Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA
DE SEGOVIA

Grado en Ingeniería Informática
de Servicios y Aplicaciones

Criptografía de curvas elípticas: ataques y análisis de robustez

---

Alumno: Eduardo Quintana Adeva
Tutor: José Ignacio Farrán Martín

Criptografía de curvas elípticas: ataques y análisis de robustez

Eduardo Quintana Adeva

Septiembre 2025

Índice general

Índice de figuras

Índice de tablas

Dedicado a todos aquellos más desesperados
que yo con este trabajo

Agradecimientos

Quiero agradecer en especial a mi compañero y amigo, Miguel Benito, por todas las divagaciones que ha tenido que aguantar por mi parte mientras revisábamos este trabajo, y en general, en cualquier quedada que hayamos tenido. De igual manera, quiero agradecerle a Juan Antonio Velasco el haber estado ahí todos estos años de carreras, convirtiéndose en un pilar fundamental no sólo para mí, sino para cientos de jóvenes que se embarcaron en la aventura que es el PEC, sin saber la tempestad que se les venía encima. A mi empresa, *Animal Data Analytics* por la flexibilidad que me ha ofrecido todos estos años con la carrera. Y como lo más importante se deja para el final, a mi madre, por ser un apoyo constante y haberme quitado de la cabeza el renunciar a las matemáticas, a pesar de todas esas tardes en las que le aburría hablando de topología.

Abstract

This work presents a mathematical analysis of elliptic curve cryptosystems, highlighting the lack of objective metrics to compare the security of different systems. To address this gap, a new metric called *Available Attack At* ($3A$) is introduced, based on the probability of finding a collision using a Pollard's Rho attack. It is shown that this metric defines a distribution function and, under two simplifying hypotheses, chosen an elliptic curve cryptosystem $(K, \mathbb{E}, P)$ is proven to converge in distribution to $1 - \exp(-\frac{N(N-1)}{2 \cdot ord(P)})$. Two experimental studies are conducted: the first compares empirical distributions obtained via Monte Carlo simulations across six different cryptosystems with the proposed theoretical distribution; the second contrasts the cryptosystems Z25519 and P384, concluding—based on $3A$—that P384 offers higher resistance against Pollard's Rho attacks.

Keywords: Elliptic Curve, Cryptography, Pollard Rho Attack, Birthday Paradox, Collision.

Resumen

Este trabajo presenta un análisis matemático de los criptosistemas de curva elíptica, destacando la ausencia de métricas objetivas que permitan comparar la seguridad entre distintos sistemas. Para abordar esta carencia, se introduce una nueva métrica llamada *Available Attack At* (3A) basada en la probabilidad de encontrar una colisión mediante un ataque Rho de Pollard. Se demuestra que dicha métrica define una función de distribución y que, bajo dos hipótesis simplificadoras, fijando un criptosistema de curva elíptica $(K, \mathbb{E}, P)$ converge en distribución hacia $1 - \exp(-\frac{N(N-1)}{2 \cdot ord(P)})$. Se llevan a cabo dos estudios experimentales: en el primero se comparan las distribuciones empíricas obtenidas mediante simulaciones de Monte Carlo en seis criptosistemas distintos con la distribución teórica propuesta; en el segundo se contrastan los criptosistemas Z25519 y P384, concluyendo —a partir de $3A$— que P384 ofrece una mayor resistencia frente a ataques de tipo Rho de Pollard.

Palabras claves: Curva Elíptica, Criptografía, Ataque Rho de Pollard, Paradoja del Cumpleaños, Colisión.

Parte I

Descripción del proyecto

Capítulo 1

Introducción

The beginning is the most important part of any work.

Platón

La creciente adopción de criptografía basada en curvas elípticas (ECC) ha impulsado el desarrollo de sistemas cada vez más complejos y robustos para proteger la información digital. Sin embargo, a medida que estos criptosistemas evolucionan, surge la necesidad de contar con herramientas que permitan evaluar y comparar su seguridad de manera cuantitativa y estandarizada. La valoración de estos sistemas de forma justa y objetiva no es una tarea sencilla, ya que exige una ardua labor analítica, que a su vez conlleva trabajar con conceptos matemáticos sumamente complejos. En este contexto, la creación de una métrica específica para medir la seguridad en criptosistemas de curva elíptica se convierte en un elemento clave para investigadores, desarrolladores y entidades reguladoras. En este trabajo se propone el diseño de una métrica integral basada en el ataque Rho de Pollard –ataque preferido a la hora de vulnerara sistemas de ECC– ofreciendo así una herramienta objetiva y útil para el análisis y la toma de decisiones en el ámbito de la seguridad criptográfica. Esta labor ha sido llevada a cabo mediante la creación de una librería en Python que permite trabajar de manera amigable con sistemas aritméticos no estándar y a su vez implementa algoritmos que conforman la base de los ataques a los sistemas de ECC. Por último, en el mismo se presenta un estudio de validación sobre la métrica propuesta y un ejemplo de uso en la comparación de sistemas utilizados en entornos de producción.

1.1. Planteamiento del Problema

Actualmente, la ECC presenta una línea de investigación sumamente importante, hecho que puede ser contrastado por la multitud de aplicaciones que esta presenta en diversos entornos de producción. Ejemplos de esto pueden ser el algoritmo de intercambio de clave en E2EE, los certificados digitales utilizados en multitud de naciones a lo largo del globo o el establecimiento de contratos inteligentes en sistemas *blockchain*.

A pesar de la alta confiabilidad que presentan todos estos sistemas, el manejo y estudio profundo de los mismos es una cuestión nada trivial que requiere de amplios conocimientos algebraicos. Es por esto que resulta increíblemente complejo comparar de forma cuantitativa y objetiva la seguridad entre dos criptosistemas de curvas elípticas.

De manera general, se acepta que, a mayor orden del elemento generador seleccionado como parámetro de dominio, mayor es la complejidad asociada a la resolución del problema del logaritmo discreto en curvas elípticas, lo cual conlleva un incremento de la seguridad del criptosistema. Sin embargo, esta premisa debe ser matizada pues existen curvas con propiedades susceptibles de ser explotadas para facilitar o acelerar la vulneración de los criptosistemas que conforman. El *National Institute of Standards and Technology* (NIST) ha establecido una serie de requisitos específicos para los criptosistemas basados en ECC, con el propósito de mitigar los riesgos derivados de métodos de ataque conocidos [NIST, 2001]. Las disposiciones establecidas por el NIST enlazan directamente con las directrices formuladas en [Tena, 2014], orientadas a la adecuada elección de las coordenadas del elemento generador, con el fin de prevenir la aceleración en la resolución del problema del logaritmo discreto.

Las directrices previamente mencionadas delimitan una categoría de criptosistemas que la comunidad considera inseguros; sin embargo, no ofrecen un marco que permita comparar de manera cuantitativa el nivel de seguridad entre un criptosistema seguro y otro que no lo es. Más aún, dicha clasificación resulta insuficiente incluso para establecer comparaciones objetivas entre dos sistemas que se consideran seguros. En consecuencia, el propósito central de este trabajo es proponer una métrica objetiva y fundamentada que posibilite la evaluación cuantitativa de la seguridad entre diferentes sistemas.

1.2. Objetivos del Proyecto

A continuación se presentan los objetivos a cumplir en el siguiente proyecto de investigación:

- **OB-01:** Elaboración de un módulo para realizar operaciones en aritmética de cuerpos finitos de forma optimizada aislando al programador de los entresijos matemáticos.
- **OB-02:** Elaboración de un módulo para realizar operaciones en aritmética de curva elíptica de forma optimizada aislando al programador de los entresijos matemáticos.
- **OB-03:** Creación de un módulo de ataques que implemente los principales ataques para la vulneración de un criptosistema de curva elíptica.
- **OB-04:** Creación de una métrica que permita contrastar la seguridad entre criptosistemas de curva elíptica.
- **OB-05:** Validación de la métrica creada sobre un *ground truth*, estableciendo un estudio estadístico pertinente.
- **OB-06:** Aplicación de la métrica obtenida realizando un estudio comparativo entre criptosistemas de curva elíptica utilizados en entornos de producción.

1.2.1. Restricciones

- **R-01:** El dominio de las curvas elípticas deberá ser el producto cartesiano de un cuerpo primo consigo mismo.
- **R-02:** La aplicación solo permite trabajar con cuerpos primos de característica estrictamente mayor a tres.
- **R-03:** La aplicación solo permitirá trabajar con curvas elípticas en forma de Weiestrass.
- **R-04:** La librería desarrollada sólo funcionara en entornos Python 3.X.

1.3. Estructura de la memoria

En adelante se expone el contenido de cada una de las secciones de esta memoria, definiendo estas brevemente e indicando un orden de lectura recomendado.

- **Sección 1 - Introducción.** Se comenta brevemente la relevancia de la ECC en las comunicaciones actuales. Se presenta el problema para comprender y evaluar este tipo de criptosistemas debido a su *background* teórico. Finalmente se introduce la herramienta desarrollada, los objetivos alcanzados a lo largo de su desarrollo y las restricciones que esta tiene.
- **Sección 2 - Planificación.** Se presenta el cuerpo del proyecto para el desarrollo de EllipticCUrVa mostrando el desglose de tareas, el presupuesto y el cronograma. De igual manera se presenta la documentación software necesaria para el diseño de la herramienta.
- **Sección 3 - Antecedentes.** Se presenta el desarrollo teórico necesario para entender todos los procesos implementados en EllipticCUrVa. Esta sección se encuentra dividida en tres áreas de conocimiento: **Álgebra - Curvas Algebraicas**, **Curvas Elípticas** y **Criptografía**.
- **Sección 4 - Propuesta de medida de seguridad.** En este apartado se define una nueva medida de seguridad basada en un ataque popular en la ECC, y se muestran métodos de calculo para esta.
- **Sección 5 - Desarrollo de la propuesta.** Se define tanto la herramienta programada para realizar los dos experimentos realizados con la nueva medida de seguridad; uno de validación y otro un caso de uso.
- **Sección 5 - Análisis y resultados.** Se muestran los resultados de los experimentos de validación y el ejemplo de aplicación con la métrica propuesta. De igual manera se establece una breve discusión comparando lo obtenido con trabajos similares.
- **Sección 6 - Conclusiones y trabajo futuro.** Se comentan los resultados del capítulo anterior y se comparan con trabajos similares. Además, se presentan mejoras que podrían ser implementadas en futuras versiones de EllipticCUrVa.

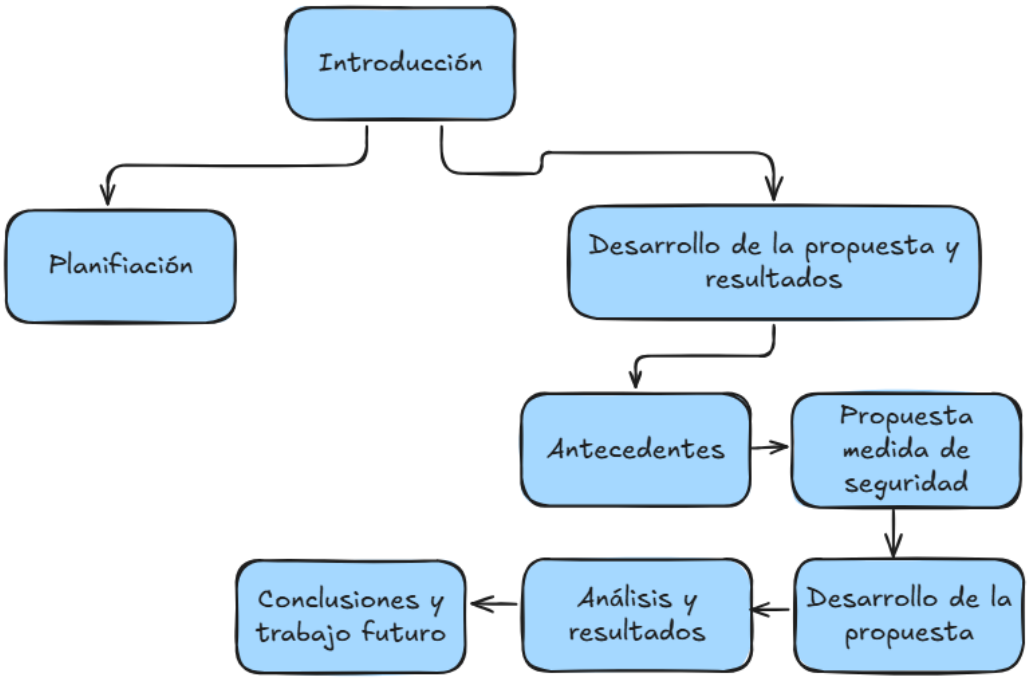


Figura 1.1: Orden de lectura recomendado

Capítulo 2

Planificacion

I always been attracted to ideas that were about revolt against authority. When you make your peace with authority you became an authority. I like ideas about the breaking away or overthrowing the estabilished order. I am interested in anything about revolt, disorder, chaos, specially activity that seems to have no meaning.

Jim Morrison

La planificación de un proyecto de investigación representa una fase esencial que orienta y estructura el desarrollo del estudio. En este capítulo se expone la estructuración en etapas del proyecto llevado a cabo, así como el cronograma planificado y las modificaciones que se han debido efectuar a lo largo del tiempo sobre este.

2.1. Metodología de trabajo

Para desarrollar este proyecto se ha utilizado un marco híbrido de trabajo basado en *Test Driven Development* (TDD), el marco de desarrollo Iterativo-Incremental y *Peer Review*. En la Sección 2.1.1 de este capítulo se definen cada una de las metodologías señaladas y posteriormente la conexión entre estas en el nuevo marco híbrido señalado. Por último, en la Sección ... se presenta una justificación del proyecto por incrementos.

2.1.1. Definición de las metodologías utilizadas

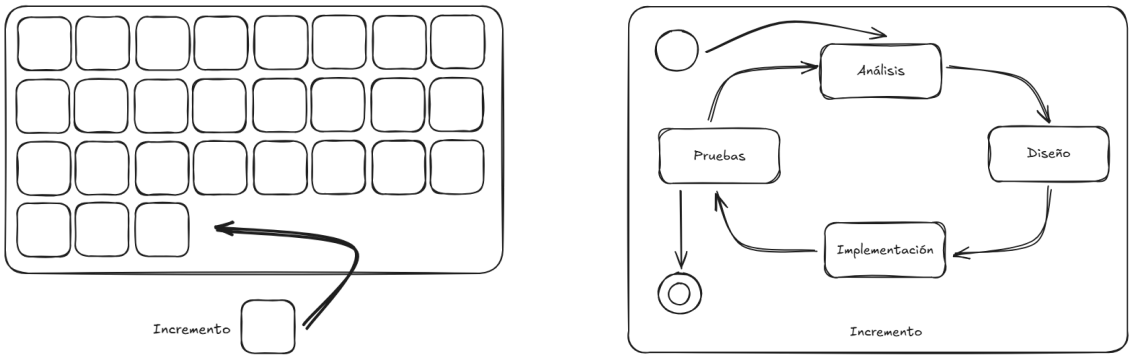
En esta sección se presentan y describen de manera detallada las distintas metodologías empleadas a lo largo del desarrollo de este proyecto.

Desarrollo Iterativo-Incremental

El desarrollo iterativo-incremental [Larman, 2004] es un enfoque de desarrollo de proyectos en el que el producto final se divide en incrementos (véase Figura 2.1a). Cada unos de estos incrementos

es conformado y refinado de forma iterativa siguiendo un patrón de cuatro fases: Análisis, Diseño, Implementación y Pruebas (véase Figura 2.1b):

- **Análisis**. En esta fase se definen las funcionalidades que deben figurar en el incremento o las propiedades que este debe satisfacer (p.e. si el incremento es la implementación de un login, aquí se definen cosas como el tipo de seguridad, las funciones de verificación o las propiedades que este debe satisfacer para adecuarse a un ISO).
- **Diseño**. En este nuevo paso, se definen cosas como la lógica que deben ejecutar las funciones anteriores, los patrones a utilizar o las relaciones entre artefactos (p.e. volviendo al ejemplo del login, se definirían los pasos a seguir dentro de la función; comprobar que nombre y usuario no están vacíos ni son susceptibles de inyección, validar contra la base de datos, devolver un 303 si son autorizados y un 401 si no lo son).
- **Implementación**. En esta nueva etapa se pasa a la escritura en código fuente de la lógica seleccionada en la fase de diseño (p.e. la creación de los DAOs y las entidades de los usuarios sobre BBDD).
- **Pruebas**. En esta última fase se ejecutan pruebas sobre las funcionalidades implementadas para cerciorar su correcto uso y ejecución (p.e. la definición de test unitarios sobre cada una de las funcionalidades implementadas).



(a) Incremento sobre proyecto (b) Iteración sobre el incremento

Figura 2.1: Fases del desarrollo iterativo-incremental

La ventaja en esta metodología es que permite evaluar de forma parcial la solución a desarrollar por parte del equipo, pudiendo iterar nuevamente en los incrementos en caso de modificaciones necesarias.

Test Driven Development (TDD)

El TDD [Beck, 2022] es una metodología de desarrollo de software centrada en las pruebas unitarias. De forma más precisa, TDD indica que antes de implementar las funcionalidades de una solución, se deben implementar las pruebas unitarias correspondientes. La implementación de estas pruebas unitarias sigue un ciclo de vida específico (véase Figura 2.2) conocido como *Red-Green-Refactor*:

- **Red**. Se escribe una prueba, asociada a la funcionalidad a implementar, que falle. El motivo de este fallo se debe a que la funcionalidad no existe o todavía no valora alguna restricción definida o caso anómalo.

- **Green**. Se implementa la funcionalidad, o se añade a esta la menos porción de código que satisfaga todos los test unitarios.
- **Refactor**. Se reescribe el código de la funcionalidad para que resulte más legible y/o eficiente. Por último, se eliminan pruebas redundantes en caso de existir.

En el desarrollo de una funcionalidad se pueden iterar tantas veces como sea necesario, con el objetivo de que esta satisfaga todas las restricciones lógicas o de negocio, y presente una ejecución segura.

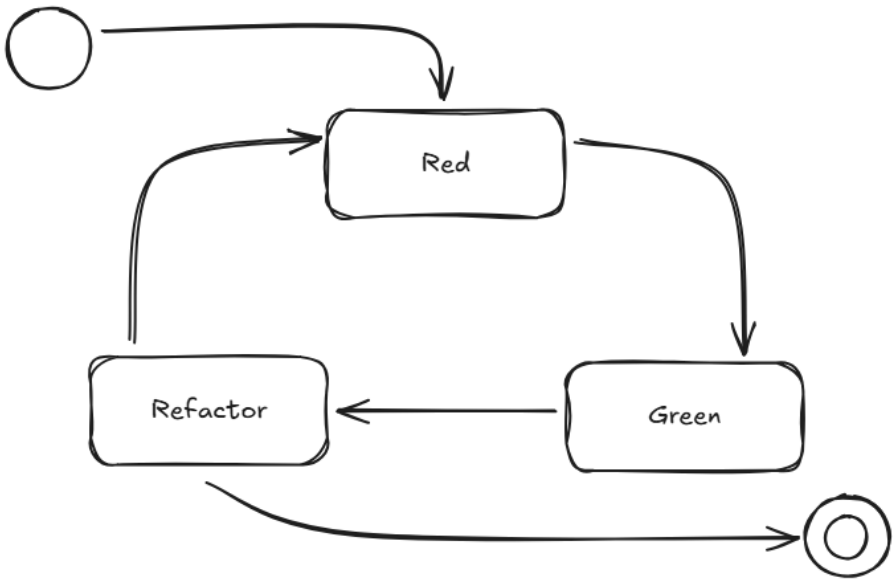


Figura 2.2: Ciclo *Red-Green-Refactor*

Muchas metodologías de trabajo (por no decir casi todas) definen a lo largo del ciclo de desarrollo software una fase de pruebas; sin embargo esta labor muchas veces se queda en el olvido por resultar un trabajo tedioso o por cercanía a la hora de presentar el proyecto. El TDD palia esta situación, pues el desarrollador se encarga de desarrollar las pruebas unitarias en la fase de implementación, no pudiendo escribir nuevo código fuente sin implementar primero un test unitario.

Peer Review

El *Peer Review* [Kelly et al., 2014] es un proceso colaborativo en el que el trabajo desarrollado, ya sea código, documentación o resultados parciales, es evaluado de manera crítica por uno o más miembros del equipo o por expertos externos. Este proceso permite identificar errores, inconsistencias y áreas de mejora que pueden no ser evidentes para el autor original, incrementando así la calidad y confiabilidad del producto final.

Marco híbrido

Para este proyecto se ha elaborado un marco de trabajo híbrido basado en las tres metodologías definidas anteriormente. Para ello se ha desglosado el proyecto en incrementos, que son valorados como tareas de segundo nivel de una **Estructura de Desglose de Trabajo** (WBS). Cada uno de estos incrementos ha sido construido de forma iterativa, siguiendo la estructura cíclica mostrado en la Figura 2.3.

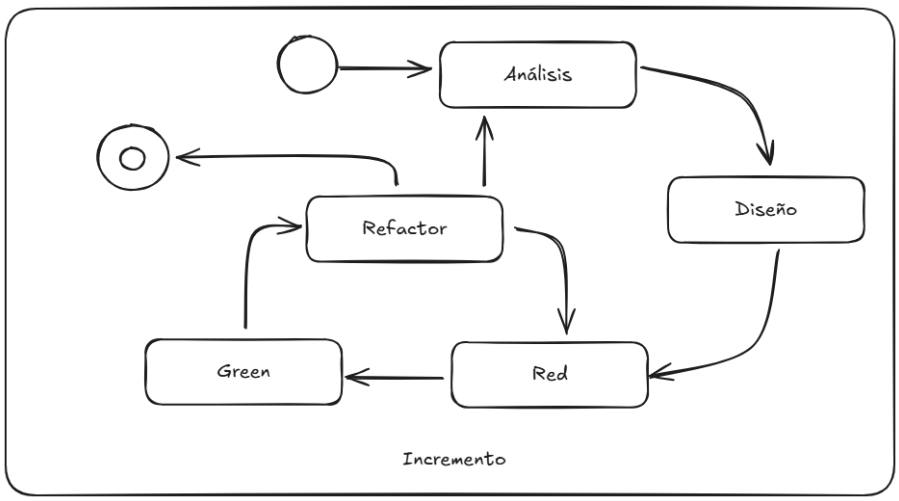


Figura 2.3: Iteraciones del marco de trabajo híbrido

A diferencia del diseño iterativo-incremental clásico, las fases de implementación y pruebas son sustituidas por una nueva fase productiva basada en TDD. Así pues el ciclo de vida de desarrollo de software empezaría por la fase de análisis, seguiría por la de diseño y por último llegaría la fase productiva, en la que implementación y pruebas irían de la mano, gracias a TDD. Dentro de esta nueva fase productiva, iteraríamos según la regla *Red-Green-Refactor* todas las veces que fuese necesario hasta cubrir por completo la implementación de todas las funcionalidades de forma segura y testada. Si hubiesen errores de análisis o diseño, avanzaríamos nuevamente desde esta fase productiva a la fase de análisis, pudiendo refactorizar el incremento por completo.

Otro punto que discrepa con el diseño iterativo-incremental clásico es un incremento se anexaba a la solución final cuando este se daba por finalizado. En este trabajo se ha añadido una fase de *Peer Review* al final de cada incremento por parte de un revisor no desarrollador. Si la revisión es satisfactoria, el incremento se añade a la solución final. Por el contrario, se opina que la calidad del código o de la documentación escrita es mejorable, o encuentran errores; el ciclo de desarrollo del incremento es reactivado para la resolución de los puntos señalados por el revisor.

2.1.2. División en incrementos

En esta sección se dividirá el proyecto en incrementos y estos incrementos serán divididos a su vez en tareas. Para la subdivisión en incrementos se ha utilizado la WBS mostrada en la Figura 2.4, conformando cada tarea de segundo nivel un incremento del proyecto. A continuación se procede a dividir cada uno de estos incrementos en las tareas que fueron llevadas a cabo para su resolución.

Incremento 1 - Documentación sobre grupos

En este primer incremento se hizo una revisión bibliográfica sobre grupos, revisándose los tópicos presentes en la Tabla 2.1 para la escritura de la Sección 3.1 de los antecedentes, que pasó el proceso de verificación por pares antes de formar parte de la memoria.

Incremento 2 - Documentación sobre anillos

En este segundo incremento se hizo una revisión bibliográfica sobre anillos, revisándose los tópicos presentes en la Tabla 2.2 para la escritura de la Sección 3.2 de los antecedentes, que pasó

Figura 2.4: Estructura de desglose de trabajo del proyecto

Tabla 2.1: Tareas del incremento 1

Id	**Descripción**
TA_1.1.	Desarrollo del concepto de grupo
TA_1.2.	Desarrollo del problema del logaritmo discreto
TA_1.3.	Desarrollo de subgrupos y grupo generado
TA_1.4.	Desarrollo de orden de grupo

el proceso de verificación por pares antes de formar parte de la memoria.

Tabla 2.2: Tareas del incremento 2

Id	Descripción
TA_2.1	Desarrollo del concepto de anillo y cuerpo
TA_2.2	Desarrollo de las potencias en anillos
TA_2.3	Desarrollo de los anillos de polinomios
TA_2.4	Desarrollo de residuos cuadráticos

Incremento 3 - Documentación sobre curvas

En este tercero incremento se hizo una revisión bibliográfica sobre curvas algebraicas –focalizando especialmente en cuervas elípticas– revisándose los tópicos presentes en la Tabla 2.3 para la escritura de las Secciones 3.3 y 3.4 de los antecedentes, que pasaron el proceso de verificación por pares antes de formar parte de la memoria.

