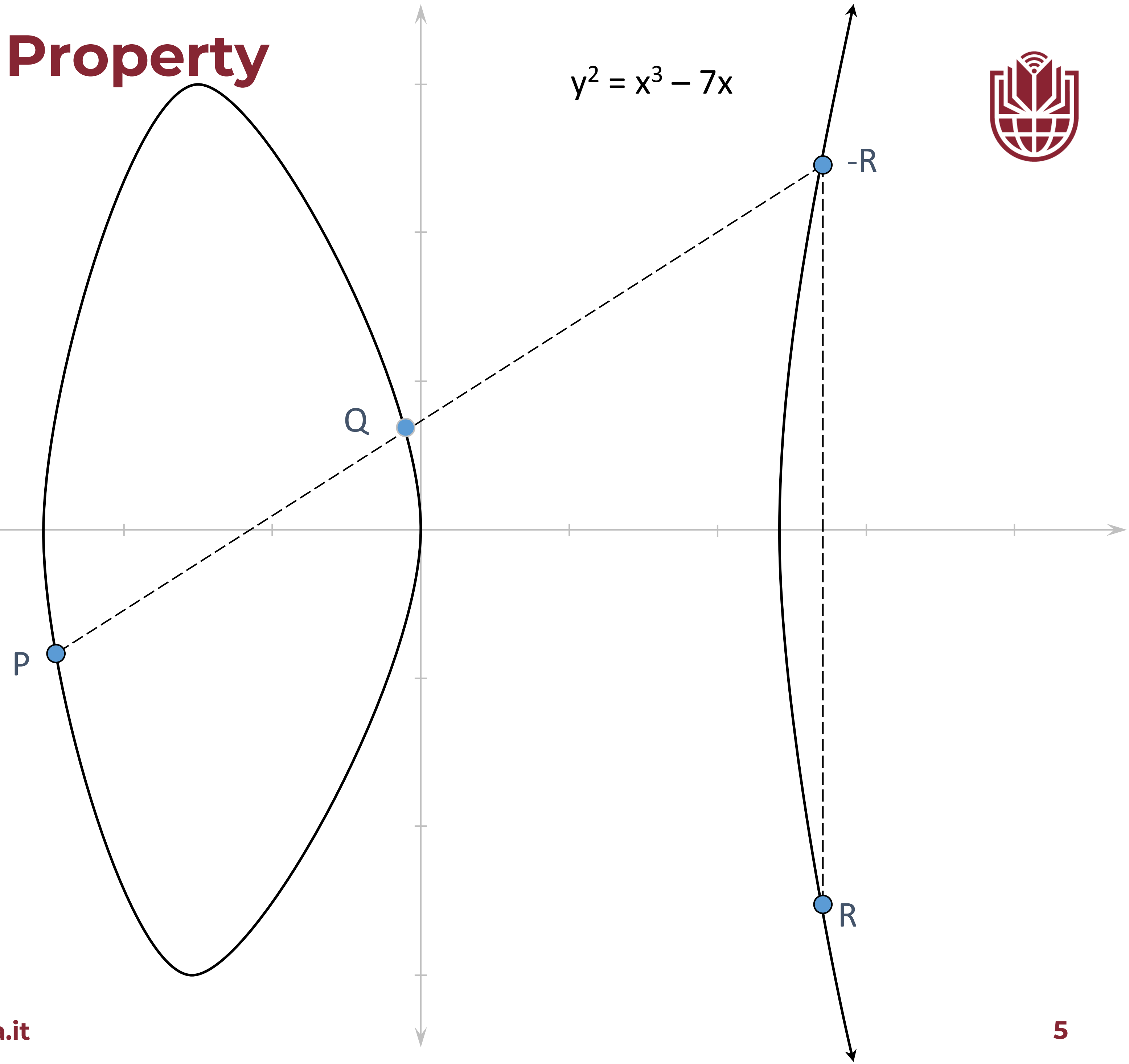
where

$s = (y_P - y_Q) / (x_P - x_Q)$ slope of line through P and Q

$x_R = s^2 - x_P - x_Q$

$y_R = -y_P + s(x_P - x_R)$

Additive Property



