Universidad de Oviedo

Facultad de Ciencias
Grado de Matemáticas

Trabajo Fin de Grado

Aplicaciones criptográficas de las curvas elípticas

Autor: David Pérez Navarro

Tutor: Ignacio Fernández Rúa

Índice general

Capítulo 1

Introducción

1.1. Motivación

Las curvas elípticas son objetos geométricos de sencilla descripción algebraica que poseen una notable riqueza matemática, ya que pueden dotarse de una operación de suma que convierte al conjunto de sus puntos en un grupo abeliano[1]. Gracias a esta propiedad, las curvas elípticas han captado el interés de diversas ramas de las matemáticas y han encontrado aplicaciones relevantes en contextos teóricos y prácticos.

Desde mediados del siglo XIX, el estudio de las curvas elípticas cobró gran importancia en diversos campos de las matemáticas. Un ejemplo destacado es su uso en 1995 por Andrew Wiles para demostrar el Último Teorema de Fermat[2], más de tres siglos después

[1]Un grupo abeliano es un grupo $(G, +)$ donde la operación $+$ es conmutativa: $a + b = b + a$ para todo $a, b \in G$

[2]El Último Teorema de Fermat conjeturado por el propio Pierre de Fermat en 1637, afirma que dado n un número entero mayor que 2, entonces no existen números enteros positivos x, y y z, tales que se cumpla la igualdad $x^n + y^n = z^n$.

de su planteamiento. Además, las curvas elípticas desempeñan un papel fundamental en la conjetura de Birch y Swinnerton-Dyer, así como en criptografía, especialmente las que son sobre cuerpos finitos.

El presente trabajo tiene como objetivo estudiar las aplicaciones criptográficas de las curvas elípticas. La criptografía, que mezcla campos de las matemáticas, como el álgebra, con la informática, entre otras ciencias, es una rama de la ciencia muy importante en la actualidad. Su objetivo principal es garantizar la seguridad de la información mediante técnicas que permiten proteger la confidencialidad, la integridad, la autenticación y el no repudio de los datos. En un mundo cada vez más interconectado, donde las transacciones digitales, las comunicaciones electrónicas y el intercambio de información sensible son constantes, la criptografía juega un papel esencial para proteger la privacidad y la seguridad.

En este contexto, las curvas elípticas han revolucionado el campo de la criptografía debido a su capacidad para proporcionar altos niveles de seguridad con claves más pequeñas en comparación con otros sistemas criptográficos clásicos, como RSA o Diffie-Hellman. Este enfoque se basa en la dificultad computacional del problema del logaritmo discreto en curvas elípticas, cuya resolución es más compleja que en grupos multiplicativos tradicionales, lo que permite mantener la seguridad con un uso más eficiente de los recursos computacionales.

El uso de criptosistemas basados en curvas elípticas no solo ofrece ventajas en términos de eficiencia, sino que también es fundamental en aplicaciones modernas como la seguridad en las comunicaciones móviles, las transacciones bancarias, los protocolos de autenticación y las firmas digitales. Estos sistemas permiten realizar operaciones criptográficas robustas incluso en dispositivos con recursos limitados.

A lo largo de este trabajo, se analizarán en detalle varios criptosistemas basados en curvas elípticas, incluyendo el protocolo de intercambio de claves de Diffie-Hellman, el esquema de cifrado ElGamal y el algoritmo de firma digital. Además, se estudiará el algoritmo de Massey-Omura, que también puede adaptarse al marco de las curvas elípticas. El estudio de estos criptosistemas es fundamental para comprender cómo las matemáticas avanzadas se aplican a la seguridad informática y cómo estas técnicas continúan evolucionando para enfrentar nuevos desafíos criptográficos.

1.2. Introducción a la Criptografía

La palabra **Criptografía** proveniente del griego kryptós, *secreto*, y graphé, *escritura*, es la rama de la criptología [3] que estudia las técnicas de cifrado de mensajes en busca de la confidencialidad.

Si recurrimos al Diccionario de la lengua española, elaborado por la RAE, encontramos que la palabra criptografía se define como:

Arte de escribir con clave secreta o de un modo enigmático.

Sin embargo, esta definición no logra explicar en términos científicos en qué consiste realmente la criptografía, por ello, se opta por esta nueva definición más precisa.

Definición 1.1. *La Criptografía consiste en el diseño y análisis de técnicas matemáticas que permiten proteger las comunicaciones ante la presencia de usuarios no legítimos (adversarios) o del uso ilegítimo por parte de usuarios legítimos.*

Esta protección se busca cumpliendo los siguientes **objetivos de seguridad**:

- **Confidencialidad**: garantizar que la información permanezca secreta para todos, excepto para aquellos individuos autorizados, impidiendo que los adversarios puedan acceder o interpretar su contenido.
- **Integridad de los datos**: asegurar que la información no ha sido modificada por usuarios no autorizados, permitiendo detectar cualquier alteración en los datos originales.
- **Autenticación del origen de los datos**: verificar que la información proviene realmente de la fuente legítima y no ha sido enviada por un usuario no autorizado.
- **Autenticación de una entidad**: corroborar la identidad de una entidad para garantizar que es quien dice ser.

[3] La disciplina que estudia los mensajes que se convierten en difíciles o imposibles de leer por entidades no autorizadas.

- **No repudio**: evitar que un usuario niegue haber realizado una acción, por ejemplo, si una empresa envía un correo electrónico a un trabajador y este se lo envía a una tercera parte neutral, la empresa no puede negar haber enviado ese correo.

El modelo básico de comunicación lo podemos representar mediante la figura 1.1, en ella A (Alice) y B (Bob) se comunican mediante un canal no seguro, donde E (Eve)[4] es capaz de interferir todas las comunicaciones. El objetivo de E es vulnerar cualquier mecanismo de seguridad que A y B implementen para proteger sus mensajes.

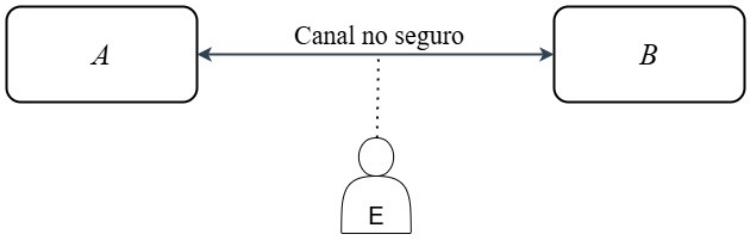


Figura 1.1: modelo básico de comunicación.

El ejemplo más sencillo, o quizás el primero que se nos viene a la mente, es una conversación de teléfono entre A y B, con E intentando escuchar la conversación. Sin embargo, A y B no tienen por qué ser personas; por ejemplo, A podría ser el buscador mediante el cual un individuo va a comprar y B el sitio web en el que se va a realizar la compra. E podría interferir en la comunicación para robar los datos de una tarjeta de crédito, o incluso hacerse pasar por A o B para robar información durante la transacción.

Respecto a E, el adversario, generalmente lo dotamos de las siguientes cualidades:

- E es capaz de leer toda la información que se transmite a través del canal.
- E puede modificar la información enviada o inyectar nueva información.

[4] A (Alice) y B (Bob) son los individuos autorizados, mientras que E (Eve) es el adversario.

- E tiene conocimiento del protocolo criptográfico[5] usado durante la comunicación. Sin embargo, E nunca conocerá las claves privadas de los usuarios.

Dentro de la criptografía moderna se distinguen dos tipos: la **criptografía de clave privada** y la **criptografía de clave pública**.

1.3. Criptografía de clave privada

La **criptografía de clave privada**, o de clave simétrica, involucra un tipo de criptosistema[6] en el que tanto A como B utilizan la misma clave secreta, es decir, desconocida para E. La seguridad de este criptosistema depende de mantener la clave en secreto; cualquier persona que la conozca es capaz de acceder a la información cifrada. Algunos de estos criptosistemas son: Data Encryption Standard (DES), RC4 o Advanced Encryption Standard (AES). También involucra códigos de autenticación de mensaje, como HMAC, que garantizan la integridad y la autenticidad de los datos.

1.4. Criptografía de clave pública

Introducida en 1976 por Whitfield Diffie, Martin Hellman y Ralph Merkle[7] la **criptografía de clave pública** surge como solución a los problemas presentes en la criptografía

[5]Un protocolo criptográfico es un conjunto de reglas y procedimientos que utilizan algoritmos criptográficos para lograr objetivos de seguridad específicos en la comunicación y el intercambio de datos.

[6]Un criptosistema es un conjunto de algoritmos matemáticos que permiten llevar a cabo procesos de cifrado y descifrado de información con el objetivo de garantizar la confidencialidad de los datos.

[7]El concepto de criptografía de clave pública fue descrito en 1969 por el británico James Ellis, sin embargo, su descubrimiento fue clasificado por el gobierno británico. Tras su muerte, en 1997, el gobierno desclasificó su trabajo sobre criptografía de clave pública.

de clave privada (o simétrica). Estos son, entre otros:

- El problema de la distribución de claves entre las partes, la necesidad de una clave común y secreta que tiene que ser compartida por ambas partes antes del intercambio de información convierte a los criptosistemas de clave privada en inseguros e inutilizables.
- El problema del manejo de claves, en un hipotético servidor con N individuos cada uno de ellos debe contar con una clave distinta para cada uno de los $N-1$ individuos restantes.

Además, estos criptosistemas nos dan la garantía de no repudio, entendiendo como no repudio que, dado un mensaje:

- El emisor no puede negar que envió un mensaje.
- El receptor no puede negar que recibió el mensaje.

En esta técnica criptográfica, cada individuo (o entidad) tendrá una clave formada por la dupla (e, d), donde e será la clave pública y d una clave privada que mantendrá cierta relación con e. La imposibilidad a nivel computacional de hallar la clave privada a partir de la clave pública permite que la criptografía de clave pública sea usada en nuestras conexiones.

El primer criptosistema de clave pública práctico fue diseñado por Rivest, Shamir y Adleman en 1978. Conocido como RSA, basa su seguridad en la dificultad de factorizar grandes números. También encontramos ElGamal, basado en la dificultad de resolver el problema del logaritmo discreto. Para más detalles sobre los criptosistemas de clave pública mencionados, consultar (6), capítulo 1, pp. 6–11.

En la criptografía de clave pública encontramos los esquemas basados en **curvas elípticas**, foco principal de este trabajo. Todo comienza en 1985, cuando de manera independiente los matemáticos americanos Neal Koblitz y Victor Miller proponen el uso de las curvas elípticas para el diseño de criptosistemas de clave pública (o asimétricos).

Comenzaremos viendo los beneficios del uso de la curva elíptica en criptografía frente a otros esquemas (sección 1.4.1), detallaremos la estructura de este objeto matemático sobre

diferentes cuerpos (sección 3), presentaremos criptosistemas basados en curvas elípticas (sección 5.2) y analizaremos cómo también existen tests de primalidad basados en esta estructura matemática (capítulo 6).

1.4.1. Beneficios de la Criptografía de curva elíptica

La mayor ventaja de los criptosistemas basados en curva elíptica es el tamaño de la clave. Para visualizar esta ventaja, compararemos algoritmos de RSA y de curvas elípticas.

Basándonos en los datos proporcionados en (1) la ECC[8] ofrece una seguridad equivalente a la del RSA con claves de menor tamaño, así se puede ver en la Tabla 1.1.

Clave simétrica	**ECC**	**RSA**
80	163	1024
112	233	2240
128	283	3072
192	409	7680
256	571	15360

Tabla 1.1: Comparación de tamaños de clave (en bits) para niveles de seguridad análogos.

Para evitar ataques exitosos a los criptosistemas de curva elíptica se recomiendan claves de 283 bits de tamaño en caso de buscar seguridad temporal, y claves de 571 bits de tamaño si se busca seguridad a largo plazo. Notar que para alcanzar esos niveles de seguridad en el RSA es necesario claves de entre 3072 y 15360 bits de tamaño.

La Tabla 1.2 compara el tiempo en generar las claves para ECC y RSA.

Podemos concluir que la generación de claves para ECC supera a RSA en todas las longitudes de clave, y es especialmente evidente a medida que aumenta la longitud de la clave. Sin embargo, aunque el tamaño de la clave es más pequeño, la complejidad de las operaciones en la curva elíptica (como la suma de puntos) es más alta que en los grupos

[8]Las siglas ECC, de Elliptic-curve cryptography, hacen referencia a la criptografia de curva elíptica.

ECC (Bits)	RSA (Bits)	ECC (Tiempo)	RSA (Tiempo)
163	1024	0.08s	0.16s
233	2240	0.18s	7.47s
283	3072	0.27s	9.80s
409	7680	0.64s	133.90s
571	15360	1.44s	679.06s

Tabla 1.2: Comparación de tamaños de clave y tiempos de generación (ECC vs. RSA).

multiplicativos finitos, como así podemos ver en la Tabla 1.3. En dicha tabla comparamos el rendimiento en la generación de firmas digitales entre ECC y RSA. La generación de firmas digitales es un proceso criptográfico por el cual una persona o entidad crea una firma que autentica un mensaje o un documento digital. Para más detalles sobre el funcionamiento de un algoritmo de firma digital consultar la sección 5.2.3.

ECC (Bits)	RSA (Bits)	ECC (Tiempo)	RSA (Tiempo)
163	1024	0.15s	0.01s
233	2240	0.34s	0.15s
283	3072	0.59s	0.21s
409	7680	1.18s	1.53s
571	15360	3.07s	9.20s

Tabla 1.3: Comparación del rendimiento en la generación de firmas (ECC vs. RSA).

La ventaja de construir un grupo abeliano utilizando curvas elípticas radica en el **Teorema de Hasse** (3.12), el cual establece que, en cuerpos finitos, el número de puntos en una curva elíptica es aproximadamente igual a q, donde q es el tamaño del cuerpo finito. A diferencia de los grupos multiplicativos, donde para cada valor de q existe únicamente un grupo multiplicativo $(\mathbb{F}_q)^*$, en el caso de las curvas elípticas es posible definir múltiples curvas distintas sobre un mismo cuerpo finito. Esta propiedad ofrece una mayor flexibilidad y diversidad de opciones al trabajar con curvas elípticas.

Otro gran beneficio de la criptografía de curva elíptica es la variedad, ya que, en criptografía de curva elíptica se pueden definir numerosas curvas distintas mediante la elección de diferentes parámetros. Esta flexibilidad permite seleccionar curvas que optimizan distintos aspectos, como la eficiencia computacional, la resistencia a ataques específicos o la facilidad de implementación en determinados entornos. Además, facilita la existencia de estándares variados y adaptados a distintas necesidades de seguridad.

La memoria se dividirá en capítulos; tras la introducción se presenta el concepto de complejidad computacional (capítulo 2), que permite evaluar la eficiencia de los algoritmos utilizados en criptografía. En cada uno de los capítulos posteriores, se desarrollarán los elementos fundamentales para los criptosistemas basados en curvas elípticas. En el capítulo 3 se estudiará la estructura de las curvas elípticas sobre distintos cuerpos: los números reales, los racionales y los cuerpos finitos. Además, se analizará el funcionamiento de la suma de puntos sobre la curva, operación clave para el desarrollo de los esquemas criptográficos.

A continuación, el capítulo 4 se dedicará al estudio de los algoritmos de conteo de puntos en curvas elípticas definidas en cuerpos finitos, destacando el algoritmo de Schoof, junto a las herramientas matemáticas necesarias para el desarrollo.

En el capítulo 5, se abordará la criptografía de curva elíptica, comenzando por el problema del logaritmo discreto y continuando con los principales ataques y algoritmos asociados. Así mismo, se expondrán los criptosistemas más utilizados, como el intercambio de claves de Diffie–Hellman, el sistema de ElGamal y algoritmos de firma digital.

El capítulo 6 presenta el uso de curvas elípticas en tests de primalidad; concretamente, se estudia el algoritmo de Goldwasser–Kilian. Se terminará en el capítulo 7 con las conclusiones.

Capítulo 2

Complejidad computacional

Definición 2.1. *Un algoritmo es un proceso computacional bien definido que toma un valor, o conjunto de valores, como* ***entrada*** *y devuelve un valor, o conjunto de valores, como* ***salida****. Es decir, un algoritmo es una secuencia de pasos computacionales que transforman la entrada en la salida.*

Un ejemplo de algoritmo es el **algoritmo de ordenamiento**:

- **Entrada:** Una secuencia de n números reales $\langle x_1, x_2, \dots x_n \rangle$.
- **Salida:** Una permutación ordenada $\langle x'_1, x'_2, \dots x'_n \rangle$ tal que $x'_1 \leq x'_2 \leq \cdots \leq x'_n$.

Se denominará tamaño de entrada al número de bits necesarios para representar la entrada del algoritmo. Por ejemplo, el algoritmo de factorización de Fermat, que factoriza un número entero n buscando expresarlo como la diferencia de dos cuadrados, tiene un tamaño de entrada de $l = \lfloor log_2 n \rfloor + 1$ bits.