Tabla 2.3: Tareas del incremento 3

Id	Descripción
TA_3.1	Desarrollo del concepto de curva algebraica
TA_3.2	Desarrollo del concepto de singularidad
TA_3.3	Desarrollo del concepto de curva elíptica
TA_3.4	Desarrollo de los grupos inducidos sobre una curva elíptica
TA_3.5	Desarrollar el orden de la curva
TA_3.6	Desarrollar el ECDLP

Incremento 4 - Documentación sobre criptografía de curva elíptica

En este cuarto incremento se hizo una revisión bibliográfica sobre anillos, revisándose los tópicos presentes en la Tabla 2.4 para la escritura de la Sección 3.5 de los antecedentes, que pasó el proceso de verificación por pares antes de formar parte de la memoria.

Tabla 2.4: Tareas del incremento 4

Id	Descripción
TA_4.1	Desarrolla preludio criptografia
TA_4.2	Desarrollar criptografia de clave publica
TA_4.3	Desarrollar ECC
TA_4.4	Desarrollar ataques de ECC

Incremento 5 - Creación del módulo de aritmética

En este quinto incremento se realizó la implementación de un módulo de aritmética no estándar en Python, utilizando para ello los conocimientos adquiridos en la revisión bibliográfica. La implementación de este módulo fue divida en tareas según se puede apreciar en la Tabla 2.5. Se remarca que, el desarrollo de este módulo siguió todas las fases del marco híbrido de desarrollo software presentado en este mismo capítulo.

Tabla 2.5: Tareas del incremento 5

Id	Descripción
TA_5.01	Crear clase entero modular
TA_5.02	Crear operadores modulares
TA_5.03	Implementar símbolos de Jacobi
TA_5.04	Implementar algoritmo Tonelli-Shanks
TA_5.05	Implementar clase curva elíptica
TA_5.06	Implementar evaluaciones de curva elíptica
TA_5.07	Implementar algoritmo de pertenencia
TA_5.08	Implementar la cota de Hasse
TA_5.09	Implementar clase punto curva
TA_5.10	Implementar operadores elípticos
TA_5.11	Implementar algoritmo de punto aleatorio

Incremento 6 - Creación del módulo de ataque

En este sexto incremento se realizó la implementación de un módulo de ataque a la ECC Python, utilizando para ello los conocimientos adquiridos en la revisión bibliográfica y el módulo de aritmética no estándar. La implementación de esta librería fue divida en tareas según se puede apreciar en la Tabla 2.6. Se remarca que, el desarrollo de este módulo siguió todas las fases del marco híbrido de desarrollo software presentado en este mismo capítulo.

Tabla 2.6: Tareas del incremento 6

Id	Descripción
TA_6.1	Algoritmo Rho de Pollard Clásico
TA_6.2	Algoritmo Rho de Pollard-Floyd

Incremento 7 - Desarrollo de la métrica teórica

En este séptimo incremento se desarrolló un marco teórico que abalase la métrica seleccionada como apta para comparar y evaluar criptosistemas de ECC. Esta vez, el incremento se dividió en las tareas que se pueden apreciar en la Tabla 2.7, dando como resultado el Capítulo 5, que fue añadido a esta memoria tras pasar revisión por pares.

Tabla 2.7: Tareas del incremento 7

Id	Descripción
TA_7.1	Definición teórica de la métrica
TA_7.2	Teorema de aproximación asintótica
TA_7.3	Algoritmo de Montecarlo

Incremento 8 - Validación de la métrica

En este octavo incremento se desarrolló un estudio de validación sobre la métrica definida en el incremento anterior. Este –el incremento octavo– se dividió en las tareas que figuran en la Tabla 2.8, dando como resultado las Secciones 5.2 y 6.1, que fueron añadidas a la memoria después de pasar la revisión por pares. Se resalta también que los experimentos han sido programados siguiendo la metodología expuesta en este mismo capítulo.

Tabla 2.8: Tareas del incremento 8

Id	Descripción
TA_8.1	Implementación del algoritmo de Montecarlo
TA_8.2	Realización de experimentos
TA_8.3	Análisis de los resultados

Incremento 9 - Aplicación real

En este noveno, y último, incremento se desarrolló un ejemplo de aplicación sobre la métrica definida en el incremento séptimo. La subdivisión de tareas para este incremento puede encontrarse en la Tabla 2.8, dando como resultado las Secciones 5.3 y 6.2 de la memoria, que fueron añadidas a esta después de pasar la revisión por pares. Se resalta también que los experimentos han sido programados siguiendo la metodología expuesta en este mismo capítulo.

Tabla 2.9: Tareas del incremento 9

Id	Descripción
TA_9.1	Comparación de Z25519 vs P389
TA_9.2	Análisis de los resultados (II)

2.2. Cronograma

El periodo de ejecución total para este periodo ha sido fijado entre el 1 de septiembre de 2024 hasta 1 de septiembre de 2025. Esto significa que el proyecto deberá empezar y acabar dentro de esta ventana temporal, no pudiendo empezar antes ni acabar después. Como fecha de comienzo se toma el 1 de septiembre de 2024 fijando como plan de trabajo jornadas diarias de una hora, dando como resultado que la fecha de finalización estimada sea el 28 de junio de 2025. En las Figuras 2.5, 2.6, 2.7 y 2.8 se muestra el plan de trabajo, a priori, dividido por incrementos.

Las siguientes secciones del capítulo muestran los ajustes que se han debido realizar sobre el cronograma a lo largo del proyecto, justificando los motivos de cada uno de estos.

Tras estos ajustes, la duración final del proyecto ha sido de 317 horas, comenzando el 1 de septiembre de 2024 y finalizando el 1 de septiembre de 2025; teniendo que ser interrumpido en dos en el mes de enero y el mes de febrero por motivos externos.

2.2.1. Ajuste primero. Enero

El primer reajuste del cronograma fue llevado a cabo el 12 de enero de 2025. En estas fechas se tuvo que pausar la implementación del módulo de aritmética bajo la necesidad de atender un proyecto totalmente ajeno al presente. En dicho reajuste, se pauso la implementación del módulo de aritmética, siendo retomada el 1 de febrero de 2025. Las consecuencias de este cambio pueden ser apreciadas en las Figuras 2.9, 2.10 y 2.11, donde se aprecia el retraso en las tareas restantes del proyecto, lo que conlleva retraso de la fecha de finalización estimada al 20 de julio de 2025.

2.2.2. Ajuste segundo. Marzo

Al igual que pasó en Enero, el 4 de marzo de 2025 se reajustó el cronograma por la dedicación intensiva a otro proyecto ajeno a este, deteniéndose nuevamente la implementación del módulo de

aritmética y reanudando la implementación el 1 de abril de 2025. Nuevamente, esto de implicó nuevos retrasos sobre el plan inicial que pueden ser apreciados en las Figuras 2.12, 2.13 y 2.14, conllevando a postergar la fecha de finalización al 10 de agosto de 2025.

2.2.3. Ajuste tercero. Ejemplo de aplicación

Tras ver en sintonía la investigación, daba la sensación de que el trabajo realizado era demasiado teórico y la contribución realizada no poseía aplicación alguna. Al estar en fecha, se decidió añadir un objetivo más a la investigación, que era utilizar la teoría desarrollada para comparar criptosistemas usados en entornos de producción (**OB-06**). Esto conllevo un nuevo reajuste del cronograma que puede ser apreciado en las Figuras 2.15 y 2.16, conllevando nuevamente a un retraso en la fecha de finalización de proyecto, esta vez al 1 de septiembre de 2025.

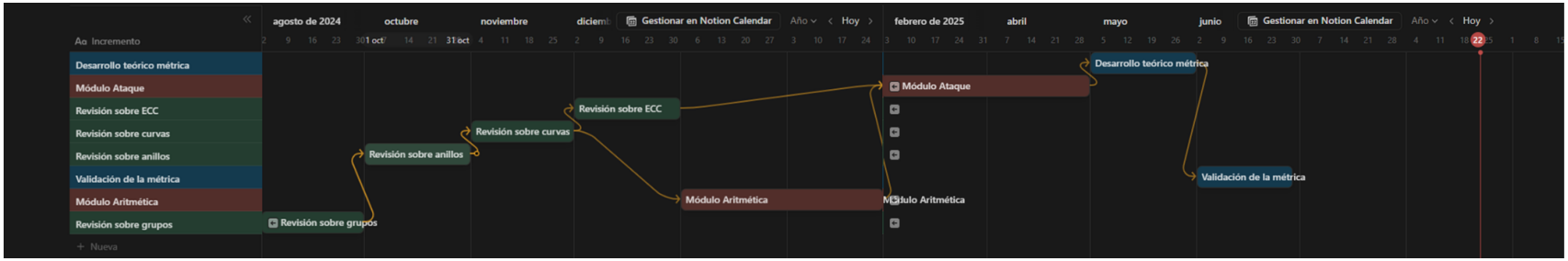


Figura 2.5: Plan de trabajo a priori por incrementos

Figura 2.6: Cronograma a priori para la revisión bibliográfica

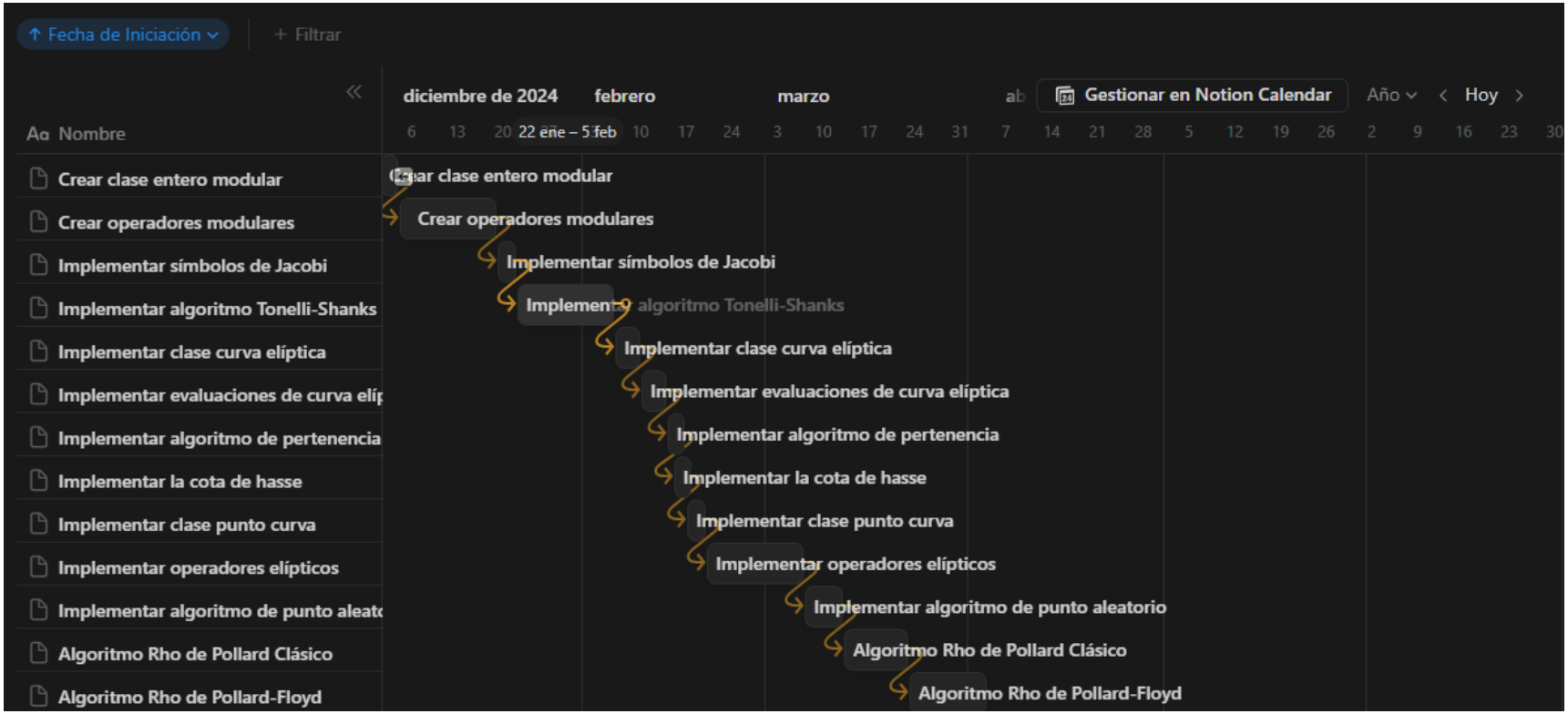


Figura 2.7: Cronograma a priori de la implementación de EllipticCUrVa

Figura 2.8: Cronograma a priori del marco experimental

Figura 2.9: Plan de trabajo tras primer reajuste por incrementos

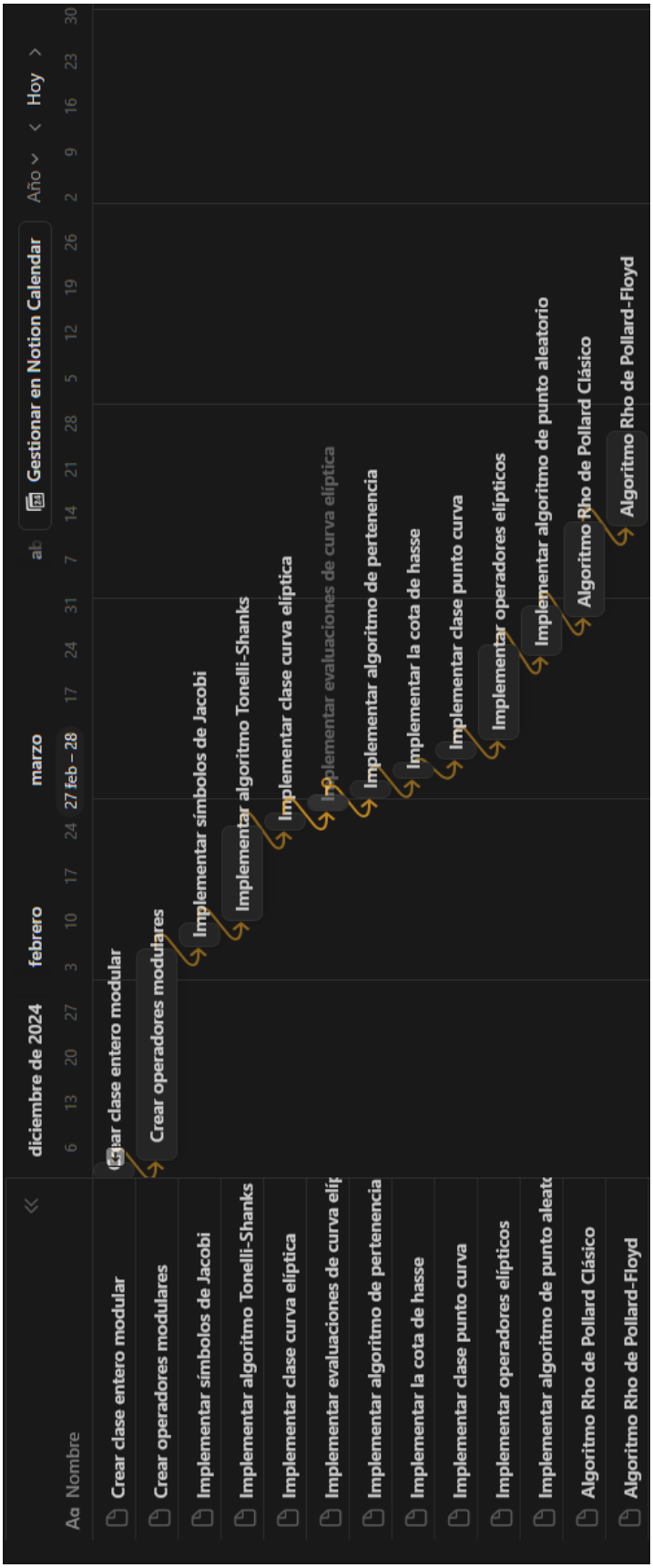


Figura 2.10: Cronograma de la implementación de EllipticCUrVa tras primer reajuste

Figura 2.11: Cronograma del marco experimental tras primer reajuste

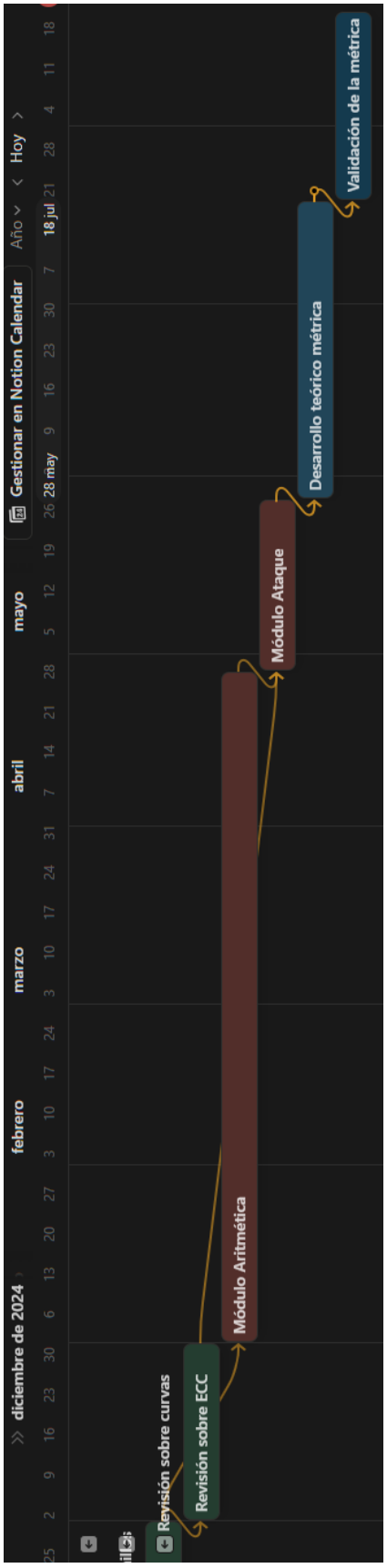


Figura 2.12: Cronograma de los incrementos tras segundo ajuste

Figura 2.13: Cronograma de la implementación de EllipticCUrVa tras primer reajuste

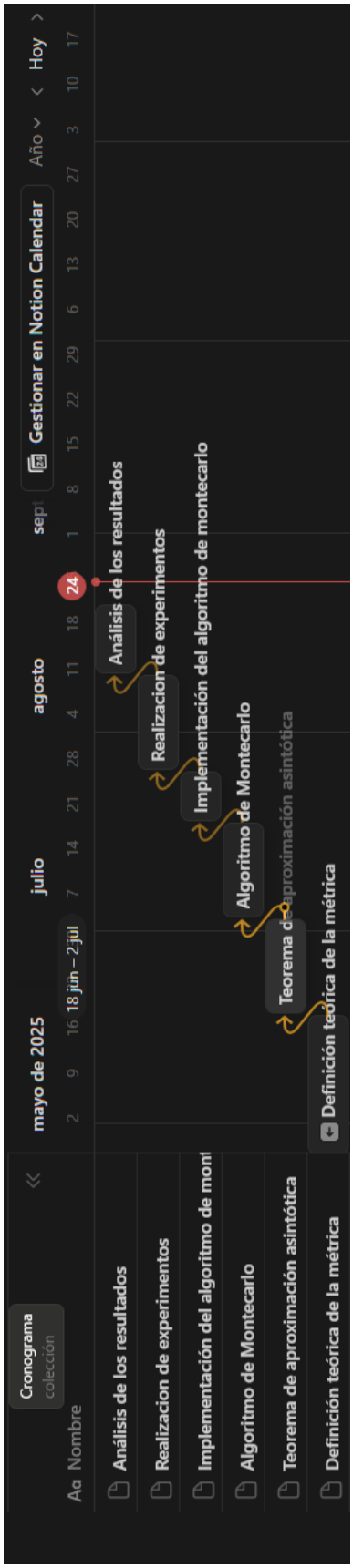


Figura 2.14: Cronograma del marco experimental tras segundo reajuste

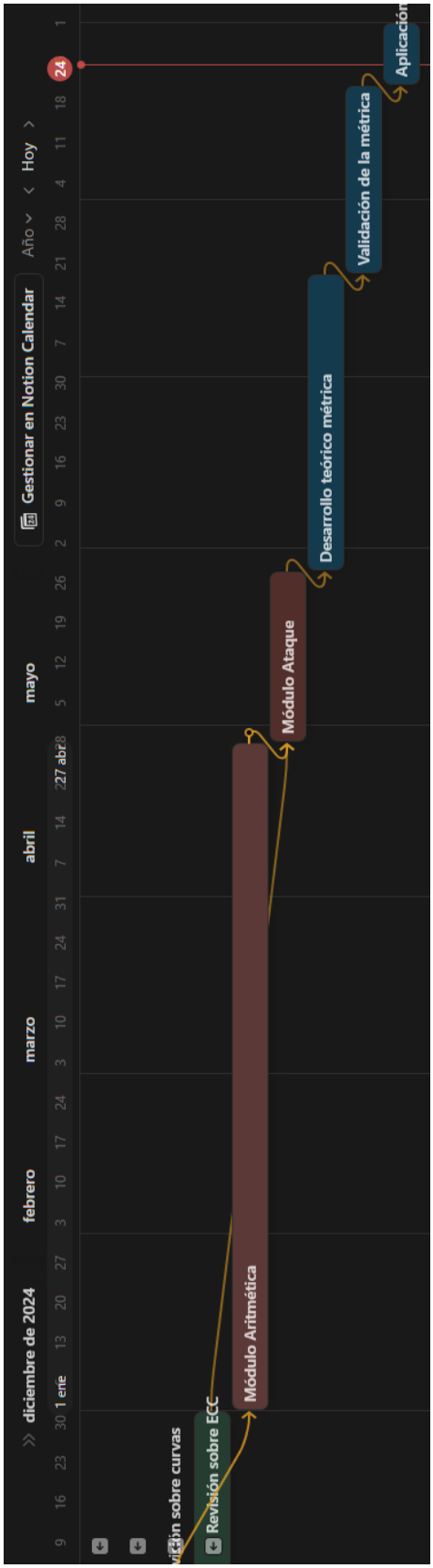


Figura 2.15: Cronograma de los incrementos tras el tercer reajuste

Figura 2.16: Cronograma del marco experimental tras el tercer reajuste

2.3. Presupuesto

En esta sección se presentan los presupuestos del estudio elaborado dividido en dos apartados: **Gastos de personal** y **Gastos fungibles**. Aunque no se ha desarrollado un gasto en material software, por utilizar licencias gratuitas y módulos *open source*, se muestra otro apartado de **Herramientas software utilizadas**.

Como apunte extra, el gasto de los suministros se prorroga durante el plazo de ejecución de todo el proyecto, del 1 de septiembre de 2024 al 1 de septiembre de 2025, previendo posibles cambios de cronograma respecto de la planificación inicial y evitando el desabastecimiento de estos.

2.3.1. Gastos de personal

Los gastos de personal se calculan teniendo en cuenta las 300 horas de trabajo en el TFG, estableciendo el pago la hora del personal no investigador en 9.93€/h (impuestos no incluido). Este precio ha sido extraído de la última nómina percibida por el autor, que puede ser encontrada en anexos.

Además, se realiza la contratación de un investigador de la UCM, Miguel Benito Parejo, con precio acordado de 1500€ por 40 horas de asesoría. De igual manera, se realiza la contratación de un investigador de la UVa, José Ignacio Farrán Martín, con precio acordado de 1500€ por 40 horas de asesoría.

La contratación del personal investigador se realiza bajo el marco legal del artículo 83 de la LOSU, encontrándose sus gastos desglosados en las Tablas 2.11 y 2.12 según indican los reglamentos específicos de cada universidad.

Tabla 2.10: Desglose de los gastos del personal

Personal	**Base Imponible (€/h)**	**Horas (h)**	**Total (€)**[1]
Eduardo Quintana	9.93€/h	300h	2,979.99€
Miguel Benito	32.73€/h	40h	1,309.50€
J.I. Farrán	31.87€/h	40h	1,275.50€
Total			**5,564.99€**

Tabla 2.11: Desglose de la contratación del personal investigador (UCM)

Descripción	**Retención (%)**	**Total (€)**
Precio de contratación		*1,500.00€*
Gastos Generales[2]	5 %	75.00€
Gestión Administrativa[2]	5 %	75.00€
Presupuesto al departamento		*1,350€*
Retención al trabajador[3]	3 %	40.50€
Retribución al trabajador		**1,309.50€**

[1] IVA incluido
[2] Sobre el precio de contratación
[3] Sobre el presupuesto al departamento

Tabla 2.12: Desglose de la contratación del personal investigador (UVa)

Descripción	**Retención (%)**	**Total (€)**
Precio de contratación		*1,500.00€*
Gestión administrativa	5 %	75.00€
Retención para el dpto.	2 %	30.00€
Retención para el centro	2 %	30.00€
Gastos de infraestructura	3 %	75.00€
Otras retenciones	3 %	75.00€
Retribución al trabajador		**1,275.50€**

Tabla 2.13: Desglose del personal trabajador

Descripción	**Retención (%)**	**Total (€)**
Precio de contratación		*4,082.19€*
Seguridad Social	6.48 %	264.53€
IRPF	20.52 %	837.67€
Retribución al trabajador		**2,979.99€**

2.3.2. Gastos fungibles

Dado que este proyecto no ha necesitado hardware específico, simplemente se calculará el internet y la luz prorrateados a lo largo del proyecto. A parte se calculará la amortización del equipo informático utilizado para llevar a cabo el trabajo.

