

# Criptografía de la curva elíptica

Curva elíptica

Problema del logaritmo discreto

ECDH

ECDSA

Curvas elípticas más utilizadas

Comparación con RSA

Demostración práctica

# Curva elíptica

# Curva elíptica

Consideramos un cuerpo  $K$  y definimos la curva elíptica  $E$  sobre él de la siguiente forma:

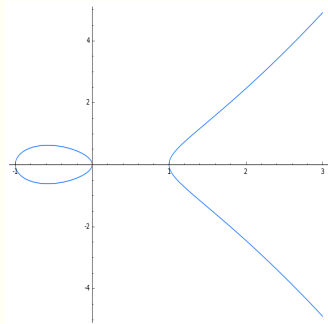
$$E = \{(x, y) \in K \times K : y^2 = x^3 + ax + b, 4a^3 + 27b \neq 0\} \cup \{\infty\}.$$

El punto  $\infty$  se añade por definición.

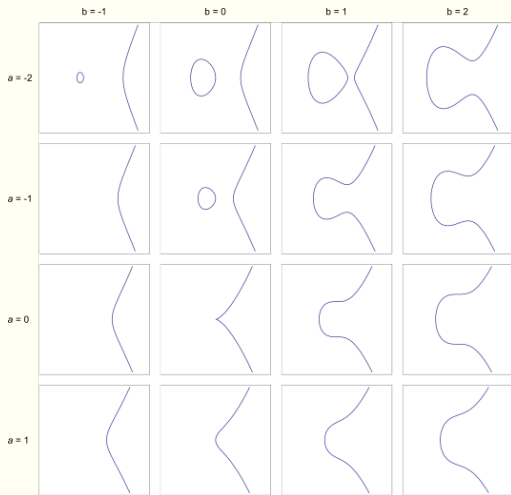
# Curva elíptica

Listing 1: Curva elíptica  $y^2 = x^3 - x$

```
sage: E = EllipticCurve(RR, [-1, 0]);  
plot(E, (-4, 3), color=hue(0.6))
```



# Curva elíptica

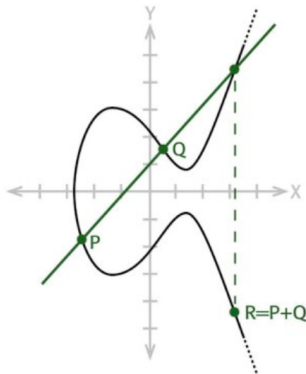


# Estructura de grupo

- ❖ Elemento neutro:  $\infty$ .
- ❖ Elemento opuesto de  $P = (x_P, y_P)$ :  
 $-P = (x_P, -y_P)$ .
- ❖ Operación de suma:  $P + Q = R$ .

# Estructura de grupo

- Operación de suma:  $P + Q = R$ .





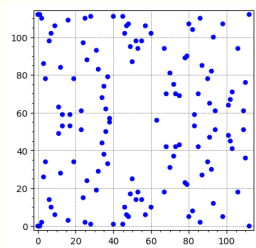
# Estructura de grupo

- ❖ Elemento neutro:  $\infty$ .
- ❖ Elemento opuesto de  $P = (x_P, y_P)$ :  
 $-P = (x_P, -y_P)$ .
- ❖ Operación de suma:  $P + Q = R$ .
- ❖ Producto por escalares,  $n \in \mathbb{Z}$ :

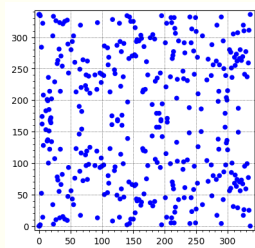
$$nP = \begin{cases} (P + \dots + P) & \text{si } n > 0 \\ 0 & \text{si } n = 0 \\ (-P - \dots - P) & \text{si } n < 0 \end{cases}$$

# EC sobre cuerpos finitos

$$E = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : y^2 \equiv x^3 + ax + b \text{ mód } p, \\ 4a^3 + 27b \not\equiv 0 \text{ mód } p\} \cup \{\infty\}.$$



$$y^2 + y = x^3 - x \text{ sobre } \mathbb{F}_{113}$$



$$y^2 + y = x^3 - x \text{ sobre } \mathbb{F}_{337}$$

# Subgrupos

- Los múltiplos de  $P$  generan un subgrupo cíclico de orden  $\min\{n \in \mathbb{N} : nP = \infty \text{ y } n|N\}$ .
- Primero elegimos el orden del subgrupo,  $n$ , y posteriormente hallamos el generador  $P$ .

# Problema del logaritmo discreto

# Problema del logaritmo

Si conocemos  $P$  y  $Q$  dos puntos de la curva elíptica, ¿podemos encontrar un  $k$  tal que  $Q = kP$ ?

# Problema del logaritmo

Si conocemos  $P$  y  $Q$  dos puntos de la curva elíptica, ¿podemos encontrar un  $k$  tal que  $Q = kP$ ?