El tiempo de ejecución de un algoritmo es útil cuando es independiente de la plataforma (software y hardware) en la cual se implementa. Es por ello que se trabaja estimando el número de operaciones elementales que se realizan. El tiempo de ejecución es, por lo tanto, una cota superior expresada en función del tamaño de entrada. Por ejemplo, el algoritmo

que factoriza un entero n probando con todos los enteros posibles hasta $\sqrt{n}$ tiene un tiempo de ejecución de aproximadamente $\sqrt{n} \approx 2^{l/2}$ divisiones realizadas.

Dada la dificultad de obtener el tiempo de ejecución exacto de un algoritmo, se recurre a la notación O que permite aproximar el tiempo de ejecución en función del tamaño de entrada.

Definición 2.2. *Si $f, g : \mathbb{N} \longrightarrow \mathbb{R}^+$ son dos funciones, se define $f = O(g)$ si existe una constante c, con $c > 0$, y $n_0 \in \mathbb{N}$ tal que $f(n) \leq cg(n)$, $\forall n \geq n_0$. Se puede decir que f no crece más rápido que g multiplicado por una constante c.*
También se usará el concepto $f = o(g)$ si, para cualquier constante c, $\exists n_o \in \mathbb{N}$ tal que $f(n) \leq cg(n)$, $\forall n \geq n_0$. En este caso se dice que $f(n)$ se convierte en insignificante en comparación con $g(n)$.

Nota 2.1. *En la definición 2.2 $f = O(g)$ se dice que f* ***es una O grande de*** *g y $f = o(g)$ se dice que f* ***es una o pequeña de*** *g.*

Según el tiempo de ejecución podemos distinguir los siguientes algoritmos.

Definición 2.3. *Sea A un algoritmo con tamaño de entrada l bits.*

1. *A es un algoritmo de tiempo polinómico si su tiempo de ejecución es $O(l^c)$ para alguna constante $c > 0$.*
2. *A es un algoritmo de tiempo exponencial si su tiempo de ejecución no es de la forma $O(l^c)$ para ningún $c > 0$.*
3. *A es un algoritmo de tiempo subexponencial si su tiempo de ejecución es $O(2^{o(l)})$ y A no es un algoritmo de tiempo polinómico.*
4. *A es un algoritmo de tiempo completamente exponencial si su tiempo de ejecución no es de la forma $O(2^{o(l)})$.*

Como regla general en criptografía se dice que un problema es "sencillo" si se resuelve en tiempo polinómico, mientras que se considera "difícil" si requiere tiempo exponencial o subexponencial.

Capítulo 3

Curva Elíptica

A lo largo de este capítulo trataremos todo lo relacionado con las curvas elípticas, sus puntos y su estructura sobre diferentes cuerpos.

3.1. Curvas algebraicas

Definición 3.1. *Una **curva algebraica** sobre un cuerpo K se define como el conjunto de puntos $(x, y) \in K^2$ que satisfacen que $p(x, y) = 0$, con $p \in K[x, y]$ un polinomio de dos variables con coeficientes en K.*

Las curvas algebraicas pueden definirse tanto sobre el plano afín, con el que trabajaremos en la mayoría de secciones de este capítulo, como sobre el plano proyectivo (ver sección 3.5).

Las curvas algebraicas se clasifican en función del grado más alto del polinomio que las define. Se distinguen los siguientes casos:

- Si el grado es 1, se denominan **curvas lineales**. Este es el caso de las rectas.

- Si el grado es 2, se denominan **cónicas**. En este grupo se encuentran las circunferencias, elipses, parábolas, e hipérbolas, entre otras.
- Si el grado es 3, se denominan **curvas cúbicas**. Las curvas elípticas pertenecen a esta categoría.
- Si el grado es 4, se denominan **curvas cuárticas**.
- Finalmente, si el grado es n, con $n \geq 5$, se denominan **curvas de grado** n.

Entre las posibles características a estudiar de las **curvas cúbicas**, para su aplicación en la criptografía, destaca el estudio de los **puntos singulares**.

Definición 3.2. *Se denomina* ***punto singular*** *de una curva cúbica al punto en el cual la curva no es suave, es decir, donde el gradiente de la función que la define se anula simultáneamente en todas las direcciones, y por tanto no existe una única tangente bien definida. La tangente en un punto se define como la recta que aproxima localmente a la curva cúbica.*

La figura 3.1 muestra un punto singular en $(0, 0)$, pues en ese punto se intersecan dos rectas tangentes distintas. Si la curva no presenta puntos singulares, diremos que es "suave".

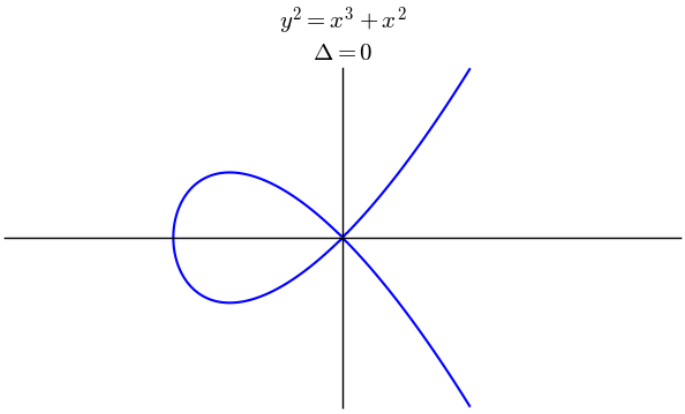


Figura 3.1: Curva cúbica real de ecuación $y^2 = x^3 + x^2$.

Δ representa el discriminante de la curva. En la sección 3.2 se detalla más sobre este valor.

3.2. Introducción a las curvas elípticas

Definición 3.3. *Una curva elíptica E sobre un cuerpo K se define como el conjunto de puntos $(x, y) \in K^2$ que verifican la ecuación:*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{3.1}$$

junto con un punto adicional, llamado punto del infinito, donde $a_1, a_2, a_3, a_4, a_6 \in K$ y $\Delta \neq 0$, donde Δ será el discriminante de E que se define como:

$$\begin{aligned} \Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \end{aligned} \tag{3.2}$$

Además, si L es una extensión del cuerpo K, es decir $L : K$, definimos al conjunto de los *puntos $L-$racionales* de E como:

$$E(L) = \left\{(x, y) \in L \times L : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\right\} \cup \{\infty\}$$

Nota 3.1. *El punto ∞ será el **punto del infinito**. También lo denotaremos como O.*

Nota 3.2. *Respecto a la definición 3.3 cabe destacar que:*

- *La curva elíptica es una curva cúbica no singular, es decir, no existen puntos $(x, y) \in K^2$ que satisfagan simultáneamente la ecuación (3.1) y las condiciones:*

$$\frac{\partial F}{\partial x}(x, y) = 0 \quad y \quad \frac{\partial F}{\partial y}(x, y) = 0,$$

  donde $F(x, y) := y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6$. Es decir, no hay puntos en los que se anulen simultáneamente las derivadas parciales de F, lo cual caracteriza a los puntos singulares (ver definición 3.2).

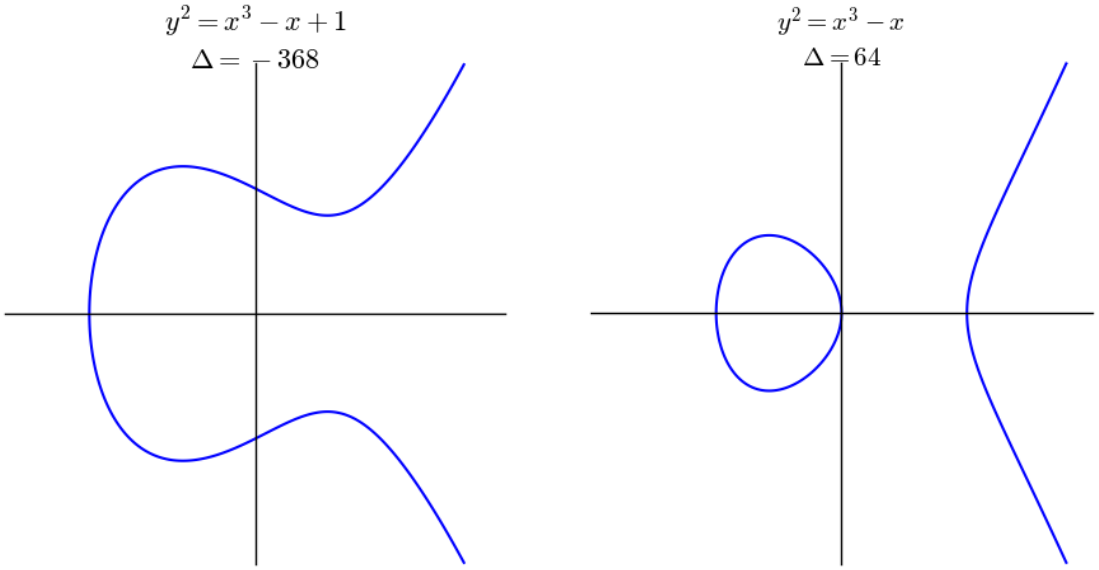
- *A la ecuación (3.1) la llamaremos ecuación de Weierstrass.*

- *El número Δ es el discriminante de la ecuación de Weierstrass.*

- *La condición $\Delta \neq 0$ nos asegura que no existen puntos singulares.*

Ejemplo 3.3. *A continuación se muestran dos curvas elípticas reales junto a su gráfica (ver figura 3.2).*

$$E_1: \quad y^2 = x^3 - x + 1, \quad \text{con discriminante } \Delta = -368$$
$$E_2: \quad y^2 = x^3 - x, \quad \text{con discriminante } \Delta = 64.$$



(a) (b)

Figura 3.2: Dos curvas elípticas sobre $\mathbb{R}$.

3.3. Simplificación de la ecuación de Weierstrass

En la práctica rara vez se trabaja con curvas definidas de la forma 3.1, pues mediante cambios de variables es posible obtener ecuaciones simplificadas.

Definición 3.4. *Sean E_1 y E_2 dos curvas elípticas definidas sobre K un cuerpo con ecuaciones:*

$$E_1 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$
$$E_2 : y^2 + b_1xy + b_3y = x^3 + b_2x^2 + b_4x + b_6.$$

*Diremos que son **isomorfas** sobre K si existen $u, r, s, t \in K$, con $u \neq 0$, tales que el cambio de variables*

$$(x, y) \to (u^2x + r, u^3y + u^2sx + t)$$

transforma la ecuación E_1 en E_2. Se llama ***cambio admisible de variables*** *a la transformación anterior.*

Según la característica del cuerpo K podemos distinguir tres casos.

1. Si la característica del cuerpo K es distinta de 2 o de 3, entonces el cambio admisible de variables

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1 x}{216} - \frac{a_1^3 + 4a_1 a_2 - 12a_3}{24} \right)$$

transforma E en la curva elíptica

$$y^2 = x^3 + ax + b \tag{3.3}$$

con $a, b \in K$. El discriminante de la curva pasa a ser $\Delta = -16(4a^3 + 27b^2)$.

2. Si la característica de K es 2, tenemos que distinguir dos casos . Si $a_1 \neq 0$, entonces el cambio admisible de variables

$$(x, y) \to \left(a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3} \right)$$

transforma E en la curva

$$y^2 + xy = x^3 + ax^2 + b \tag{3.4}$$

con $a, b \in K$. Diremos que este tipo de curvas son **no supersingulares** u **ordinarias**, su discriminante es $\Delta = b$.

Si $a_1 = 0$, entonces el cambio admisible de variables

$$(x, y) \to (x + a_2, y)$$

transforma E en la curva

$$y^2 + cy = x^3 + ax + b \tag{3.5}$$

con $a, b, c \in K$. Este tipo de curvas se denotan curvas **supersingulares**, su discriminante es $\Delta = c^4$.

3. Si la característica de K es 3, volvemos a distinguir dos casos. Si $a_1^2 \neq -a_2$, entonces el cambio admisible de variables

$$(x, y) \to \left(x + \frac{d_4}{d_2}, y + a_1 x + a_1 \frac{d_4}{d_2} + a_3 \right),$$

con $d_2 = a_1^2 + a_2$ y $d_4 = a_4 - a_1 a_3$, transforma E en la curva

$$y^2 = x^3 + ax^2 + b \tag{3.6}$$

con $a, b \in K$. De nuevo diremos que este tipo de curvas son **no supersingulares** u **ordinarias**, su discriminante es $\Delta = -a^3 b$.

Si $a_1^2 = -a_2$, entonces el cambio admisible de variables

$$(x, y) \to (x, y + a_1 x + a_3)$$

transforma E en la curva

$$y^2 = x^3 + ax + b \tag{3.7}$$

con $a, b \in K$. Este tipo de curvas se denotan curvas **supersingulares**, su discriminante es $\Delta = -a^3$.

3.4. Curvas elípticas sobre los números reales

Antes de trabajar sobre cuerpos finitos, hablaremos sobre la estructura de grupo abeliano que siguen los puntos de una curva elíptica. Para poder visualizarlo, será necesario trabajar sobre $K = \mathbb{R}$, donde la curva elíptica se representa como una curva en el plano, como se muestra en la Figura 3.2.

Sea E una curva elíptica dada por su ecuación de Weierstrass $y^2 = x^3 + ax + b$. La operación binaria que define dicha estructura de grupo se basa en la siguiente regla geométrica:

Tres puntos alineados sobre una curva elíptica suman cero, entendiendo como cero el punto del infinito.

Así, podremos definir a partir de P y Q, dos puntos de la curva E, el elemento opuesto de P, es decir, $-P$ y la suma $P + Q$. Distinguiremos los siguientes casos:

1. Si $P = O$ el punto del infinito, entonces $-P$ será también O y $P + Q$ será Q, pues la recta formada por los puntos P y Q interseca a la curva en el punto $-Q$, tenemos por tanto que $O + Q + (-Q) = O$, o equivalentemente, $O + Q = Q$. Por tanto O actúa como elemento neutro.

2. Supongamos a partir de ahora que ni P ni Q son el punto del infinito. El opuesto de P es aquel con la misma coordenada x y el opuesto en la coordenada y, es decir, si tomamos $P = (x, y)$, entonces $-P = -(x, y) = (x, -y)$, que viendo la forma de la ecuación de Weierstrass es evidente que $-P$ esta en la curva elíptica.
   Veamos que el resultado es cierto, sea $Q = (x, -y)$ la recta formada por los puntos P y Q es la recta vertical, y como toda recta vertical interseca a O (ver sección 3.5), por lo tanto $P + Q + O = O$, que significa que $P + Q = O$ y por tanto $Q = -P$.

3. Si P y Q tienen diferente coordenada x entonces la recta formada por ambos puntos interseca en un tercer punto R de la curva elíptica. $P + Q$ será $-R$ como podemos deducir de la igualdad $P + Q + R = O$. Este caso se muestra en la Figura 3.3.

4. Finalmente, si $P = Q$ tomaremos la recta tangente a la curva en el punto P y R el único punto de la recta tangente que interseca a la curva elíptica, así $P + Q = 2P$ será $-R$.

La Figura 3.3 situada a la derecha muestra gráficamente cómo funciona la suma de dos puntos sobre la curva $y^2 = x^3 + 2x + 3$. Dados dos puntos $P = (-1, 0)$ y $Q = (\frac{1}{4}, \frac{15}{8})$, se representa la recta que pasa por esos dos puntos y se toma como R la intersección entre la curva elíptica y la recta, en este caso $R = (3, 6)$. Como $P+Q+R = O$ tenemos que $P+Q = -R$, es decir, $P + Q = (3, -6)$.

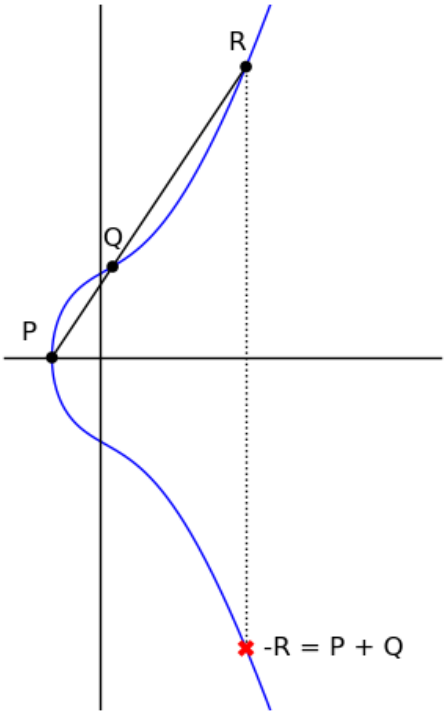


Figura 3.3: Suma de puntos sobre la curva elíptica.

Tras explicar las reglas que rigen la suma de puntos en una curva elíptica definida sobre $\mathbb{R}$ y visualizar gráficamente esta operación, vamos a desarrollar una fórmula que nos permita calcular R, el tercer punto de la curva elíptica que interseca a $\mathbf{r}$[1], y poder así deducir las coordenadas de $P + Q$.