Amortización de equipos informáticos

Se toma como vida útil de equipos informáticos, programas y demás herramientas para el tratamiento de la información un periodo de cinco años (véase [Tributaria, 2022]). Se señala que el sistema utilizado para este proyecto es un *Lenovo IdeaPad 320-15IKB - 81BG00KUSP* con un precio de salida al mercado de 874.00€ (IVA incluido). Se establece amortización lineal sobre el equipo señalado, multiplicando el costo del mismo por el porcentaje de horas en su vida útil utilizadas para el proyecto:

$$\text{Amortizacion} = \frac{300h}{43,800h} \cdot 874{,}00€ = 5{,}99€$$

Nota: En proyectos de investigación reales, la factura del equipo debería ser anexada a modo de justificación; sin embargo, como este documento es considerado información sensible y no hay firmado un NDA, se obvia este paso.

Prorrateo del internet

La cuota de internet contratada asciende a 27.99€ al mes, gasto acreditado mediante las facturas incluidas en el Anexo E. Dado que el servicio de internet está disponible las 24 horas del día, se ha optado por un prorrateo proporcional al tiempo de trabajo efectivo dedicado al proyecto. Considerando que este posee plazo de ejecución del 1 de septiembre de 2024 al 1 de septiembre de 2025 y que se han previsto jornadas de 1 hora diaria, lo que supone 365 horas de uso frente a las 8,760 horas anuales de disponibilidad del servicio. Aplicando este criterio de proporcionalidad, el gasto imputable al proyecto es:

$$\text{Gasto Internet} = \frac{365\text{ h}}{8,760\text{ h}} \cdot 335{,}88\text{ €} \approx 13{,}99\text{ €}$$

Prorrateo consumo eléctrico

El suministro eléctrico asociado al proyecto se encuentra bajo un plan estable con una tarifa fija de 0,151803 €/kWh, dato que puede ser contrastado en las facturas aportadas en el Anexo E. El equipo informático empleado en la ejecución presenta un consumo máximo de 45 Wh, siempre que se encuentre funcionando y conectado a la red. Considerando que el período de ejecución del proyecto abarca del 1 de septiembre de 2024 al 1 de septiembre de 2025, y que se han planificado jornadas de trabajo de una hora diaria, el gasto imputable en electricidad se obtiene multiplicando el número total de horas de dedicación por el consumo horario del equipo y el precio del kWh.

$$\text{Gasto electrico} = 365\text{ h} \cdot 0{,}151803\text{€/kWh} \cdot 0{,}045\text{kWh} \approx 2{,}49\text{€}$$

2.3.3. Herramientas software utilizadas

En la Tabla 2.15 se muestra un listado de herramientas software utilizadas. Están no se valoran dentro del presupuesto por ser *open source* o hacer uso de estas a través de una versión gratuita.

2.3.4. Presupuesto General

La Tabla 2.14 presenta de forma general los gastos de personal –desglosados en la Sección 2.3.1– y los gastos fungibles –desglosados en la Sección 2.3.2–.

Tabla 2.14: Presupuesto General

Gastos de Personal	7,082.19€
Gastos Fungibles	22.77€
Total	**7,104.96€**

Tabla 2.15: Herramientas software utilizadas

Nombre	**Descripción**	**Licencia**	**Versión**
Notion	Notion es una herramienta todo en uno para organizar notas, tareas y proyectos, que combina escritura, bases de datos y colaboración en equipo en un mismo espacio.	Gratuita	4.17.0.
VS Code	Visual Studio Code (VS Code) es un editor de código fuente gratuito y multiplataforma, desarrollado por Microsoft, que destaca por su rapidez, personalización mediante extensiones y soporte para múltiples lenguajes de programación.	Gratuita	1.103.1.
Python	Python es un lenguaje de programación de alto nivel, interpretado y multiparadigma, conocido por su sintaxis clara y fácil de leer.	Open Source	3.11.9.
Pandas	Pandas es una biblioteca de Python para manipulación y análisis de datos, especialmente útil para trabajar con tablas y series temporales.	Open Source	2.2.2.
NumPy	NumPy es una biblioteca de Python para cálculos numéricos y operaciones con arreglos multidimensionales.	Open Source	2.1.0.
SymPy	SymPy es una biblioteca de Python para matemática simbólica, permitiendo manipular y resolver expresiones algebraicas, ecuaciones y cálculos simbólicos.	Open Source	1.13.3
Plotly	Plotly es una biblioteca de Python para visualización interactiva de datos, creando gráficos y dashboards dinámicos.	Gratuita	6.1.2.
Overleaf	Es una plataforma en línea para editar documentos LaTeX de forma colaborativa, basada en la web.	Gratuita	Tex Live 2023

Parte II

Desarrollo de la propuesta y resultados

Capítulo 3

Antecedentes

Who controls the past controls the future. Who controls the present controls the past.

George Orwell

En este apartado se presentará el *background* teórico necesario para entender, tanto las operaciones que realiza EllipticCUrVa a bajo nivel, como el estudio realizado con esta misma.

Para una mayor organización, se ha dividido este punto en tres secciones separadas. La Secciones 3.1, 3.2 y 3.3 introducen las estructuras matemáticas necesarias sobre las que se fundamentan las curvas algebráicas, la aritmética de curva elíptica y el problema del logaritmo discreto generalizado (DLP). Este preámbulo se ve directamente complementado por la Sección 3.4, en el que se desarrolla de forma teórica tanto el concepto de curva elíptica como la ley de grupo inducida por estas, y se resaltan alguno de los resultados más notables. Finalizamos con la Sección 3.5 en la que se pone en práctica todo lo aprendido, introduciendo distintos criptosistemas y ataques sobre ECC.

3.1. Grupos

El principal cometido del álgebra es la creación de estructuras matemáticas con propiedades especiales asociadas. Estas estructuras generalizan conceptos cotidianos y las operaciones que se pueden ejercer sobre estos, como por ejemplo los números y cualquier tipo de operación aritmética sobre estos; o las palabras y la concatenación de estas. Entre todas estas estructuras, los **grupos** siempre han brillado por su alta aplicabilidad en todo tipo de ciencias, y en particular la criptografía.

Definición 3.1.1 (Grupo)**.** *Un **grupo** es una estructura matemática conformada por una dupla $(G, \cdot)$ .dónde G es un conjunto y $\cdot : G \times G \to G$ es una operación binaria satisfaciendo las siguientes propiedades:*

- ***Elemento Neutro:*** *Existe $e \in G$ tal que $a \cdot e = e \cdot a = a$ para todo $a \in G$*
- ***Elemento Inverso:*** *Para todo $a \in G$ existe $a^{-1} \in G$ tal que $a \cdot a^{-1} = a^{-1} \cdot a = e$, siendo e el elemento neutro mentado anteriormente.*
- ***Propiedad Transitiva:*** *Para todos $a, b, c \in G$ se tiene que $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.*

Si el par de la Definición 3.1.1 además cumple la **propiedad conmutativa**, para todo $a, b \in G$ se tiene que $a \cdot b = b \cdot a$, se dice que el grupo es **conmutativo** o **abeliano**.

Notación 3.1.1. *Sea $(G, \cdot)$ un grupo, a lo largo del escrito se cometerán los siguientes abusos de notación:*

- *Siempre que no suponga ninguna duda al lector, denotaremos el grupo $(G, \cdot)$ simplemente por G.*
- *El elemento neutro de un grupo general será referido mediante el carácter e.*
- *Dado un elemento $a \in G$, denotaremos su elemento inverso por a^{-1}.*
- *Dado un elemento $a \in G$ y un natural n, se define: a^n como el producto de a consigo mismo n veces, a^0 como e y a^{-n} como $(a^{-1})^n$.*
- *Dados $a, b \in G$, ante omisión se entenderá que $ab = a \cdot b$.*

Notación 3.1.2. *Sea $(G, \cdot)$ un grupo abeliano, a lo largo del escrito se cometerán los siguientes abusos de notación:*

- *El operador $\cdot$ será denotado por $+$.*
- *El elemento neutro asociado al grupo será denotado por 0.*
- *Dado un elemento $a \in G$, denotaremos su elemento inverso por $-a$.*
- *Dados $a, b \in G$, ante omisión, $a - b$ será interpretado como $a + (-b)$.*
- *Dados $a \in G$ y n natural se definen: na como el producto de a consigo mismo n veces, $0a$ como 0 y $-na$ como $n(-a)$.*

Mientras que la Notación 3.1.1 se puede aplicar a todo tipo de grupo, la Notación 3.1.2 se encuentra reservada única y exclusivamente a grupos abelianos. Se trabajará con una o con otra según sea conveniente a lo largo de la sección. Para asentear bien estos conceptos, se presentan algunos ejemplos típicos de grupos importantes en el ámbito de la criptografía.

Ejemplo 3.1.1 (Enteros)**.** *El conjunto de los números enteros junto a la suma conforman un grupo abeliano. Sin embargo, éste mismo junto al producto no conforma ni siquiera un grupo, pues por ejemplo el elemento inverso de 2 sería $1/2 \notin \mathbb{Z}$.*

Ejemplo 3.1.2 (Grupos cíclicos)**.** *Dado un entero positivo $n \in \mathbb{Z}$, se define conjunto $\mathbb{Z}_n := \{z \in \mathbb{Z} : 0 \leq z < n\}$. Sobre este conjunto se define la aplicación $\oplus : \mathbb{Z}_n \times \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ tal que $z_1 \oplus z_2 = (z_1 + z_2)$ mod n. El par $(\mathbb{Z}_n, \oplus)$ conforma un grupo abeliano conocido como **grupo cíclico** de n elementos.*

Gran parte de los algoritmos de clave pública se basan en la siguiente familia de grupos abelianos.

Ejemplo 3.1.3 (Grupos primos)**.** *Dado un primo positivo p, se define $\mathbb{F}_p^* = \{z \in \mathbb{Z} : 0 < z < p-1\}$ y $\odot : \mathbb{F}_p^* \times \mathbb{F}_p^* \longrightarrow \mathbb{F}_p^*$ de forma que $z_1 \odot z_2 = z_1 \cdot z_2$ mod p. El par $(\mathbb{F}_p^*, \odot)$ conforma un grupo abeliano.*

Se puede probar que para todo primo p existe un elemento $\alpha \in \mathbb{F}_p^*$ cuyas potencias sucesivas generan todos los elementos del grupo.

El siguiente ejemplo es de vital importancia para la criptografía de clave simétrica, sin embargo, necesitamos introducir el concepto de aplicación biyectiva antes.

Definición 3.1.2 (Función biyectiva)**.** *Dados dos conjuntos A y B; y $f : A \longrightarrow B$ una aplicación entre estos, diremos que es* ***biyectiva*** *si:*

- *Para cada $b \in B$ existe un elemento $a \in A$ tal que $f(a) = b$.*
- *Para cada par de elementos distintos $a_1, a_2 \in A$, $f(a_1) \neq f(a_2)$.*

Ejemplo 3.1.4 (Grupo simétrico)**.** *Sea S_n al conjunto de biyecciones de un conjunto de n elementos en si mismo y por o a la composición de funciones. El par (S_n, o) conforma un grupo no conmutativo conocido como* ***grupo de permutaciones*** *o* ***grupo simétrico*** *de n elementos.*

Gran parte de los algoritmos de clave simétrica se basan en el grupo de permutaciones. En estos, el texto a cifrar se divide en bloque de n bits y a cada uno de estos se les aplica un conjunto de permutaciones seguras sobre S_n. Un ejemplo concreto se puede encontrar en el RFC [Karn et al., 1995] donde se desarrollan todos los tecnicismos asociados al algoritmo **Triple Data Encryption Standard** (3DES). En este se divide el texto a cifrar en bloques de 64 bits y se aplican permutaciones asociadas a S_{64}, intercalando entre medias funciones no lineales, que ayudan a mejora la seguridad del cifrado.

3.1.1. El problema del logaritmo discreto

La criptografía de curva elíptica se basa en inducir un grupo abeliano sobre los elementos de una curva elíptica y escoger un punto generador sobre el cual resolver el *problema del logaritmo discreto* sea computacionalmente inviable (véase Definición 3.5.5). En esta sección se ofrecerá una visión abstracta de este problema asentando los principios para las Secciones 3.4.3 y 3.5 donde se tratará en mayor profundidad el problema del logaritmo discreto de curva elíptica (ECDLP) y se mostrará la influencia de este sobre las primitivas relativas a la criptografía de curva elíptica.

Definición 3.1.3 (Logaritmo Discreto)**.** *Sea G un grupo abeliano y $g, h \in G$ elementos. Se define el* ***logaritmo discreto*** *en base g de h como el menor entero no negativo n que cumple $ng = h$. En caso de que ningún entero no negativo satisfaga $ng = h$ convendremos que $\log_g h = -1$.*

Se introducen un par de ejemplos para mejora la comprensión de la definición anterior.

Ejemplo 3.1.5. *En $\mathbb{F}_7^*$ el logaritmo discreto en base 5 de 2 es 4, puesto que $5^2 = 4 \ (mod\ 7)$, $5^3 = 6 \ (mod\ 7)$, $5^4 = 2 \ (mod\ 7)$. Por otro lado, en $\mathbb{Z}/4\mathbb{Z}$ el logaritmo discreto en base 2 de 3 es -1, esto se debe a que $2 \cdot 2 = 0 \ (mod\ 4)$ y $3 \cdot 2 = 2 \ (mod\ 4)$.*

No existen reglas de resolución generales para el problema del logaritmo discreto, pudiendo ser abordado únicamente mediante fuerza bruta o algoritmos probabilísticos. El número de iteraciones promedio necesarias para la convergencia del algoritmo es directamente proporcional al número de elementos distintos generados por las potencias del elemento base. Es por esto que a medida que este crece, más difícil será de resolver el problema. En la Secciones 3.1.2 y 3.1.3 se explorarán la estructura que tiene el conjunto de potencias de los elementos de un grupo y su cardinal. De igual manera, en la Sección 3.5 hablaremos de estrategias para resolver específicamente el problema de curva elíptica.

3.1.2. Subgrupos y grupos generados

Definición 3.1.4 (Subgrupo)**.** *Sea $(G, \cdot)$ un grupo y $H \subset G$. Diremos que H es subgrupo de G si para todos $g_1, g_2 \in H$ se tiene que $g_1 g_2^{-1} \in H$ y lo expresaremos como $H < G$.*

Proposición 3.1.1. *Sea $(G, \cdot)$ un grupo y $H < G$. El par $(H, \cdot)$ posee estructura de grupo.*

La prueba de la Proposición 3.1.1 puede encontrarse en [Judson, 2020]. Los dos conceptos presentados son de vital importancia pues sirven para probar que, dado un grupo G y un elemento a sobre este tomado sin pérdida de generalidad, el conjunto de potencias conforma un subgrupo de G. Efectivamente, denotemos por $\langle a \rangle$ al conjunto de potencias de a y tomemos $b_1, b_2 \in \langle a \rangle$. Por caracterización, existen $n_1, n_2 \in \mathbb{Z}$ tales que $b_1 = a^{n_1}$ y $b_2 = a^{n_2}$, aplicando esto se tiene que:

$$b_1 b_2^{-1} = \underbrace{a \cdot ... \cdot a}_{n_1 \text{ veces}} \cdot (\underbrace{a \cdot ... \cdot a}_{n_2 \text{ veces}})^{-1} = a^{n_1 - n_2}$$

Una vez probado esto, de la Proposición 3.1.1 se sigue que $\langle a \rangle$ posee estructura de grupo, además, si G era abeliano entonces $\langle a \rangle$ lo será también por herencia.

Notación 3.1.3 (Grupo generado)**.** *Sea G un grupo y $a \in G$ denotamos por $\langle a \rangle$ al grupo generado por las potencias de a.*

3.1.3. El orden de un grupo

En esta última sección sobre grupos se expondrán resultados fundamentales sobre el cardinal asociado a su soporte, conocido como *orden*, en este contexto particular.

Definición 3.1.5 (Orden)**.** *El **orden** de un grupo G es la cantidad de elementos que este posee y se denota por $|G|$. Por otro lado, el orden de un elemento $g \in G$ se define como el menor entero positivo n cumpliendo $g^n = e$ y se denota por $ord(g)$. Por conveniencia, diremos que $ord(g) = \infty$ si $g^n \neq e$ para todo entero positivo n.*

Los elementos con orden finito sobre un grupo son especiales pues tienden a cumplir propiedades no convencionales. Por ejemplo, dado un grupo G sin pérdida de generalidad, sus elementos de orden finito conforman un subgrupo *normal*[1]. De igual manera, el conjunto de elementos de orden menor o igual m conforman un subgrupo normal de G.

Definición 3.1.6 (Torsión)**.** *El conjunto de elementos de orden finito sobre un grupo G se conoce **torsión** y es denotado por $\tau(G)$. De igual manera, el conjunto de elementos de orden menor o igual que m sobre G se conoce como **m-torsión** y se denota por $G[m]$.*

Es claro que si $H < G$, entonces, por contención, $|H| < |G|$. Lo que no es tan obvio es que, el vínculo existente entre el orden de un grupo y el de sus subgrupos no queda ahí. El Teorema de Lagrange es un fuerte resultado que explora la relación entre estos factores, demostrando que el orden de un subgrupo siempre es divisor del orden del grupo:

Teorema 3.1.1 (Teorema de Lagrange)**.** *Dados $H < G$, el orden de H divide al de G, es decir, $|H| \mid |G|$.*

La demostración de este resultado no es compleja, pero se necesita trabajar con el concepto de *clase lateral* y esto queda fuera de los objetivos de este trabajo. El lector interesado podrá encontrar en [Judson, 2020] la prueba del mismo.

A modo de corolario, del resultado anterior se puede probar que $ord(a) = |\langle a \rangle|$. Efectivamente, denotemos por $ord(a) = m$, por $|\langle a \rangle| = n$ y razonemos por reducción al absurdo.

- Supongamos que $m > n$, entonces por caracterización $e, a, a^2, ..., a^{m-1}$ deberían estar en $\langle a \rangle$. Por otro lado, como $n < m$ entonces existen $0 < n_1 < n_2 \leq m$ tales que $a^{n_1} = a^{n_2}$. Multiplicando ahora por a^{-n_1} a ambos lados de la igualdad anterior se tiene que $a^{n_2 - n_1} = e$ y dado que $n_2 - n_1 \leq m - n_1 < m$ llegamos a absurdo pues $ord(a) = m$.

[1] Se dice que $N < G$ es normal si para todos $g \in G$ y $n \in N$ se cumple $gng^{-1} \in N$.

- Ahora supongamos que $n > m$. Fijémonos que como $ord(a) = m$, entonces toda potencia puede ser reescrita[2] como a^k con $0 \leq k < m$. Tomemos $g_1, ..., g_n$ los n elementos de $\langle a \rangle$. Por caracterización existen esteros $n_1, ..., n_n$ satisfaciendo $g_1 = a^{n_1}, ..., g_n = a^{n_n}$. A su vez, por la afirmación anterior, existen enteros $m_1, ..., m_n$ entre 0 y m cumpliendo $g_1 = a^{n_1} = a^{m_1}, ..., g_n = a^{n_n} = a^{m_n}$. Ahora bien, como tenemos n elementos, estos pueden ser caracterizados de m formas distintas y $n > m$ entonces, por el Principio del Palomar, dos deben ser el mismo, llegando así a absurdo.

Esto implica que, el número de elementos distintos engendrados por un elemento a es igual al orden del mismo. Este resultado justifica que, cuanto más grande sea el orden del elemento base escogido, más complejo será el problema del logaritmo discreto.

Otro importante resultado sobre órdenes de grupos finitos conmutativos es el *teorema de caracterización.*

Teorema 3.1.2 (Caracterización de grupos finitos abelianos)**.** *Todo grupo finito abeliano es isomorfo a un producto de grupos cíclicos. Es decir, sea G un grupo abeliano finito entonces existen primos $p_1, ..., p_n$ (no necesariamente distintos) y naturales $\alpha_1, ..., \alpha_n$ tales que:*

$$G \simeq \mathbb{Z}_{p_1^{\alpha_1}} \times ... \times \mathbb{Z}_{p_n^{\alpha_n}}$$

Que dos grupos sean isomorfos significa que existe un isomorfismo entre estos. Un isomorfismo de grupos es una aplicación biyectiva entre los elementos de dos grupos, $\phi : G_1 \longrightarrow G_2$, respetando la operación de grupos, es decir para cada $g_1, g_2 \in G_1$ se cumple $\phi(g_1 \cdot g_2) = \phi(g_1)\phi(g_2)$. En [Judson, 2020] puede encontrarse la prueba de este teorema. Esta demostración no va a ser desarrollada dentro de la sección puesto que se necesita del *teorema de isomorfía de Noether*, resultado altamente abstracto que se escapa de los objetivos de este trabajo.

El Teorema 3.1.2 ofrece una vía para la elección de elementos base sobre el problema del logaritmo discreto. Para ilustrar esto, se muestra el siguiente ejemplo:

Ejemplo 3.1.6. *Sea G un grupo abeliano finito y supongamos que existen coprimos n_1, n_2 tales que $G \simeq \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, es decir, existe un isomorfismo $\phi : \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \longrightarrow G$. Es claro que el elemento $(1, 1) \in \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ posee orden maximal así pues, la imagen del isomorfismo ϕ sobre este, $\phi(1, 1)$ dará lugar a un elemento de orden maximal sobre G.*

En el ejemplo anterior, es claro que se necesita conocer la descomposición en grupos cíclicos asociada al grupo de estudio. Softwares como **Groups, Algorithms, Programming (GAP)** [GAP, 2024] han tratado esta cuestión desde hace años, ofreciendo funcionalidades específicas para calcular la descomposición de grupos abelianos finitos en producto de cíclicos.

3.2. Anillos y Cuerpos

Una vez hemos entendido el concepto de grupo, debemos pasar a entender el concepto de curva. Sin embargo, antes de entender esto deberán introducirse el concepto de anillo y cuerpo. El concepto de anillo es relativamente reciente, utilizado primeramente por Hilebert en el año 1897 en su libro *Theorie der algebraischen Zahlkörper.*

Definición 3.2.1 (Anillo)**.** *Sean A un conjunto, $+ : A^2 \longrightarrow A$ una operación binaria denotada **suma** y $\cdot : A^2 \longrightarrow A$ una operación binaria denotada **producto**. Se dice que la terna $(A, +, \cdot)$ posee estructura de **anillo** si cumple las siguientes propiedades:*

[2]Efectivamente, sea h un entero cualquiera, si $h \geq m$ entonces este se puede reescribir como $h = pm + t$, con p y t enteros y $0 \leq t < m$, entonces $a^h = a^{pm+t} = a^{pm}a^t = (a^m)^p a^t = e^p a^t = a^t$. Supongamos ahora que $h < 0$, aplicando el mismo razonamiento que antes, existe p y t enteros con $0 \leq t < m$ tales que $|h| = pm + t$, desarrollando esto tenemos que $a^{-pm-t} = e^{(p+1)m}a^{-pm-t} = a^{(p+1)m}a^{-m-t} = a^{m-t}$ y claramente $0 < m - t \leq m$.

- *El par* $(A, +)$ *posee estructura de grupo abeliano.*
- *El producto satisface la propiedad transitiva, es decir:*

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c, \quad \forall\, a, b, c \in A$$

- *El producto cumple la propiedad distributiva respecto de la suma, es decir:*

$$a \cdot (b + c) = a \cdot b + a \cdot c, \quad \forall\, a, b, c \in A$$

$$(a + b) \cdot c = a \cdot c + b \cdot c, \quad \forall\, a, b, c \in A$$

Cuando además de estas propiedades existe un elemento neutro para el producto, es decir, existe $1 \in A$ tal que $1 \cdot a = a \cdot 1 = a, \quad \forall\, a \in A$ se dice que el anillo es **unitario**. De igual manera, se dice que A es **conmutativo** cuando el producto cumple la propiedad conmutativa, es decir $a \cdot b = b \cdot a, \quad \forall\, a, b \in A$.

Notación 3.2.1. *Sea* $(A, +, \cdot)$ *un anillo genérico. A lo largo de la sección se cometerán los siguientes abusos de notación, siempre que no dé lugar a confusión:*

- *El anillo* $(A, +, \cdot)$ *será denotado simplemente por* A.
- *Sean* A *un anillo y* $a, b \in A$ *elementos cualesquiera. Ante omisión se entenderá que* $ab = a \cdot b$.
- *El inverso de un elemento* $a \in A$ *para la suma será denotado por* $-a$ *y será referido como* ***opuesto*** *de* a.
- *El elemento neutro para la suma será denotado por* 0.
- *En caso de existir, el elemento neutro para el producto será denotado por* 1.
- *Sean* $a \in A$ *y* $n \in \mathbb{N}$ *se define* $na = \underbrace{a + ... + a}_{n \text{ veces}}$, $0a = 0_A$ *y* $-na = \underbrace{(-a) + ... + (-a)}_{n \text{ veces}}$.
- *Sean* $a \in A$ *y* $n \in \mathbb{N}$ *se define* $a^0 = 1$ *y* $a^n = \underbrace{a \cdot ... \cdot a}_{n \text{ veces}}$

Es lícito preguntarse si en un anillo unitario A puede existir el inverso del producto. Para contestar esta pregunta aparentemente sencilla se creó una disciplina conocida como **Teoría de Galois**. En esta se introduce el concepto de cuerpo siendo un anillo en el que todo elemento salvo el cero es inversible para el producto. A pesar de lo que indica la razón, los cuerpos son conceptos anteriores los anillos, habiendo sido anticipados por **Évariste Galois** en el año 1832 a través de una definición alternativa a la utilizada en este texto.