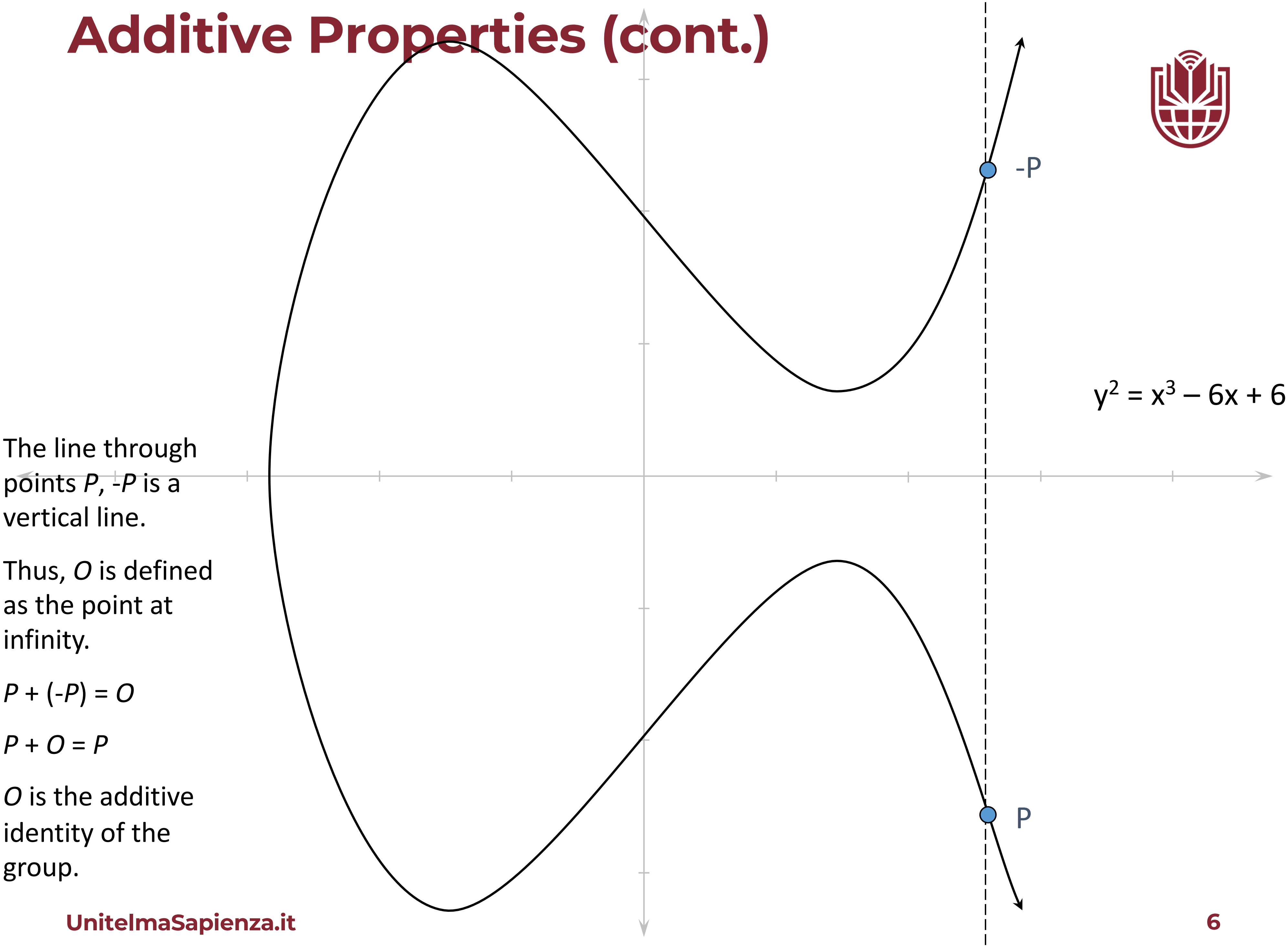
The group operation of addition is defined here geometrically.

Here is the addition of distinct points *P*, *Q* and *Q* ≠-*P*.