Dados $P = (x_1, y_1), Q = (x_2, y_2)$ buscamos expresar las coordenadas $(x_3, y_3) = P + Q$ en función de x_1, y_1, x_2, y_2. Si nos encontramos en el caso donde P y Q tienen diferente coordenada x, e $y = \alpha x + \beta$ es la ecuación de la recta $\mathbf{r}$, entonces $\alpha = (y_2 - y_1)/(x_2 - x_1)$ y $\beta = y_1 - \alpha x_1$. Ambas igualdades se deducen fácilmente de las ecuaciones $y_1 = \alpha x_1 + \beta$ y $y_2 = \alpha x_2 + \beta$.

Un punto $(x, \alpha x + \beta)$ de $\mathbf{r}$ se encuentra en la curva elíptica si y solo si $(\alpha x + \beta)^2 = x^3 + ax + b$, por lo tanto, tenemos un punto intersección por cada raíz cúbica de $x^3 - (\alpha x + \beta)^2 + ax + b$. Sabiendo que x_1 y x_2 son dos raíces, pues $(x_1, \alpha x_1 + \beta)$ y $(x_2, \alpha x_2 + \beta)$

[1] $\mathbf{r}$ es la recta formada por los puntos P y Q.

son los puntos P y Q en la curva y aplicando una de las relaciones de Cardano-Vieta [2] deducimos que la tercera raíz es $x_3 = \alpha^2 - x_1 - x_2$. Así $P + Q = (x_3, y_3)$ tiene como coordenadas:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2,$$
$$y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3).$$

Con x_3 y y_3 en función de x_1, y_1, x_2, y_2. El caso $P = Q$ no dista del anterior, α se obtiene mediante diferenciación implícita de la ecuación de Weierstrass en P. Esto es:

$$2ydy = (3x^2 + a)dx$$

y despejando,

$$m = \frac{dy}{dx} = \frac{3x_1^2 + a}{2y_1}.$$

Obtenemos así que $P + Q = (x_3, y_3)$ tiene como coordenadas:

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1,$$
$$y_3 = -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3).$$

El uso de esta operación sobre el conjunto de los puntos racionales[3] (ver sección 3.6) de E da lugar a un grupo abeliano. Es decir, dados dos puntos de E somos capaces de obtener un nuevo punto racional en E verificando los axiomas siguientes:

---

[2]La relación de Cardano-Vieta establece que para un polinomio de grado n de la forma:

$$a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = 0$$

las raíces $r_1, r_2, \ldots, r_n$ satisfacen la siguiente relación para la suma de las raíces:

$$r_1 + r_2 + \cdots + r_n = -\frac{a_{n-1}}{a_n}$$

Dado que en nuestro caso el polinomio es mónico, la suma de sus raíces es $-a_{n-1}$.

[3]Los puntos racionales de una curva elíptica son aquellos puntos cuyas coordenadas pertenecen a $\mathbb{Q}$

- $(P + Q) + R = P + (Q + R)$ para todos los puntos racionales P, Q, R en E.
- Existe un punto racional O [4] en E tal que $P + O = P$ y $O + P = P$ para todos los puntos racionales P en E.
- Para cualquier punto racional P en E, existe un punto Q (también llamado $-P$) tal que $P + Q = O$ y $Q + P = O$.
- $P + Q = Q + P$ para todos los puntos racionales P y Q en E.

Notación: Sea $n \in \mathbb{Z}$, diremos que nP es P sumado consigo mismo n veces si n es positivo, y $-P$ sumado consigo mismo $|n|$ veces si n es negativo.

El punto del infinito O es el **elemento neutro** del grupo abeliano de la curva elíptica. Para visualizar de manera más efectiva este punto, vamos a introducir las **coordenadas proyectivas** sobre un cuerpo K.

3.5. Coordenadas proyectivas

Sea K un cuerpo y sean c y d dos números naturales. Definimos una relación de equivalencia $\sim$ en el conjunto $K^3 \setminus \{(0,0,0)\}$ de la siguiente manera:

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \text{ si } X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2, Z_1 = \lambda Z_2 \text{ para algún } \lambda \in K \setminus \{0\}.$$

Denotaremos a la clase de equivalencia que contiene $(X, Y, Z) \in K^3 \setminus \{(0,0,0)\}$ como:

$$(X : Y : Z) = \left\{ \left(\lambda^c X, \lambda^d Y, \lambda Z\right) : \lambda \in K \setminus \{0\} \right\}$$

[4] El punto del infinito ya definido anteriormente.

Llamaremos **punto proyectivo** a $(X : Y : Z)$ y (X, Y, Z) será un **representante** [5] de la clase de equivalencia. $\mathbb{P}(K)$ es el conjunto de todos los **puntos proyectivos**. Si $Z \neq 0$ existe un único representante de la clase de equivalencia con $Z = 1$. Dicho de otra forma, existe una correspondencia biyectiva entre el conjunto

$$\mathbb{P}(K)^* = \{(X : Y : Z) : X, Y, Z \in K, Z \neq 0\}$$

y el conjunto de los **puntos afines**

$$\mathbb{A}(K) = \{(x, y) : x, y \in K\}.$$

El único representante de la clase $(X : Y : Z)$ de equivalencia con $Z = 1$ es $(X/Z^c, Y/Z^d, 1)$. El conjunto de los puntos proyectivos

$$\mathbb{P}(K)^0 = \{(X : Y : Z) : X, Y, Z \in K, Z = 0\}$$

representan la **recta en el infinito**. Trabajemos un ejemplo que nos permitirá ver la diferencia del punto del infinito O con el resto de puntos.

Ejemplo 3.4. *Fijamos $c = 1$ y $d = 1$[6]. La ecuación de la curva elíptica 3.1*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

definida sobre K tiene como ecuación en coordenadas proyectivas:

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3.$$

Si $Z = 0$, sustituyendo en la ecuación vemos que el único punto proyectivo que encontramos en la ecuación es el $(0 : 1 : 0)$, este punto será nuestro O ya definido anteriormente. Notar que es el único punto sobre E que tiene $Z = 0$, el resto de puntos proyectivos tendrá como coordenada Z al uno.

En la sección anterior se afirmó que la suma de los puntos $P = (x, y)$ y $-P = (x, -y)$ de una curva elíptica E es O, el punto del infinito. Esta afirmación se justifica porque la recta que pasa por ambos puntos es vertical, y en el contexto de curvas elípticas, toda recta

---

[5] Cualquier elemento de la clase de equivalencia puede actuar como representante, pues si $(\tilde{X}, \tilde{Y}, \tilde{Z}) \in (X : Y : Z)$ entonces $(\tilde{X} : \tilde{Y} : \tilde{Z}) = (X : Y : Z)$

[6] Las coordenadas con estos valores de c y d se conocen como **coordenadas proyectivas estándar.**

vertical se considera que interseca al punto en el infinito. Esta interpretación se formaliza al trabajar con coordenadas proyectivas.

En coordenadas proyectivas, los puntos P y $-P$ se expresan como $P = (X : Y : Z)$ y $-P = (X : -Y : Z)$, respectivamente. Al aplicar el algoritmo de suma de puntos en coordenadas proyectivas (ver (6)), se obtiene que:

$$P + (-P) = (X_3 : Y_3 : Z_3) = (0 : 8Y^3Z^5 : 0)$$

Y como todo punto con $Z = 0$ en coordenadas proyectivas se encuentra en el infinito. Entonces:

$$P + (-P) = (0 : 1 : 0) = O$$

Nota 3.5. *Otro ejemplo de coordenadas proyectivas son las coordenadas de Jacobi, en las cuales $c = 2$ y $d = 3$. El punto O bajo estas coordenadas es $(1 : 1 : 0)$.*

El propio programa de `SageMath`[7] utiliza este tipo de coordenadas cuando trabajamos sobre cuerpos finitos, pues la suma de puntos en coordenadas afines implica el cálculo de inversos modulares para elementos del cuerpo finito sobre el cual está definida la curva. La inversión modular puede realizarse utilizando el algoritmo extendido de Euclides, aunque su costo es aproximadamente 100 veces mayor que el de una simple suma o multiplicación en el cuerpo finito.

Sin embargo, la suma de puntos en coordenadas proyectivas y jacobianas no requiere el cálculo de ningún inverso modular. Por lo tanto, el costo de sumar dos puntos en una curva elíptica utilizando coordenadas proyectivas o jacobianas es significativamente menor que en la representación afín.

Ejemplo 3.6. *A continuación, se presentará el algoritmo para la duplicación de puntos en coordenadas de Jacobi. Sea $P = (X : Y : Z)$, un punto de la curva elíptica E con ecuación de Weierstrass $y^2 = x^3 + ax + b$. Expresando el punto P como $P = \left(\frac{X}{Z^2} : \frac{Y}{Z^3} : 1\right)$ se puede utilizar la fórmula (ver sección 3.4) para obtener $2P$ en coordenadas afines, esto es:*

---

[7] `SageMath`, conocido anteriormente como Sage, es un sistema algebraico computacional (en inglés CAS) que destaca por estar construido sobre paquetes matemáticos ya contrastados como NumPy, Sympy, PARI/GP o Maxima y por acceder a sus potencias combinadas a través de un lenguaje común basado en Python. Texto extraído de Wikipedia, disponible en https://es.wikipedia.org/wiki/SageMath.

$$X_2 = \left(\frac{3\left(\frac{X}{Z^2}\right)^2 + a}{2\left(\frac{Y}{Z^3}\right)}\right)^2 - 2\frac{X}{Z^2} = \frac{\left(3\left(\frac{X^2}{Z^4}\right) + a\right)Z^8 - 8XY^2}{4Y^2Z^2} = \frac{(3X^2 + aZ^4)^2 - 8XY^2}{4Y^2Z^2}$$

$$Y_2 = -\frac{Y}{Z^3} + \left(\frac{3\left(\frac{X}{Z^2}\right)^2 + a}{2\left(\frac{Y}{Z^3}\right)}\right)\left(\frac{X}{Z^2} - X_2\right) = \frac{3X^2 + aZ^4}{2YZ}\left(\frac{X}{Z^2} - X_2\right) - \frac{Y}{Z^3}.$$

Para eliminar los denominadores y expresar $2P$ en coordenadas de Jacobi, se define $X_3 = X_2Z_3^2$ y $Y_3 = Y_2Z_3^3$ donde $Z_3 = 2YZ$. Así, obtenemos que $2P$ en coordenadas de Jacobi es $2P = (X_3 : Y_3 : Z_3)$ con:

$$\begin{aligned} X_3 &= \left(3X^2 + aZ^4\right)^2 - 8XY^2 \\ Y_3 &= \left(3X^2 + aZ^4\right)\left(4XY^2 - X_3\right) - 8Y^4 \\ Z_3 &= 2YZ. \end{aligned}$$

En capítulos posteriores, más concretamente en el capítulo 5, sección 5.3, se verá que las curvas elípticas utilizadas en criptografía suelen emplear como parámetro $a = -3$. Bajo esta condición, al calcular X_3 en el algoritmo de duplicación de puntos, la expresión $(3X^2 - 3Z^4)$ se puede factorizar como $3(X - Z^2)(X + Z^2)$. Esta factorización permite reducir el número de operaciones requeridas en el algoritmo: de cuatro multiplicaciones y seis elevaciones al cuadrado a cuatro multiplicaciones y cuatro elevaciones al cuadrado.

3.6. Curvas elípticas sobre los números racionales

El estudio de las curvas elípticas sobre $\mathbb{Q}$ ha dado lugar a numerosos resultados fundamentales. A lo largo de esta sección, presentaremos algunos de los más importantes.

El grupo $E(\mathbb{Q})$ puede tener un número finito o infinito de elementos, pero podemos asegurar que el grupo es generado por un número finito de elementos.

Teorema 3.7 (Mordell, 1922)**.** *Sea E una curva elíptica definida sobre $\mathbb{Q}$, dada por una ecuación de la forma*

$$E : y^2 = x^3 + ax + b, \quad \text{con } a, b \in \mathbb{Q}.$$

Entonces, el grupo de los puntos racionales $E(\mathbb{Q})$ es un grupo abeliano finitamente generado, es decir,

$$E(\mathbb{Q}) \cong \mathbb{Z}^r \oplus T,$$

*donde T es un subgrupo abeliano finito (llamado el **subgrupo de torsión** de $E(\mathbb{Q})$) y r es un número entero no negativo (llamado el **rango** de E).*

La estructura del subgrupo de torsión es también conocida.

Teorema 3.8 (Mazur, 1977)**.** *Sea E una curva elíptica definida sobre $\mathbb{Q}$. Entonces, el grupo de torsión T de $E(\mathbb{Q})$ es isomorfo a uno de los siguientes grupos abelianos finitos:*

- *Un grupo cíclico de orden N, con $1 \leq N \leq 10$ ó $N = 12$, es decir,*

$$\mathbb{Z}/N\mathbb{Z}, \quad \text{para } N \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12\}.$$

- *Un grupo producto de dos grupos cíclicos de la forma $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/2N\mathbb{Z}$, con $1 \leq N \leq 4$, es decir,*

$$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2N\mathbb{Z}, \quad \text{para } N \in \{1, 2, 3, 4\}.$$

No existen otros grupos de torsión posibles para curvas elípticas sobre $\mathbb{Q}$.

El **rango** de una curva representa el número de puntos independientes con orden infinito, es decir, cuántos números racionales existen que me permiten generar infinitos puntos racionales sobre la curva E. Veamos ejemplificados algunos casos.

Ejemplo 3.9. *Trabajaremos primero sobre la curva:*

$$E_1 : y^2 = x^3 - x$$

Para ello usaremos las siguientes líneas de código en `SageMath`*:*

```
Entrada: E = EllipticCurve([-1, 0]) # Definimos la curva
         E.torsion_points()  # Calculamos los puntos de torsión
         E.rank()   # Calculamos su rango
```

```
Salida: [(0 : 1 : 0), (-1 : 0 : 1), (0 : 0 : 1), (1 : 0 : 1)]
        0
```

Estamos ante una curva con rango 0*, lo que implica que no existen puntos racionales con orden infinito; todos los puntos son de torsión. El código empleado nos permite saber que los puntos de torsión son* $P = (-1, 0)$*,* $Q = (0, 0)$*,* $R = (1, 0)$ *y* O *(el punto del infinito).*

Para determinar la estructura de T*, el* ***subgrupo de torsión****, observamos que contiene exactamente cuatro elementos. Según el* ***teorema de Mazur*** *(ver Teorema 3.8), un subgrupo de torsión de orden* 4 *en una curva elíptica sobre* $\mathbb{Q}$ *puede ser isomorfo a* $\mathbb{Z}/4\mathbb{Z}$ *o a* $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ *(el grupo de Klein). Apoyándonos en el código*

```
Entrada: P=E(-1,0),Q=E(0,0),R=E(1,0) #Definimos sobre la curva los
         #puntos P,Q y R
         P+P, Q+Q, R+R  # Comprobamos que cada punto es su propio inverso
         P+Q, P+R, Q+R  # Verificamos la suma entre puntos distintos
```

```
Salida: ((0 : 1 : 0), (0 : 1 : 0), (0 : 1 : 0),
        (1 : 0 : 1), (0 : 0 : 1), (-1 : 0 : 1))
```

podemos confirmar que nos encontramos en el segundo caso, pues:

- *Cada elemento distinto de la identidad es su propio inverso.*
- *La suma de dos puntos distintos a la identidad da lugar al tercero.*
- *El teorema 3.7 nos asegura que el subgrupo de torsión es abeliano.*

Conclusión, el subgrupo de torsión T *es isomorfo al* ***grupo de Klein*** $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$*. En la Figura 3.4 podemos encontrar los puntos* $P = (-1, 0)$*,* $Q = (0, 0)$ *y* $R = (1, 0)$ *representados sobre* E_1 *definida sobre los números reales y en la Figura 3.5 se encuentra la representación del subgrupo de torsión de* E_1 *sobre los racionales.*

Ejemplo 3.10. *Buscando una curva elíptica con* ***rango*** *no nulo encontramos la curva*

$$E_2 : y^2 = x^3 - 200x + 211$$

Y usando las siguientes líneas de código:

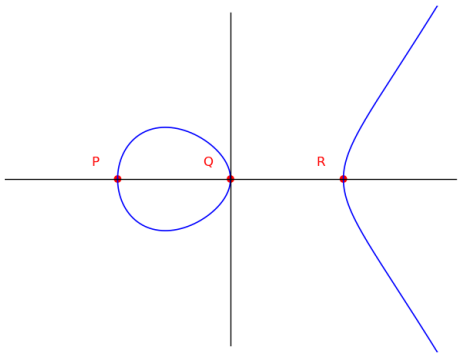


Figura 3.4: Representación de E_1 sobre $\mathbb{R}$.