Definición 3.2.2 (Unidad)**.** *Sea* A *un anillo unitario. Diremos que un elemento* $a \in A$ *es* ***unidad****, si existe* $b \in A$ *tal que* $ab = ba = 1$. *En este caso diremos que el elemento* b *es el* ***inverso*** *de* a *y lo denotaremos por* a^{-1}.

Definición 3.2.3 (Cuerpo)**.** *Un anillo unitario* K *es un* ***cuerpo*** *si todo elemento salvo* 0 *es unidad.*

Para ilustrar mejor las definiciones anteriores se mostrarán una serie de ejemplos:

Ejemplo 3.2.1 (Enteros)**.** *El conjunto de los números entero junto a la suma y productos usuales conforman un anillo conmutativo y unitario en el que no todo elemento tiene inverso.*

Ejemplo 3.2.2 (Pares)**.** *El conjunto de los números pares junto a la suma y producto usual conforman un anillo conmutativo no unitario.*

Ejemplo 3.2.3 ($\mathbb{Z}$ módulos)**.** *Los conjuntos $\mathbb{Z}_n$ (véase Ejemplo 3.1.2) junto a las suma $\oplus : \mathbb{Z}_n \times \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ y el producto $\odot : \mathbb{Z}_n \times \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$, definidas como $z_1 \oplus z_2 := (z_1 + z_2)\ mod\ n$ y $z_1 \odot z_2 := z_1 z_2\ mod\ n$, conforman un anillo conmutativo y unitario.*

Ejemplo 3.2.4 (Cuerpo primo)**.** *Se puede probar en el ejemplo anterior que, si p es primo, entonces el anillo $\mathbb{Z}_p$ es un cuerpo conmutativo.*

Para todo primo p, los cuerpos del Ejemplo 3.2.4 serán denotados por $\mathbb{F}_p$ y nos referiremos a estos por **cuerpos primos** de p elementos.

Los cuerpos conforman un sistema aritmético completo[3] sobre el que vamos a construir el concepto de curva, pero para ello, antes se debe construir el concepto de ecuación. En la siguiente sección se tratarán los anillos de polinomios, preámbulo que servirá de introducción al concepto de curva.

3.2.1. Potencias en cuerpos primos

Aunque el módulo presentado esta preparado para escalar a otro tipo cuerpos, en esta primera versión sólo se trabajará con cuerpos primos, aprovechando muchas de las propiedades existentes en estos.

Teorema 3.2.1 (Pequeño Teorema de Fermat)**.** *Sea $\mathbb{F}_p$ un cuerpo primo y $z \in \mathbb{F}_p$ distinto de cero, entonces se cumple la siguiente identidad:*

$$z^{p-1} = 1\ mod\ p$$

Esta propiedad es de vital importancia en la optimización de los algoritmos de calculo de potencias modulares. El lector astuto se percatará que $z^n = z^{n \bmod p-1}$ para todo elemento $z \in \mathbb{F}_p$.

Aplicando esta primera idea y un esquema recursivo, tal y como presentan [Menezes et al., 2018] y como se muestra en el Figura 3.1, se puede calcular la n-ésima potencia de un número z módulo p en $O(\log_2(n \bmod p))$ operaciones. Resaltamos que este mismo algoritmo es aplicable a cualquier cuerpo conmutativo, sin el factor de aceleración dado por el Teorema 3.2.1.

[3]Esto significa que sobre este sistema aritmético se puede sumar, restar, multiplicar y dividir.

```
def potencia(base : int,
             exponente : int,
             caracteristica : int) -> int:

    exponente %= (caracteristica - 1)

    if exponente == 0:
        return 1

    elif exponente == 1:
        return base

    elif (exponente % 2) == 0:
        return potencia((base * base) % caracteristica,
                        exponente // 2,
                        caracteristica)
    else:
        return potencia((base * base) % caracteristica,
                        exponente // 2,
                        caracteristica) * base
```

Figura 3.1: Algoritmo de exponenciación modular

3.2.2. Anillos de polinomios

Aunque los conceptos de anillo y cuerpo ofrecen un marco general para definir sistemas aritméticos, no son suficientes para definir el concepto de curva. En ramas analíticas, las curvas vienen definidas local o totalmente en términos de expresiones implícitas o paramétricas. Por esto mismo que necesitamos construir expresiones algebraicas abstractas sobre estos sistemas aritméticos.

Los anillos de polinomios son estructuras algebraicas que generalizan las expresiones polinómicas sobre un sistema aritmético inducido por un anillo o un cuerpo. En lo que sigue se definirá formalmente este concepto y se trabajará con alguna de sus propiedades principales.

Definición 3.2.4 (Anillo de Polinomios)**.** *Sea A un anillo, se define el* ***anillo de polinomios*** *con coeficientes sobre A, como el conjunto de sucesiones de elementos de A en las que solo un número finito de elementos son distintos de 0. Además los elementos sobre $A[X]$ son conocidos como* ***polinomios*** *con coeficientes en A.*

Para trabajar de forma más amena con el concepto de polinomio se introduce la siguiente notación:

Notación 3.2.2. *Sea A un anillo, $A[X]$ su anillo de polinomios asociado y $\{a_i\}_{i\in\mathbb{Z}^+} \in A[X]$. Sean además $i_1, ..., i_n$ todos los índices sobre los que no se anulan los elementos de $\{a_i\}_{i\in\mathbb{Z}^+}$, se establece la siguiente notación para el polinomio P:*

$$P(X) = a_{i_1}X^{i_1} + ... + a_{i_n}X^{i_n}$$

Además, por conveniencia se define $X^0 = 1_A$ y $0_A \cdot X = 0_A$.

Definición 3.2.5 (Suma de polinomios)**.** *Sean $A[X]$ un anillo de polinomios y $P, Q \in A[X]$ elementos sobre este tales que:*

$$P(X) = a_0 + a_1X + ... + a_nX^n, \quad a_0, ..., a_n \in A$$

$$Q(X) = b_0 + b_1X + ... + b_mX^m, \quad b_0, ..., b_m \in A$$

Se define la ***suma de polinomios*** *como:*

$$P(X) \oplus Q(X) := \sum_{i=0}^{\text{máx}\{n,m\}} (a_i + b_i)X^i$$

Definición 3.2.6 (Producto de polinomios)**.** *Sean $A[X]$ un anillo de polinomios y $P, Q \in A[X]$ elementos sobre este tales que:*

$$P(X) = a_0 + a_1X + ... + a_nX^n, \quad a_0, ..., a_n \in A$$

$$Q(X) = b_0 + b_1X + ... + b_mX^m, \quad b_0, ..., b_m \in A$$

Se define el ***producto de polinomios*** *como:*

$$P(X) \odot Q(X) := \bigoplus_{\substack{0 \leq i \leq n \\ 0 \leq j \leq m}} (a_i + b_j)X^{i+j}$$

Entendiendo $\bigoplus$ como sumatorio aplicando la suma de polinomios.

Teorema 3.2.2. *Para todo anillo A se cumple que la terna $(A[X], \oplus, \odot)$ posee estructura de anillo. Además, si A es conmutativo o unitario, $A[X]$ también lo será, respectivamente.*

La demostración del teorema anterior puede ser encontrada en [Judson, 2020].El teorema anterior construye un marco para operar sobre expresiones polinómicas con coeficientes en un anillo genérico. Fijémonos que puedo aplicar el razonamiento anterior de forma recursiva de manera que, si sobre todo anillo de polinomios $A[X]$, se puede definir a su vez un anillo de polinomios $A[X, Y] := A[X][Y]$ sobre este. Esto permite trabajar en sistemas aritméticos con expresiones polinómicas con tantas indeterminadas como sea necesario.

Ahora que ya tenemos expresiones, debemos pasar a poder evaluarlas. Para ello introducimos el concepto de morfismo de evaluación:

Definición 3.2.7 (Morfismo de Evaluación)**.** *Sea $A[X]$ y $a \in A$, el* ***morfismo de evaluación*** *inducido por a se define como:*

$$\phi_{\boldsymbol{a}} : A[X] \to A, \quad P(X) = \sum_{i=0}^{n} b_iX^i \mapsto \sum_{i=1}^{n} b_ia^i$$

Construidos los anillos de polinomios, pasamos a introducir el concepto de grado que será vital para definir las curvas elípticas.

Definición 3.2.8 (Grado)**.** *Sea $P(X_1, ..., X_n) \in A[X_1, ..., X_n]$ se define su grado como la mayor suma de exponentes de cada producto de indeterminadas que lo conforma. Este valor se denota por $\delta(P)$.*

Para ilustrar mejor la definición anterior, pongamos un ejemplo:

Ejemplo 3.2.5. *Supongamos que tenemos el polinomio $P(X) = 3X^2 + 5X \in \mathbb{R}[X]$, entonces $\delta(P) = 2$. Por otro lado, si tengo el polinomio $Q(X, Y) = 7X^2Y^2 + 3Y^3 + X^2 \in \mathbb{R}[X, Y]$ entonces su grado es $\delta(Q) = 4$ puesto que la suma de exponentes en $7X^2Y^2$ es mayor que en $3Y^3$ y que X^2.*

3.2.3. Radicales sobre cuerpos primos

Introducidos las potencias y los polinomios sobre anillos, restan los radicales, es decir, dado un cuerpo primo $\mathbb{F}_p$ y un elemento t sobre este, los radicales de t serán las soluciones del polinomio $X^2 + t$.

Se debe resaltar, que al igual que ocurre con los reales, no todo elemento $t \in \mathbb{F}_p$ admite radical. Introducimos así la siguiente definición.

Definición 3.2.9. *(Residuo Cuadrático) Dado un $t \in \mathbb{F}_p$ diremos que es un **Residuo Cuadrático** si existe $a \in \mathbb{F}_p$ tal que $a^2 = t$.*

En el contexto de teoría de números, los **Símbolos de Legendre** son una herramienta fundamental que nos permite evaluar cuando un elemento o no residuo cuadrático.

Definición 3.2.10 (Símbolo de Legendre)**.** *Dado $t \in \mathbb{F}_p$ definimos su **Símbolo de Legendre** como:*

$$\left(\frac{t}{p}\right) = \begin{cases} 0, & si\ t = 0\ mod\ p \\ 1, & si\ t\ es\ residuo\ cuadratico \\ -1, & si\ t\ no\ es\ residuo\ cuadratico \end{cases}$$

A nivel formal, los símbolos de Legendre son sumamente útiles para enunciar resultados, pero es necesario un método de cálculo que nos permita conocer su valor. Introducimos de esta manera el **Criterio de Euler**.

Teorema 3.2.3 (Criterio de Euler)**.** *Sea $t \in \mathbb{F}_p$, su símbolo de Legendre viene dado por la siguiente expresión:*

$$\left(\frac{t}{p}\right) = t^{(p-1)/2}\ mod\ p$$

Aquel lector que desee estudiar más profundamente los símbolos de Legendre, y su generalización, los símbolos de Jacobi, en [Martín, 2024].

Los resultados enunciados hasta el momento, nos sirve para identificar los residuos cuadráticos sobre un cuerpo primo particular, pero no conforman ningún método de cálculo para su radical. Nuevamente, en [Martín, 2024] podemos encontrar un algoritmo que responde a esta necesidad, el algoritmo de Shanks-Tonelli. La idea subyacente detrás de este algoritmo es construir una sucesión de valores sobre $\mathbb{F}_p$ que converge a un radical de t en sentido estacionario[4].

Algoritmo 3.2.1 (Shanks-Tonelli)**.**

Input: *El residuo cuadrático a y la característica p del cuerpo.*

Output: *El radical del elemento a.*

- ***Paso 1.*** *Si $a = 1$ o $a = 0$ se devuelve a.*
- ***Paso 2.*** *Se calcula el criterio de Euler sobre a, si el resultado es -1 se lanza una excepción controlada.*
- ***Paso 3.*** *Se factoriza $p - 1$ como $s \cdot 2^e$.*

[4]Una sucesión de valores $\{x_n\}_{n\in\mathbb{N}}$ converge en sentido estacionario a x si existe N natural tal que $x_n = x$ para todo $n > N$.

- ***Paso 4.*** *Se calcula el primer elemento q que no sea residuo cuadrático entre* $[2, p-1]$.
- ***Paso 5.*** *Se inicializan, en el orden expuesto, las siguientes variables:*

$$x = t^{\lfloor (s+1)/2 \rfloor} \ mod\ p$$

$$b = t^s \ mod\ p$$

$$g = q^s \ mod\ p$$

$$r = e$$

- ***Paso 6.*** *Se inicializan una variable* $m = 0$.
- ***Paso 7.*** *Si* $b = 1$ *se devuelve* x.
- ***Paso 8.*** *Si* $b \neq 1$ *se inicializa* $aux = b$.
- ***Paso 9.*** *Se actualizan las siguientes variables:*

$$aux = aux^2 \ mod\ p$$

$$m = m + 1$$

- ***Paso 10.*** *Se repite el paso 9 hasta que* $aux \neq 1$ *o* $m = r$.
- ***Paso 11.*** *Si* $m \neq 0$ *se actualizan las siguientes variables:*

$$x = x \cdot g^{2^{r-m-1}} \ mod\ p$$

$$g = g^{2^{r-m}} \ mod\ p$$

$$b = b \cdot g \ mod\ p$$

- ***Paso 12.*** *Si* $b \neq 1$ *se actualiza la variable* $r = m$.
- ***Paso 13.*** *Si* $m \neq 0$ *y* $b \neq 1$ *se vuelve al Paso 6, manteniendo los valores actuales.*

Con este algoritmo se da por acabada la sección actual, presentando un marco completo para trabajar con aritmética sobre cuerpos finitos. Se recomienda la lectura de [Apostol, 2020] a aquellos interesados en este tópico y que desean profundizar más aún en la teoría de números.

3.3. Curvas algebraicas

Una vez definidas una forma de aritmética abstracta y expresiones polinómicas sobre esta, se cumplen las condiciones para introducir el concepto de curva algebraica pero antes de esto se debe aclarar que **a partir de ahora todo anillo con el que se trabaje será un cuerpo conmutativo**.

Definición 3.3.1 (Relación de polinomios equivalentes)**.** *Sea* $K[X_1, ..., X_n]$ *un anillo de polinomios, se define la relación* $\mathcal{R}$ *sobre estos como:*

$$\mathcal{R} : P \sim Q \iff \exists\ a \in K \mid aP = Q$$

Se puede probar que la relación presentada en la definición anterior es de equivalencia y se invita al lector interesado a consultar [Fulton, 2008] para prestar atención a los detalles.

Definición 3.3.2 (Curva algebraica plana)**.** *Sea $K[X,Y]$ un anillo de polinomios, una **curva algebraica** sobre K^2 es una clase de equivalencia sobre $K[X,Y]/\mathcal{R}$, siendo $\mathcal{R}$ la relación de equivalencia definida en la anterior.*

En la definición anterior los representantes canónicos toman el nombre de **ecuaciones implícitas**. Se cometerá un abuso de notación identificando las curvas con sus ecuaciones implícitas. Por ejemplo, si $\overline{P(X,Y)} = \overline{X+Y+7}$ es una curva, la denotaremos simplemente por $P(X,Y) = X+Y+7$.

Desde este punto de vista, las curvas se antojan un concepto puramente algebraico, sin embargo se puede construir un concepto geométrico equivalente a través de los puntos que satisfacen las ecuaciones de una curva.

Definición 3.3.3 (Soporte)**.** *Sea $\mathcal{C}$ una curva sobre K^2 y $P(X,Y) \in \mathcal{C}$, su **soporte** se define como:*

$$\mathcal{C}(K) := \{(a,b) \in K^2 : P(a,b) = 0\}$$

Cabe destacar que, el soporte de una curva es totalmente independiente de su representante canónico seleccionado. Efectivamente, sea $\mathcal{C}$ una curva plana y $P, Q \in \mathcal{C}$. Por caracterización se tiene que $P = aQ$ para cierto $a \in K$. Sea $(x,y) \in \mathcal{C}(K)$ entonces $aQ(x,y) = P(x,y) = 0$, multiplicando por a^{-1} a ambos lados se llega a que $P(x,y) = 0$, probando que la definición de soporte es independiente del representante canónico elegido.

En [Fulton, 2008] se prueba la existencia de una biyección entre las curvas algebraicas y su soporte por medio del *teorema de los ceros de Hilbert*. Esto permite establecer una identificación entre curvas y conjuntos, así pues, a lo largo del texto no se hará distinción entre los conceptos de curva y soporte.

3.3.1. Regularidad de curvas y singularidades

Sea $\mathcal{C} \in K^2$ una curva definida por el polinomio $P(X,Y)$. Fijémonos que por ser P un polinomio, el concepto de derivada se puede extender de forma natural del análisis matemático.

Definición 3.3.4 (Derivada en sentido Euler)**.** *Sea $P(X) = a_0 + ... + a_n K^n \in K[X]$ un polinomio, se define la derivada en sentido Euler de este como:*

$$P_X(X) = \partial P/\partial X = \sum_{i=1}^{n} i a_i X^{i-1}$$

El concepto de derivada se generaliza a anillos de polinomios en varias indeterminadas naturalmente puesto que $A[X_1,...,X_n] = A[X_1,...,X_{i-1},X_{i+1},...,X_n][X_i]$. Veamos un ejemplo:

Ejemplo 3.3.1. *Sea $P(X,Y) = X^4 + 5XY^3 + 3Y^3 + X^7$ sobre $\mathbb{F}_7[X,Y]$. Su derivada respecto de X sería $P_X(X,Y) = 4X^3 + 5Y^3$, de igual manera, su derivada respecto de Y sería $P_Y(X,Y) = XY^2 + 2Y^2$.*

Al igual que el concepto de derivada se puede extender naturalmente del análisis matemático, podemos extender los conceptos de recta tangente a una curva sobre un punto y singularidad.

Definición 3.3.5 (Singularidad)**.** *Sea $\mathcal{C}$ una curva plana sobre K^2 con ecuación implícita $P(X,Y)$, diremos que posee una singularidad sobre $(x,y) \in \mathcal{C}(K)$ si se cumple $P_X(x,y) = P_Y(x,y) = 0$.*

Se dice que una curva es **regular** si carece de singularidades. En particular, las curvas elípticas por definición (véase Definición 3.4.2) De igual manera, diremos que una singularidad es **simple** si alguna de sus segundas derivadas es no nula sobre esta.

Definición 3.3.6 (Recta tangente)**.** *Sea $\mathcal{C}$ una curva plana sobre K^2 con ecuación implícita $P(X,Y)=0$ y $(x,y)\in K^2$ un punto no singular sobre esta, se define la recta tangente a $P(X,Y)$ en (x,y) como:*

$$T_P(\mathcal{C}) : P_X(x,y)(X-x)+P_Y(x,y)(Y-y)$$

El lector habido se dará cuenta que sobre una singularidad no se puede definir la recta tangente de la forma enunciada en la Definición 3.3.6. Dado que sólo nos interesa trabajar con curvas elípticas y estas son regulares, no se desarrollará el concepto de **espacio tangente** sobre conjuntos de puntos singulares.

Ahora bien, sí que será de interés reconocer cuando una curva es singular o no. Para ello se introducirá el concepto de resultante de dos polinomios:

Definición 3.3.7 (Resultante (I))**.** *Sean $P,Q\in K[X]$ dos polinomios, se define su resultante como:*

$$Res(P,Q)=\prod_{\{x:P(x)=0\}}\ \prod_{\{y:Q(y)=0\}}(x-y)$$

Prestando atención a la definición anterior, se tiene que dos polinomios poseen resultante cero si y sólo si poseen al menos una raíz en común. Dado que la definición anterior es muy poco manejable, se introduce la siguiente definición equivalente:

Definición 3.3.8 (Resultante (II))**.** *Sean $P,Q\in K[X]$ dos polinomios de la forma $P(X)=a_0+a_1X+...+a_nX^n$ y $Q(X)=b_0+b_1X+...+b_mX^m$. La resultante de sendos polinomios se define como el determinante de su matriz de Sylvester asociada, es decir:*

$$Res(P,Q)=\left|\begin{array}{ccccc} a_n & \dots & a_0 & \dots & 0 \\ 0 & \ddots & \vdots & \ddots & 0 \\ 0 & \dots & a_n & \dots & a_0 \\ b_m & \dots & b_0 & \dots & 0 \\ 0 & \ddots & \vdots & \ddots & 0 \\ 0 & \dots & b_n & \dots & b_0 \end{array}\right| \quad \left.\begin{array}{l}\\ \\ \\ \end{array}\right\} m \text{ filas} \quad \left.\begin{array}{l}\\ \\ \\ \end{array}\right\} n \text{ filas}$$

Dado que este concepto para el segundo caso es arduo de entender, se expone el siguiente ejemplo:

Ejemplo 3.3.2. *Sean $P(X)=X^2-1$ y $Q(X)=X^3-X$ sobre $\mathbb{R}[X]$. Es claro que el espectro de raíces en ambos casos es $\{-1,1\}$ y $\{-1,0,1\}$, respectivamente. Si calculamos la resultante por la primera definición se tendría:*

$$Res(P,Q)=(-1+1)\cdot(-1-0)\cdot(-1-1)\cdot(1+1)\cdot(1-0)\cdot(1-1)=0$$

Aplicando la segunda definición, tendríamos que la matriz de Sylvester asociada a los polinomios es:

$$Res(P,Q)=\left|\begin{array}{ccccc} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{array}\right|=0$$

El concepto de resultante se extiende de forma natural a anillos de polinomios en varias indeterminadas, pues se recuerda que $K[X_1,...,X_n]=K[X_1,...,X_{i-1},X_{i+1},...,X_n][X_i]$. En este caso,

si tengo dos polinomios $P, Q \in K[X_1, ..., X_{i-1}, X_{i+1}, ..., X_n][X_i]$, se tiene que $Res(P, Q; X_i) = Res(P, Q) \in K[X_1, ..., X_{i-1}, X_{i+1}, ..., X_n]$.

La resultante de un polinomio con respecto a su derivada se conoce como **discriminante** y se denota como $\Delta P = Res(P, P_X)$. El siguiente resultado relaciona las singularidades de un polinomio con su discriminante.

Teorema 3.3.1 (Criterio del discriminante)**.** *El polinomio $P(X) \in K[X]$ posee puntos singulares si y solo si cumple $\Delta P = 0$.*

Corolario 3.3.1. *Dado un punto $(x, y) \in K^2$ y $P(X, Y) \in K[X, Y]$, si $P_Y(x, y) = 0$ (resp. $P_X(x, y) = 0$) y $\Delta P(X, y) = 0$ (resp. $\Delta P(x, Y) = 0$), entonces la curva presenta singularidades.*

Muchas fuentes aplican un razonamiento erróneo en esta demostración asumiendo que $Res(P_Y(a, Y), P(a, Y); Y) = Res(P_Y, P; Y)(a)$, lo que en general es falso, pero en este caso sí que se cumple. Se invita al lector a consultar [Smith, 2013] para concretar los detalles en la prueba del resultado anterior.

3.4. Curvas Elípticas

Las anteriores secciones presentan el preámbulo algebraico necesario para entender cualquier libro de criptografía basada en aritmética de curva elíptica. A continuación se presentarán las curvas elípticas, la estructura de grupo inducida sobre estas y se tratarán los métodos de resolución del problema del logaritmo discreto de curva elíptica. Se advierte al lector que, al igual que pasaba en la sección anterior, K se referirá a un cuerpo en lo que sigue.

Definición 3.4.1 (Curva elíptica (I))**.** *Sea $P(X, Y) \in K[X, Y]$ una curva de grado tres diremos que es **elíptica** si posee género 1.*

En la definición anterior el concepto de género le puede sonar desconocido al lector. Informalmente se puede decir que el *género* de una curva es el número de ciclos que esta da sobre sí misma. El **Teorema de Riemann-Roch** (véase [Smith, 2013]) nos muestra que toda curva plana $\mathcal{C}$ de grado g con S singularidades simples posee género:

$$gen(C) = \frac{(d-1)(d-2)}{2} - S \tag{3.1}$$

De aquí se deduce que toda curva de grado tres (**cúbica**) sin singularidades posee género 1. Por ende, se puede tomar la siguiente definición equivalente:

Definición 3.4.2 (Curva elíptica (II))**.** *Diremos que una curva plana de grado tres es **elíptica** si no posee singularidades.*

Aunque esta definición empieza a ser más manejable, las cúbicas sobre un anillo de polinomios bivariado siguen tomando una multitud de formas:

$$P(X, Y) = a_{00} + a_{10}X + a_{01}Y + a_{11}XY + a_{20}X^2 + a_{02}Y^2 +$$
$$+a_{21}X^2Y + a_{12}XY^2 + a_{30}X^3 + a_{03}Y^3,$$
$$a_{00}, a_{10}, a_{01}, a_{11}, a_{20}, a_{02}, a_{21}, a_{12}, a_{30}, a_{03} \in K$$

La complejidad acaecida a esta situación se resuelve gracias al siguiente resultado:

Teorema 3.4.1 (Forma de Weiestrass)**.** *Sea $\mathbb{E} \in K[X,Y]$ una curva de grado tres sin singularidades y con $char(K) \neq 2,3$. Existe un cambio de referencia que nos permite expresar esta como:*

$$\mathbb{E} : Y^2 = X^3 + AX + B; \quad A, B \in K$$

Los detalles de la demostración del resultado anterior pueden ser consultados en [Silverman, 2009]. Esta prueba se sale de las competencias de este trabajo, aunque sí que comentaremos que, nuevamente, se utiliza el teorema de Riemann-Roch para demostrar la existencia de dicho cambio de variable a través de los divisores de la curva.

