Trabajo Teoría

# Curva Elíptica

24 de noviembre de 2019

Sofía Almeida Bruno
Pedro Manuel Flores Crespo
María Victoria Granados Pozo

# Índice

**Secciones** **Página**

# 1. Introducción

# 2. Curva Elíptica

Antes de entrar en cómo usar las curvas elípticas en criptografía debemos definirlas y estudiar su estructura de grupo.

Una curva elíptica será el conjunto de puntos que verifiquen la ecuación:

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

donde $a$ y $b$ son constantes y verifican $4a^3 + 27b \neq 0$[1]. Esta ecuación es la **ecuación de Weierstrass** para curvas elípticas. Normalmente, $a, b, x, y$ toman valores en un cuerpo. Por ejemplo: los números reales $\mathbb{R}$, los números complejos $\mathbb{C}$, los números racionales $\mathbb{Q}$, un cuerpo finito $\mathbb{F}_p$ para un primo $p$ (este es el que se utiliza en criptografía habitualmente), un cuerpo finito $\mathbb{F}_q$ donde $q = p^k$, $k \geq 0$, ...

De forma general, consideramos un cuerpo $K$ y definimos la curva elíptica $E$ sobre él de la siguiente forma:

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

El punto $\infty$ lo incluimos por definición. Lo podemos imaginar como un punto que se encuentra en la parte alta del eje y, pero también en la parte de abajo. Una línea que pase por este punto será una línea vertical, luego dos líneas verticales distintas coinciden en este punto.

Podemos usar Sage para visualizar algunos ejemplos de curvas elípticas. En el Código 1 observamos la forma genérica de definir en Sage curvas elípticas.

```
sage:  E = EllipticCurve(K, [a, b]); E
Elliptic Curve defined by y^2  = x^3 + x + 3 over a field K
```

Código 1: Curva elíptica en Sage

El Código 2 es el necesario para crear la curva con ecuación de WeiErstrass $y^2 = x^3 - 1$.

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

Código 2: Curva elíptica $y^2 = x^3 - x$

Podemos observar la salida obtenida en la Figura 1.

[1]Imponemos esta condición para que la curva no presente raíces múltiples, que pueden dar problemas en la teoría general que expondremos. Se pueden tratar ambos casos de forma individual aunque nosotros no incluiremos estas excepciones en nuestro trabajo

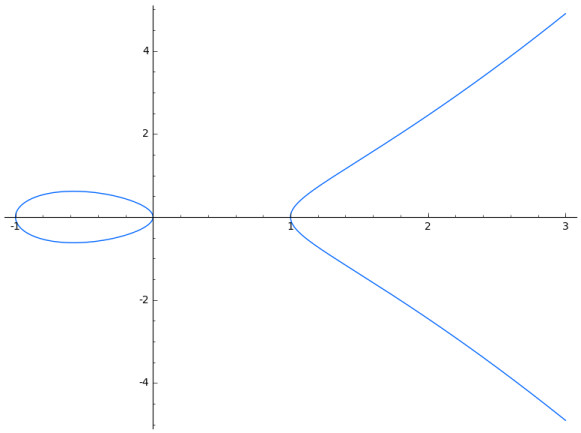


Figura 1: Salida Código 2

Programamos de forma similar el código para visualizar la gráfica de la curva dada por la ecuación $y^2 = x^3 + x$, véase el Código 3, cuya salida se encuentra en la Figura 2

```
sage: E =  EllipticCurve(RR, [1, 0]);
      plot(E, (-2, 4), color=hue(0.8))
```

Código 3: Curva elíptica $y^2 = x^3 + x$

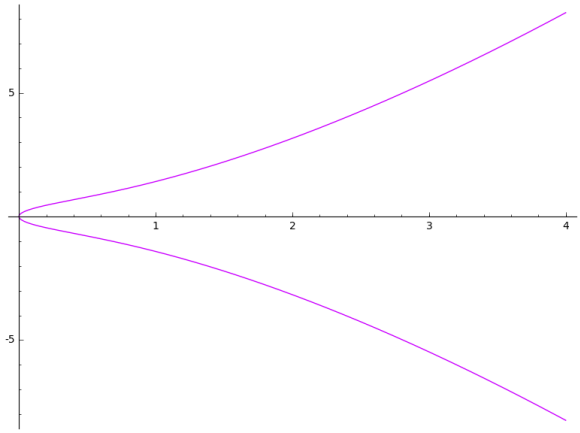


Figura 2: Salida Código 3

En la Figura 3 tenemos una tabla con la forma de las curvas elípticas para valores enteros de $a$ entre -2 y 1 y $b$ entre -1 y 2.