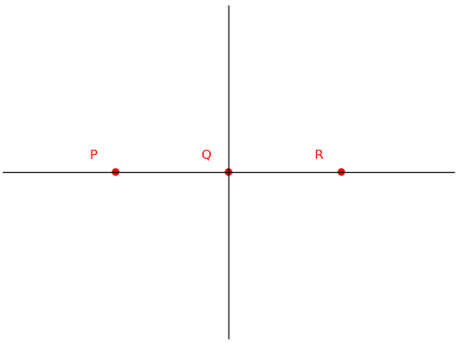


Figura 3.5: Subgrupo de torsión sobre $\mathbb{Q}$.

```
Entrada: E = EllipticCurve([-200, 211]) # Definimos la curva
         E.torsion_points()  # Calculamos los puntos de torsión
         E.rank()   # Calculamos su rango
         E.gens()  # generadores de puntos con orden infinito
```

```
Salida: [(0 : 1 : 0)]
         2
        [(-3 : 28 : 1), (73/4 : 411/8 : 1)]
```

Vemos que la curva tiene ***rango*** 2*, su único punto de torsión es* O *el punto del infinito y tiene como generadores de puntos con orden infinito a* $P = (-3, 28)$ *y a* $Q = (\frac{73}{4}, \frac{411}{8})$*. Si tomamos como generador el punto* $P = (-3, 28)$ *obtenemos:*

$$2P = \left(\frac{48745}{3136}, \frac{5143221}{175616}\right)$$

$$3P = \left(-\frac{42403596339}{3381771409}, -\frac{5376314620080460}{196660152747577}\right)$$

3.7. Curvas elípticas sobre cuerpos finitos

A lo largo de esta sección trabajaremos con K un cuerpo finito $\mathbb{F}_q$ con $q = p^r$[8].Sea E una curva elíptica definida sobre $\mathbb{F}_q$ queremos, como mínimo, aproximar el número de puntos de $E(\mathbb{F}_q)$, es decir, cuántos pares $(x, y) \in \mathbb{F}_q^2$ son solución de la ecuación E.

Nota 3.11. *Recordar que si la característica de K es 2 ó 3, entonces la ecuación simplificada de Weierstrass tiene que ser de la forma 3.4, 3.5, 3.6 ó 3.7.*

Como cada valor de $x \in \mathbb{F}_q$ da lugar como máximo a dos puntos $y \in \mathbb{F}_q$, una cota superior es

$$\#E(\mathbb{F}_q) \leq 2q + 1$$

Sin embargo, dado que una ecuación cuadrática escogida de manera aleatoria tiene, aproximadamente, un 50 % de probabilidad de tener solución (dos valores de y) para cada elemento de $\mathbb{F}_q$ (ver anexo A), podemos esperar que $\#E(\mathbb{F}_q)$ esté próximo a q. Pese a que el número exacto de puntos puede variar, el teorema de Hasse proporciona un intervalo, llamado **intervalo de Hasse** o $\mathcal{H}(q)$, que limita el rango de puntos que podemos encontrar en E sobre $\mathbb{F}_q$.

Teorema 3.12 (Hasse)**.** *Sea E una curva elíptica definida sobre un cuerpo finito $\mathbb{F}_q$ con q elementos. Entonces, el número de puntos de E sobre $\mathbb{F}_q$, denotado como $\#E(\mathbb{F}_q)$, satisface la siguiente desigualdad:*

$$|\#E(\mathbb{F}_q) - (q + 1)| \leq 2\sqrt{q}.$$

Además, podemos describir la estructura de grupo que tiene $E(\mathbb{F}_q)$.

Teorema 3.13. *Sea E una curva elíptica definida sobre $\mathbb{F}_q$. Entonces, $E(\mathbb{F}_q)$ es isomorfo a $\mathbb{Z}/N_1\mathbb{Z} \times \mathbb{Z}/N_2\mathbb{Z}$, donde N_1 y N_2 son enteros positivos determinados de manera única, tales que N_2 divide a N_1 y a $q - 1$.*

Ejemplo 3.14. *Sea $q = p = 23$ y $E : y^2 = x^3 + x - 1$ la curva elíptica definida sobre $\mathbb{F}_{23}$. Se comprueba fácilmente con Sage que $\#E(\mathbb{F}_{23}) = 20$ y que, por lo tanto, nos encontramos dentro del intervalo de Hasse.*

[8] p siempre será un número primo.

Sea $P = (1, 1)$, punto generador aleatorio de la curva. Comprobando que su orden es 20, podemos afirmar que a partir de P podremos generar todos los puntos de la curva E sobre $\mathbb{F}_{23}$. Aplicando el teorema 3.13 entonces $E(\mathbb{F}_{23})$ es isomorfo al grupo cíclico $\mathbb{Z}/20\mathbb{Z}$.

No es cierto que cualquier punto genere a todos los restantes de la curva elíptica. Buscando entre los puntos encontramos el $Q = (8, 6)$ que tiene orden 5 y genera únicamente los puntos en rojo de la Figura 3.6. A partir de P generamos todos los puntos de $E(\mathbb{F}_{23})$,

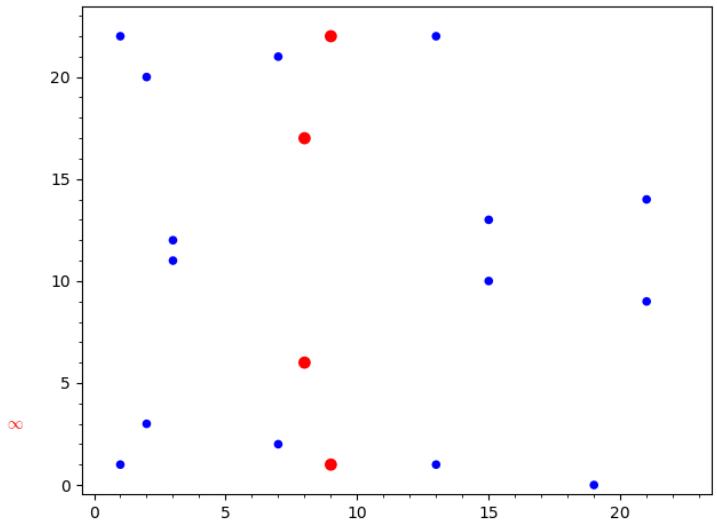


Figura 3.6: Representación de curva elíptica $y^2 = x^3 + x - 1$ sobre $\mathbb{F}_{23}$.

que son:

$0P = \infty$	$5P = (3, 11)$	$10P = (19, 0)$	$15P = (3, 12)$
$1P = (1, 1)$	$6P = (21, 14)$	$11P = (15, 10)$	$16P = (9, 1)$
$2P = (2, 20)$	$7P = (7, 2)$	$12P = (8, 6)$	$17P = (13, 22)$
$3P = (13, 1)$	$8P = (8, 17)$	$13P = (7, 21)$	$18P = (2, 3)$
$4P = (9, 22)$	$9P = (15, 13)$	$14P = (21, 9)$	$19P = (1, 22)$

Todos estos puntos se encuentran representados en la figura 3.6.

Resulta interesante ver alguna curva con la que se trabaja en la realidad, así podemos ver con qué parámetros nos encontramos en curvas usadas en criptografía.

Ejemplo 3.15. *La curva secp256k1, definida sobre el cuerpo finito* $\mathbb{F}_p$ *con:*

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

y $p = 2^{256} - 2^{32} - 977$ *número primo, es una curva usada en el mundo tecnológico; destaca por su eficiencia y seguridad en sus aplicaciones criptográficas. Es usada para validar direcciones y transacciones de criptomonedas, o en sistemas de votación digital, para garantizar la seguridad y autenticidad de los votos.*

3.7.1. Extensiones de cuerpos finitos y el teorema de Weil

Finalmente, trabajemos sobre extensiones de cuerpos finitos y la conjetura de Weil[9]. Sea E una curva elíptica sobre $\mathbb{F}_p$, es evidente que E está también definida sobre $\mathbb{F}_{p^r}$ $\forall r \in \mathbb{N}$, por ello, se pueden estudiar los puntos en $\mathbb{F}_{p^r}$ que son solución de la curva elíptica E. Es decir, los puntos $\mathbb{F}_{p^r}$ racionales.

Denotaremos por $N = N_1$ al número de puntos de E sobre $\mathbb{F}_p$ y, de manera análoga, por N_r al número de puntos de E sobre $\mathbb{F}_{p^r}$.

A partir de la serie de los posibles N_r creamos la función generadora[10] (conocida como función zeta local[11]) $Z(T, E/\mathbb{F}_p)$ sobre $\mathbb{Q}[[T]]$, definida por:

$$Z(T, E/\mathbb{F}_p) = e^{\sum_{r=1}^{\infty} \frac{N_r T^r}{r}}, \tag{3.8}$$

con T la indeterminada.

---

[9] André Weil, en 1949, propuso una serie de conjeturas relativas al número de puntos en variedades algebraicas definidas sobre cuerpos finitos. Entre las variedades algebraicas encontramos a las curvas elípticas.

[10] Se dice función generadora a la función que representa una sucesión de números mediante una serie de potencias.

[11] Se dice función zeta local a la función cuya derivada logarítmica es una función generadora.

Teorema 3.16 (Teorema de Weil para curvas elípticas)**.** *La función zeta local es una función racional de T con la siguiente expresión:*

$$Z\left(T, E/\mathbb{F}_q\right) = \frac{1 - aT + qT^2}{(1-T)(1-qT)} \tag{3.9}$$

donde solo el entero a depende de la curva elíptica particular E. El valor a está relacionado con $N = N_1$ de la siguiente manera: $N = q+1-a$. Además, el discriminante del polinomio cuadrático del numerador es negativo (es decir, $a^2 < 4q$, de donde deducimos el Teorema de Hasse) y, por lo tanto, el polinomio cuadrático tiene dos raíces complejas conjugadas α, β, ambas de valor absoluto $\sqrt{q}$.

Demostración. La demostración completa se puede consultar en (17), páginas 140-144. □

Corolario 3.17. *Conocido $N_1 = N$, el número de puntos de E sobre $\mathbb{F}_q$, podemos determinar el número de puntos sobre cualquier cuerpo extensión de $\mathbb{F}_q$ a partir de la relación*

$$N_r = q^r + 1 - \alpha^r - \beta^r, \quad \forall r \in \mathbb{N}.$$

Demostración. Partimos de la ecuación (3.9) y escribimos el numerador como $(1-\alpha T)(1-\beta T)$, sabemos por el teorema anterior que tanto α como β son raíces del polinomio. Por lo tanto, tenemos la siguiente expresión:

$$Z\left(T, E/\mathbb{F}_q\right) = \frac{1 - aT + qT^2}{(1-T)(1-qT)}.$$

Además, tenemos la expresión de la función zeta de (3.8).

$$\frac{1 - aT + qT^2}{(1-T)(1-qT)} = e^{\sum_{r=1}^{\infty} \frac{N_r T^r}{r}}.$$

Tomando el logaritmo en ambos lados y desarrollando la expresión, deducimos la igualdad:

$$\sum_{r=1}^{\infty} \frac{N_r T^r}{r} = \log(1-\alpha T) + \log(1-\beta T) - \log(1-T) - \log(1-qT).$$

Por el desarrollo de la serie de Taylor sabemos que $\log(1-cT) = -\sum_{r=1}^{\infty} \frac{c^r T^r}{r}$, entonces

$$\sum_{r=1}^{\infty} \frac{N_r T^r}{r} = -\sum_{r=1}^{\infty} \frac{\alpha^r T^r}{r} - \sum_{r=1}^{\infty} \frac{\beta^r T^r}{r} + \sum_{r=1}^{\infty} \frac{T^r}{r} + \sum_{r=1}^{\infty} \frac{q^r T^r}{r},$$

y por tanto $\forall r \in \mathbb{N}$ se cumple:

$$N_r = q^r + 1 - \alpha^r - \beta^r.$$

□

Ejemplo 3.18. *Vamos a trabajar la función zeta de la curva elíptica*

$$E : y^2 + y = x^3 - x + 3$$

sobre $\mathbb{F}_2$.

Teniendo en cuenta que $\#E(\mathbb{F}_2) = 1$, *y aplicando el teorema 3.16 es claro que:*

$$1 = N = 2 + 1 - a \Rightarrow a = 2.$$

Sustituyendo en la ecuación (3.9):

$$Z(T, E/\mathbb{F}_2) = \frac{1 - 2T + 2T^2}{(1-T)(1-2T)}.$$

Despejando en el numerador obtenemos como raíces $\frac{1}{2} \pm \frac{1}{2}i$.

Es decir[12],

$$\begin{cases} \frac{1}{\alpha} = \frac{1}{2} + \frac{1}{2}i & \Rightarrow \alpha = (1-i) \\ \frac{1}{\beta} = \frac{1}{2} - \frac{1}{2}i & \Rightarrow \beta = (1+i). \end{cases}$$

[12]Se aplica que el inverso de un número complejo $z = a + bi$ es $z^{-1} = (a - bi)/(a^2 + b^2)$

Por lo tanto,

$$N_r = 2^r + 1 - \alpha^r - \beta^r = 2^r + 1 - (1-i)^r - (1+i)^r.$$

Con el código siguiente se prueba que los cálculos son correctos. Primero calculamos los puntos mediante la función proporcionada por **Sage.**

```
# Definir el cuerpo finito F(2^7)
F.<a> = GF(2^7, modulus='primitive')

# Definir la curva elíptica y^2 + y = x^3 - x + 1 sobre F(2^7)
E = EllipticCurve(F, [0, 0, 1, -1, 1])

# Calcular el número de puntos (incluyendo el punto al infinito)
numero_puntos = E.cardinality()

print(f"Número de puntos en la curva: {numero_puntos}")
Número de puntos en la curva: 113
```

Y lo comparamos con nuestro N_7 *(se ha escogido* $r = 7$*, pero se puede tomar cualquier* $r \in \mathbb{N}$*).*

```
# Definir los números complejos
z = (1 - 1*i);
w = (1 + 1*i);

# Definir el exponente
r = 7;

#Calcular el resultado
resultado = 2^r + 1 - z^r - w^r;
print(f"Número de puntos en la curva: {resultado}")
Número de puntos en la curva: 113
```

Capítulo 4

Algoritmos para el conteo de puntos en curvas elípticas sobre cuerpos finitos

4.1. Introducción

A lo largo de esta sección se desarrollarán los algoritmos para el conteo de puntos de una curva elíptica sobre un cuerpo finito, se comenzará presentando los métodos más sencillos, hasta llegar al algoritmo de Schoof, el primer algoritmo capaz de calcular $\#E(\mathbb{F}_q)$ en tiempo polinómico.

Sea E una curva elíptica definida sobre un cuerpo finito $\mathbb{F}_q$, con $q = p^r$, donde p es un número primo y $r \in \mathbb{N}$, y cuya ecuación de Weierstrass es $y^2 = x^3 + ax + b$. Calcular $\#E(\mathbb{F}_q)$, el número de puntos de la curva elíptica sobre $\mathbb{F}_q$, es imprescindible para los algoritmos presentados en los capítulos 5 y 6. Cada $x \in \mathbb{F}_q$ tiene $0, 1$ ó 2 posibles soluciones[1] en E. Más concretamente, hay $1 + \left(\frac{x^3+ax+b}{\mathbb{F}_q}\right)$ soluciones para cada $x \in \mathbb{F}_q$,

[1]Las posibles soluciones para cada $x \in \mathbb{F}_q$ son los puntos del conjunto $E(\mathbb{F}_q)$ cuya primera coordenada toma el valor x.

donde $\left(\frac{\cdot}{\mathbb{F}_q}\right)$ es la extensión del símbolo de Legendre (ver anexo A) para $\mathbb{F}_q$. Esto es:

$$\left(\frac{a}{\mathbb{F}_q}\right) = \begin{cases} 1 & \text{si } x^2 = a \text{ tiene dos soluciones en } \mathbb{F}_q, \\ -1 & \text{si } x^2 = a \text{ no tiene solución en } \mathbb{F}_q, \\ 0 & \text{si } x^2 = a \text{ tiene una única solución en } \mathbb{F}_q. \end{cases}$$

Entonces,

$$\#E(\mathbb{F}_q) = 1 + \sum_{x \in \mathbb{F}_q} \left(1 + \left(\frac{x^3 + ax + b}{\mathbb{F}_q}\right)\right) = 1 + q + \sum_{x \in \mathbb{F}_q} \left(\frac{x^3 + ax + b}{\mathbb{F}_q}\right).$$

Por lo tanto, el problema de calcular $\#E(\mathbb{F}_q)$ es equivalente a calcular $\sum_{x \in \mathbb{F}_q} \left(\frac{x^3+ax+b}{\mathbb{F}_q}\right)$. Sin embargo, el cálculo directo del sumatorio anterior en la práctica (para valores de q grandes) es ineficiente, además, no nos aprovechamos del teorema de Hasse (ver teorema 3.12), el cual nos da un intervalo de longitud $4\sqrt{q}$ en el que se encuentra $\#E(\mathbb{F}_q)$.

La complejidad computacional de este método es $O(q^{1+\epsilon})$ para todo $\epsilon > 0$[2]. Por ejemplo, en el caso en que $q = p$ un número primo, el algoritmo tiene una complejidad computacional de $O(p \log p)$ donde p se corresponde al número de puntos $x \in \mathbb{F}_p$ sobre los que se itera, y $\log p$ refleja el coste de determinar, usando el símbolo de Legendre, si x es un residuo cuadrático.