No existe demostración que pruebe su dificultad.

Algoritmo *Baby-step, giant-step*, complejidad  $O(\sqrt{n})$ .

# Algoritmos

# Parámetros

- ❖  $p$ : número primo.
- ❖  $a, b$ : coeficientes de la curva elíptica.
- ❖  $G$ : generador del subgrupo.
- ❖  $n$ : orden del subgrupo.
- ❖  $h$ : cofactor del subgrupo,  $h = N/n$ .



# Claves

- ❖ Clave privada: entero aleatorio  $d$  elegido en el intervalo  $[1, n - 1]$ .
- ❖ Clave pública: punto de la curva  $H = dG$ .

# ECDH

# ECDH

1. Generación de claves:

Claves de Alice:  $d_A$  y  $H_A = d_A G$ .

Claves de Bob:  $d_B$  y  $H_B = d_B G$ .

2. Alice y Bob intercambian sus claves públicas.

3. Cálculo de clave compartida:

Alice calcula  $C = d_A H_B$ .

Bob calcula  $C = d_B H_A$ .

$$C = d_A H_B = d_A (d_B G) = d_B (d_A G) = d_B H_A = C.$$

# ECDSA

# ECDSA - Firma

1. Elegimos  $k$  entero aleatorio en el conjunto  $\{1, \dots, n-1\}$ .
2. Calculamos  $P = kG$ .
3. Hallamos el número  $r \equiv x_P \pmod n$ .
4. Si  $r$  es 0 volvemos al paso 1.
5. Calculamos  $s \equiv k^{-1}(z + rd_A) \pmod n$ .
6. Si  $s$  es 0 volvemos a 1.

La firma será:  $(r, s)$ .

# ECDSA - Verificación

1. Calculamos  $u_1 \equiv s^{-1} z \bmod n$ .
2. Calculamos  $u_2 \equiv s^{-1} r \bmod n$ .
3. Calculamos el punto  $P = u_1 G + u_2 H_A$ .

La firma es válida si  $r \equiv x_P \bmod n$ .

# ECDSA - Importancia de $k$

1.  $r_1$  es igual a  $r_2$ , puesto que  $r \equiv x_P \pmod n$  y  $P = kG$  es el mismo para las dos firmas.
2.  $(s_1 - s_2) \pmod n \equiv k^{-1}(z_1 - z_2) \pmod n$ .
3.  $k(s_1 - s_2) \pmod n \equiv (z_1 - z_2) \pmod n$ .
4.  $k \equiv (z_1 - z_2)(s_1 - s_2)^{-1} \pmod n$ .
5. Despejamos  $d_S$  de  $s \equiv k^{-1}(z + rd_S) \pmod n$  y obtenemos  $d_S \equiv r^{-1}(sk - z) \pmod n$ , todos valores conocidos.

# Curvas elípticas más utilizadas



# Curvas elípticas más utilizadas

- ❖ Spec256k1 (o Curve25519): se utiliza para ECDSA en el modelo criptográfico de Bitcoin. Viene dada por:

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

Se suele utilizar como punto base

$$G = (0x79be667ef9dcbbac55a06295ce870b07029bfc0  
0x483ada7726a3c4655da4fbfc0e1108a8fd17b44$$

# Curvas elípticas más utilizadas



que nos aporta un grupo de orden

$$n = 2^{256} - 0x14551231950b75fc4402da1732fc9bebf.$$

- ❖ NIST P-256: es la que trae *OpenSSL* por defecto y existen una gran cantidad de métodos para optimizar su uso.
- ❖ Curve25519: es una de las curvas más rápidas que existen actualmente y su implementación de referencia es de dominio público. Viene dada por

$$y^2 = x^3 + 48662x^2 + x$$

# Curvas elípticas más utilizadas



definida sobre el cuerpo finito con  $2^{255} - 19$  elementos el punto base tal que  $x = 9$ . Esto nos aporta un subgrupo de orden

$2^{252} + 0x14def9dea2f79cd65812631a5cf5d3ed$ .

- Curve448: ofrece potencialmente 224 bits de seguridad y aporta un gran rendimiento. Su implementación base está disponible bajo una licencia MIT.

# Comparación con RSA

# Comparación con RSA

	RSA	EC
<b>Problema en que basa su seguridad</b>	Factorización	Logaritmo discreto
<b>Almacenamiento de claves</b>	1024	160
	153600	521

# Demostración práctica

# OpenSSL

```
$ openssl ecparam -list_curves
$ openssl ecparam -name secp256k1 -out secp256k1.pem
$ cat secp256k1.pem
$ openssl ecparam -in secp256k1.pem -genkey -noout -out secp256k1-key.pem
$ cat secp256k1-key.pem
$ openssl ecparam -name secp256k1 -genkey -noout -out secp256k1-key.pem
$ cat secp256k1-key.pem
$ openssl ec -in secp256k1-key.pem -pubout -out ecpubkey.pem
```