Sea $P(X,Y) = X^3 + AX + B - Y^2$ una cúbica en forma de Weiestrass sobre un cuerpo K, es claro que $P_Y(X,Y) = -2Y$ y por ende, si $char(K) \neq 2$, un punto será singularidad sobre P si y sólo si $Y = 0$. Aplicando el Teorema 3.3.1, y más concretamente el Corolario 3.3.1, se tiene que:

$$\Delta P(X,0) = \begin{vmatrix} 1 & 0 & A & B & 0 \\ 0 & 1 & 0 & A & B \\ 3 & 0 & A & 0 & 0 \\ 0 & 3 & 0 & A & 0 \\ 0 & 0 & 3 & 0 & A \end{vmatrix} = \begin{vmatrix} 1 & A & B & 0 \\ 3 & A & 0 & 0 \\ 0 & 0 & A & 0 \\ 0 & 3 & 0 & A \end{vmatrix} + 3 \begin{vmatrix} 1 & A & B & 0 \\ 0 & 0 & A & B \\ 3 & A & 0 & 0 \\ 0 & 3 & 0 & A \end{vmatrix} =$$

$$= A \begin{vmatrix} 1 & A & B \\ 3 & A & 0 \\ 0 & 0 & A \end{vmatrix} + 3 \left(\begin{vmatrix} 0 & A & B \\ A & 0 & 0 \\ 3 & 0 & A \end{vmatrix} + 3 \begin{vmatrix} A & B & 0 \\ 0 & A & B \\ 3 & 0 & A \end{vmatrix} \right) =$$

$$= A(A^2 - 3A^2) + 3(-A^3 + 3(A^3 + 3B^2)) = 27B^2 - 4A^3$$

Del anterior calculo se deduce que, si $char(K) \neq 2,3$ entonces la curva $P(X,Y)$ poseerá singularidades si y sólo si sus coeficientes satisfacen la ecuación $27B^2 - 4A^3 = 0$. Este resultado permite una nueva definición para las curvas elípticas en características distintas de dos y tres.

Definición 3.4.3 (Curva elíptica (III))**.** *Una curva elíptica en característica distinta de 2 y 3 es una cúbica de la forma $P(X,Y) : Y^2 = X^3 + AX + B$ satisfaciendo $27B^2 - 4A^3 \neq 0$.*

A lo largo del texto y para facilidad del lector se trabajará únicamente con esta definición y se asumirá que la característica del cuerpo utilizado es distinta de 2 y 3.

3.4.1. Grupo inducido por una curva elíptica

La excelente regularidad de la que gozan las curvas elípticas infiere sobre estas una infinidad de propiedades muy interesantes. Gracias a estas se han conseguido hitos impresionantes como la demostración del último Teorema de Fermat (véase [Wiles, 1995]) o la creación de algoritmos de factorización (véase [Lenstra Jr, 1987]). Una de las propiedades más llamativas es que las manipulaciones geométricas pueden inducir una ley de grupo conmutativo sobre los puntos de una curva. A lo largo de esta sección se desarrollará esta ley de grupo y todas las consecuencias en las que ha derivado.

Desarrollaremos la ley de grupo a través del *método de la cuerda y la tangente* pero antes necesitamos definir los siguientes términos:

Notación 3.4.1. *Para toda curva elíptica $\mathbb{E}$ sobre un cuerpo K se adoptará la siguiente notación:*

(a) Por conveniencia se asumirá que el punto $\mathcal{O}$ pertenece al soporte de la curva.

(b) Sean $P, Q \in \mathbb{E}$ denotamos a la recta que pasa por estos mediante $\overline{PQ}$.

(c) La reflexión de un punto $(x, y) = P \in \mathbb{E}$ sobre la recta $Y = 0$ se denota mediante $r_Y(P) = (x, -y)$.

(d) Dado $P \in \mathbb{E}$, la recta $\overline{\mathcal{O}P}$ se define como la recta[5] *$\overline{r_Y(P)P}$.*

(e) La recta tangente a $\mathbb{E}$ por $\mathcal{O}$ no corta a $\mathbb{E}$ en ningún otro punto.

Sobre la Notación 3.4.1, los puntos (b) y (c) pueden ser tomados como definiciones. Sin embargo, los puntos (a), (d) y (e) no son impuestos por capricho. La prueba formal de estos enunciados pasa por sumergir la curva en un plano proyectivo e imponer que $\mathcal{O}$ es el único punto de corte que tiene esta con la recta del infinito, en [Quintana, 2022] se pueden encontrar todos los detalles.

Definición 3.4.4. *Dada una curva elíptica $\mathbb{E}$ sobre un cuerpo K, se define la siguiente operación:*

$$\eta : \mathbb{E}^2 \longrightarrow \mathbb{E}; \quad \eta(P, Q) = \begin{cases} \overline{PQ} \cap \mathbb{E} \setminus \{P, Q\}, & P \neq Q \\ T_P(\mathcal{C}) \cap \mathbb{E} \setminus \{P\}, & P = Q \end{cases}$$

Se recuerda que, por la Definición 3.3.6, $T_P(\mathbb{E})$ se refiere a la recta tangente a la curva $\mathbb{E}$ por el punto P. Nuevamente, en [Quintana, 2022] puede encontrarse la prueba de la función η se encuentra bien definida. A través de la función auxiliar definida anteriormente se puede construir la ley de grupo inducida por una curva elíptica.

Definición 3.4.5 (Ley de grupo inducida)**.** *Para toda curva elíptica $\mathbb{E}$ sobre un cuerpo K el siguiente operador binario define una ley de grupo conmutativo sobre el soporte de la misma.*

$$\oplus : \mathbb{E}^2 \longrightarrow \mathbb{E}; \quad P \oplus Q := \eta(\eta(P, Q), \mathcal{O})$$

En la construcción anterior, la función η se refiere al operador definido en la Definición 3.4.4. Las referencias [Fulton, 2008, Quintana, 2022, Silverman, 2009] utilizan distintos métodos de demostración para probar que el par $(\mathbb{E}, \oplus)$ satisface la definición de grupo (véase Definición 3.1.1).

Esta ley de grupo es el principal pilar en el que se basa la aritmética de curva elíptica. Se puede probar con suma facilidad a través de esta diversas propiedades como la existencia de elemento neutro:

Proposición 3.4.1 (Elemento neutro)**.** *Sea $\mathbb{E}$ una curva elíptica, el punto $\mathcal{O}$ cumple la propiedad de elemento neutro para la ley de grupo inducida.*

Efectivamente, el elemento el punto $\mathcal{O}$ satisface la propiedad de elemento neutro sobre $\oplus$. Si se tima un punto $P \in \mathbb{E}$ sin pérdida de generalidad y se desarrolla se obtienen las siguientes identidades:

$$P \oplus \mathcal{O} = \eta(\eta(P, \mathcal{O}), \mathcal{O}) = \eta(r_Y(P), \mathcal{O}) = P$$

$$\mathcal{O} \oplus P = \eta(\eta(\mathcal{O}, P), \mathcal{O}) = \eta(r_Y(P), \mathcal{O}) = P$$

Una vez demostrada la existencia del elemento neutro para el par $(\mathbb{E}, \oplus)$ se puede enunciar sin dificultad la fórmula para calcular el elemento inverso.

Proposición 3.4.2 (Elemento inverso)**.** *Sea $\mathbb{E}$ una curva elíptica y $P \in \mathbb{E}$ un punto sobre esta tomado sin pérdida de generalidad, siempre se cumple que $-P = r_Y(P)$.*

[5] Si la curva $\mathbb{E}$ se encuentra en forma de Weiesstrass se puede probar sin dificultad que $\pi_Y(P) \in \mathbb{E}$ para todo $P \in \mathbb{E}$.

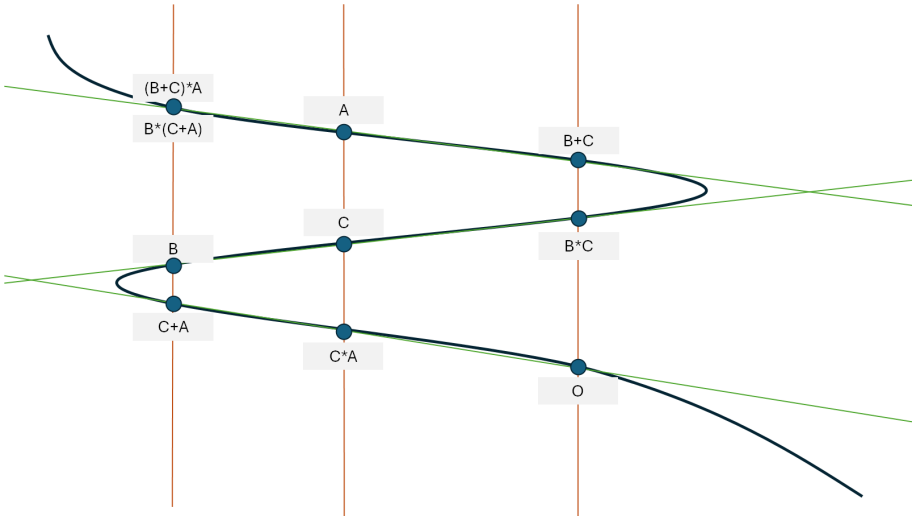


Figura 3.2: Idea gráfica de la demostración sintética

La demostración de esto es sumamente simple, guiándose por las indicaciones presentadas en Notación 3.4.1. Se toma P sin pérdida de generalidad, desarrollando la suma de este con su reflejo sobre el eje de las Y se llega a la siguiente identidad:

$$P \oplus r_Y(P) = \eta(\eta(P, r_Y(P)), \mathcal{O}) = \eta(\mathcal{O}, \mathcal{O}) = \mathcal{O}$$

Para la última igualdad en la identidad anterior, el lector debe recordar que $\mathcal{O}$ posee multiplicidad tres sobre $\mathbb{E}$ (véase el punto e de la Notación 3.4.1). De esta proposición se puede deducir fácilmente que, si $\mathbb{E}$ se encuentra en forma de Weiestrass, entonces todo punto $P = (x, y)$ sobre esta con $y = 0$ es su propio inverso.

Corolario 3.4.1. *Sea $\mathbb{E}$ una curva de Weiestrass y $P \in \mathbb{E}$ con $P = (x, 0)$ entonces se cumple que $P = -P$ para $\mathcal{O}$.*

Por herencia de la Definición 3.4.4 se tiene el cumplimiento de la propiedad conmutativa sobre el par $(\mathbb{E}, \oplus)$, es decir $P \oplus Q = Q \oplus P$ para todos $P, Q \in \mathbb{E}$.

Proposición 3.4.3 (Propiedad Conmutativa)**.** *Fijada una curva elíptica $\mathbb{E}$, la operación inducida $\oplus$ conmuta sobre el soporte de esta.*

Finalmente, se debe probar la propiedad asociativa. Existen dos demostraciones para este hecho. La primera, que puede ser encontrada en [Silverman, 2009], es analítica y pasa por parametrizar la operación $\oplus$ en ecuaciones ver que la nueva expresión satisface asociatividad entre los puntos del soporte. La segunda, a ser encontrada en [Quintana, 2022], es sintética y pasa por el **Teorema de Cayley-Bacharach**. Este resultado muestra que para todo par de cúbicas C_1 y C_2 que se cortan en nueve puntos distintos, toda cúbica C_3 que pase por ocho de estos puntos, pasará también por el noveno. Tras aplicar este resultado, entendiendo cada 3 upla de rectas involucrada en la definición de $\oplus$ como una cúbica (véase la Figura 3.2) se llega a que $\eta(A, B + C) = \eta(A + B, C)$ para todos A, B, C sobre una curva elíptica.

A pesar de la belleza que subyace en la demostración sintética, la demostración analítica nos ofrece un una vía rápida para computar todos estos resultados. Cabe resaltar que sólo expondremos las ecuaciones para curvas sobre cuerpos con característica distinta de dos y de tres, sin embargo, en [Silverman, 2009] se puede ver que este resultado es generalizable a todo cuerpo.

Teorema 3.4.2. *Sea $\mathbb{E} : Y^2 = X^3 + AX + B$ una ecuación elíptica en forma de Weiestrass sobre un cuerpo K con característica k distinta a cero, dos y tres. Dados dos puntos $A, B \in \mathbb{E}$ con $A = (x_1, y_1)$ y $B = (x_2, y_2)$, la ley de grupo inducida se encuentra parametrizada*[6] *por:*

$$s = \begin{cases} (3x_1^2 + A) \cdot (2y_1)^{-1}, & A = B \\ (y_2 - y_1) \cdot (x_2 - x_1)^{-1} & A \neq B \end{cases}$$

$$A \oplus B = \begin{cases} A, & B = \mathcal{O} \\ B, & A = \mathcal{O} \\ \mathcal{O}, & x_1 = x_2 \;\; y \;\; y_1 = -y_2 \\ (s^2 - x_1 - x_2, \quad s \cdot (x_2 - s^2) - y_1), & \textit{otros casos} \end{cases}$$

3.4.2. El orden de una curva elíptica

Habiendo introducido los grupos inducidos por curvas elípticas y proporcionando herramientas para trabajar con ellos, se hablará de su orden (véase la Definición 3.1.5) y las principales propiedades que este satisface.

En primer lugar, cabe destacar que el grupo inducido por toda curva elíptica $\mathbb{E}$ sobre un cuerpo finito K posee orden finito. Este hecho es inmediato desde que $\mathbb{E} \subset K^2$.

Proposición 3.4.4. *El soporte de una curva elíptica sobre un cuerpo finito es finito.*

La proposición anterior junto al Teorema 3.1.2 nos permite establecer que el grupo inducido por una curva elíptica sobre un cuerpo finito puede ser caracterizado como suma directa de grupos cíclicos, aunque el **Teorema de Ruck** [Lynn, 2024] ofrece una representación mucho más precisa.

Teorema 3.4.3 (Teorema de Ruck)**.** *Sea $\mathbb{E}$ una curva elíptica sobre un cuerpo finito $\mathbb{F}_q$, su grupo inducido es isomorfo al siguiente producto de grupos cíclicos:*

$$\mathbb{E}(\mathbb{F}_q) \simeq \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}, \quad n_1 | n_2, \; n_2 | (q-1)$$

El Teorema 3.4.3, también conocido como Teorema de Cassels, describe la estructura algebraica de los grupos inducidos por las curvas elípticas. Sin embargo, a primera vista no proporciona herramientas para cuantificar el número de elementos que estos grupos contienen. Por su parte, el Teorema de Hasse establece la primera cota conocida para la cantidad de puntos de una curva elíptica, en función de la característica del cuerpo sobre el que se define.

Teorema 3.4.4 (Teorema de Hasse)**.** *Sea $\mathbb{F}_q$ un grupo y $\mathbb{E}$ una curva elíptica sobre este, el orden de esta curva obedece la siguiente cota:*

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

3.4.3. Logaritmo discreto de curva elíptica

En esta sección se desarrollará el problema del logarítmo discreto de curva elíptca (ECDLP), enlazando directamente con el preámbulo expuesto en la Sección 3.1.1.

[6] Las operaciones utilizadas en la parametrización de la ley de grupo son la suma y producto asociados a K como cuerpo.

Definición 3.4.6 (ECDLP)**.** *Dada una curva elíptica* $\mathbb{E}$ *y dos puntos* $P, Q \in \mathbb{E}$*. El logarítmo discreto en base* P *de* Q *se basa en encontrar el menor entero positivo* d *tal que* $d \cdot P = Q$*. Denotándose esto por:*

$$\log_P(Q) = \text{mín}\{d \in \mathbb{Z}^+ : d \cdot P = Q\}$$

En la definición anterior sólo tiene sentido calcular $\log_P(Q)$ cuando $Q \in \langle P \rangle$, en caso contrario se entenderá que dicho valor no existe. Como se puede apreciar, este concepto es una generalización natural de la Definición 3.1.3 adaptado a los grupos abelianos inducidos sobre el soporte de una curva elíptica.

En la mayoría de casos, no existe un algoritmo determinista para el cálculo de este valor, siendo necesario una parte heurística que itera por las potencias de P, implicando esto que la dificultad en el cálculo de este valor crece con el orden de P. En la Sección 3.5.4 se desarrollan distintos algoritmos –o ataques– que permiten calcular $\log_P(Q)$ de forma eficiente, vulnerando así criptosistemas basados en ECC.

3.5. Criptografía de Curva Elíptica

Antes de definir la criptografía es conveniente encontrarse familiarizado con la *Seguridad de la Información*. La ISO 27001 marca los tres principios fundamentales que debería cumplir cualquier sistema informático para ser considerado seguro:

Confidencialidad Garantiza que la información sea solo accesible para aquellos usuarios o entidades autorizados.

Integridad Se basa en la confiabilidad de la información, buscando evitar cualquier modificación, alteración o corrupción no autorizada de los datos.

Disponibilidad Trata de asegurar el acceso a la información o servicios del sistema, siempre que un usuario lo necesite.

A esta tríada de principios, conocida como tríada CIA o CID, autores como Menezes añaden un supuesto a mayores, la **Autenticación**. Este supuesto se basa en verificar la identidad de la entidad antes de concederle acceso a una plataforma.

Existe una ingente cantidad de protocolos, técnicas y mecanismos encargados de hacer que los sistemas informáticos cumplan estas propiedades. La criptografía sería uno más de estos artefactos teóricos utilizados en el ámbito de la seguridad como se presenta en la Figura 3.3. Más concretamente, [Menezes et al., 2018] proponen la siguiente definición:

Definición 3.5.1 (Criptografía)**.** *La criptografía se define como el conjunto de técnicas matemáticas (primitivas) utilizadas en el ámbito de la seguridad para cerciorar los supuestos de confidencialidad, integridad, autenticación y disponibilidad.*

Estas primitivas se categorizan en tres grupos bien diferenciados cuya estructura se presenta en la Figura 3.4. En esta sección se prestará especial atención a las primitivas de clave pública desarrolladas a través de aritmética de curva elíptica.

3.5.1. Introducción a la terminología

La notación debe ser un lenguaje técnico y fácil de comprender que permita al escritor desarrollar de forma completa o parcial una o varias técnicas. A lo largo de este punto se introducirá

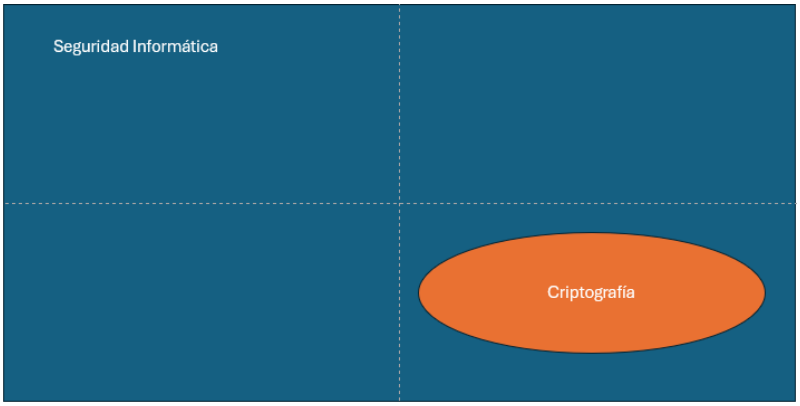


Figura 3.3: Seguridad Informática vs Criptografía

la notación utilizada en criptosistemas para el correcto desarrollo de las primitivas implementadas sobre criptografía de curva elíptica.

Sea $\mathcal{A}$ un conjunto o **alfabeto**. Dado un alfabeto de definición $\mathcal{A}$, su conjunto de sucesiones sobre los números naturales ($\mathcal{A}^{\mathbb{N}}$) conformarán el **espacio de mensajes** o **espacio de texto plano** (denotado por $\mathcal{M}$) y el **espacio de texto cifrado** (denotado por $\mathcal{C}$). Los elementos pertenecientes a los espacios anteriores toman el nombre de **mensaje** o **texto cifrado**.

Mediante esta notación, el espacio de mensajes representaría el conjunto de archivos, textos o elementos a los cuales se les quiere aplicar una primitiva. El resultado obtenido de la transformación anterior sería un elemento del espacio de texto cifrado.

Cabe destacar que el espacio de texto cifrado puede estar definido sobre un alfabeto distinto que el espacio de mensajes. Se presentará ahora el espacio de clave, de cuyos elementos se derivan las transformaciones entre los mensajes en claro y sus cifrados correspondientes.

Definición 3.5.2 (Espacio de clave)**.** *Sea $\mathcal{K}$ un conjunto genérico, $\mathcal{M}$ un espacio de texto plano y $\mathcal{C}$ un espacio de texto cifrado. Diremos que $\mathcal{K}$ es un* ***espacio de claves*** *entre $\mathcal{M}$ y $\mathcal{C}$ si cada elemento $e \in \mathcal{K}$ induce una aplicación $E_e : \mathcal{M} \longrightarrow \mathcal{C}$, que será referida como* ***función de cifrado****.*

En función del tipo de primitiva puede interesar revertir o no la transformación realizada. En estos casos, se exige que las aplicaciones inducidas sobre los elementos de $\mathcal{K}$ sean **biyectivas**. Así cada elemento $d \in \mathcal{K}$ también induce una biyección $D_d : \mathcal{C} \longrightarrow \mathcal{M}$, conocida como función de descifrado. La idea detrás de estas funciones es que para cada $e \in K$ existe un $d \in K$ tal que $D_d(E_e(m)) = m$, para todo $m \in \mathcal{M}$.

Definición 3.5.3 (Esquema de Cifrado)**.** *Sean $\mathcal{M}$ y $\mathcal{C}$ espacios de texto plano y de cifrado sobre los alfabetos $\mathcal{A}$ y $\mathcal{A}'$, respectivamente. Sea a su vez $\mathcal{K}$ un espacio de clave entre $\mathcal{M}$ y $\mathcal{C}$. Se denota por* ***esquema de cifrado*** *a los conjuntos $\{E_e : \mathcal{M} \longrightarrow \mathcal{C} \mid e \in \mathcal{K}\}$ y $\{D_d : \mathcal{C} \longrightarrow \mathcal{M} \mid d \in \mathcal{K}\}$.*

Definición 3.5.4 (Criptosistema)**.** *Fijados espacios de texto plano, de texto cifrado y de claves $\mathcal{M}$, $\mathcal{C}$ y $\mathcal{K}$, respectivamente. Se define el* ***criptosistema*** *asociado a estos como la tupla $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \{E_e : e \in \mathcal{K}\}, \{D_d : d \in \mathcal{K}\})$*

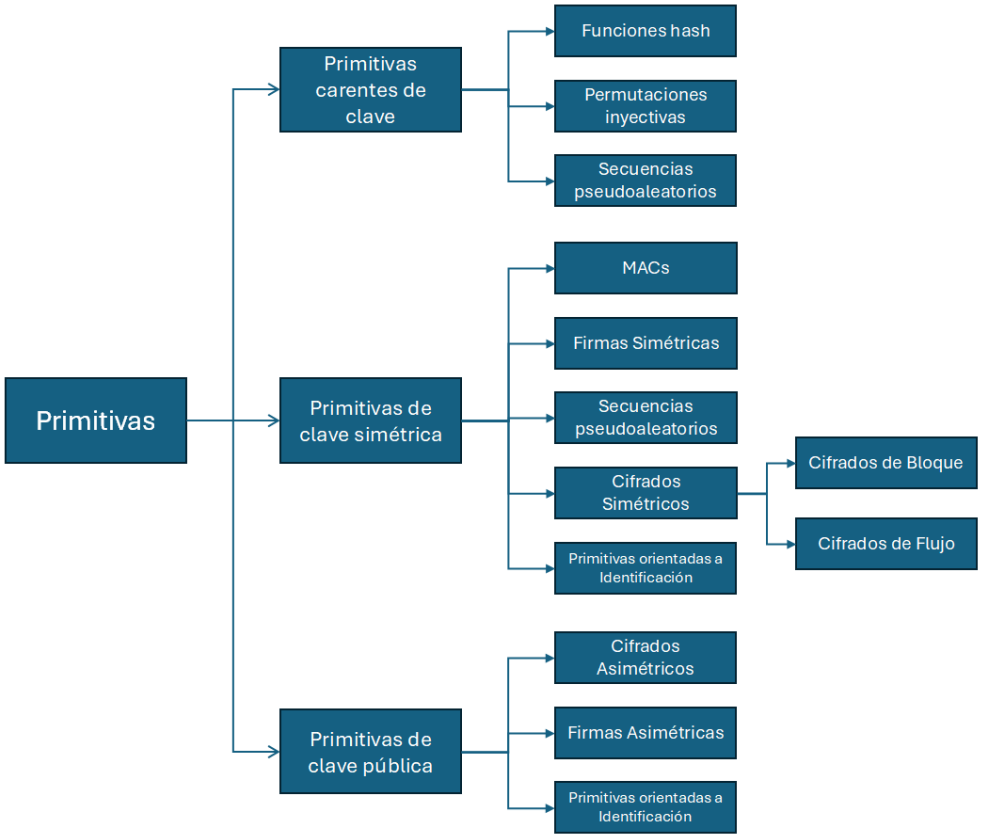


Figura 3.4: Estratificación de Primitivas

3.5.2. Criptografía de Clave Pública

Antes de desarrollar esta nueva sección se introducirán los conceptos de inviabilidad computacional, función unidireccional y función trampa; vitales para entender mucho mejor la idea subyacente tras la criptografía de clave pública.