La ineficiencia de este método motiva la búsqueda de nuevos métodos más eficientes para calcular $\#E(\mathbb{F}_q)$. Se comenzará presentando el algoritmo baby-step giant-step (también conocido como algoritmo de Shanks), que reduce la complejidad computacional del algoritmo anterior a $O(q^{1/4+\epsilon})$. A continuación, se estudiará el anillo de endomorfismos y, en particular, el endomorfismo de Frobenius, fundamentales para el desarrollo final del algoritmo de Schoof, un algoritmo capaz de calcular $\#E(\mathbb{F}_q)$ en tiempo polinómico.

---

[2] $O(q^{1+\epsilon})$ se presenta como una forma de indicar que el algoritmo no es estrictamente lineal, esto se debe a la necesidad de calcular la extensión del símbolo de Legendre $\left(\frac{\cdot}{\mathbb{F}_q}\right)$.

4.2. Algoritmo baby-step giant-step

Se sabe, por el teorema 3.12, que $\#E(\mathbb{F}_q) \in \left[q+1-2\sqrt{q}, q+1+2\sqrt{q}\right]$, por lo tanto, en este algoritmo vamos a buscar, a partir de $P=(x,y)$ un punto aleatorio[3] de la curva elíptica $E(\mathbb{F}_q)$, un $N \in \mathcal{H}(q)$[4] tal que $NP=O$. Si no existe otro $\tilde{N} \in \mathcal{H}(q)$ cumpliendo que $\tilde{N}P=O$, entonces $N=\#E(\mathbb{F}_q)$. Si existiesen $\tilde{N}$ y N tal que $NP=O=\tilde{N}P$, entonces, $(\tilde{N}-N)P=O$, y se sabría que d (el orden de P) divide a $\tilde{N}-N$. Sin embargo, no se conocería $\#E(\mathbb{F}_q)$. En esta situación basta con tomar un segundo punto P aleatorio y repetir el algoritmo que se va a presentar a continuación.

El algoritmo *baby-step giant-step* proporciona una manera de buscar el N. Se comienza tomando un entero s tal que $s \approx \sqrt[4]{q}$ y verificando que $s(2s+1) \geq 4\sqrt{q}$ (la longitud de $\mathcal{H}(q)$). A continuación, se definen los *baby-steps*, que son $P, 2P, \ldots, sP$. Notar que, tras este cálculo, conocemos $2s+1$ puntos de $E(\mathbb{F}_q)$, estos son: $O, \pm P, \pm 2P, \ldots, \pm sP$, pues el inverso de cada uno de estos puntos es el mismo cambiando el signo de la segunda coordenada.

A continuación, se calculan $Q=(2s+1)P$ y $R=(p+1)P$ y se definen los *giant-steps* como $R, R \pm Q, \ldots, R \pm tQ$, donde $t=\left[\frac{2\sqrt{q}}{2s+1}\right]$[5]. Para cada punto del *giant-step* $R+iQ$, con $i \in \{0, \pm 1, \pm 2, \ldots, \pm t\}$, se busca si existe un $j \in \{0, \pm 1, \pm 2, \ldots, \pm s\}$ tal que

$$R+iQ=jP \Leftrightarrow R+iQ-jP=O,$$

sustituyendo, y dado que $NP=O$, se concluye que

$$N=p+1+i(2s+1)-j.$$

La colisión $R+iQ=jP$ existe gracias al parámetro s seleccionado, pues cualquier entero de $\mathcal{H}(q)$ puede expresarse como $p+1+i(2s+1)-j$ para ciertos $i \in \{0, \pm 1, \pm 2, \ldots, \pm t\}$

---

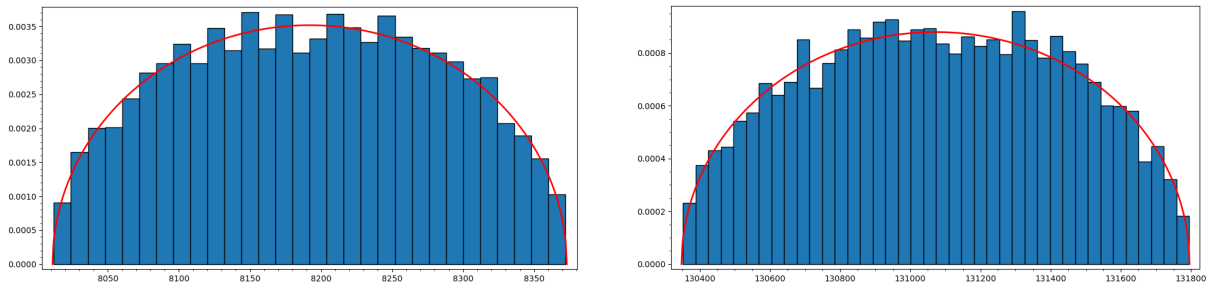
[3] El punto P se genera tomando, de manera aleatoria, un $x \in \mathbb{F}_q$ y comprobando si x^3+ax+b es un cuadrado en $\mathbb{F}_q$. El proceso se itera hasta encontrar un x válido.

[4] $\mathcal{H}(q)$ hace referencia a $\left[q+1-2\sqrt{q}, q+1+2\sqrt{q}\right]$ el intervalo de Hasse para el número q.

[5] $[x]$ hace referencia a la parte entera del número x.

y $j \in \{0, \pm 1, \pm 2, \ldots, \pm s\}$. Esto es posible porque s se elige suficientemente grande para que el número total de combinaciones posibles de i y j cubra todo el intervalo de Hasse. Los pasos grandes permiten recorrer $\mathcal{H}(q)$ en saltos de tamaño $2s + 1$, mientras que los pasos pequeños cubren los enteros dentro de cada salto.

Nota 4.1. *Notar que el algoritmo empieza a buscar las colisiones en el punto $R = (p+1)$, el punto intermedio de $\mathcal{H}(q)$, esto responde a una medida de optimización. La conjetura de Sato-Tate formula que, en promedio*[6]*, $\#E(\mathbb{F}_q)$ no se distribuye de manera uniforme sobre $\mathcal{H}(q)$, sino que sigue una distribución semicircular, siendo por lo tanto, los enteros más próximos a los extremos del intervalo los menos probables. En la Figura 4.1 se muestran las distribuciones de $\#E(\mathbb{F}_q)$ para $q = 8191$ y $q = 131071$, ambos números primos de Mersenne, a partir de la generación aleatoria de curvas elípticas.*



(a) Distribución para $\mathbb{F}_{8191}$

(b) Distribución para $\mathbb{F}_{131071}$

Figura 4.1: Distribución del número de puntos en curvas elípticas sobre diferentes cuerpos finitos.

El algoritmo reduce el coste computacional de calcular $\#E(\mathbb{F}_q)$ a $O(q^{1/4+\epsilon})$ para todo $\epsilon > 0$. Esto se debe a que el número de pasos necesarios es proporcional a $\sqrt[4]{q}$, tanto en la construcción de los *baby-steps* como en la búsqueda entre los *giant-steps*.

Sin embargo, el algoritmo tiene como condición clave que no exista otro $\tilde{N} \in \mathcal{H}(q)$ tal que $\tilde{N}P = O$. El problema fue abordado de manera elegante en el artículo original de René Schoof. El artículo puede consultarse en (16).

[6]En promedio significa que, aunque la conjetura podría no valer para cada curva elíptica individual, sí es cierta cuando se promedian los resultados sobre familias grandes de curvas.

4.3. Algoritmo de Schoof

A lo largo de esta sección se presentará el primer algoritmo de conteo de puntos con tiempo polinómico, el algoritmo de Schoof (presentado por René Schoof en 1985). Actualmente, el algoritmo de Schoof en su forma original no se utiliza en la práctica debido a su elevado coste computacional. En su lugar, se emplea una versión optimizada conocida como **Schoof-Elkies-Atkin (SEA)**, desarrollada por Atkin y Elkies a principios de los años 1990.

Primero se introducirán los conceptos teóricos necesarios para comprender el algoritmo, y posteriormente se describirá el propio algoritmo.

4.3.1. Endomorfismos

Definición 4.1. *Se define el endomorfismo de una curva elíptica E sobre $\mathbb{F}$ un cuerpo como la aplicación $\alpha\colon E(\bar{\mathbb{F}}) \to E(\bar{\mathbb{F}})$, con α definida por funciones racionales con coeficientes en $\bar{\mathbb{F}}$[7]. Respecto a la definición de endomorfismo, se cumple:*

1. *α está definida por funciones racionales: existen f, g funciones que se pueden expresar como cociente de dos polinomios con coeficientes en $\mathbb{F}$ tales que para todo $P = (x, y) \in E(\bar{\mathbb{F}})$,*
$$\alpha(P) = (f(x, y), g(x, y)) \in E(\bar{\mathbb{F}}).$$

2. *Preserva la operación de grupo, pues es un endomorfismo entre grupos:*
$$\alpha(P + Q) = \alpha(P) + \alpha(Q) \quad \forall P, Q \in E(\bar{\mathbb{F}}).$$

3. *Mapea el punto del infinito a sí mismo: $\alpha(O) = O$.*

[7] $\bar{\mathbb{F}}$ representa la clausura algebraica del cuerpo $\mathbb{F}$. Es decir, $\bar{\mathbb{F}}$ es el cuerpo más pequeño extensión de $\mathbb{F}$ en el cual se escinden todos los polinomios no constantes con coeficientes en $\mathbb{F}$.

Ejemplo 4.2. *Un caso de endomorfismo muy importante es el* ***endomorfismo producto****, denotado por* $[n]$*, donde* $n \in \mathbb{Z}$*.El endomorfismo asocia a cada punto* $P \in \bar{\mathbb{F}}$ *el punto* nP*, es decir:*

$$[n]P = \underbrace{P + P + \cdots + P}_{n \text{ veces}}.$$

La expresión del endomorfismo α puede generalizarse. Trabajando sobre E, una curva elíptica definida por su ecuación de Weierstrass simplificada (ver sección 3.3), dado que para todo $P \in E(\bar{\mathbb{F}})$ se tiene que $\alpha(P) \in E(\bar{\mathbb{F}})$, entonces, cualquier potencia de y de la forma y^{2n}, con $n \in \mathbb{N}$, puede sustituirse por un polinomio de x. Del mismo modo, cualquier potencia de y de la forma y^{2n+1}, con $n \in \mathbb{N}$, puede sustituirse por un polinomio de x multiplicado por y.

Además, dado que $\alpha(P) = \alpha(x, y) \in E(\bar{\mathbb{F}})$, se cumple (ver sección 3.4) que $-\alpha(x, y) = \alpha(-(x, y)) = \alpha(x, -y)$. De esta relación se deduce la siguiente proposición:

Proposición 4.3. *Sea* $\alpha \colon E(\bar{\mathbb{F}}) \to E(\bar{\mathbb{F}})$ *un endomorfismo de una curva elíptica* E *sobre el cuerpo* $\mathbb{F}$*. Entonces* α *es de la forma*

$$\alpha(P) = (f(x), y \cdot g(x)),$$

donde f *y* g *son funciones racionales.*

Notar entonces que $f(x) = \frac{f_1(x)}{f_2(x)}$, con $f_1(x)$ y $f_2(x)$ dos polinomios sin factores comunes y con coeficientes en $\mathbb{F}$. De esta expresión de f se deduce el grado de un endomorfismo.

Definición 4.2. *Sea* $\alpha \colon E(\bar{\mathbb{F}}) \to E(\bar{\mathbb{F}})$ *un endomorfismo de una curva elíptica* E *sobre el cuerpo* $\mathbb{F}$*. Se define el grado de* α *como el grado máximo entre los polinomios* $f_1(x)$ *y* $f_2(x)$*. Es decir,*

$$gr(\alpha) = \max\{gr(f_1(x)), gr(f_2(x))\}.$$

4.3.2. Anillo de endomorfismos

Definición 4.3. *Sea* E *una curva elíptica definida sobre un cuerpo* $\mathbb{F}$*, el anillo de los endomorfismos de* E*, denotado* $End(E)$*, es el conjunto de todos los endomorfismos de* E *definidos sobre* $\bar{\mathbb{F}}$*, con estructura de anillo inducida por la suma y cuya multiplicación se define como la composición de aplicaciones. Para* $\alpha, \beta \in End(E)$ *la suma se define como:*

$$(\alpha + \beta)(P) = \alpha(P) + \beta(P), \quad \forall P \in E(\bar{\mathbb{F}}).$$

Definición 4.4. *Dado n un número natural, se define el subgrupo de n-torsión como*

$$E[n] = \{P \in E(\bar{\mathbb{F}}) \mid nP = O\}.$$

Es decir, el núcleo del endomorfismo producto $[n]$.

En el anillo de endomorfismos es clave el concepto de $\hat{\alpha}$ el endomorfismo dual.

Teorema 4.4. *Para cualquier endomorfismo $\alpha\colon E(\bar{\mathbb{F}}) \to E(\bar{\mathbb{F}})$ existe un único endomorfismo $\hat{\alpha}\colon E(\bar{\mathbb{F}}) \to E(\bar{\mathbb{F}})$ tal que $\hat{\alpha}\circ\alpha = [n]$, donde $n = gr(\alpha)$. A $\hat{\alpha}$ se le llama endomorfismo dual.*

El endomorfismo dual $\hat{\alpha}$ permite definir una noción análoga a la traza, tal como ocurre en el caso de matrices. Esta analogía es esencial en el desarrollo del algoritmo de Schoof y en el estudio de la acción del endomorfismo de Frobenius (ver sección 4.3.2) sobre los puntos de E.

Definición 4.5. *El entero $tr(\alpha) = \alpha + \hat{\alpha}$, donde $\alpha + \hat{\alpha} = 1 + gr(\alpha) - gr(\alpha - 1)$, es la traza del endomorfismo α.*

Endomorfismo de Frobenius

Cuando se trabaja con una curva elíptica E definida sobre un cuerpo finito $\mathbb{F}_q$, el endomorfismo de Frobenius es un caso especial y fundamental dentro del anillo de endomorfismos de E.

Definición 4.6. *Sea E una curva elíptica definida sobre un cuerpo finito $\mathbb{F}_q$. Se define el endomorfismo de Frobenius como la aplicación*

$$\begin{aligned} \varphi_q\colon E(\bar{\mathbb{F}}_q) &\longrightarrow E(\bar{\mathbb{F}}_q) \\ (x, y) &\longmapsto (x^q, y^q). \end{aligned}$$

El endomorfismo de Frobenius es clave en el estudio del conjunto $E(\mathbb{F}_q)$.

Proposición 4.5. *El endomorfismo de Frobenius cumple que $\ker(\varphi_q - 1) = E(\mathbb{F}_q)$, además, $\#E(\mathbb{F}_q) = gr(\varphi_q - 1)$.*

Notar como se está relacionando el número de puntos de la curva elíptica con el endomorfismo de Frobenius. Por lo tanto, la desigualdad del teorema 3.12 se puede reformular. $\#E(\mathbb{F}_q)$ cumple que:

$$\#E(\mathbb{F}_q) = q + 1 - t, \text{ con } -2\sqrt{q} \leq t \leq 2\sqrt{q},$$

donde t es la traza del endomorfismo de Frobenius.

Además, t es el único entero para el cual la ecuación

$$\left(x^{q^2}, y^{q^2}\right) - t\left(x^q, y^q\right) + q\left(x, y\right) = O, \quad \forall P = (x, y) \in E(\bar{\mathbb{F}}_q).$$

Llamaremos ecuación característica de Frobenius a la ecuación anterior.

4.3.3. Polinomios de división

Sea $P = (x : y : 1)$ un punto de la curva elíptica E expresado en coordenadas proyectivas (ver sección 3.5) para el que se calcula $P, 2P, \ldots, nP$. Entonces, nP tendrá la fórmula $(\phi_n : \omega_n : \psi_n)$ en coordenadas jacobianas, donde ϕ_n, ω_n y ψ_n son polinomios con x, y, a y b como variables. Si se transforma la fórmula anterior a coordenadas afines, el punto nP pasa a ser de la forma

$$nP = \left(\frac{\phi_n}{\psi_n^2}, \frac{\omega_n}{\psi_n^3}\right).$$

Notar que ϕ_n, ω_n y ψ_n se reducen módulo la ecuación de Weierstrass de E, por lo tanto, el grado de y en los polinomios será a lo sumo uno.

Definición 4.7. *El polinomio ψ_n se define como el n-ésimo polinomio de división. Se define de manera recursiva como:*

$$\begin{cases} \psi_1 = 1, \\ \psi_2 = 2y, \\ \psi_3 = 3x^4 + 6ax^2 + 12bx - a^2, \\ \psi_4 = 4y\left(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - a^3 - 8b^2\right). \end{cases}$$

Además, si $n > 4$, las funciones se definen recursivamente como:

$$\begin{cases} \psi_{2n+1} = \psi_{n+2}\psi_n^3 - \psi_{n-1}\psi_{n+1}^3, \\ \psi_{2n} = \dfrac{1}{2y}\psi_n\left(\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2\right). \end{cases}$$

A continuación se muestra uno de los resultados necesarios para el algoritmo de Schoof.