P + *Q* = *R*

For distinct points *P*, *Q* where *P* ≠ -*P*, the line through *P* and *Q* will intersect the curve at exactly one other point (-*R*).

Additive Properties (cont.)



The line through points *P*, *-P* is a vertical line.

Thus, *O* is defined as the point at infinity.

P + (-*P*) = *O*

P + *O* = *P*

O is the additive identity of the group.

Doubling Algebraically

Doubling the point P (when y_P is not 0),
2P = R

where

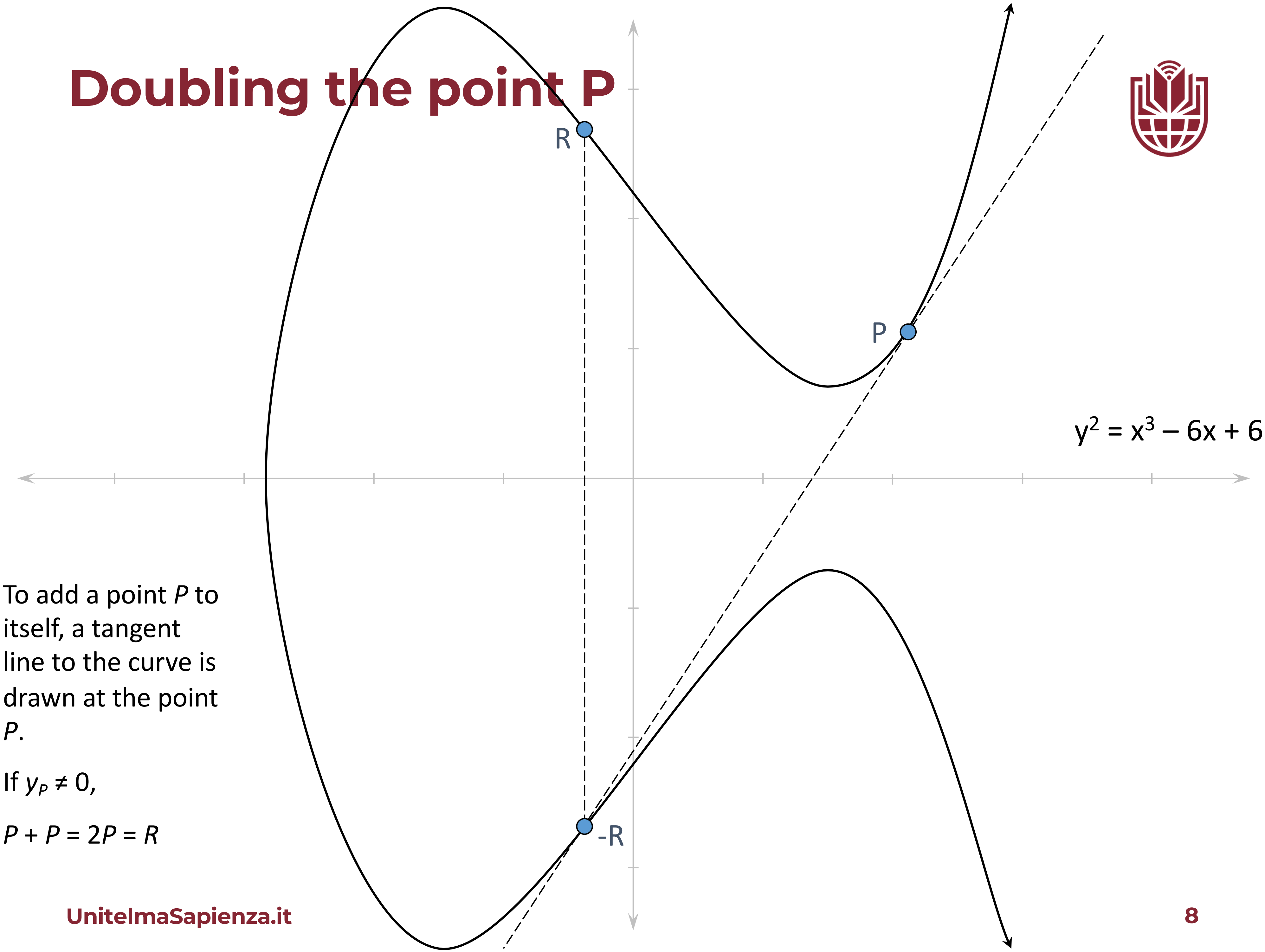
$$s = (3x_P^2 + a) / (2y_P)$$

$$x_R = s^2 - 2x_P$$

$$y_R = -y_P + s(x_P - x_R)$$

Recall that **a** is one of the parameters chosen with the elliptic curve and that **s** is the slope of tangent to the curve at the point P.

Doubling the point P

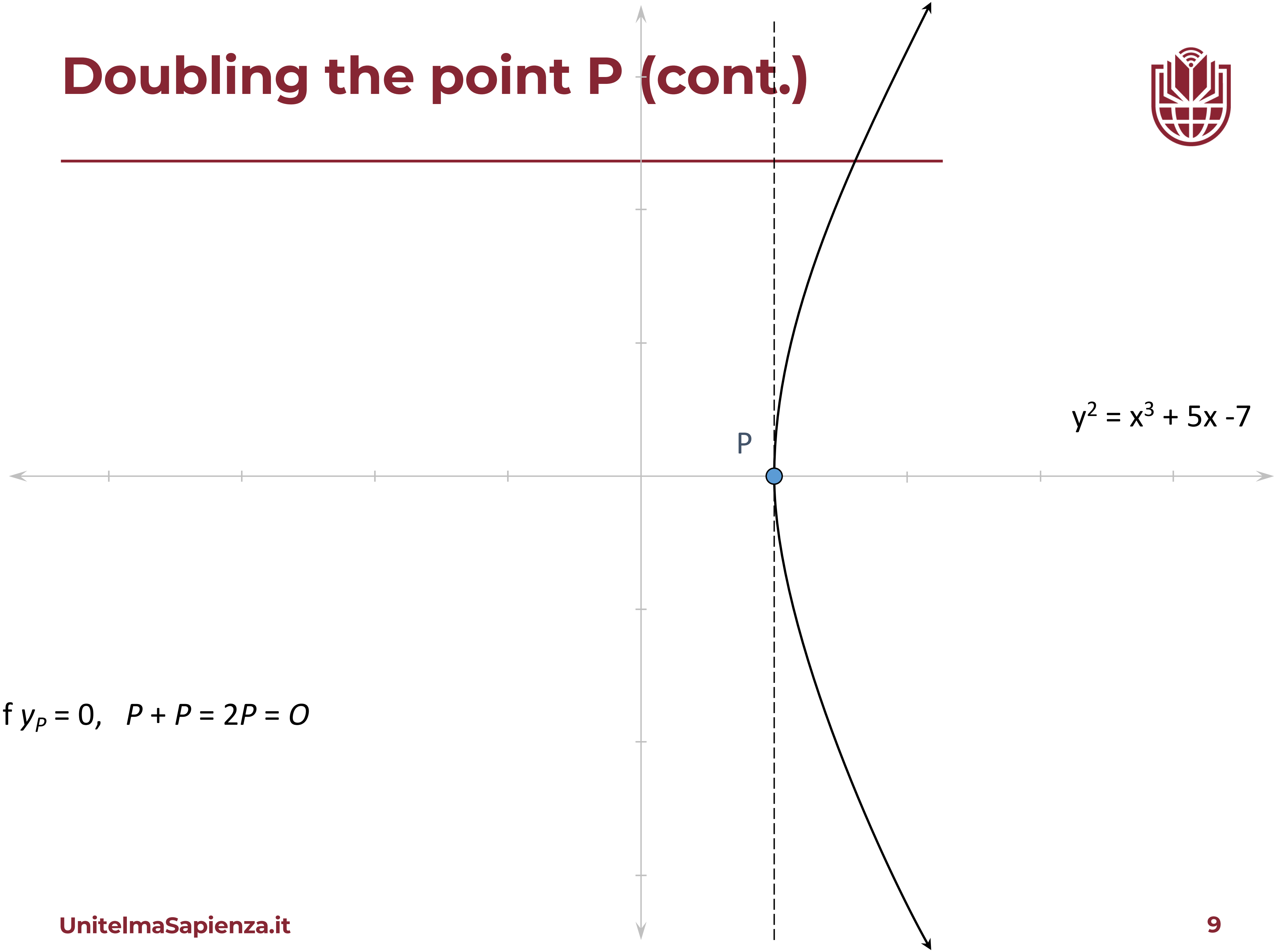


To add a point P to itself, a tangent line to the curve is drawn at the point P.