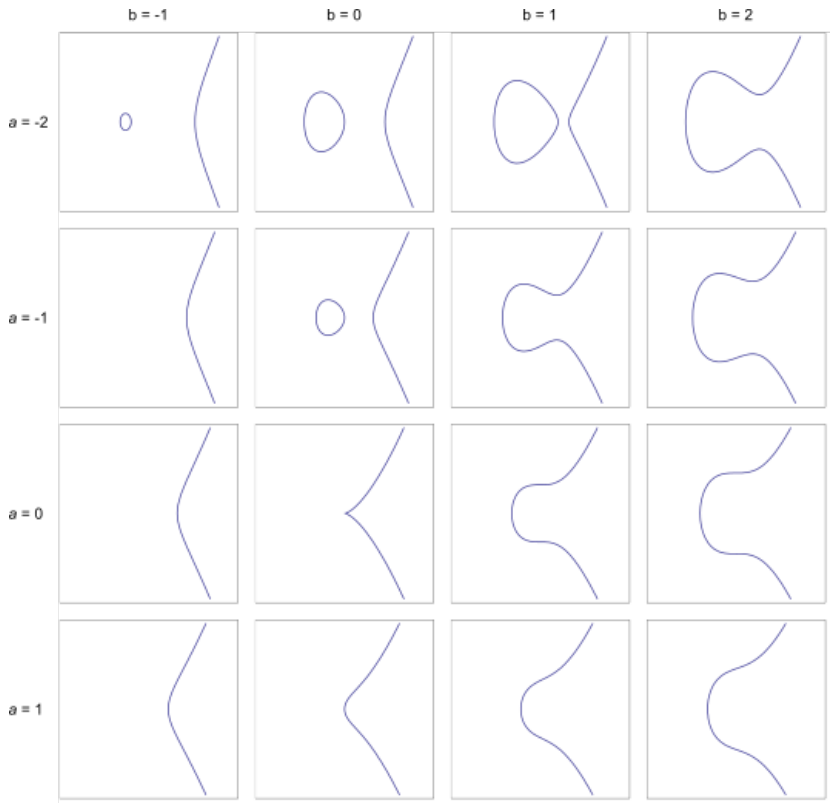


Figura 3: Ejemplos de curvas elípticas

## 2.1. Estructura de grupo

Dada una curva elíptica podemos definir un grupo abeliano formado por puntos de la misma.

1. Tomamos como elemento neutro el punto del infinito $\{\infty\}$.

2. El opuesto de un punto $P$ viene dado por su simétrico por el eje $X$. Más concretamente, si $P = (x_P, y_P)$ entonces $-P = (x_P, -y_P)$.

3. En general, dados dos puntos de la curva $P = (x_P, y_P)$ y $Q = (x_Q, y_Q)$ con $x_P \neq y_P$, su suma es el punto $-R$ que viene definida por:

   $$(x_R, -y_R) = (m^2 - x_P - x_Q, -y_P - m(x_R - x_P)) = (m^2 - x_P - x_Q, -y_Q - m(x_R - x_Q)),$$

   donde

   $$m = \frac{y_P - y_Q}{x_P - x_Q}.$$

   Puede causar asombro que definamos $P + Q = -R$. Esto se deba al sentido geométrico de la suma en la curva. Esta operación debe seguir la siguiente regla: "dados tres puntos $P$, $Q$ y $R$ alineados y no nulos su suma $P + Q + R$ debe ser el elemento neutro". Por

lo tanto, $P + Q$ es el opuesto al punto de corte de la recta que une ambos puntos. De ahí también notamos la presencia del signo menos en la segunda coordenada tal y como indicamos en el punto anterior. También notar que el sentido geométrico nos aporta intuitivamente la conmutatividad ya que la recta tangente es la misma en el caso $P+Q$ y $Q + P$ y la asociatividad ya que $(P + Q) + R = P + (Q + R) = (P + R) + Q$ (y las demás posibilidades).

No podemos usar estas fórmulas en algunos casos especiales, poe ejemplo cuando $P = Q$. En este caso no existe una sola recta que pase por "ambos" puntos. Sin embargo, si nos aproximamos a $Q$ mediante puntos $Q'$ con $Q' \neq P$ en seguida vemos que la recta que obtendríamos es la tangente en $P$. Así, definimos su suma como el opuesto del punto de intersección de la recta tangente en $P$. Otra situación interesante es cuando $P \neq Q$ pero no hay más puntos de corte. Nos encontramos en un caso parecido al anterior ya que uno de los puntos es tangente a la curva. Si suponemos que $P$ es donde se da la tangencia, por el caso anterior tememos que $P + P = -Q$ por lo que $P + Q = -P$. En estas situaciones de tangencia tenemos que:

$$m = \frac{3x_P^2 + a}{2y_P}.$$

En ambos casos debemos comprobar que la suma pertenece a la curva y que los tres puntos están alineados.