Lema 4.6. *Para todo número n entero se cumple que:*

1. *ψ_n pertenece a:*

   a) $\mathbb{Z}[x, a, b]$, *si n es impar.*

   b) $2y\mathbb{Z}[x, a, b]$, *si n es par.*

2. *ϕ_n pertenece a $\mathbb{Z}[x, a, b]$.*

3. *ω_n pertenece a:*

   a) $y\mathbb{Z}[x, a, b]$, *si n es impar.*

   b) $\mathbb{Z}[x, a, b]$, *si n es par.*

4.3.4. El algoritmo de Schoof

La idea del algoritmo de Schoof es sencilla, se calcula la traza del endomorfismo de Frobenius módulo l, para l números primos pequeños hasta cumplir que $\prod_l l > 4\sqrt{q}$, donde $4\sqrt{q}$ es la longitud de $\mathcal{H}(q)$. Posteriormente se aplica el teorema Chino de los restos y se obtiene t[8], la traza del endomorfismo de Frobenius.

El caso $l = 2$, dado que q es impar[9], entonces $\#E(\mathbb{F}_q)$ es divisible entre dos si y solo si contiene algún punto de orden dos. Un punto tiene orden dos si $2P = O$, o equivalentemente, $P = -P$. La igualdad $P = (x, y) = -P = (x, -y)$ se cumple si la

[8]Se toma $M = \prod_l l > 4\sqrt{q}$ para que, tras aplicar el Teorema Chino de los restos, t quede unívocamente determinado. Supongamos que exista $t \leq t' \in [-2\sqrt{q}, 2\sqrt{q}]$ verificando la misma congruencia, es decir, $t \equiv t' \mod M$, entonces $M \mid (t - t')$. Pero, sabemos que $|t|, |t'| \leq 2\sqrt{q}$, por lo tanto, $|t - t'| \leq |t| + |t'| \leq 4\sqrt{q} < M$, así, $M \mid (t - t') \iff t - t' = 0 \iff t = t'$.

[9]Se trabaja bajo la suposición de q impar. Si $q = 2^n$, con $n \in \mathbb{N}$, se calcula $\#E(\mathbb{F}_2)$ y se aplica el teorema 3.16.

coordenada y vale cero[10]. La manera más eficiente de comprobar si $\#E(\mathbb{F}_q)$ es divisible entre dos es viendo si:

$$m.c.d.(x^q - x, x^3 + ax + b) \neq 1, \text{en } \mathbb{F}_q[x].$$

Caso $l \neq 2$. Recordar que el endomorfismo de Frobenius tiene como ecuación característica $\varphi_q - t\varphi_q + q = 0$[11].

La restricción de φ_q al subgrupo de l-torsión, $E[l]$, sigue cumpliendo la ecuación característica

$$\varphi_l - t_l\varphi_l + q_l = 0$$

en el anillo $End(E[l])$, donde $t_l \equiv t \mod l$ y $q_l \equiv q \mod l$.

Se busca expresar φ_l como se muestra en la proposición 4.3. Para ello se toma ψ_l como el l-ésimo polinomio de división de la curva elíptica E. Por el lema 4.6, como l es impar, ψ_l no depende de la variable y, entonces, $\psi_l \in \mathbb{F}_q[x]$. Además, teniendo en cuenta que $P = (x_0, y_0) \in E(\bar{\mathbb{F}}_q)$ pertenece a $E[l]$ si y solo si $\varphi_l(x_0) = 0$[12]. Por lo tanto, dado que los endomorfismos en $End(E[l])$ son funciones racionales, los polinomios que forman dichas funciones son elementos del anillo $\mathbb{F}_q[x, y]/(\psi_l(x), y^2 - f(x))$, con $f(x) = x^3 - ax - b$ la ecuación de Weierstrass que define a E.

Así, en el caso del endomorfismo de Frobenius se tiene que:

$$\varphi_l = (x^q \mod \psi_l(x), y^q \mod (\psi_l(x), y^2 - f(x))) =$$

$$= (x^q \mod \psi_l(x), y \cdot (f(x)^{(q-1)/2} \mod \psi_l(x))).$$

---

[10] Así la ecuación $x^3 + ax + b$ que define a E tiene un cero.

[11] Se entiende a 0 como el endomorfismo trivial, es decir, el endomorfismo que envía que envía todos los puntos $P \in E(\bar{\mathbb{F}}_q)$ al punto del infinito O.

[12] Dado que el único punto en coordenadas proyectivas (ver sección 3.5) con $Z = 0$ es el punto del infinito O y teniendo en cuenta que $lP = O$, pues $P = (x_0, y_0) \in E[l]$, entonces se cumple que $\psi_l(x_0) = 0$.

La segunda igualdad se deduce sustituyendo $y^2 = f(x)$ y recordando que q es impar. De manera análoga

$$\varphi_l^2 = (x^{q^2} \mod \psi_l(x), y \cdot (f(x)^{(q^2-1)/2} \mod \psi_l(x))).$$

Por lo tanto, hemos expresado el endomorfismo de Frobenius como $(f(x), y \cdot g(x))$, donde f y g son polinomios en el anillo $\mathbb{F}_q[x]/(\psi_l(x))$.

Dado que ya se ha expresado el endomorfismo de Frobenius como muestra la proposición 4.3, solo falta calcular t_l la traza del endomorfismo Frobenius módulo l. Para ello se calculan ψ_l, el l-ésimo polinomio de división, φ_l y φ_l^2. Se busca un $c \in \{0, 1, \ldots, l-1\}$ tal que $c\varphi_l = \varphi_l^2 + q_l$, dicho $c = t_l$ es la traza del endomorfismo Frobenius módulo l. El proceso se repite para los l números primos necesarios (l números primos pequeños hasta cumplir que $\prod_l l > 4\sqrt{q}$), se realiza el Teorema Chino de los restos y se obtiene t.

Pese a ser un algoritmo con tiempo de ejecución polinómico, el algoritmo de Schoof es ineficiente en la práctica. No fue sino hasta 1991, con las mejoras introducidas por Elkies y Atkin, que dicho algoritmo se volvió realmente eficiente. Para más información del algoritmo de Schoof-Elkies-Atkin (SEA) consultar (4).

Capítulo 5

Criptografía de curva elíptica

A lo largo de este capítulo se introducirá la criptografía de curva elíptica. Se comenzará presentando el problema del logaritmo discreto en curvas elípticas, cuya dificultad, bajo ciertas condiciones, garantiza la seguridad de los esquemas criptográficos que se describirán a continuación. Además, se analizarán dos de los ataques más comunes a este problema. Posteriormente, presentaremos los esquemas criptográficos basados en curva elíptica más conocidos y usados, como puede ser el esquema de intercambio de claves de Diffie-Hellman o el algoritmo de firma digital basado en curvas elípticas.

5.1. Logaritmo discreto en curvas elípticas

Definición 5.1. *Sea E una curva elíptica definida sobre $\mathbb{F}_q$, $P \in E(\mathbb{F}_q)$ un punto de orden n, y $Q \in \langle P \rangle$*[1]*. El* ***problema del logaritmo discreto*** *sobre una curva elíptica consiste en encontrar el número entero $l \in [0, n-1]$ tal que:*

$$Q = lP.$$

[1]Donde $\langle P \rangle$ es el subgrupo de $E(\mathbb{F}_q)$ generado por el punto P.

Denotaremos el logaritmo discreto de Q en base P como $l = \log_P Q$.

La seguridad de los criptosistemas basados en curva elíptica recae en la dificultad del problema del logaritmo discreto. Es por ello que se escogen curvas con parámetros resistentes a los ataques conocidos.

A diferencia de los algoritmos para la factorización de enteros, como los empleados en el RSA (ver sección 1.4), cuyo coste computacional es subexponencial, los mejores algoritmos conocidos para resolver el problema del logaritmo discreto para curvas elípticas tienen coste computacional completamente exponencial.

El algoritmo más básico que se puede utilizar es calcular la secuencia $P, 2P, 3P, ...$ hasta encontrar Q. En el peor de los casos se darán n pasos, y de media $n/2$ pasos. Este tipo de ataque se evita tomando un n suficientemente grande ($n \geq 2^{160}$ es suficiente). El mejor ataque conocido al problema del logaritmo discreto sobre curvas elípticas es una mezcla del algoritmo de Pohlig-Hellman (ver 5.1.1) y el algoritmo Rho de Pollard (ver 5.1.2), que tiene una complejidad exponencial de $O(\sqrt{p})$ con p el primo divisor de n más grande. Por lo tanto, debemos recurrir a parámetros de las curvas elípticas para los cuales n sea divisible por p un primo suficientemente grande; $p \geq 2^{160}$ hace intratable el problema. Es por ello que en la práctica el problema del logaritmo discreto se considera intratable en la actualidad.

5.1.1. Algoritmo de Pohlig-Hellman

Mediante el algoritmo se reduce el problema del logaritmo discreto de un grupo de orden n a varios subproblemas más pequeños, resolviendo el logaritmo discreto módulo cada factor primo de $\langle P \rangle$; luego se combinan los resultados usando el **Teorema Chino de los Restos**.

Dado P de orden n, con $n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ su factorización, el algoritmo busca calcular $l_i = l \bmod p_i^{e_l}$ para cada $1 \leq i \leq r$, y resolver el sistema de congruencias con solución

única (por el **Teorema Chino de los Restos**).

$$\begin{cases} l \equiv l_1 & \text{mód } p_1^{e_1} \\ l \equiv l_2 & \text{mód } p_2^{e_2} \\ \quad \vdots & \\ l \equiv l_r & \text{mód } p_r^{e_r} \end{cases}$$

con $n \in [0, n-1]$. La obtención de cada l_i se reduce al cálculo de e_i logaritmos discretos del subgrupo de $\langle P \rangle$ con orden p_i. Se representa l_i en base p_i, es decir:

$$l_i = z_0 + z_1 p_i + z_2 p_i^2 + \cdots + z_{e_i-1} p_i^{e_i-1}$$

con $z_i \in [0, p_i - 1]$. El cálculo de los coeficientes $z_0, z_1, ..., z_{e_i-1}$ se realiza de la siguiente manera, definimos $P_0 = \frac{n}{p_i} P$ y $Q_0 = \frac{n}{p_i} Q$. Dado que P_0 tiene orden p_i, se cumple que:

$$Q_0 = \frac{n}{p_i} Q = l \left(\frac{n}{p_i} P \right) = l P_0 = z_0 P_0.$$

Una vez calculado el logaritmo discreto $z_0 = \log_{P_0} Q_0$ se define $Q_1 = \frac{n}{p_i^2}(Q - z_0 P)$. Desarrollando la expresión de Q_1 llegamos a:

$$\begin{aligned} Q_1 &= \frac{n}{p_i^2} (Q - z_0 P) = \frac{n}{p_i^2} (l - z_0) P = (l - z_0) \left(\frac{n}{p_i^2} P \right) \\ &= (z_0 + z_1 p_i - z_0) \left(\frac{n}{p_i^2} P \right) = z_1 \left(\frac{n}{p_i} P \right) = z_1 P_0. \end{aligned}$$

Por lo tanto, $z_1 = \log_{P_0} Q_1$. De manera análoga, se calcula $z_t = \log_{P_0} Q_t$ con

$$Q_t = \frac{n}{p_i^{t+1}} \left(Q - z_0 P - z_1 p_i P - z_2 p_i^2 P - \cdots - z_{t-1} p_i^{t-1} P \right).$$

Es por eso que se recurre a puntos P con orden n divisible por un primo de gran tamaño. En general, las curvas estandarizadas para uso criptográfico tienen orden $q \cdot h$ con q un primo de gran tamaño y h un número pequeño (ver 5.3).

Efecto del Orden de los Puntos en el Algoritmo de Pohlig-Hellman

Analizaremos cómo la diferencia entre trabajar con un punto P cuya orden es divisible por un primo de gran tamaño, frente a otro punto que no cumple esta condición, afecta el tiempo de ejecución del algoritmo de Pohlig-Hellman. Para este propósito, aunque la función *discrete_log* de SAGE implementa el algoritmo de Pohlig-Hellman, utilizaremos la función desarrollada en (14). Trabajaremos con dos curvas elípticas, ambas definidas sobre $\mathbb{F}_p$ con $p = 4516284508517$:

$$E_1 : y^2 = x^3 + 7x + 1$$
$$E_2 : y^2 = x^3 + 7x + 15.$$

```
import time

m = 21345332
p = 4516284508517
E = EllipticCurve(GF(p), [7, 1])
P = E.gens()[0]
mP = m * P
print(E.abelian_group())
# Imprimir la factorización del orden del grupo
print(E.order().factor()) #4.516.285.972.627=11 * 13 * 31582419389

start_time = time.time()  # Obtener el tiempo de inicio

# Calcular el logaritmo discreto usando Pohlig-Hellman
mRec = PolligHellman(P, mP)

end_time = time.time()

# Imprimir el resultado del logaritmo discreto y su tiempo de ejecución
print(mRec)
print(f"Tiempo de ejecución: {end_time - start_time} segundos")
```

En el primer fragmento de código, trabajamos con la curva E_1, la cual definimos en `SageMath`, y elegimos como punto P al generador del grupo cíclico isomorfo a $\mathbb{Z}/4516285972627\mathbb{Z}$. Por lo tanto, el orden del punto P es 4516285972627, cuya factorización es:

$$4516285972627 = 11 \cdot 13 \cdot 31582419389.$$

En este caso, el tiempo de ejecución para encontrar $m = 21345332$ utilizando el **algoritmo de Pohlig-Hellman** es de **14,37 segundos**.

```
import time


m = 21345332
p = 4516284508517
F = GF(p)
E = EllipticCurve(GF(p), [7, 15])
print(E.abelian_group())
E.gens()
Q = E(4105247933079 , 658434611787)
# punto de orden 1.129.071.439.224 = 2^3 * 3^3 * 113 * 1913 * 24181
mQ = m * Q

# Imprimir la factorización del orden del grupo
print(E.order().factor())

start_time = time.time()  # Obtener el tiempo de inicio

# Calcular el logaritmo discreto usando Pohlig-Hellman
mRec = PolligHellman(Q, mQ)

end_time = time.time()

# Imprimir el resultado del logaritmo discreto
print(mRec)
print(f"Tiempo de ejecución: {end_time - start_time} segundos")
```

En el segundo fragmento de código, trabajamos con la curva E_2. Aquí, el grupo de puntos es isomorfo a $\mathbb{Z}/1129071439224\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$. Elegimos el punto Q que se presenta en el código, cuyo orden es:

$$1129071439224 = 2^3 \cdot 3^3 \cdot 113 \cdot 1913 \cdot 24181,$$

lo que implica que carece de factores primos de gran tamaño. En este caso, el tiempo de ejecución para encontrar $m = 21345332$, utilizando el **algoritmo de Pohlig-Hellman**, es de **0,04768 segundos**.

A pesar de que el orden de Q es **solo 4 veces más pequeño** que el de P, el tiempo de ejecución es aproximadamente **300 veces más rápido**.

5.1.2. Algoritmo Rho de Pollard

A continuación se asumirá que n, el orden de P, es un número primo. El algoritmo se basa en encontrar dos duplas distintas (c', d') y (c'', d'') de números enteros módulo n tales que:

$$c'P + d'Q = c''P + d''Q$$

$$\Updownarrow$$

$$(c' - c'')P = (d'' - d')Q = (d'' - d')lP$$

$$\Updownarrow$$

$$(c' - c'') \equiv (d'' - d')l \mod n$$

Podemos resolver así el problema del logaritmo discreto calculando

$$l = (c' - c'')(d'' - d')^{-1} \mod n$$

La manera más sencilla de encontrar (c', d') y (c'', d'') es generando $c, d \in [0, n-1]$ de manera aleatoria y almacenando $(c, d, cP+dQ)$ en una tabla hasta que se dé una **colisión**[2]. Debido a la paradoja del cumpleaños[3], el número esperado de iteraciones hasta dar con una colisión es de $\sqrt{\pi n/2}$.

---

[2]Se dice colisión cuando un punto $cP + dQ$ se obtiene por segunda vez.

[3]La paradoja del cumpleaños se refiere a la probabilidad de que, en un grupo de n personas, al menos dos compartan el mismo cumpleaños. Si asumimos que los cumpleaños están distribuidos uniformemente en los 365 días del año (ignorando años bisiestos), para $n = 23$, esta probabilidad es aproximadamente 0.507, es decir, hay más del 50 % de probabilidad de que dos personas compartan cumpleaños, lo cual es sorprendentemente alto dado el tamaño del grupo. El número esperado de iteraciones hasta la primera colisión (en este caso que se repita un cumpleaños con $n = 365$) es de $\sqrt{\pi n/2}$.

El algoritmo rho de Pollard encuentra los pares (c', d') y (c'', d'') en, aproximadamente, el mismo tiempo que la manera aleatoria, pero no necesita apenas almacenamiento.