Definición 3.5.5 (Inviabilidad Computacional)**.** *Un problema a tratar es* ***inviable computacionalmente*** *si no se puede resolver en tiempo polinómico o con la memoria disponible.*

Definición 3.5.6 (Función unidireccional)**.** *Una función* $f : X \longrightarrow Y$ *es* ***unidireccional*** *si cumple las siguientes propiedades:*

- *Calcular* $f(x)$ *tiene costo a lo sumo polinómico para todo* $x \in X$*.*
- *Para cada casi todo*[7] $y \in Im(f)$*, encontrar un* $x \in X$ *tal que* $f(x) = y$ *es computacionalmente inviable, en caso contrario se considera* ***computacionalmente viable****.*

Definición 3.5.7 (Función trampa)**.** *Diremos que una función* $f : X \longrightarrow Y$ *es* ***trampa*** *si es unidireccional, pero conociendo cierta información extra (****información trampa****) se tiene que, fijado*

[7]Esto implica que el conjunto de valores en $Im(f)$ que no cumplen esta propiedad posee medida nula en el sentido de Lebesgue.

$y \in Im(f)$ calcular $x \in X$ tal que $f(x) = y$ es un problema computacionalmente viable.

Una vez desarrollados estos conceptos, tomemos un criptosistema $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \{E_e : e \in K\}, \{D_d : d \in K\})$ y tomemos una clave (e, d) sobre este. Supongamos conocida la clave e y además supongamos que para cada $c \in \mathcal{C}$ encontrar un $m \in \mathcal{M}$ tal que $E_e(m) = c$ es un problema computacionalmente inviable. En los términos superiores, esto implicaría que E_e es una función trampa y su inversa sería computacionalmente abordable tomando la clave d como información extra. Aplicando esta idea a través del esquema presentado en la Figura 3.5, se puede establecer una comunicación sin peligro de que ningún adeversario vulnere el supuesto de confidencialidad. Puesto que E_e es una función trampa, aunque un adversario incaute la clave e o un mensaje cifrado $c \in \mathcal{C}$, sería computacionalmente inviable que acceda al contenido m.

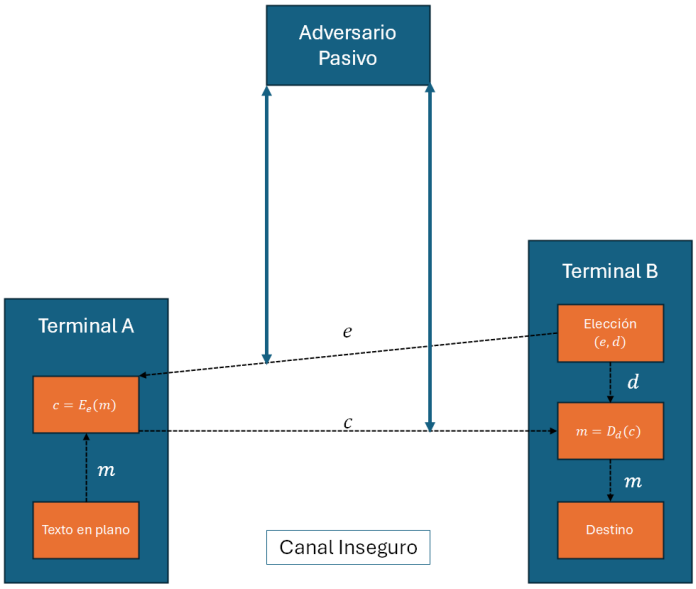


Figura 3.5: Esquema de comunicación segura en clave pública

El nombre de **clave pública** proviene de la compartición de la clave e entre terminales. Fijémonos que para que este cifrado sea seguro, la clave d no debe poder ser calculada a través de e (o al menos, su calculo debe ser computacionalmente inviable). Naturalmente esta propiedad implica la diferencia $e \neq d$, razón por la que este tipo de primitivas también toma el nombre de **criptografía de clave asimétrica**.

Necesidad de autenticación en los sistemas de clave pública

A pesar de parecer sistemas perfectos, las comunicaciones a través de un criptosistema de clave pública pueden ser fácilmente vulnerados. Por ejemplo, la Figura 3.6 muestra un esquema de ataque típico, en el que un adversario activo consigue descifrar el mensaje trasmitido por el terminal A, modificarlo y enviar al terminal B texto diferente haciéndose pasar por el terminal A. Para evitar este tipo de situaciones, la compartición de la clave pública debe estar condicionada a un proceso de autenticación.

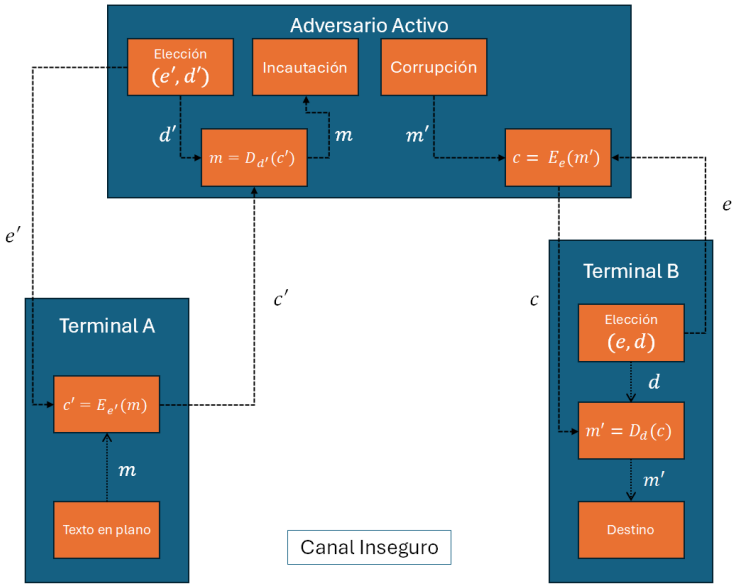


Figura 3.6: Adversario activo en esquema de comunicación con clave pública

En las secciones posteriores se desarrollarán técnicas para el manejo de clave, nuevamente focalizadas en criptografía de clave pública. No obstante, se advierte al lector de la existencia de primitivas basadas en criptografía de clave simétrica y se invita a su lectura en los Capítulos 12 y 13 de [Menezes et al., 2018].

3.5.3. Traducción a curvas elípticas

En los anteriores capítulos se presentaba un escenario general y abstracto que permitía englobar el trabajo con todo tipo de primitiva, no obstante, no debemos olvidar nuestro foco, la criptografía de curva elíptica. A lo largo de esta sección se desarrollarán los conceptos fundamentales para entender esta y para comprender mejor esto, se expondrán y desarrollarán alguna de las primitivas más utilizadas.

Todo criptosistema basado en aritmética de curvas elípticas comienza por fijar un cuerpo K,

una curva elíptica $\mathbb{E}$ sobre este y un punto $P \in \mathbb{E}$ distinto del punto del infinito. A la terna conformada por estos tres valores se le conoce con el sobrenombre de **parámetros de domino** del criptosistema.

Fijados estos parámetros de dominio, el soporte de la curva conformará el espacio de texto plano y el espacio de texto cifrado, es decir, $\mathcal{M} = \mathcal{C} = \mathbb{E}(K)$. Para construir el espacio de clave, se debe recurrir a una función auxiliar $\Theta_P : \mathbb{N} \to \mathbb{E}$ de forma que $\Theta_P(d) = d \cdot P$. Mediante estas premisas se define el espacio de clave como $\mathcal{K} = G(\Theta_P)$, es decir el grafo de la función Θ_P. Así pues, si se tiene un par $(d, Q) \in G(\Theta_P)$, el número d será considerado clave privada y el punto Q clave pública, siendo claro que es computacionalmente inviable obtener d a través de Q, pues sería necesario resolver un problema de logaritmo discreto de curva elíptica.

Afirmar que $\mathcal{M} = \mathbb{E}$ puede parecer contradictorio, pues ejemplos como el algoritmo de ElGamal –desarrollado más adelante– pueden cifrar cualquier mensaje[8] $m \in \{0,1\}^*$. Bajo estos supuestos, el mensaje m suele ser descompuesto en bloques de n-bits, $m_1, ..., m_k$; cuya concatenación resulta en m más una mantisa de ceros. Bajo esta construcción, los sistemas informáticos implementan una transformación inyectiva[9] $\Phi : \{0,1\}^n \to \mathbb{E}$. Esta transformación es utilizada para transformar cada uno de los bloques m_i en puntos M_i sobre $\mathbb{E}(K)$, que son posteriormente cifrados en puntos P_i. Cada uno de los puntos P_i es mandado al terminal receptor, teniendo este que descifrar los puntos, aplicar la transformación inversa Φ^{-1} y para calcular m_i y reconstruir el mensaje.

En tanto a los esquemas de cifrado y descifrado, es necesario seguir siendo bastante generales, pues varían mucho en función de la primitiva utilizada. Con independencia de la primitiva, para todo par de claves $(d, Q) \in \mathcal{K}$ existen funciones $E_Q : \mathbb{E} \to \mathbb{E}$ y $D_d : \mathbb{E} \to \mathbb{E}$ satisfaciendo $D_d \; o \; E_Q = Id$, el conjunto de estas funciones componen los esquemas de cifrado y descifrado.

Destacamos que la construcción de las funciones E_Q y D_d dado un par de claves es información de dominio público. Por esto mismo, es importante elegir un punto P adecuado dentro de los parámetros de dominio; si $\log_P(Q) = d$ es fácilmente computable entonces un *man in the middle* podría computar fácilmente la clave privada y por consiguiente D_d, rompiendo así el criptosistema.

Hasta ahora, se puede observar que el criptosistema en su totalidad puede ser construido a través de los parámetros de dominio. Por esta razón, de ahora en adelante cometeremos el siguiente abuso de notación.

Notación 3.5.1. *A lo largo de la memoria cometeremos el abuso de notación de identificar los criptosistemas basados en curvas elípticas con los parámetros de dominio, siempre que no de lugar a error.*

A continuación se muestran dos de los ejemplos más típicos entre las primitivas de curva elíptica para asentar conceptos. El algoritmo de intercambio de clave de Diffie-Hellmann y el cifrado de ElGamal.

Cifrado ElGamal

Dados un cuerpo K, una curva elíptica $\mathbb{E}$ y un punto $P \neq \mathcal{O}$ de esta y una transformación inyectiva $\Phi : \mathcal{M} \to \mathbb{E}$. Supongamos que el terminal A desea enviar un mensaje m al terminal B a través de un canal cifrado mediante el algoritmo de ElGamal, entonces se siguen los siguientes pasos:

[8] En $\{0,1\}^*$, el símbolo * representa la estrella de *Kleene*. Esta notación se refiere a toda combinación finita de ceros y unos, incluido el vacío.

[9] En la memoria no se desarrollan este tipo de transformaciones, pero para aquellos interesados, se resalta el algoritmo propuesto en [Koblitz, 1987] y su posterior refinamamiento que puede ser encontrado en [Yan, 2013].

- 1. El terminal B selecciona un entero positivo $d \in [1, ord(P)]$ (**clave privada**).
- 2. A continuación, el terminal B calcula el punto $Q = d \cdot P$ (**clave pública**) y se lo remite al terminal A.
- 3. El terminal A calcula $\Phi(m) = M \in \mathbb{E}$. Tras esto, escoge un entero positivo aleatorio k y calcula los puntos $M' = k \cdot Q + M$ y $Q' = k \cdot P$ (esta transformación sería E_Q); remitiendo el par (M', Q') a B.
- 4. El terminal B recibe el par (M', Q') y calcula $M = M' - d \cdot Q'$ (esta sería la transformación D_d). Por último computa $m = \Phi^{-1}(M)$.

En el algoritmo anterior se puede comprobar que todo funciona adecuadamente gracias a la conmutatividad:

$$M' - d \cdot Q' = (M + k \cdot Q) - d \cdot (k \cdot P) = M + \cancel{(k \cdot d) \cdot P} - \cancel{(k \cdot d) \cdot P} = M$$

Elliptic Curve Diffie-Hellman (ECDH)

Supongamos que dos terminales, A y B, desean establecer un secreto compartido de forma segura sin que este viaje por ningún canal de comunicación. Fijados un cuerpo K, una curva elíptica $\mathbb{E}$ sobre K y un punto $P \in \mathbb{E}$ con $P \neq \mathcal{O}$, los terminales pueden establecer el secreto a través de los siguientes pasos.

- 1. El terminal A selecciona un entero positivo $d_a \in [1, ord(P)]$ (**clave privada de A**). Y calcula el punto $Q_a = d_a \cdot A$ (**clave pública de A**).
- 2. El terminal B selecciona un entero positivo $d_b \in [1, ord(P)]$ (**clave privada de B**). Y calcula el punto $Q_b = d_b \cdot A$ (**clave pública de B**).
- 3. Los terminales intercambian sus claves públicas.
- 4. El terminal A calcula $S = d_a \cdot Q_b$. Además, el terminal B calcula $S = d_b \cdot Q_a$. El punto S se convierte en el secreto compartido entre terminales.

Nuevamente, es sumamente fácil comprobar a través de la propiedad conmutativa que el algoritmo anterior funciona correctamente:

$$S = d_a \cdot Q_b = (d_a \cdot d_b) \cdot P = (d_b \cdot d_a) \cdot P = d_b \cdot Q_a = S$$

3.5.4. Ataques en criptosistemas de curva elíptica

Como ya hemos resaltado anteriormente, la seguridad en los sistemas criptográficos de curva elíptica depende prácticamente por exclusivo de la dificultad en la resolución del ECDLP asociado. A lo largo de esta sección abordaremos en líneas generales los ataques existentes en la ECC, ahondando particularmente en el ataque Rho de Pollard.

Antes de entrar de lleno, resaltaremos que este tipo de sistemas no son los únicos que basan su seguridad en el logaritmo discreto. Por ejemplo, gran parte de la criptografía basada en cuerpos finitos se basa también en este problema, sin embargo, estos presentan una mayor vulnerabilidad debido al ataque del *Index Calculus*. Este ataque no es aplicable –directamente– sobre las curvas elípticas y esto se debe a que, a diferencia de los cuerpos finitos, las curvas elípticas no poseen un sistema aritmético completo. El no poder aplicar este tipo de ataque conlleva a que la longitud

de clave –el orden del elemento generador– pueda ser mucho menor que en otro tipo de sistemas, ofreciendo la misma seguridad.

Empezamos presentando los ataques por rango de vulneración. Cualquier criptosistema permite ser atacado mediante **fuerza bruta**. Para realizar este ataque se debe calcular el soporte de la curva en cuestión y probar punto por punto hasta lograr dar con aquel que resuelva el problema del logaritmo discreto. La probabilidad de vulneración en este tipo de algoritmos es de $k/|\mathbb{E}|$ donde k es el número de intentos establecidos. Existe una **variante estocástica** de este método que consiste en probar aleatoriamente puntos generados de la curva, la probabilidad en este caso de obtener un acierto tras k ejecuciones sigue una ley geométrica de parámetro $1/|\mathbb{E}|$, es decir, es $1-(1-1/|\mathbb{E}|)^k$. Este es uno de los raros casos en los que el algoritmo del museo británico no se cumple; pues encontramos que la esperanza es de $|\mathbb{E}|$ y $(|\mathbb{E}|+1)/2$ ejecuciones en la versión estocástica y determinista, respectivamente.

Los algoritmos presentados hasta ahora son aplicables a cualquier criptosistema cuya seguridad se base en la resolución del logaritmo discreto. Sin embargo, existen un par de algoritmos destacables en ciertas clases de curvas elípticas que veremos a continuación.

Se comienza con el **ataque de curva anómala**; dada una curva elíptica $\mathbb{E}$, cuando el número de puntos sobre su soporte $\mathbb{E}(K)$ es igual que el número de elementos de K, entonces se dice que la curva es *anómala*. En estos casos, existe un isomorfismo de grupos natural entre $\mathbb{E}(K)$ y K, de tal forma que el ECDLP asociado al criptosistema se puede reducir a un problema de logaritmo discreto sobre cuerpos finitos. Este es uno de los extraños casos en los que, se puede aplicar el *Index Calculus* para resolver el problema del logaritmo discreto en una curva elíptica.

Otro ataque conocido, es el **ataque MOV** (Menezes-Okamoto-Vanstone). Este ataque requiere de una abstracción algebraica muy grande, por lo que se introducirá muy sutilmente al lector. El soporte de una curva elíptica $\mathbb{E}(\mathbb{F}_q)$ y p' es un primo que divide a $|\mathbb{E}|$. El **grado de inmersión** de la curva sobre p' se define como el menor entero positvo k que satisface $p'|q^k-1$. Bajo estas condiciones, $\mathbb{E}(\mathbb{F}_q)$ puede ser entendido como un subgrupo sobre $\mathbb{F}_{q^k}^*$, más concretamente, el **par de Weil** presentaría una función $\psi:\mathbb{E}(\mathbb{F}_q)^2\to\mathbb{F}_{q^k}^*$, satisfaciendo $\psi(d\cdot P,Q)=\psi(P,Q)^d$ para todos $P,Q\in\mathbb{E}(F_q)$. Este ataque permite convertir nuevamente el ECDLP en un logaritmo discreto sobre cuerpos finitos, sin embargo suele ser contraproducente aplicar este ataque por la magnitud que alcanza a tener el grado de inmersión.

A continuación se detallarán los detalles del algoritmo Rho de Pollard, considerado el más eficiente en general para resolver el ECDLP.

Algoritmo Rho de Pollard

Antes de empezar a desarrollar este algoritmo, introduciremos la siguiente definición:

Definición 3.5.8 (Función aleatoria)**.** *Sea $\mathcal{F}$ un espacio de funciones uniformemente distribuidas y $\hat{\mathcal{F}}$ un subespacio (manejable) de este. Diremos que una función $\hat{f}$ tomada aleatoriamente sobre $\hat{\mathcal{F}}$ es **pseudoaleatoria** sobre $\mathcal{F}$, si para todo discriminador D siguiente cantidad es despreciable:*

$$\left| \underset{\hat{f}\leftarrow\hat{\mathcal{F}}}{P}(D^{\hat{f}}(\omega)=1) - \underset{f\leftarrow\mathcal{F}}{P}(D^{f}(\omega)=1) \right|$$

La anterior definición es, sin lugar a dudas, la más técnica de toda la memoria. Es por esto que pasamos a explicarla en términos simples. Dado un conjunto $\Omega=C_1\sqcup C_2$, un *discriminador* D es un algoritmo probabilístico capaz de identificar si un elemento $\omega\in\Omega$ pertenece a C_1 o pertenece a C_2, devolviendo 1 o 0, respectivamente. En este caso, $\hat{\mathcal{F}}$ sería C_1, siendo $C_2=\mathcal{F}-\hat{\mathcal{F}}$. La notación D^f implica que el discriminador D actúa como un **oráculo** sobre la función f, es decir, que puede evaluar esta a pesar de no conocerla. La cantidad presentada se refiere a que cualquier

discriminador D es incapaz de identificar si una función proviene de $\hat{\mathcal{F}}$ o no, incluso pudiendo evaluarla.

Visto esto, se presenta la función clásica del algoritmo Rho de Pollard para la resolución del problema del logaritmo discreto de curva elíptica.

Algoritmo 3.5.1 (Rho de Pollard)**.**

***Input**: El cuerpo K, la curva elíptica $\mathbb{E}$, el punto base P, el punto a evaluar Q y el orden de P.*

***Output**: El entero d tal que $Q = d \cdot P$.*

- *1. El atacante toma una función pseudoaleatoria f de la forma:*

$$f(X) = \begin{cases} P + X, & X \in H_1 \\ X + Q, & X \in H_2 \\ 2X, & X \in H_3 \end{cases}$$

  Conformando H_1, H_2 y H_3 una partición uniforme distribuida de $\mathbb{E}$.

- *2. Se calcula el punto $X_{i+1} = f(X_i)$, siendo $X_0 = P$.*

- *3. Se repite el paso 2 hasta que se encuentre una colisión. Esto implica que el nuevo valor X_{i+1} es igual a X_j para algún $0 \leq j \leq i$.*

- *4. La colisión encontrada en el paso 3 induce una igualdad del tipo:*

$$X_{i+1} = X_j \Leftrightarrow m_{i+1}P + n_{i+1}Q = m_jP + n_jQ \Leftrightarrow (m_{i+1} - m_j)P = (n_j - n_{i+1})Q$$

- *5. Si $(n_j - n_{i+1}) = 0 \mod ord(P)$, entonces la ecuación del 4 paso cuatro posee infinitas soluciones y no proporciona información para la resolución del ECDLP. Sino d se puede calcular despejando la incógnita x sobre la ecuación diofántica $(m_{i+1} - m_j) \cdot x + ord(P) \cdot y = (n_j - n_{i+1})$.*

En esta primera versión del algoritmo de Pollard, encontramos un problema fundamental y es la detección de la colisión en el paso 3. A parte de tener que guardar en memoria todos los valores X_i, se tienen que evaluar de forma repetitiva para localizar una la igualdad entre el nuevo punto calculado y uno de estos, lo que computacionalmente no es eficiente. Para paliar este problema, se entiende la sucesión obtenida como nodos de un grafo dirigido y se aplica el algoritmo de Floyd para buscar un ciclo sobre este. Gracias a esto ahorramos costos operacionales y en memoria, a cambio de que la encontrada no sea necesariamente la primera colisión establecida.

Algoritmo 3.5.2 (Rho de Pollard con aceleración de Floyd)**.**

***Input**: El cuerpo K, la curva elíptica $\mathbb{E}$, el punto base P, el punto a evaluar Q y el orden de P.*

***Output**: El entero d tal que $Q = d \cdot P$.*

- *1. El atacante toma una función pseudoaleatoria f de la forma:*

$$f(X) = \begin{cases} P + X, & X \in H_1 \\ X + Q, & X \in H_2 \\ 2X, & X \in H_3 \end{cases}$$

  Conformando H_1, H_2 y H_3 una partición uniforme distribuida de $\mathbb{E}$.

- *2. Se calculan las sucesiones $X_{i+1} = f(X_i)$ e $Y_{i+1} = f(f(Y_i))$, siendo $X_0 = Y_0 = P$.*

- *3. Se repite el paso 2 hasta que* $Y_i = X_i$ *para algún* i *natural.*
- *4. La igualdad encontrada en el paso 3 induce una igualdad del tipo:*

$$X_i = Y_i \Leftrightarrow m_X P + n_X Q = m_Y P + n_Y Q \Leftrightarrow (m_X - m_Y)P = (n_Y - n_X)Q$$

- *5. Si* $(n_X - n_Y) = 0 \mod ord(P)$*, entonces la ecuación del* 4 *paso cuatro posee infinitas soluciones y no proporciona información para la resolución del ECDLP. Sino* d *se puede calcular despejando la incógnita* x *sobre la ecuación diofántica* $(m_X - m_Y) \cdot x + ord(P) \cdot y = (n_Y - n_X)$*.*

En este ámbito el proceso aplicado al algoritmo anterior se conoce como *baby step - giant step* o algoritmo tipo de liebre-tortuga; pues si nos fijamos la sucesión X_i paulatinamente tomará los mismos valores que la sucesión Y_i.

Capítulo 4

Propuesta de medida de seguridad

Originality is the one thing which unoriginal minds cannot feel the use of.

John Stuart Mill

Medir la seguridad en un criptosistema de curva elíptica es una ardua tarea que muchas veces pasa por un estudio teórico-matemático muy profundo. Aunque existen recomendaciones generales para poner un criptosistema de curva elíptica en un sistema de producción [Chen et al., 2019], no existe ningún métrica que permita cuantificar la seguridad que presenta un sistema frente a otro. Es claro que la seguridad de un sistema estará directamente ligada a la dificultad en la resolución del ECDLP asociado; acudiendo a la literatura [Lynn, 2024, Smith, 2013, Silverman, 2006] encontramos que el algoritmo más frecuentemente utilizado para resolver este problema es Rho de Pollard. En virtud de esta idea, se define el siguiente índice:

4.1. El índice 3A

Definición 4.1.1 (Índice 3A)**.** *Dadas una curva elíptica $\mathbb{E}$ sobre un cuerpo K, un punto $P \in \mathbb{E}$ y un entero positivo N; se define el índice **Available Attack At** (3A) como la probabilidad de romper un cifrado de curva elíptica a través de un ataque Rho de Pollard en a lo sumo N iteraciones.*

La formalización del índice anterior versa por preámbulos básicos en teoría de categorías, lo que se escapa de los objetivos de este trabajo; sin embargo, estableceremos una construcción semiformal de estos conceptos: Se definen los objetos de la clase *EllipBase* como las ternas $(K, \mathbb{E}, P)$ donde K es un cuerpo, $\mathbb{E}$ es una curva elíptica definida sobre K y P es un punto sobre $\mathbb{E}$. De igual manera, se define $\mathbb{P}_\rho(x \leq n|(K, \mathbb{E}, P))$ como la probabilidad de romper un cifrado de curva elíptica mediante el ataque Rho de Pollard en a lo sumo n iteraciones, condicionado a que se están utilizando la curva $\mathbb{E}$ y el punto P sobre el cuerpo K. Se define de esta manera el índice $3A$ como la función:

$$\begin{array}{rccc} 3A: & \mathit{EllipBase} \times \mathbb{N} & \longrightarrow & [0,1] \\ & (K, \mathbb{E}, P, n) & \longmapsto & \mathbb{P}_\rho(x \leq n|(K, \mathbb{E}, P)) \end{array}$$

Al ser adimensional, este índice permite establecer criterios claros de superioridad entre cifrados. Para ilustrar mejor esta afirmación se muestran los siguientes ejemplos:

Ejemplo 4.1.1. *Sobre dos cifrados de curva elíptica con parámetros de dominio $(K_1, \mathbb{E}_1, P_1)$ y $(K_2, \mathbb{E}_2, P_2)$, respectivamente, si $3A(K_1, \mathbb{E}_1, P_1, n) > 3A(K_2, \mathbb{E}_2, P_2, n)$ entonces el cifrado sobre la*

curva $\mathbb{E}_2$ es objetivamente más seguro que el cifrado sobre la curva $\mathbb{E}_1$ por tener una probabilidad de ataque efectivo más baja.