If $y_P \neq 0$,

$P + P = 2P = R$

Doubling the point P (cont.)



If $y_P = 0$, $P + P = 2P = O$

Elliptic Curves over Finite Fields

- To provide fast and precise arithmetic for cryptographic applications, finite fields are used in place of the real numbers. Variables and coefficients are finite integers
- For example, $y^2 = x^3 + ax + b$ equality intended mod p with a and b in F_p.
- The algebraic formulas derived from the geometric arithmetic of elliptic curves over real numbers can be adapted for elliptic curves over finite fields.
- Need large number of points on the curve

Finite Elliptic Curves

have two families commonly used:

1) prime curves $E_p(a,b)$ defined over Z_p
   - $y^2 \bmod p = (x^3 + ax + b) \bmod p$ where $4a^3 + 27b^2 \bmod p \neq 0$
   - use integers modulo a **prime p** (ranging between 112-521 bits) for both variables and coefficients
   - best in software
   - Example: P=(3,10), Q=(9,7), in $E_{23}(1,1)$
     - P+Q = (17,20)
     - 2P = (7,12)

Elements (points) of the group

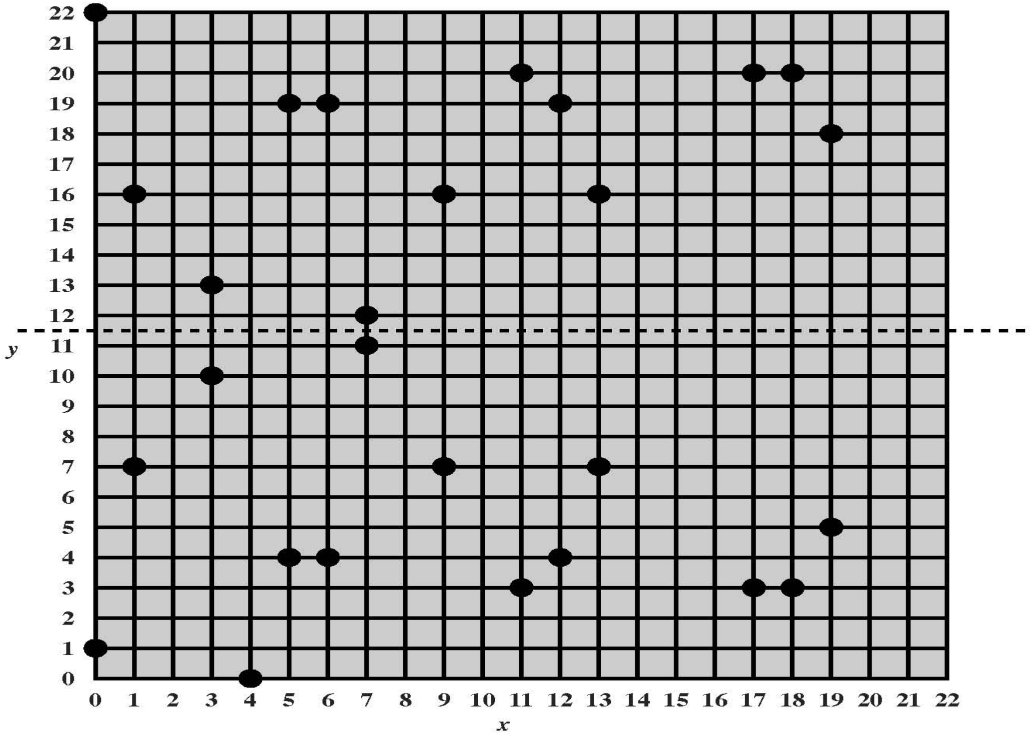


Figure 10.10 The Elliptic Curve $E_{23}(1,1)$

Finite Elliptic Curves

2) binary curves $E_{2^m}(a,b)$ defined over GF(2^m)