La idea del algoritmo subyace en definir una función iterada[4] $f : \langle P \rangle \longrightarrow \langle P \rangle$ tal que dado un $X \in \langle P \rangle$ y $c, d \in [0, n-1]$ cumpliendo que $X = cP + dQ$, el punto $\tilde{X} = f(X)$ y $\tilde{c}, \tilde{d} \in [0, n-1]$ tales que $\tilde{X} = \tilde{c}P + \tilde{d}Q$ se calcule de manera rápida y sencilla.

Dado un punto $X_0 \in \langle P \rangle$, determina una secuencia $\{X_i\}_{i \geq 0}$ de puntos tales que $X_i = f(X_{i-1})$. Como el orden de $\langle P \rangle$ es finito, la secuencia acabará colisionando y dando lugar a un bucle. Es decir, existe t cumpliendo que $X_t = X_{t+s}$. Se representa como se ve en la Figura 5.1.

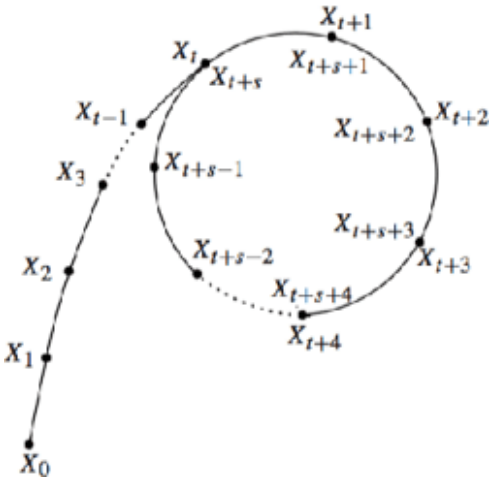


Figura 5.1: Secuencia $\{X_i\}_{i \geq 0}$ descrita con forma de ρ.

Uno de los algoritmos empleados para buscar colisiones es **el algoritmo de detección de ciclos de Floyd**[5]. El funcionamiento del algoritmo es sencillo, $\forall i \in \mathbb{N}$ se calcula la dupla (X_i, X_{2i}) hasta que $X_i = X_{2i}$.

El método original de Pollard busca generar secuencias lo suficientemente complejas, de tal manera que puedan considerarse aleatorias. Para ello:

---

[4]Se define una función iterada como una función que se compone consigo misma.

[5]También conocido como el algoritmo de la liebre y la tortuga.

- Se comienza dividiendo el conjunto $E(\mathbb{F}_q)$ en tres conjuntos disjuntos S_1, S_2, S_3 de aproximadamente el mismo tamaño.

- Se generan c_0 y d_0 dos enteros aleatorios en el intervalo $(0, n)$ y se define el punto $X_0 = c_0P + d_0Q$, así como la función

$$X_{i+1} = f(X_i) = \begin{cases} Q + X_i & \text{si } X_i \in S_1 \\ 2X_i & \text{si } X_i \in S_2 \\ P + X_i & \text{si } X_i \in S_3. \end{cases}$$

- Los valores c_i y d_i se calculan también en función del S_i al que pertenecen, esto es:

$$(c_{i+1}, d_{i+1}) = \begin{cases} (c_i + 1, d_i) & \text{si } X_i \in S_1 \\ (2c_i, 2d_i) & \text{si } X_i \in S_2 \\ (c_i, d_i + 1) & \text{si } X_i \in S_3. \end{cases}$$

- Se generan puntos X_j hasta encontrar una colisión mediante el algoritmo de detección de ciclos de Floyd. Una vez se encuentra un punto X_j tal que $X_i = X_{2i}$ se toma $l = (c_{2j} - c_j)(d_{2j} - d_j)^{-1} \mod n$.

Sin embargo, la función $f(X_i)$ usada no es lo suficientemente aleatoria, es por ello que se recurre a dividir más el conjunto $E(\mathbb{F}_q)$, $s \approx 20$ se considera una buena opción. Se definen las particiones S_i del grupo $E(\mathbb{F}_q)$ como:

$$S_i = \{X = (x, y) \in E(\mathbb{F}_q) \mid x \mod s = i\}, \quad \text{para } i = 0, 1, \ldots, s - 1.$$

El proceso es igual al descrito en el método original de Pollard, únicamente hay q definir una nueva función $f(X_i)$.

5.2. Criptosistemas basados en Curvas Elípticas

5.2.1. Intercambio de Claves de Diffie-Hellman

El presente algoritmo permite el intercambio seguro, entre Alice y Bob, de un punto común [6] de una curva elíptica E. El proceso que se sigue es el siguiente.

1. Alice y Bob deciden un cuerpo finito $\mathbb{F}_q$, una curva elíptica E definida sobre ese mismo cuerpo y un punto P de la curva. Se busca que el subgrupo generado por P sea del mismo orden que el tamaño de E (entendiendo como tamaño el número de puntos de E sobre $\mathbb{F}_q$). De no ser posible, se busca que al menos el orden de P sea grande.
2. Alice escoge un entero a y toma $A = aP$ un punto de la curva elíptica.
3. Bob escoge un entero b y toma $B = bP$ un punto de la curva elíptica.
4. Alice y Bob intercambian los valores de A y B. El intercambio puede producirse a través de un canal no seguro.

Nota 5.1. *El cuerpo finito $\mathbb{F}_q$, la curva elíptica E definida sobre ese mismo cuerpo y el punto P de la curva suelen ser preseleccionados por la entidad correspondiente, como puede ser el NIST (Instituto Nacional de Estándares y Tecnología).*

Es importante señalar que el orden de P tiene que ser divisible por un primo grande para evitar el ataque de Pohlig-Hellman (ver sección 5.1.1).

El punto abP puede usarse para derivar una clave secreta en un criptosistema simétrico, como puede ser el AES [7].

---

[6] El punto es desconocido por ambos al inicio del algoritmo.

[7] Advanced Encryption Standard (AES) es un esquema de clave simétrica de cifrado por bloques (ver 1.3) y estandarizado por el *NIST*.

Eve, el adversario, conoce los puntos P, $A = aP$ y $B = bP$ sobre la curva elíptica E definida en el cuerpo finito $\mathbb{F}_q$. Su objetivo [8] es calcular el punto abP sin conocer los valores de a ó b.

Sin embargo, la única manera conocida de resolver el problema es conociendo a ó b, por ello es necesario resolver el problema del logaritmo discreto asociado a la curva elíptica E. La seguridad del algoritmo se fundamenta en la dificultad computacional de resolver este problema.

Nota 5.2. *En un intercambio de claves usando curvas elípticas, no es necesario que Alice y Bob compartan ambas coordenadas, x e y, de un punto en la curva. Esto se debe a que, en una curva elíptica, la coordenada x determina la coordenada y con una ambigüedad de signo (± 1). Por lo tanto, si Bob envía solo la coordenada x, Alice puede calcular las dos posibles opciones de y extrayendo la raíz cuadrada en el cuerpo finito $\mathbb{F}_q$. Para resolver la ambigüedad, Bob también debe enviar un **bit adicional** que indique cuál de las dos raíces corresponde al punto correcto. Este procedimiento, conocido como **compresión de punto** en criptografía, permite reducir la cantidad de datos transmitidos sin perder información.*

5.2.2. Criptosistema de Clave Pública Elgamal

A diferencia del esquema anterior, este criptosistema le permitirá a Bob enviar un mensaje a Alice sin necesidad de comunicación previa. El algoritmo funciona de la siguiente manera:

1. Alice y Bob deciden un cuerpo finito $\mathbb{F}_q$, una curva elíptica E definida sobre ese mismo cuerpo y un punto P de la curva.

2. Alice escoge un entero a y toma $A = aP$ un punto de la curva elíptica.

3. Alice publica el punto A como su **clave pública**. Mientras que el entero a será su **clave privada**.

[8] A este problema se le conoce como el problema de Diffie-Hellman sobre curvas elípticas.

4. Bob toma un mensaje M, con $M \in E(\mathbb{F}_q)$, y un entero k escogido de manera aleatoria. Bob genera los puntos pertenecientes a la curva elíptica B_1 y B_2 tales que:
$$B_1 = kP \text{ y } B_2 = M + kA$$

5. Bob envia ambos puntos a Alice a través de un canal no seguro.

6. Alice utiliza a, su clave privada para obtener $B_2 - aB_1$. Notar que $B_2 - aB_1 = M$, y por tanto, el punto de la curva elíptica que se pretendía enviar.

Se prueba de manera sencilla que la igualdad final es cierta.

$$B_2 - aB_1 = (M + kA) - akP = M + kaP - akP = M$$

Nota 5.3. *El mensaje en claro enviado por Bob es un punto M de la curva elíptica; sin embargo, el texto cifrado está formado por dos puntos,B_1 y B_2 de la curva elíptica. Pese a la* ***compresión de puntos****, Bob tiene que enviar 2 bits de información por cada bit de mensaje. Es por ello que este criptosistema es* ***menos eficiente*** *que el RSA (ver sección 1.4).*

Al igual que en la sección 5.2.1, Eve, un adversario, tiene que resolver el problema de Diffie-Hellman sobre curvas elípticas. Dado que ya conoce $A = aP$ y $B_1 = kP$, si puede resolver el problema de Diffie-Hellman, podrá calcular $akP = kA$. Además, como ya conoce B_2, podrá obtener M.

5.2.3. Algoritmo de firma digital basado en Curvas Elípticas

A continuación se presentará un esquema de firma digital.

Definición 5.2. *Un esquema de firma digital es un sistema criptográfico que permite a Alice, mediante una clave privada, firmar un documento digital. Bob, utilizando la clave pública de Alice, puede verificar que la firma es válida, garantizando así la autenticidad del remitente y la integridad del mensaje.*

El esquema a seguir es el siguiente.

1. Alice y Bob deciden un cuerpo finito $\mathbb{F}_q$, una curva elíptica E definida sobre ese mismo cuerpo y un punto P de la curva con orden un número primo N.

2. Alice escoge un entero a y toma $A = aP$ un punto de la curva elíptica.

3. Alice publica el punto A, dicho punto se denota **clave pública de verificación.** El punto a será su **clave privada de firma**.

4. Alice firma un documento digital $d \mod N$. Además, selecciona un entero k de manera aleatoria, calcula kP y define[9]:

$$s_1 \equiv x(kP) \mod N \text{ y } s_2 \equiv (d + as_1)k^{-1} \mod N$$

   Alice publica (s_1, s_2) como la firma del documento d.

5. Bob calcula

$$v_1 \equiv ds_2^{-1} \mod N \text{ y } v_2 \equiv s_1 s_2^{-1} \mod N$$

   A partir de esa información calcula $v_1P + v_2A$ un punto de la curva elíptica que verifica

$$x(v_1P + v_2A) \equiv s_1 \mod N$$

Proposición 5.4. *Siguiendo los procesos descritos en* 4. *y* 5.*, el punto que calcula Bob en* 5. *cumple la igualdad* $x(v_1P + v_2A) \equiv s_1 \mod N$.

Demostración. Desarrollando la expresión inicial, llegamos a

$$v_1P + v_2A = (ds_2^{-1})P + (s_1 s_2^{-1})aP = s_2^{-1}(d + as_1)P = kP.$$

Y por lo tanto,

$$x(v_1P + v_2A) = x(kP) \equiv s_1 \mod N.$$

□

Nota 5.5. *Para que el algoritmo de firma digital funcione correctamente, es necesario que los enteros* k *y* s_2 *sean invertibles módulo* N.

[9]La notación $x(\cdot)$ hace referencia a la coordenada x del punto "$\cdot$"

5.2.4. Algoritmo de Massey-Omura

También conocido como algoritmo de tres envíos, es un criptosistema diseñado para el envío de mensajes. El esquema a seguir es el siguiente.

1. Alice y Bob deciden un cuerpo finito $\mathbb{F}_q$, una curva elíptica E definida sobre ese mismo cuerpo y un punto P de la curva con orden un número primo N.
2. Alice y Bob seleccionan de forma secreta enteros e_A y e_B, respectivamente, entre 1 y N, tales que m.c.d.$(e_A, N) = 1$ y m.c.d.$(e_B, N) = 1$. Luego, calculan sus inversos $d_A = e_A^{-1}$ mód N y $d_B = e_B^{-1}$ mód N.
3. Alice quiere enviar un mensaje P (un punto en la curva E) a Bob. Para ello:
   a) Alice calcula $e_A P$ y lo envía a Bob.
   b) Bob recibe $e_A P$ y lo multiplica por su clave e_B, obteniendo $e_B e_A P$. Luego, envía este resultado de vuelta a Alice.
   c) Alice recibe $e_B e_A P$ y lo multiplica por su clave privada d_A, obteniendo $e_B P$. Este resultado lo envía a Bob.
   d) Bob recibe $e_B P$ y lo multiplica por su clave privada d_B, obteniendo finalmente el mensaje original P.

5.3. Curvas del NIST

5.3.1. Curvas de Weierstrass

Definición 5.3. *Dada una curva con ecuación de Weierstrass $E : y^2 = x^3 + ax + b$ sobre $\mathbb{F}_q$ un cuerpo finito se definen los* ***parámetros de dominio*** *como $D = (p, h, n, a, b, G, \{Semilla, c\})$ con:*

1. *q el orden (número de elementos) del cuerpo.*
2. *h el cofactor.*

- $h = 1$ *para curvas generadas de manera pseudoaleatoria.*
- $h > 1$ *para otro tipo de curvas especiales.*

3. *n representa el orden del subgrupo generado por el punto base G.*

4. *$a = -3$ para curvas pseudoaleatorias por razones de eficiencia (ver sección 3.5). Bajo ciertas premisas fijar el valor de a no compromete la seguridad de la curva.*

5. *El parámetro b, que para curvas pseudoaleatorias cumple $b^2c \equiv -27 \mod p$.*

6. *G es el punto base, este es un punto fijo y público en la curva que se usa como punto de partida. G_x representa a la coordenada x del punto y G_y representa a la coordenada y.*

7. *La semilla[10] es una entrada de 160 bits de utilizada como valor inicial para el algoritmo empleado en la generación pseudoaleatoria de los parámetros de la curva. (Para más información sobre el algoritmo consulte (3), páginas $38 - 39$).*

8. *El valor c es la salida del algoritmo. Es necesario para calcular el valor de b.*

Las curvas utilizadas en protocolos criptográficos reales no se seleccionan de forma arbitraria. En su lugar, se emplean algoritmos de generación pseudoaleatoria que crean curvas siguiendo criterios estrictos de seguridad y eficiencia. Este enfoque garantiza que las curvas no solo cumplan con las propiedades matemáticas necesarias, sino que también sean verificables y resistentes a ataques conocidos.

El uso de curvas generadas de manera pseudoaleatoria ofrece varias ventajas clave:

- Transparencia y verificabilidad: Cualquiera puede reproducir y validar cómo se generaron los parámetros, evitando la introducción de puertas traseras o manipulaciones maliciosas.

[10]Dado que la semilla es salida de un algoritmo de **hash** se presenta únicamente en hexadecimal. El hexadecimal es la forma estándar para representar los valores hash, pues es la forma más compacta y legible de representar los valores; notar que un dígito hexadecimal representa 4 bits.

- Mayor seguridad: Estas curvas están diseñadas para resistir ataques criptográficos avanzados, como el problema del logaritmo discreto o la reducción a problemas más fáciles en cuerpos finitos.
- Optimización del rendimiento: La elección de parámetros específicos, como $a = -3$, mejora la eficiencia de las operaciones.

Ejemplo 5.6. *La curva P-256 es una de las presentes en (3). Se trata de una curva con seguridad de* 128 *bits. Los parámetros de la curva son*[11]*:*

- *p (el primo de la curva):*

$$p = 115792089210356248762697446949407573530\backslash \\ 086143415290314195533631308867097853951.$$

- *h (el cofactor de la curva, que es 1 para P-256):*

$$h = 1.$$

- *n (el orden del subgrupo generado por el punto base G):*

$$n = 115792089210356248762697446949407573529\backslash \\ 996955224135760342422259061068512044369.$$

- *a (coeficiente a en la ecuación de Weierstrass):*

$$a = -3.$$

- *b (coeficiente b en la ecuación de Weierstrass):*

$$b = 410583637251521421293261297800472684 09\backslash \\ 114441015993725554835256314039467401291.$$

- *G (el punto base):*

$$G = (x_G, y_G)$$

donde:

[11]Para representar números cuya longitud excede el margen de la hoja, se utiliza el símbolo \. Aunque aparezcan divididos en dos líneas, se está representando un único número.

$$x_G = 48439561293906451759052585252797914202\backslash$$
$$76294952604174799584408071708240463528 6.$$

$$y_G = 36134250956749795798585127919587881956\backslash$$
$$61110667298501507187719825356841440510 9.$$

- $\{Semilla, c\}$:

$$Semilla = 0xc49d360886e704936a6678e1139d26b7819f7e90,$$

$$c = 57436011470200155964173534038266061871\backslash$$
$$44042624415903817595594730946459579034 9.$$

5.3.2. Curvas de Koblitz

Definición 5.4. *Las* ***curvas de Koblitz***[12] *son curvas elípticas definidas sobre* $\mathbb{F}_2$ *con ecuación*

$$E_a : y^2 + xy = x^3 + ax + 1$$

donde $a = 0$ *ó* $a = 1$.