Ejemplo 4.1.2. *De igual manera, sobre dos cifrados de curva elíptica con parámetros de dominio $(K_1, \mathbb{E}_1, P_1)$ y $(K_2, \mathbb{E}_2, P_2)$, respectivamente, si $n > m$ y $3A(K_1, \mathbb{E}_1, P_1, n) = 3A(K_2, \mathbb{E}_2, P_2, m)$ entonces el cifrado basado en la curva $\mathbb{E}_1$ es más seguro por tener la misma probabilidad de ataque efectivo que el cifrado $\mathbb{E}_2$, pero necesitar de un mayor número de iteraciones para ello.*

Ejemplo 4.1.3. *Por último, fijando un cuerpo K y una curva $\mathbb{E}$ sobre este, y tomando dos puntos distintos $P_1, P_2 \in \mathbb{E}$, si $3A(K, \mathbb{E}, P_1, n) > 3A(K, \mathbb{E}, P_2, n)$ entonces el punto P_1 es más inseguro para establecer el criptosistema que el punto P_2, puesto que la probabilidad de ataque efectivo en a lo sumo n iteraciones es mayor en el primer caso que en el segundo.*

Los ejemplos anteriores son sólo una muestra de como se puede caracterizar la seguridad de un método de cifrado en función de sus parámetros de dominio. Esta idea intuitiva será expresada formalmente a través de los siguientes resultados, demostrados en el Anexo C:

Proposición 4.1.1. *Para cada $n \in \mathbb{N}$ fijo, la función $3A(\cdot, \cdot, \cdot, n)$ induce una **relación de orden parcial** (véase [Dugundji, 1966]) entre los distintos elementos de EllipBase.*

Proposición 4.1.2. *Fijados un cuerpo K, una curva $\mathbb{E}$ sobre K y un punto $P \in \mathbb{E}$, la función $3A(K, \mathbb{E}, P, \cdot)$ es monótona creciente.*

Corolario 4.1.1. *Dados $\mathcal{E}_1, \mathcal{E}_2 \in$ EllipBase y dos enteros positvos $n > m$. Si $3A(\mathcal{E}_1, n) = 3A(\mathcal{E}_2, m)$ entonces $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_2$ o no se pueden comparar a través del índice $3A$ para n iteraciones.*

Este índice es el claro candidato para comparar cifrados basados en curva elíptica, ahora bien, su cálculo puede ser algo no trivial a priori. En las siguientes secciones se proponen algoritmos para estimar estos valores.

4.2. Estimación del índice 3A

El índice $3A$ sobre un esquema de cifrado para un número de iteraciones fijo no deja de ser una probabilidad y esta puede ser estimada a través del método de Montecarlo (véase [Gentle, 1998]).

Algoritmo 4.2.1 (Simulación de Montecarlo para $3A(K, \mathbb{E}, P, n)$)**.**

1. *Se toma un cuerpo K, una curva elíptica $\mathbb{E}$ sobre K, un punto base P sobre $\mathbb{E}$, un entero n y una lista vacía L.*

2. *Se fija un número máximo de iteraciones permitidas sobre el algoritmo Rho de Pollard, un real positivo α entre $(0, 1)$ que simboliza el nivel de confianza $1 - \alpha$ y un real positivo ε que simboliza el error máximo permitido.*

3. *Se toma un entero positivo aleatorio m y se calcula $Q = mP$.*

4. *Se aplica el esquema Rho de Pollard limitado a n iteraciones para intentar resolver el problema del logaritmo discreto. Si el algoritmo converge se concatena un 1 a la lista L, sino un 0.*

5. *Se repiten los pasos 3 y 4 dos veces más.*

6. *Se repiten los pasos 3 y 4 un total de $\left\lceil \frac{1}{\varepsilon^2 \alpha} \right\rceil$. Tras esto, el valor buscado será $avg(L)$ asumiendo un error menor a ε con un $(1 - \alpha) \cdot 100\,\%$ de probabilidad.*

En el Algoritmo 4.2.1 se generará una masa aleatoria simple (MAS) $(X_i)_{i\leq N} \sim Bernouilli(p)$, con $p = 3A(K, \mathbb{E}, P, n)$. En virtud de la **Ley de los Grandes Números** (véase [Billingsley, 1995]) dicha MAS cumple la siguiente propiedad:

$$\lim_{N\to\infty} \frac{\sum_{i=1}^{N} X_i}{N} = 3A(K, \mathbb{E}, P, n) \tag{4.1}$$

La Ecuación 4.1 muestra un resultado de convergencia asintótico. Este resultado nos dice que cuanto mayor sea la MAS generada, más ajustada será la simulación del valor $3A(K, \mathbb{E}, P, n)$, sin embargo no da ninguna cota explícita para controlar el error en función de N. Para lograr esta labor, se acude a la **Desigualdad de Tchebyshev**:

Teorema 4.2.1 (Desigualdad de Tchebyshev, [Billingsley, 1995])**.** *Sea f una función monótona creciente, X una variable aleatoria no negativa y $\varepsilon > 0$ un real, se cumple la siguiente cota:*

$$f(\varepsilon) \cdot P(X > \varepsilon) \leq E[f(X)]$$

Corolario 4.2.1. *Sean $X_1, ..., X_n$ una colección de variables aleatorias independientes igualmente distribuidas con media μ y desviación σ, se cumple la siguiente cota:*

$$P\left(\frac{|\sum(\frac{X_i}{n} - \mu)|}{\sigma\sqrt{n}} > \varepsilon\right) \leq \frac{\sigma^2}{n\varepsilon^2}$$

Corolario 4.2.2. *Sean $x_1, ..., x_n$ una masa aleatoria simple proveniente de una distribución Bernouilli de parámetro p independientes, $\varepsilon > 0$ y $\alpha \in (0, 1)$. Si $n \geq \lceil \frac{1}{\varepsilon^2\alpha} \rceil$ entonces $\left|\frac{\sum x_i}{n} - p\right| < \varepsilon$ con una confianza mayor igual que $1 - \alpha$.*

El Corolario 4.2.2 define la condición de parada sobre el Algoritmo 4.2.1–en el sexto paso– asegurando así un error menor a ε con una probabilidad del $(1 - \alpha) \cdot 100$ por ciento.

Aunque claramente el algoritmo anterior permite estimar teóricamente el valor $3A(K, \mathbb{E}, P, n)$, su aplicabilidad directa es inviable sobre cuerpos grandes. Bajo la Hipótesis 4.3.1 que veremos a continuación, se puede aplicar la **paradoja del cumpleaños** teniéndose que $3A(K, \mathbb{E}, P, O(\sqrt{|\mathbb{E}(K)|}))$ $\approx 0{,}5$ (véase [Paar and Pelzl, 2010]). Aplicando el Teorema de Hasse (3.4.4) a este razonamiento se sigue que, para una curva $\mathbb{E}(\mathbb{F}_p)$, tras $O(\sqrt{p})$ iteraciones del algoritmo Rho de Pollard hay un cincuenta por ciento de posibilidades de encontrar una colisión.

En la siguiente sección se presentará una idea para acelerar este algoritmo y se dará una explicación intuitiva de su funcionamiento.

4.3. Aproximación del índice 3A para cuerpos grandes

A lo largo de la presente sección, y con el objetivo de facilitar la exposición de las ideas principales, se adoptará la siguiente hipótesis simplificadora:

Hipótesis Simplificadora 4.3.1. *Dado un criptosistema de curva elíptica $(K, \mathbb{E}, P)$ y una función pseudoaleatoria $f : \mathbb{E} \to \mathbb{E}$ la sucesión de puntos:*

$$X_1 = f(P), \quad X_n = f(X_{n-1})$$

Conforma una masa aleatoria simple uniformemente distribuida sobre los elementos de $\langle P \rangle$.

La asunción de la Hipótesis 4.3.1 no es descabellada; en [Hildebrand, 2005] podemos encontrar que la distribución de la masa aleatoria simple $(X_i)_{i\leq N}$ tiende a una uniforme a medida que crece $ord(P)$, si N es suficientemente grande.

Tómese un criptosistema $(K, \mathbb{E}, P)$ y supóngase que se pretende resolver la resolución del problema del logaritmo discreto asociado a $H = mP$ bajo la estrategia de Pollard. Bajo la Hipótesis 4.3.1 la **paradoja del cumpleaños** marca que la probabilidad de encontrar una colisión en n iteraciones es:

$$3A(K, \mathbb{E}, P, n) = 1 - \prod_{i=0}^{n-1} \frac{ord(P) - i}{ord(P)} \tag{4.2}$$

donde $ord(P)$ se refiere al orden del elemento generador P (véase Sección 3.1.2). Este resultado se entiende mucho mejor si se piensa por complementario:

Ejemplo 4.3.1. *En un experimento se tienen cinco bolas en una caja, teniendo color único cada una. Una persona toma una bola al azar con los ojos vendados y posteriormente la vuelve a dejar en su sitio. Nos interesa calcular la probabilidad de tomar tres consecutivas de color distinto. Llamando a cada una de las selecciones X_1, X_2 y X_3, esto se traduce en términos probablísticos como:*

Tabla 4.1: Casos totales y favorables del Ejemplo 4.3.1

Selección	C. Totales	C. Favorables
X_1	5	5
X_2	5	4
X_3	5	3

$$P(Suceso) = P(X_1 \text{ sin repeticion}) \cdot P(X_2 \text{ sin repeticion}) \cdot P(X_3 \text{ sin repeticion}) =$$

$$\frac{5}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} = \prod_{i=0}^{2} \frac{5-i}{5} = 0{,}48$$

De igual manera, la probabilidad del suceso complementario, es decir, que al menos se tome una bola repetida sería $P(\neg Suceso) = 1 - P(Suceso) = 1 - \prod_{i=0}^{2} \frac{5-i}{5}$. Cambiando las bolas por los elementos del subgrupo generado $\langle P \rangle$ y las selecciones por los elementos generados durante la ejecución del esquema de Pollard, llegamos a la expresión anterior.

En el Ejemplo 4.3.1 cabe preguntarse por el número de elementos veces que tenemos que repetir el experimento con una caja de N bolas para tomar dos veces la misma pelota. Empíricamente se establece que el número de repeticiones necesarias para que se dé este suceso es una $O(\sqrt{N})$. En [Paar and Pelzl, 2010, Silverman, 2006] traducen este hecho de forma directa, estableciendo que se necesita iterar en el algoritmo de Pollard $O(\sqrt{p})$ veces, siendo p la característica del cuerpo con el que se está trabajando.

4.3.1. Simplificación asintótica del índice 3A

La expresión dada por la Ecuación 4.2 sigue siendo intratable computacionalmente. Al estar trabajando con factores tan grandes, podemos establecer una equivalencia asintótica entre dicho valor y una expresión analítica explícita. El siguiente teorema original presenta el resultado fundamental de este trabajo:

Teorema 4.3.1 (Aproximación asintótica del índice 3A)**.** *Bajo la Hipótesis 4.3.1, dado un criptosistema $(K, \mathbb{E}, P)$ y un natural n fijo se tiene que:*

$$\lim_{ord(P) \to \infty} \frac{1 - 3A(K, \mathbb{E}, P, n)}{\exp\left(-\frac{(n-1) \cdot n}{2 \cdot ord(P)}\right)} = 1$$

Demostración del Teorema 4.3.1

Antes de entrar de lleno en la demostración, introduciremos los siguientes resultados que serán utilizados como lemas para la prueba del teorema 4.3.1.

Lema 4.3.1 ([Spivak, 2006])**.** *Sea $\{x_n\}_{n\in\mathbb{N}}$ una sucesión de puntos decreciente a cero entonces se tiene la siguiente equivalencia:*

$$\lim_{n\to\infty} \frac{1+x_n}{e^{x_n}} = 1$$

Demostración. Sea $(K, \mathbb{E}, P)$ un criptosistema de curva elíptica. Fijemos un entero n sin pérdida de generalidad y asumamos que se cumple la Hipótesis 4.3.1 sobre el sistema.

Bajo estas condiciones, aplicando la Ecuación 4.2 se tiene que:

$$3A(K,\mathbb{E},P,n) = 1 - \prod_{i=0}^{n-1} \frac{ord(P)-i}{ord(P)} = 1 - \prod_{i=0}^{n-1} \left(1 - \frac{i}{ord(P)}\right)$$

Teniendo en cuenta la igualdad superior se procede a trabajar con la expresión enunciada en hipótesis. Pero antes se resalta la independencia de n respecto de P, por lo n no se ve afectado al tomar límites cuando $ord(P)$ tiende a infinito.

$$\lim_{ord(P)\to\infty} \frac{1-3A(K,\mathbb{E},P,n)}{\exp(-\frac{(n-1)n}{2\cdot ord(P)})} = \lim_{ord(P)\to\infty} \frac{\prod_{i=0}^{n-1}\left(1-\frac{i}{ord(P)}\right)}{\exp(-\frac{(n-1)n}{2\cdot ord(P)})} =$$

$$= \lim_{ord(P)\to\infty} \prod_{i=0}^{n-1} \frac{\left(1-\frac{i}{ord(P)}\right)}{\exp(-\frac{(n-1)n}{2\cdot ord(P)})} \cdot \frac{\exp(-\frac{i}{ord(P)})}{\exp(-\frac{i}{ord(P)})} =$$

$$= \lim_{ord(P)\to\infty} \frac{\exp(-\frac{\sum_{i=0}^{n-1} i}{ord(P)})}{\exp(-\frac{(n-1)n}{2\cdot ord(P)})} \cdot \left(\prod_{i=0}^{n-1} \frac{1-\frac{i}{ord(P)}}{\exp(-\frac{i}{ord(P)})}\right)$$

Dado que n es independiente de P se puede dividir el término derecho en n límites sin afectar el resultado pues el límite del producto es el producto de los los límites. Estos nuevos límites toman el valor uno en virtud del Lema 4.3.1. Reducimos así la expresión superior al término izquierdo que puede ser simplificada en virtud de la formula de Euler para progresiones armónicas, concluyendo así:

$$\lim_{ord(P)\to\infty} \frac{\exp(-\frac{\sum_{i=0}^{n-1} i}{ord(P)})}{\exp(-\frac{(n-1)n}{2\cdot ord(P)})} \cdot \left(\prod_{i=0}^{n-1} \frac{1-\frac{i}{ord(P)}}{\exp(-\frac{i}{ord(P)})}\right) =$$

$$= \left(\lim_{ord(P)\to\infty} \frac{\cancel{\exp(-\frac{(n-1)n}{2\cdot ord(P)})}}{\cancel{\exp(-\frac{(n-1)n}{2\cdot ord(P)})}}\right) \cdot \prod_{i=0}^{n-1} \underbrace{\left(\lim_{ord(P)\to\infty} \frac{1-\frac{i}{ord(P)}}{\exp(-\frac{i}{ord(P)})}\right)}_{1} = 1$$

Probando así que la expresión propuesta conforma una expresión asintótica del índice 3A bajo hipótesis simplificadoras.

□

Al revisar la demostración anteriormente citada, uno puede percatarse que la convergencia del teorema 4.3.1 no es uniforme, sino puntual –o en términos probabilísticos, en ley– implicando esto, que el error en la aproximación del índice $3A$ no es homogéneo para todo n. Siendo precisos, el error será más ajustado cuanto más bajo sea el valor de n . De igual manera, a medida que aumenta el orden del elemento generador, el error se homogeniza sobre la estimación.

4.4. Breves comentarios

La métrica introducida en este capítulo permite evaluar la seguridad en criptosistemas de ECC midiendo la probabilidad de vulneración en n iteraciones del algoritmo Rho de Pollard. No solo esto, la relación de orden inducida por esta métrica y expuesta en la Proposición 4.1.1permite establecer *benchmark* entre los elementos de la clase de criptosistemas de curva elíptica.

Dado que el cálculo de este valor no es trivial, se exponen dos métodos al lector que deberá escoger según la ocasión. Si desea establecer una comparativa cuando el elemento generador de los parámetros de dominio tiene orden pequeño, entonces deberá acudidir a la simulación estadística aplicando el Algoritmo 4.2.1. Si por el contrario el generador posee un orden de gran magnitud entonces puede acudir a la fórmula representación asintótica del índice expuesta en el Teorema 4.3.1, bajo la asunción de la hipótesis simplificadora 4.3.1.

De esta manera, las incertidumbres expuestas en el capítulo introductorio se ven resueltas a través del desarrollo lógico y razonado establecido en este capítulo.

Capítulo 5

Desarrollo de la propuesta

If you are looking for different results, do not do the same thing.

Albert Einstein

El objetivo principal de este trabajo será evaluar el rendimiento de la métrica $3A$, definida en la sección anterior. Para ello estableceremos dos estudios separados:

- En el primero tomaremos criptosistemas basados en la curva $Y^2 = X^3 + 3X + 5$ sobre cuerpos primos "pequeños" y calcularemos el índice $3A$ de forma empírica y teórica, mediante el Algoritmo 4.2.1 y el Teorema 4.3.1, comparando los resultados obtenidos.
- En el segundo de los experimentos utilizaremos el índice $3A$ para comparar los criptosistemas inducidos por las curvas $Z25519$ y $P384$, cuantificando la diferencia entre la seguridad que estos presentan.

Este estudio se llevará a cabo en Python, utilizando para ello la librería `EllipticCUrVa`, –de creación propia– que permite trabajar con aritmética de curva elíptica de forma amigable en este entorno de programación.

5.1. EllipticCUrVa

El módulo `EllipticCUrVa` es una librería de aritmética no estándar, divida en dos grandes desarrollos. El primero implementa un marco para trabajar con aritmética modular sobre cuerpos finitos. Por otro lado, el segundo se apoya en este para implementar un marco de trabajo con aritmética de curvas elípticas.

5.1.1. Aritmética modular

Antes de entrar de lleno en esta sección, se debe resaltar que para que este proyecto sea utilizable y no puramente académico, se debe trabajar con enteros de longitud arbitraria. Las versiones de Python 3.X manejan cualquier tamaño de entero (véase [Foundation, 2025]) de forma automática y los cálculos con estos se encuentran optimizados, es por esto que este módulo se encuentra restringido a estos entornos.

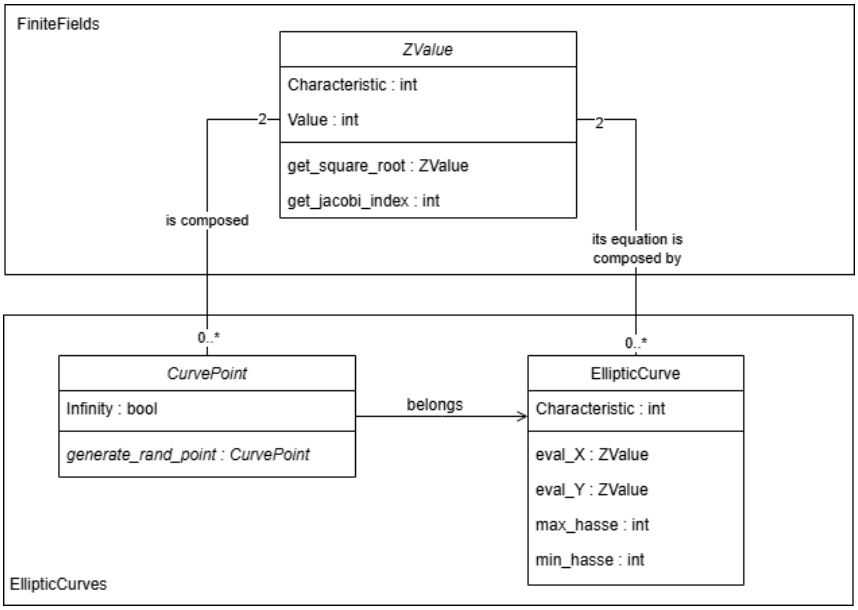


Figura 5.1: Diagrama de clases del módulo de aritmética

Se valoró manejar enteros de precisión arbitraria a bajo nivel utilizando para ello representaciones numéricas a bajo nivel, similares a las presentes en el Capítulo 4 de [Knuth, 2014]. No obstante, este acto se aplazó a futuras versiones de la solución para no retrasar el lanzamiento de la versión preliminar.

Toda la implementación de la aritmética modular se ha conseguido encapsular en una única clase llamada `ZValue`, manejando todas las operaciones a través de sobrecarga de operadores [Foundation, 2025]. Para ilustrar este párrafo mejor refiérase al Ejemplo A.1 en el Anexo A:

Dentro de los ejemplos anteriores cabe resaltar alguna de las operaciones realizadas. Por ejemplo, las potencias se encuentran implementadas a través de la función `pow` de Python [Foundation, 2025] que a bajo nivel implementa una estrategia parecia a la descrita en el algoritmo de la Figura 3.1.

De igual manera, el cálculo de los Símbolos de Jacobi es computado mediante el Criterio de Euler . Utilizando el cálculo de los símbolos, se ha implementado el algoritmo de Tonelli-Shanks para determinar los radicales de un elemento. Si el símbolo de Jacobi es -1, la rutina encargada de ejecutar el algoritmo lanza una excepción controlada.

5.1.2. Aritmética de curva elíptica

Al igual que antes, se han encapsulado todas las operaciones ligadas a la aritmética de curva elíptica en dos clases. La primera, `EllipticCurve`, representa el concepto de curva elíptica. Por otro lado tenemos la clase `CurvePoint` que simboliza un punto cualquiera sobre una curva elíptica. Estas clases se encuentran relacionadas entre sí y a su vez a se encuentran relacionadas con la clase `ZValue` por razones operacionales.

EllipticCurve

La clase `EllipticCurve` posee dos atributos de tipo `ZValue` representando los parámetros A y B de la ecuación de la curva en forma de Weiestrass (véase Teorema 3.4.1). De igual manera, posee otro atributo llamado `Characteristic` representando la característica del cuerpo primo sobre el que se está trabajando.

```
from EllipticCurva.Arithmetic.FiniteFields import ZValue
from EllipticCurva.Arithmetic.EllipticCurves import EllipticCurve

# Curva en forma de Weiestrass con forma
# Y^2 = X^3 + 2X + 3 sobre Z/5Z

curve = EllipticCurve(characterisic = 5,
                      A=2,
                      B=3)
print(curve) # Y^2 = X^3 + 2X + 3 mod 5

# Curva en forma de Weiestrass con forma
# Y^2 = X^3 + 2X + 3 sobre Z/5Z con ZValues

A = ZValue(2,5)
B = ZValue(3,5)
curve = EllipticCurve(5,A,B)
print(curve) # Y^2 = X^3 + 2X + 3 mod 5
```

Figura 5.2: Ejemplo implementación curva elíptica

En el Figura 5.2 se puede mostrar como se trabaja con la clase `EllipticCurve`.

CurvePoint

La clase `CurvePoint` posee dos atributos tipo `ZValue` representando las coordenadas X e Y del punto sobre el cuerpo primo en el que se está operando, un atributo `bool` llamado `Infinity` que representa si el punto en cuestión es el Punto del Infinito (véase Notación 3.4.1) o no, y una instancia de `EllipticCurve` referenciando la curva a la que este punto pertenece. Al igual que pasaba con la clase `ZValue`, todas las operaciones se encuentran codificadas bajo la técnica de *operator overwriting* [Foundation, 2025]. En el Ejemplo A.2 se ilustra sobre el manejo de dichos operadores.

5.1.3. Validación

Como durante el desarrollo de los módulos anteriores se ha aplica metodología TDD (Test Driven Development) [Koskela, 2007], todos los métodos han sido probados a mediante test unitarios utilizando para ello la librería `unittest`.

5.2. Rendimiento de 3A en sistemas pequeños

Este experimento se realiza con afán de contrastar el rendimiento de la cota asintótica razonada versus la distribución original, calculada empíricamente mediante simulación de Montecarlo.

Para este experimento se ha tomado la curva $\mathbb{E} : Y^2 = X^3 + 3X + 5$ sobre los cuerpos $\mathbb{F}_{31}, \mathbb{F}_{101}, \mathbb{F}_{503}, \mathbb{F}_{1009}$ y $\mathbb{F}_{2003}$. Para ello se ha calculado su dominio y de entre todos los puntos se ha escogido uno con orden maximal para ser usado como generador.