- $y^2 + xy = x^3 + ax^2 + b$, where $b \neq 0$
- elements of the finite field are integers of length at most **m** bits
- **m** ranging between 113-571 bits
- best in hardware
- Take a slightly different form of the equation
- Different close forms for addition

Elliptic Curve Cryptography

- ECC addition on the curve is analog of modulo multiplication
- ECC repeated addition is analog of modulo exponentiation

need “hard” problem equivalent to discrete log

- `Q=k×P`, where Q,P belong to a prime curve
- It is “easy” to compute Q given k, P
- but “hard” to find k given Q, P
- known as the *elliptic curve logarithm problem*

ECC Diffie-Hellman

There is key agreement analogous to standard D-H

- users select a suitable curve $E_p(a,b)$
- select base point $G=(x_1,y_1)$ which has large order n s.t. $n \times G=O$
- Users A and B select private keys $priv_A<n$, $priv_B<n$
- Each computes corresponding public keys: $Pub_A=priv_A \times G$, $Pub_B=priv_B \times G$
- compute shared key: $K=priv_A \times P_B$, $K=priv_B \times P_A$
  - same since $K=(priv_A * priv_B) \times G$

ECC Encryption/Decryption

several alternatives, consider simplest (**El Gamal**)

- select suitable elliptic curve P_m and point G as in D-H
- encode message M as (one coordinate of) a point on the curve P_m
- each user chooses private key $\mathtt{priv}_A < n$
- compute public key $\mathtt{Pub}_A = \mathtt{priv}_A \times G$
- encrypt P_m : $C_m = \{ k \times G,\ P_m + k \times \mathtt{Pub}_B \}$, k random
- decrypt C_m by computing:

$$P_m + k \times \mathtt{Pub}_B - \mathtt{priv}_B \times (k \times G) =$$

$$P_m + k \times (\mathtt{priv}_B \times G) - \mathtt{priv}_B \times (k \times G) = P_m$$

ECC Security

- relies on elliptic curve logarithm problem
  - Best known algorithms for discrete version or for factorization do not work on EC
- compared to factoring, can use much smaller key sizes than with RSA
  - ECC public key between 160 and 571 bit claimed to be as safe as RSA public key in 2048-4096 bit range
- for equivalent key lengths, computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

Comparison of Public-key crypto

Time to break (in MIPS- years)	RSA key size (in bits)	ECC key size (in bits)
10**4	512	106
10**8	768	132
10**11	1024	160
10**20	2048	210
10**78	21000	600

ECDSA

A version of the DSA based on the elliptic curve logarithm problem, NIST standard with recommended curves (!)

used by Bitcoin with Parameters

- Private key 256 bits
- Public key uncompressed 512 bit
- Public key compressed 256 bits
- Message to be signed 256 bits (after hash function)
- Signature 256 bits

Needs good source of randomness

ECDSA: generate signature

Given a private key d, a message m to sign, and its hash value z=h(m)

- Select random $0 < k < n$ (n order of G)
- Compute $k \times G = (x1, y1)$
- Compute $r = x1 \bmod n$ (if r=0, select different random)
- Compute $s = k^{-1} (z + r*d) \bmod n$ (if s=0 select different k)
- The signature is (r, s)

NOTE: is (r,s) is a valid signature, so is (r, -s mod n)

ECDSA: verify signature

Given message m, public point $Q = d \times G$ and signature (r, s) for m

- Verity that r and s are in interval (0, n) (if not, reject)
- Compute hash value of message z = h(m)
- Compute the inverse $w = s^{-1} \bmod n$ of s
- Compute the values u1 = z*w mod n and u2 = r*w mod n
- Compute the point $X = u1 \times G + u2 \times Q$
  - If $X=O$ reject
  - If X=(x1, y1) and r = x1 mod n, accept