4. Finalmente, dado $n \in \mathbb{Z}$, el producto por escalares lo definimos como:

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

Parámetros de dominio para los algoritmos (p, a, b, G, n, h) donde:

- p: primo
- a, b: Coeficientes de la curva elíptica, hay que elegirlos con cuidado para que el algoritmo sea seguro
- G: Generador del grupo
- n: Orden del grupo
- h: Cofactor del subgrupo

## 2.2. El problema del logaritmo discreto

La seguridad de la criptografía de la curva elíptica reside en la dificultad para calcular logaritmos discretos. Este problema es análogo el de otros criptosistemas como el Digital

Signatura Algorithm (DSA) o el intercambio de llaves mediante el mecanismo de Diffie-Hellman.

En nuestro caso, el problema consiste en lo siguiente: "si conocemos $P$ y $Q$ dos puntos de la curva elíptica, ¿podemos encontrar un $k$ tal que $Q = kP$? ". Vemos que no estamos calculando un logaritmo como tal pero se sigue usando ese término por analogía a otros sistemas como se ha mencionado anteriormente.

Actualmente, no existe ninguna demostración matemática que efectivamente pruebe la dificultad de dicho cálculo. Solo sabemos que es difícil pero no podemos estar seguros. Con el siguiente ejemplo ilustramos su complejidad. Para ello usamos un algoritmo "Baby-step, giant-step" que se basa en el hecho de que cualquier entero $x$ puede expresar como $x = am+b$ con $a, m$ y $b$ también enteros. Escribimos entonces:

$$\begin{aligned} Q &= xP \\ &= (am + b)P \\ &= amP + bP. \end{aligned}$$

Así, podemos expresar $Q - amP = bP$. El algoritmo consiste en en calcular algunos valores de $bP$ y otros de $Q - amP$ hasta que encontremos una correspondencia. El algoritmo también indica que debemos escoger $m = \sqrt{n}$ y los números $a$ y $b$ se mueven entre 0 y $m$ por lo que mientras $bP$ tiene incrementos pequeños ("baby") $amP$ los tiene grandes ("huge"). Este método nos aporta una complejidad tanto en tiempo como en memoria de $O(\sqrt{n})$ . Este ataque tiene entonces complejidad exponencial pero es mejor que uno de fuerza bruta. De todos modos sigue siendo intratable ya que para un $n$ tal que $\sqrt{n} = 7{,}922816251426434 \times 10^{28}$ el almacenamiento de datos que podemos llegar a necesitar es de $2,5 \times 10^{30}$ bytes de memoria (la capacidad de almacenamiento mundial es de aproximadamente $10^{21}$ bytes).

# 3. Algoritmos

## 3.1. ECDSA

ECDSA es un algoritmo de firma digital, Elliptic Curve Digital Signature Algorithm, es una variante del algoritmo DSA (Digital Signature Algortihm) aplicado a curvas elípticas. Trabaja con el hash del mensaje en lugar de con el propio mensaje. La elección de la función hash es importante, de esto dependerá la seguridad del sistema criptográfico. El hash del mensaje tendrá una longitud de $n$ bit.

Imaginemos que Alice quiere firmar un mensaje con su llave privada ($d_A$), y la otra persona, Bob, quiere validar la firma con la llave pública de Alice ($H_A$). Alice es la única que puede producir las firmas válidas, sin embargo todo el mundo que tenga su llave pública puede verificarlas.

Alice y Bob están usando los parámetros del dominio. El hash truncado lo denotaremos por $z$.

Algoritmo de firma del mensaje de Alice, a partir de $k$ y $z$ se genera la firma con la clave privada de Alice:

1. Tomamos un entero k de forma aleatorio en el conjunto { 1, ..., n-1}.

2. Calcular $P = kG$.
3. Calcular el número $r = x_P$.
4. Si $r$ es 0 entonces se toma otro $k$ y se intenta de nuevo.
5. Se calcula $s = k^{-1}(z + rd_A) \bmod n$ con $k^{-1}$ el inverso multiplicativo de $k$ módulo $n$.
6. Si $s$ es 0 entonces se elige otro $k$ y se vuelve al principio.

Al final obtendremos la firma que será la pareja ($r$, $s$)

Ahora entra el juego Bob que para validar la firma, a partir del mensaje firmado y de $z$ con la clave pública de Alice

### 3.2. ECDH

### 3.3. Comparación con RSA

## 4. Conclusiones

# I. Glosario