Notar que sobre $\mathbb{F}_2$, E_0 tiene un total de 4 puntos y E_1 cuenta con 2 (contando, obviamente, al punto del infinito). Por lo tanto, $\#E_0(\mathbb{F}_{2^m})$ será múltiplo de 4 y $\#E_1(\mathbb{F}_{2^m})$ múltiplo de 2.

Definición 5.5. *Una curva de Koblitz* E_a *tiene orden "casi primo"sobre* $\mathbb{F}_{2^m}$ *si el número de puntos se puede representar como* $\#E_a(\mathbb{F}_{2^m}) = h \cdot n$, *con* n *primo y* $h = \#E_a(\mathbb{F}_2)$.

Las curvas de Koblitz de orden casi primo definidas en $\mathbb{F}_2$ tienen gran utilidad en la criptografía de clave pública basada en curvas elípticas. Los algoritmos de multiplicación definidos sobre estas curvas evitan el uso de la duplicación de puntos. Veamos un ejemplo recomendado por el *NIST* y que podemos encontrar en (3).

[12]También son conocidas como curvas binarias anómalas.

Ejemplo 5.7. *La curva K-233 es una curva de Koblitz definida sobre* $\mathbb{F}_{2^m}$ *con* $m = 233$ *número primo y* $a = 0$. *La curva cuenta con orden* $h \cdot n$ *y parámetros:*

- *El polinomio irreducible que define el cuerpo finito sobre el cual se construye la curva elíptica:*
$$f(z) = z^{233} + z^{74} + 1.$$
- h *(el cofactor de la curva):*
$$h = 4.$$
- n*:*
$$n = 3450873173395281893717377931138512760570940988862252126328087024741343.$$
- G *(el punto base expresado en base polinómica y hexadecimal):*
$$G = (x_G, y_G)$$
donde:
$$x_G = 0x17232ba853a7e731af129f22ff4149563a419c26bf50a4c9d6eefad6126$$
$$y_G = 0x1db537dece819b7f70f555a67c427a8cd9bf18aeb9b56e0c11056fae6a3.$$

Capítulo 6

Test de primalidad por curvas elípticas

Definición 6.1. *Un **test de primalidad** es un algoritmo que, dado un número, determina si es primo o compuesto.*

6.1. Algoritmo de Goldwasser-Kilian

A lo largo de esta sección se trabajará el test de primalidad por curvas elípticas análogo al test de Pocklington-Lehmer. El test de Pocklington-Lehmer se deriva de la siguiente proposición:

Proposición 6.1. *Sea n un entero positivo impar y q un número primo que divide a $n-1$ tal que $q > \sqrt{n} - 1$. Si existe a un número entero tal que:*

$$\begin{cases} a^{n-1} \equiv 1 \mod n \\ m.c.d.(a^{(n-1)/q} - 1, n) = 1 \end{cases}$$

entonces n es primo.

Demostración. Supongamos, por reducción al absurdo, que n no es primo, entonces existe p, un primo tal que $2 < p \leq \sqrt{n}$ y $p \mid n$. Como $q > p - 1$, pues por hipótesis $q > \sqrt{n} - 1$,

se tiene que $m.c.d.(q, p-1) = 1$ (como q es primo, entonces $m.c.d.(q, p-1) = q$ ó 1, pero sabemos que $q > p-1 > 1$). Por lo tanto, existe u entero tal que $uq \equiv 1 \mod p-1$. Entonces,

$$a^{(n-1)/q} \equiv a^{uq(n-1)/q} = a^{u(n-1)} \equiv 1 \mod p \Leftrightarrow a^{(n-1)/q} - 1 \equiv 0 \mod p$$

Entonces $m.c.d.(a^{(n-1)/q}-1, n) = p$, contradicción con la hipótesis $m.c.d.(a^{(n-1)/q}-1, n) = 1$. Conclusión, n es primo.[1] □

El análogo de la proposición anterior para curvas elípticas es:

Proposición 6.2. *Sea n un entero positivo impar, E el conjunto definido por la ecuación $y^2 = x^3 + ax + b \mod p$ y m un entero. Supongamos que existe q un número primo que divide a m tal que $q > (n^{1/4}+1)^2$. Si existe P un punto cumpliendo:*

$$\begin{cases} mP = O \\ (m/q)P \text{ está definido y es distinto de } O, \end{cases}$$

entonces n es primo.

Demostración. Supongamos, de nuevo, que n no es primo, entonces existe un $p \leq \sqrt{n}$ tal que $p \mid n$. Definimos E' como la curva elíptica con la misma ecuación que E, pero módulo p y tomamos m' el orden del grupo E'. Aplicando el teorema de Hasse (ver 3.12), tenemos que

$$m' \leq p + 1 + 2\sqrt{p} = (\sqrt{p}+1)^2 \leq \left(n^{1/4}+1\right)^2 < q.$$

Por lo tanto, $m.c.d.(q, m') = 1$ y existe u entero tal que $uq = 1 \mod m'$. Tomando $P' \in E'$ el punto P, ya definido anteriormente, pero módulo p, sobre E', se cumple:

$$(m/q)P' = uq(m/q)P' = umP' = O \text{ pues } mP = O \text{ y } p \mid n\ .$$

Sin embargo, esto es una contradicción, pues $(m/q)P$ está definido y es distinto de O nos lleva a que $(m/q)P' \neq O$. □

---

[1] Para probar que $a^{(n-1)/q} \equiv a^{uq(n-1)/q}$ se usa el pequeño teorema de Fermat.
Para $a^{u(n-1)} \equiv 1$ se usa $p \mid q$ y $a^{n-1} \equiv 1 \mod n$.

Notar que el razonamiento empleado en la demostración es análogo al de la proposición anterior. El **algoritmo de Goldwasser-Kilian** para comprobar que n es primo se construye a partir de la proposición 6.2 de la siguiente manera:

A partir de tres enteros seleccionados al azar a, x, y y la relación $b \equiv y^2 - x^3 - ax$ mód n se construye $P = (x, y)$ punto de E, la curva elíptica definida por la ecuación de Weierstrass $y^2 = x^3 + ax + b$. Se aplica un algoritmo para contar el número de puntos de E (ver el capítulo 4) y se obtiene m el número de puntos E sobre $\mathbb{F}_n$ (solo si n es primo). Si no se puede expresar m como $m = kq$ con $k \geq 2$ un entero pequeño y q un "primo probable"[2], entonces se selecciona otro triple a, x, y hasta que se cumpla la condición. Posteriormente, se calculan mP y kP. Si durante el proceso obtenemos una expresión indefinida entonces n no es primo.

Supongamos que no se ha obtenido una expresión indefinida. Si $mP \neq O$ entonces n es compuesto. Si $kP = O$ se descarta la curva elíptica E y se comienza desde el principio. Finalmente, si $mP = O$ y $kP = (m/q)P \neq O$ sabemos por 6.2 que es primo.

Encontrar una curva E verificando $m = kq$ con las características solicitadas en el algoritmo es otro problema. Es necesario conocer algo sobre la distribución de los primos en el intervalo de Hasse $[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$ en el que se encuentra m. Dado que la longitud del intervalo es relativamente pequeña, no existe un teorema que nos asegure una alta probabilidad de encontrar E en un tiempo eficiente.

Posteriormente se han desarrollado otras técnicas basadas también en la proposición 6.2 y que evitan el algoritmo de Schoof para el conteo de puntos en la curva elíptica, la prueba de primalidad de la curva elíptica de Atkin-Morain es un ejemplo. Dicha técnica se basa en la multiplicación compleja para construir la curva elíptica. El desarrollo del algoritmo se sale de la temática y complejidad del proyecto; para más información, consultar el artículo original publicado por Atkin y Morain (2).

[2]Un primo probable es un número entero positivo que pasa uno o varios test de primalidad probabilísticos, es decir, algoritmos que pueden determinar con alta probabilidad (pero no certeza absoluta) si un número es primo.

Capítulo 7

Conclusiones

A lo largo de la memoria se ha visto cómo el álgebra abstracta nos permite construir algoritmos criptográficos sobre los que recae la seguridad informática moderna. Así mismo, se han estudiado las numerosas propiedades algebraicas de las curvas elípticas, su estructura de grupo y su implementación en protocolos criptográficos, demostrando su competitividad frente a los métodos tradicionales como el RSA. En conclusión, se ha recorrido un camino completo a través de los elementos matemáticos necesarios para construir las bases teóricas de la criptografía moderna.

Durante este desarrollo, se han presentado tanto los fundamentos teóricos como las aplicaciones más relevantes de la criptografía basada en curvas elípticas. Se ha profundizado en el problema del logaritmo discreto sobre curvas elípticas, cuya dificultad computacional es clave para garantizar la seguridad de los protocolos, y se han presentado los ataques más eficientes al mismo. Además, se ha explicado cuáles son los parámetros óptimos en las curvas elípticas para garantizar la máxima seguridad frente a ataques criptográficos.

Posteriormente, se han explorado los algoritmos de conteo de puntos sobre curvas elípticas, desde métodos elementales hasta el algoritmo de Schoof. El uso necesario de estos algoritmos en la criptografía de curva elíptica y en los tests de primalidad basados en curvas elípticas hacía necesario el desarrollo de un algoritmo eficiente (con un tiempo de ejecución polinómico), capaz de resolver este problema. El uso del álgebra abstracta, en este caso mediante el cálculo de la traza del endomorfismo de Frobenius y el apoyo del teorema de Hasse, ha sido clave en la resolución de este problema.

El estudio también ha incluido una visión de los principales esquemas criptográficos, como el intercambio de claves de Diffie-Hellman basado en curvas elípticas y el algoritmo de firma digital basado en curvas elípticas. Además del desarrollo de algoritmos, como el de Goldwasser-Kilian, capaces de detectar si un número es primo o no.

Las curvas elípticas representan un equilibrio entre la matemática teórica y la utilidad práctica en criptografía. Su evolución debe acompañarse de rigurosidad en la selección de parámetros para mantener la seguridad. El estudio de las curvas elípticas para su uso criptográfico sigue siendo un campo de investigación activo, especialmente ante la amenaza que suponen los futuros ordenadores cuánticos. Aunque los esquemas recogidos no ofrecen resistencia cuántica, el análisis de las curvas elípticas tradicionales ha dado lugar a variantes y esquemas que se perfilan como candidatas prometedoras dentro de la criptografía post-cuántica. Es la denominada criptografía de isogenias.

Bibliografía

[1] Arrendondo, B. and Jansma, N., *Performance Comparison of Elliptic Curve and RSA Digital Signatures*, IPCSIT vol. 4, IACSIT Press, Singapore (2011).

[2] Atkin, A. O. L. and Morain, F., *Elliptic Curves and Primality Proving*, 1993, Mathematics of Computation, 61(203), 29-68, http://dx.doi.org/10.1090/S0025-5718-1993-1199989-X.

[3] Chen, L., Moody, D., Randall, K., Regenscheid, A., and Robinson, A., *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*, NIST Special Publication 800-186, 2023. Disponible en https://doi.org/10.6028/NIST.SP.800-186.

[4] Dewaghe, L., *Remarks on the Schoof-Elkies-Atkin Algorithm*, Mathematics of Computation, Vol. 67, No. 223 (1998), pp. 1247-1252. Disponible en http://www.jstor.org/stable/2585182.

[5] Ezzouak, S., Elamrani, M., y Azizi, A., *A Variant of Pollard's Rho Attack on Elliptic Curve Cryptosystems*, Journal of Computer Science, Vol. 10, No. 8 (2014), pp. 1575-1581. Disponible en https://thescipub.com/abstract/jcssp.2014.1575.1581

[6] Hankerson, D. Menezes, A. J., y Vanstone, S., *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2003. Disponible en: https://doi.org/10.1007/b97644

[7] Jansma, N. y Kaliski, B. S., *Performance Comparison of Elliptic Curve and RSA Digital Signatures*, 2004. Disponible en: fog.misty.com/perry/ccs/ec/KF

[8] Koblitz, N., *Towards a Quarter-Century of Public Key Cryptography*, en: Koblitz, N. (ed.), *Mathematics of Public Key Cryptography*, Springer, 2000. Disponible en: https://doi.org/10.1007/978-1-4757-6856-5

[9] Koblitz, N., *A Course in Number Theory and Cryptography*, Graduate Texts in Mathematics, Vol. 114, Springer, 2ª edición, 1987. Disponible en: https://doi.org/10.1007/978-1-4419-8592-7

[10] Koblitz, N., *Elliptic Curve Cryptosystems*, 1987, Mathematics of Computation, 48, 203-209, http://dx.doi.org/10.1090/S0025-5718-1987-0866109-5.

[11] Koblitz, N., Menezes, A. J., Wu, Y.-H., and Zuccherato, R. J., *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics, Vol. 3, Springer, 1998. Disponible en: https://doi.org/10.1007/978-3-662-03642-6

[12] Magons, K., *Applications and Benefits of Elliptic Curve Cryptography*, Conference on Current Trends in Theory and Practice of Informatics, 2016. https://api.semanticscholar.org/CorpusID:11290144.

[13] MIT OpenCourseWare, *18.783 Elliptic Curves - Lecture Notes*, Spring 2015, https://opencw.aprende.org/courses/mathematics/18-783-elliptic-curves-spring-2015/lecture-notes/.

[14] Novotney, P., *Weak Curves In Elliptic Curve Cryptography*, 2010, http://wstein.org/edu/2010/414/projects/novotney.pdf.

[15] Oetiker, T., *The (Not So) Short Introduction to LaTeX 2ε*, Version 6.4, March 09, 2021. https://ctan.org/tex-archive/info/lshort/english/lshort.pdf.

[16] Schoof, R., *Counting Points on Elliptic Curves over Finite Fields*, Journal de Théorie des Nombres de Bordeaux, Vol. 7, No. 1 (1995), pp. 219-254. Disponible en: https://www.numdam.org/item/JTNB_1995__7_1_219_0/

[17] Silverman, J. H., *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, Springer, Dordrecht, 2009. Disponible en: https://doi.org/10.1007/978-1-4757-1920-8

Anexos

Anexo A

Teoría de los símbolos de Legendre y Jacobi

Definición A.1. *Sea p un número primo impar. Un entero a se dice residuo cuadrático módulo p si existe un entero x tal que:*

$$x^2 \equiv a \mod p.$$

En caso contrario, a se llama no residuo cuadrático módulo p.

Si $a \in \mathbb{F}_p^*$ verifica la congruencia $x^2 \equiv a \mod p$, entonces a tiene dos raíces, $\pm x$. Para encontrar los $a \in \mathbb{F}_p^*$ que son cuadrados es necesario calcular $x^2 \mod p$ para $x \in \{1, 2, \ldots, (p-1)/2\}$, pues el resto son la segunda raiz $-x$ para algún x de los anteriores. Bajo la suposición de que p es primo, se cumple que $(p-1)/2$ son residuos cuadráticos módulo p y a la otra mitad son no residuo cuadrático módulo p.

A continuación se define una notación para diferenciar los residuos cuadráticos de los que no lo son.

Definición A.2. *Sea p un primo impar y a un entero. Para determinar si un número es residuo cuadrático, se define el símbolo de Legendre:*

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \textit{si } a \textit{ es un residuo cuadrático módulo } p, \\ -1 & \textit{si } a \textit{ es un no residuo cuadrático módulo } p, \\ 0 & \textit{si } p \mid a. \end{cases}$$

Como una extensión del símbolo de Legendre, se define el símbolo de Jacobi.

Definición A.3. *Sea a un entero y n un entero impar con descomposición en factores primos $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, se define el símbolo de Jacobi como:*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \dots \left(\frac{a}{p_k}\right)^{e_k}.$$

Aunque el símbolo de Jacobi extiende la notación del símbolo de Legendre, no siempre indica si a es un residuo cuadrático módulo n.

De esta teoría deducimos la expresión: "Sin embargo, dado que una ecuación cuadrática escogida de manera aleatoria tiene, aproximadamente, un 50 % de probabilidad de tener solución (dos valores de y) para cada elemento de $\mathbb{F}_q$."

Vamos a precisar este dato, sea χ la aplicación tal que $\chi(x) = (\frac{x}{p})$ el símbolo de Legendre $\forall x \in \mathbb{F}_p^*$. Así, el número de soluciones y en $\mathbb{F}_p$ a la ecuación $y^2 = x^3 + ax + b$ es:

$$1 + \sum_{x \in \mathbf{F}_p} \left(1 + \chi\left(x^3 + ax + b\right)\right) = q + 1 + \sum_{x \in \mathbf{F_p}} \chi\left(x^3 + ax + b\right).$$

Bajo la suposición de p primo se espera que $\chi(x^3 + ax + b)$ de como resultado 1 ó -1 con la misma probabilidad. Esta suma se puede ver como un camino aleatorio, por lo tanto, la distancia recorrida tras p pasos es del orden de $\sqrt{p}$. Esta suma se acota por $2\sqrt{p}$ como se ha visto en el Teorema de Hasse 3.12.