Tras esto, se ha aplicado el Algoritmo 4.2.1 para calcular empíricamente el índice $3A$. En el cálculo se marca un error máximo permitido de un 0.01 con un nivel de confianza del 95 %. Aunque a priori este error se controlaba calculando el tamaño muestral mediante la desigualdad de Tchebyshev, para estos experimentos se ha utilizado la desigualdad Dvoretzky–Kiefer–Wolfowitz (DKW) (véase [Popiński, 2022]):

$$P(|F(x) - F_n(x)| > \varepsilon) \leq 2 \cdot e^{-2 \cdot \varepsilon^2 \cdot n}, \quad \forall \varepsilon > 0 \tag{5.1}$$

consiguiendo el mismo nivel de confianza que nos aseguraba con muchas menos iteraciones, ahorrando así costo computacional.

Tras esto, se ha calculado la aproximación teórica al índice $3A$ de los criptosistemas aplicando el Teorema 4.3.1, representándose gráficamente los dos resultados obtenidos.

5.2.1. Definición de los criptosistemas

Se ha calculado el dominio de cada una de las curvas, usando para ello un algoritmo de búsqueda exhaustiva sobre los puntos de los planos $\mathbb{F}_p^2$. De igual manera, se ha calculado el orden de los índices de forma secuencial hasta lograr dar con un elemento de orden maximal. Para el cálculo de los órdenes se ha seguido una estrategia aditiva en la que se sumaba el elemento objetivo consigo mismo hasta obtener el punto del infinito, sumando uno en cada una de las iteraciones a una variable contador, inicializada previamente en 1. En la Tabla 5.1 figuran los parámetros seleccionados para los experimentos.

Tabla 5.1: Parámetros de los sistemas elegidos

Cuerpo	$\mathbb{F}_{31}$	$\mathbb{F}_{101}$	$\mathbb{F}_{503}$	$\mathbb{F}_{1009}$	$\mathbb{F}_{2003}$
Generador X	0	0	1	0	1816
Generador Y	6	45	3	24	6
Generador Orden	38	115	493	1023	1020
Curva Orden	38	115	493	1023	2040

5.2.2. Definición del muestreo aleatorio

Para cada uno de los criptosistemas en la Tabla 5.1, el Algoritmo 4.2.1 se efectúa tal y como se detalla a continuación: Denotemos por G al generador del criptosistema y tomemos una muestra aleatoria simple de enteros uniformemente distribuidos $d_1, ...d_N$ en $[1, ord(G) - 1]$. Para cada d_i se calcula la clave pública $P_i = d_i \cdot G$ y se resuelve el ECDLP asociado utilizando para ello una estrategia de Pollard **sin aplicar la aceleración de Floyd**. Para cada una de las estrategias, se utiliza una función pseudoaleatoria $\psi : \mathbb{E} \to \mathbb{E}$ definida como:

$$\psi(P) = \begin{cases} 2P, & P \in \Omega_0 \\ P + G, & P \in \Omega_1 \\ P + P_i, & P \in \Omega_2 \end{cases}$$

Donde $\Omega_0 \sqcup \Omega_1 \sqcup \Omega_2 = \mathbb{E}$. Para definir la pertenencia de P a una de las particiones, se toma una función lineal, $f_i(x) = a \cdot x + b$, aleatoriamente generada y se aplica la regla $P \in \Omega_j \Leftrightarrow f_i(P_x) = j \bmod 3$. Tras encontrar la primera colisión por la estrategia de Pollard se devuelve el número de iteraciones ejecutado n_i. La muestra $n_1, ..., n_N$ son los valores utilizados para la simulación de Montecarlo.

5.3. Comparación Z25519 vs. P384

En esta sección se establece un análisis comparativo de los criptosistemas inducidos por las curvas Z25519 y P384 utilizando para ello el índice $3A$. Dado que el orden del elemento generador en sendos criptosistemas es suficientemente alto, los resultados se aproximan mediante el Teorema 4.3.1 y se representan posteriormente mediante escala logarítmica, por facilidad de comprensión. A parte de esto se aplica la siguiente fórmula:

$$\left| \int_0^{\infty} x \cdot \frac{\partial}{\partial x} \left(3A(P384, x) - 3A(Z25519, x) \right) dx \right| \tag{5.2}$$

Simbolizando la cantidad vista en la Ecuación 5.2 la diferencia en el número de iteraciones que el algoritmo Rho de Pollard requeriría, en promedio, para comprometer la seguridad de los criptosistemas. El código de este experimento se encuentra en el Apéndice D.

5.3.1. Parámetros de dominio de los criptosistemas

Para ejecutar los experimentos sobre los criptosistemas Z25519 y P384 se deben utilizar los parámetros de domino asociados. Aunque estos son públicos y se pueden encontrar en [Chen et al., 2019], exponemos estos en la Tabla 5.2.

Tabla 5.2: Parámetros de dominio de los criptosistemas

	Z25519	**P384**
Característica	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$	$2^{255} - 19$
A	-3	19298681539552699237261830834781317975544997444273427339909597334573241639236
B	55751746669818908907645289078257140818241103727901012315294400837956729358436	27580193559959705877849011840389048093056905856361568521428707301988689241309860865136260764883745107765439761230575
Generador X	19298681539552699237261830834781317975544997444273427339909597334652188435546	26247035095799689268623156744566981891852923491109213387815615900925518854738050089022388053975719786650872476732 08
Generador Y	43114425171068552920764898935933967039370386198203806730763910166200978582548	83257109614890299855467512895201081792878530488613155947092059024805031998844192244386437603929473330780865116278 71
Generador Orden	72370055773322622139731865630429942408571163593799076060019509382854542509 89	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$

Capítulo 6

Análisis y resultados

Sometimes you need to stop the train. You have to take a step back, because the work has to come from deep within you.

Steve Aoki

En la siguiente sección se presentan los resultados obtenidos en los dos experimentos descritos en la Sección 5, ofreciéndose una visión interpretada de los hallazgos obtenidos y una breve explicación de sus implicaciones.

6.1. Rendimiento de 3A en cuerpos pequeños

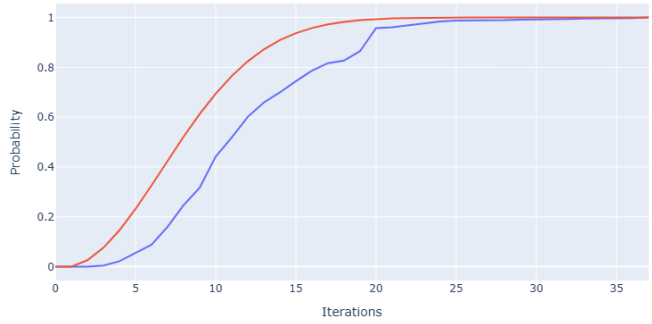
En la Figura 6.1 se presenta el índice $3A$ calculado empíricamente mediante simulación de Montecarlo y asintóticamente mediante la fórmula dada por el Teorema 4.3.1, para los criptosistemas presentados en la Tabla 5.1.

En todas las gráficas se aprecia el efecto de la convergencia del índice $3A$ a la función propuesta, acercándose la distribución empírica cada vez más a la asintótica. Numéricamente también se puede apreciar este acercamiento entre las distribuciones por medio de la divergencia de Kullback-Leiber, cuyos valores están expuestos en la Tabla 6.1. Entrando más de lleno en esta convergencia, se aprecia en los experimentos realizados que la similitud entre la expresión asintótica y la distribución empírica se maximiza o bien cuando la probabilidad de ataque efectivo es muy pequeña, o bien cuando el numero de iteraciones efectuadas son muy grandes. Esto posee lógica pues, como se anticipó en la Sección 4.3.1, la demostración del Teorema 4.3.1 pasa por la aplicación consecutiva de inifinitésimos equivalentes sobre la probabilidad de ataque efectivo, a medida que la probabilidad crece, el error sobre la cota asintótica crece; sin embargo, este error luego se vuelve a ver corregido dado que sendas expresiones, empírica y asintótica, son funciones monótonas crecientes acotadas superiormente por uno.

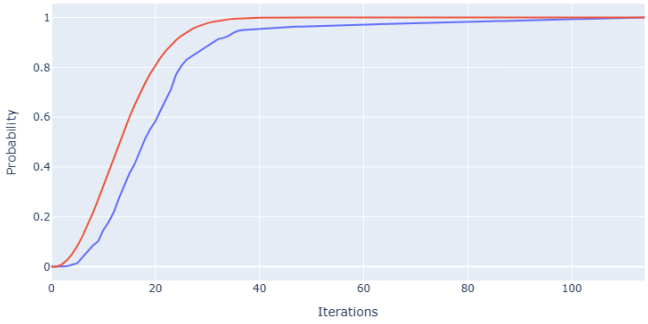
Tabla 6.1: Divergencia de Kullback-Leiber entre las distribuciones empíricas y asintóticas

Cuerpo	$\mathbb{F}_{31}$	$\mathbb{F}_{101}$	$\mathbb{F}_{503}$	$\mathbb{F}_{1009}$	$\mathbb{F}_{2003}$
Divergencia KL	0,5149	0,5188	0,2048	0,2254	0,2058

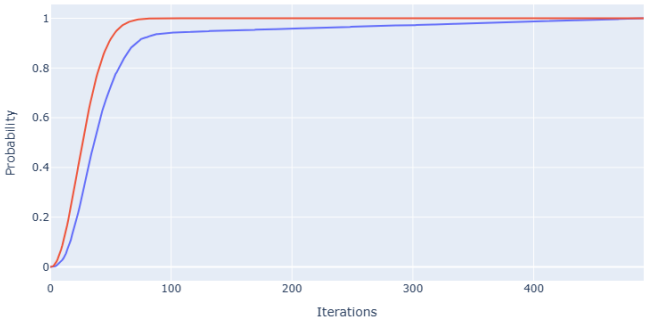
Basándonos en este tipo de convergencia, como la representación asintótica sigue la misma tendencia que la representación empírica cuando la probabilidad de ataque efectivo por el algoritmo



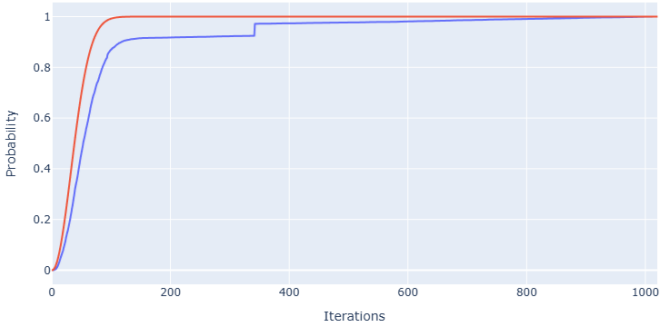
(a) Experimento sobre el cuerpo $\mathbb{F}_{31}$



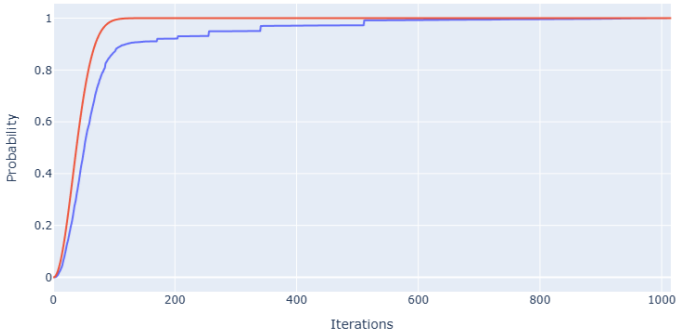
(b) Experimento sobre el cuerpo $\mathbb{F}_{101}$



(c) Experimento sobre el cuerpo $\mathbb{F}_{503}$



(d) Experimento sobre el cuerpo $\mathbb{F}_{1009}$



(e) Experimento sobre el cuerpo $\mathbb{F}_{2003}$

Figura 6.1: Comparativa del índice $3A$ asintótico (rojo) y empírico (azul) con la curva $Y^2 = X^3 + 3X + 5$ sobre distintos cuerpos

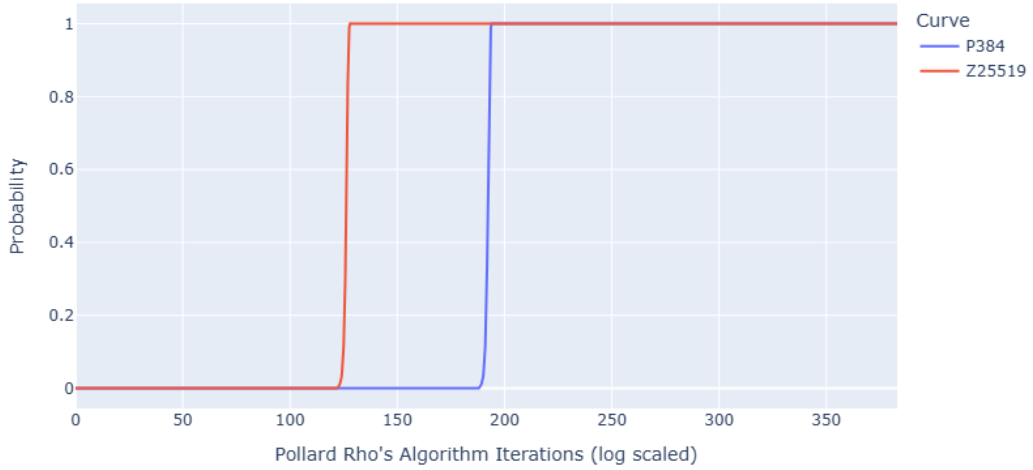


Figura 6.2: Evaluación del $3A$ asintótico sobre los sistemas $Z25519$ y $P384$

Rho de Pollard es baja, el Teorema 4.3.1 podría ser utilizado para estimar el número de iteraciones a partir del cuál un criptosistema empezaría a ser vulnerable por el ataque Rho de Pollard, dando esto sentido al siguiente experimento.

6.2. Comparación de Z25519 vs. P384

En este segundo experimento se comparan los criptosistemas asociados a las curvas $Z25519$ y $P384$ a través del índice $3A$. Mientras la Figura 6.2 muestra la representación del Teorema 4.3.1 asociada a cada uno de los criptosistemas, reflejando el numero de iteraciones necesarias en escala logarítmica en base dos para mejor comprensión del lector, la Tabla 6.2 tabula los valores de la métrica $3A$, escalados en logaritmo base dos. En sendas representaciones es palpable que $P384$ parece superior a $Z25519$ como criptosistema en lo que a seguridad se refiere.

Puesto que la representación asintótica del índice $3A$ conforma una muy buena aproximación sobre criptosistemas de gran cuando la probabilidad de ataque efectivo es pequeña, se pueden estudiar los valores $3A(Z25519, 2^{124,87}) = 0{,}1$ y $3A(P384, 2^{190,88}) = 0{,}1$ con un nivel de confianza elevado. Esto implica el criptosistema asociado a $Z25519$ comienza a ser vulnerable por el ataque Rho de Pollard mucho antes que el asociado a $P384$.

Por otro lado, si se asume la convergencia asintótica de las distribuciones, a través de la Ecuación 5.2 –cuyo desarrollo en este caso particular puede ser encontrado en el Anexo D – llegaríamos al diferencia promedio que necesitan los criptosistemas para ser vulnerados, resultando todo esto en $\sqrt{\frac{\pi \cdot ord(P384)}{2}} - \sqrt{\frac{\pi \cdot ord(Z25519)}{2}} \approx 2^{192,33}$. Permitiendo deducir que el criptosistema $Z25519$ es considerablemente más débil que el criptosistema $P384$, cuantificando la diferencia de seguridad entre criptosistemas.

Tabla 6.2: Métrica $3A$ para los criptosistemas $P384$ y $Z25519$ con iteraciones escaladas logarítmicamente en base dos

3A	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Z25519	124.87	125.42	125.76	126.02	126.24	126.44	126.63	126.84	127.10	128
P384	190.88	191.42	191.76	192.02	192.24	192.44	192.63	192.84	193.10	194

6.3. Discusión de los resultados

Los resultados obtenidos en este estudio ofrecen una métrica sólida para cuantificar y comparar la seguridad de un criptosistemas de curva elíptica.

Aunque el Teorema 4.3.1 se basa en una aproximación bajo la Hipótesis 4.3.1, esta se puede ver reforzada por [Bos et al., 2014]. Dado un criptosistema $(K, \mathbb{E}, P)$ el Teorema 2.1 del trabajo citado ofrece la siguiente cota superior al índice $3A$:

$$3A(K, \mathbb{E}, P, n) \geq 1 - \exp\left(\frac{-n^2 \cdot e}{ord(P) \cdot \log(ord(P)) \cdot (2+\delta)}\right) - \frac{1}{ord(P)^{\gamma}} \tag{6.1}$$

$$\delta, \gamma > 0;\ n > \sqrt{\frac{ord(P) \cdot \log(ord(P)) \cdot (2+\delta)}{e}}\log(2)$$

Siendo inmediato comprobar que la expresión asintótica en el Teorema 4.3.1 satisface la cota dada por la Ecuación 6.1.

Por otro lado, en [Kuhn and Struik, 2001] se presenta un estudio similar al nuestro en el que se expone la equivalencia asintótica $1-3A(K, \mathbb{E}, P, n) \approx \exp\left(\frac{-n^2}{2 \cdot ord(P)}\right)$, que a su vez es un infinitésimo equivalente de la expresión del Teorema 4.3.1. En este mismo estudio se expone que el numero de iteraciones promedio necesarias para encontrar una colisión es $\sqrt{\frac{\pi \cdot ord(P)}{2}}$, correspondiendose con la expresión expuesta en el segundo experimento. Además, el autor indica que el error relativo a la hora de utilizar esta aproximación se comporta como una $O(ord(P)^{-1/2})$.

Se debe resaltar que el Teorema 4.3.1 es muy dependiente de la hipótesis simplificadora. Esto se podría paliar haciendo uso de los resultados expuestos en [Hildebrand, 2005] donde se demuestran distintos resultados referentes a la convergencia del rango de caminatas aleatorias sobre grupos abelianos finitos a la distribución uniforme.

Capítulo 7

Conclusiones y trabajo futuro

At last, my love has come along. My lonely days are over and life is like a song

Etta James

En el siguiente epígrafe se presentan las conclusiones del estudio, relacionando estas con los objetivos propuestos la Sección 1.2, y se proponen futuras vías de trabajos basadas en la teoría desarrollada.

7.1. Conclusiones

El proyecto ha culminado con la creación del módulo EllipticCurVa en Python, diseñado para trabajar con aritmética no estándar de forma intuitiva y amigable. La estructura de esta librería se ha concebido para abordar los objetivos definidos en el plan de trabajo.

En primer lugar, se ha satisfecho el objetivo **OB-01** gracias a un submódulo integrado en EllipticCurVa que permite realizar cálculos en aritmética de cuerpo finito. De igual manera, se ha cumplido con el objetivo **OB-02** al incluir otro apartado dentro de la misma dedicado al manejo de aritmética de curva elíptica. Por último, el objetivo **OB-03** también se ve cumplido gracias a la librería al incorporar un submódulo de ataque que implementa diferentes variaciones del ataque Rho de Pollard.

La contribución original de este trabajo radica en la introducción de la métrica $3A$, la cual se presenta y detalla en el Capítulo 4. Esta métrica permite evaluar la seguridad de un criptosistema frente versión primigenia del ataque Rho de Pollard, lo que justifica la consecución del objetivo **OB-04**. Para estimar los valores de esta, se ha desarrollado el Algoritmo 4.2.1, un método de simulación de Montecarlo que aproxima los valores de esta métrica; y adicionalmente, se ha demostrado el Teorema 4.3.1, que establece una expresión asintótica para el índice $3A$ bajo ciertas hipótesis simplificadoras.

El rendimiento de la métrica ha sido confirmada a través de un estudio de validación en el que se ha observado que los distintos métodos de cálculo convergen a medida que crece el orden del elemento generador, cumpliendo así con el objetivo **OB-05**. Del estudio anterior, se desprende una conclusión notable a modo de hallazgo complementario: cuando la probabilidad de un ataque exitoso es baja ($< 0{,}01$), los diferentes métodos para estimar el índice $3A$ producen resultados prácticamente idénticos. Esto sugiere que el índice $3A$ puede ser una herramienta que permita determinar el número de iteraciones necesarias en un ataque Rho de Pollard a partir del cual

existe riesgo de vulneración.

Para finalizar, se ha aplicado la métrica $3A$ para comparar los criptosistemas Curve25519 y P389. Los resultados del análisis demostraron que el criptosistema P389 es considerablemente más resiliente a los ataques que Curve25519, mostrando así la utilidad de la métrica como herramienta de evaluación de seguridad.

7.2. Trabajo futuro

A continuación se introducen posibles líneas de trabajo nuevas basadas en los resultados obtenidos en este trabajo.

Cabe destacar que este estudio se encuentra fuertemente vinculado a la Hipótesis Simplificadora 4.3.1. Es por esto que se plantea como una futura línea de trabajo la generalización del Teorema 4.3.1 por medio de la investigación desarrollada en [Hildebrand, 2005] sobre convergencia a la distribución uniforme del rango de una caminata aleatoria sobre un grupo abeliano finito, evitando así el uso de hipótesis simplificadoras.

De igual manera, los cálculos para estimar la métrica $3A$ se han realizado sin tener en cuenta la aceleración de Floyd. Dado que el algoritmo Rho de Pollard y la detección de ciclos por el algoritmo de Floyd siempre suelen ir de la mano en la literatura, sería más que conveniente la reformulación de una nueva métrica que tenga en cuenta este efecto.

Por último, esta métrica sólo tiene en cuenta el ataque Rho de Pollard por ser, en general, el más eficiente para romper criptosistemas de curva elíptica. Sin embargo, hay casos conocidos en los que adoptar otras estrategias es preferible; por ejemplo, si la curva posee el mismo número de puntos que su soporte, es preferible el *ataque de curva anómala*; si el grado de inmersión es bajo, la estrategia MOV suele ser preferible; y si la curva factoriza como producto de grupos con factores primos pequeños, el ataque de Polihg-Hellman suele ser más eficiente que el Rho de Pollard. Por esto mismo, se propone el ajuste de estas métricas para todos estos algoritmos, y derivar en una medida compuesta de todas ellas.

Finalmente, se valora llevar EllipticCUrVa a un entorno productivo. Para ello, se pretenden añadir algunas funcionalidades como la implementación del algoritmos de Schoof para el cálculo del número de puntos en el soporte de una curva. De igual manera, se plantearía la optimización de alguno de los métodos de la librería por medio de paralelización y reescribiendo alguno de estos a bajo nivel en el lenguaje de programación C. En tanto al submódulo de ataque, se valoraría la implementación de nuevos ataques, estableciendo un método final de ataque automático que ejecute uno u otro, en función de las propiedades intrínsecas de la curva, que intente vulnerar esta de la forma más eficiente posible.

Siguiendo con EllipticCUrVa, una posible rama de estudio sería la anexión de un nuevo módulo de ataque basado en computación cuántica con la librería de Pennylane, que implemente los ataques cuánticos al ECDLP. Una vez se implementen estos, se puede establecer una metodología similar a la utilizada en este trabajo para ajustar nuevas métricas basadas en este nuevo tipo de ataques.

Por último, y en consonancia con la cuántica, está tomando notoriedad un nuevo cifrado postcuántico basado en volcanes de l-isogénias. Este se encuentra estrechamente relacionado con la criptografía de curva elíptica, pues las isogenias son isomorfismos entre curvas elípticas que respetan la ley de grupo. La generalización de lo realizado a esta nueva teoría podría suponer un marco de trabajo sumamente innovador en lo que criptografía postcuántica se refiere.

Parte III

Anexos

Anexo A

Ejemplos de implementación

```python
from EllipticCurva.Arithmetic.FiniteFields import ZValue

# Creacion de 4 y 7 sobre Z/5Z
z4 = ZValue(value=4, characteristic=5)
z7 = ZValue(7, 5)

print(z4) # 4 mod 7
print(z7) # 2 mod 7

# Operaciones modulares
suma = z4 + z7
print(suma) # 6 mod 7

producto = z4 * z7
print(producto) # 1 mod 7

resta = z4 - z7
print(resta) # 2 mod 7

division = z4 / z7
print(division) # 2 mod 7

# Operaciones con enteros
suma = z4 + 7
print(suma) # 6 mod 7

producto = z4 * 7
print(producto) # 1 mod 7

resta = z4 - 7
print(resta) # 2 mod 7

division = z4 / 7
print(division) # 2 mod 7

potencia = z4 ** 6
print(potencia) # 1 mod 7

# Otras operaciones especiales

jacobi = z4.get_jacobi_index()
print(jacobi) # 1

square_root = z4.get_square_root()
print(square_root) # 2 mod 7
```

Figura A.1: Ejemplo uso ZValue

```
from EllipticCurva.Arithmetic.EllipticCurves import CurvePoint
from EllipticCurva.Arithmetic.EllipticCurves import EllipticCurve

# Curva en forma de Weiestrass con forma
# Y^2 = X^3 + 2X + 3 sobre Z/5Z

curve = EllipticCurve(characterisic = 5,
                      A=2,
                      B=3)
print(curve) # Y^2 = X^3 + 2X + 3 mod 5

# Tomamos dos puntos aleatorios de la curva

p1 = CurvePoint.generate_rand_point(curve)
p2 = CurvePoint.generate_rand_point(curve)

# Suma y resta de puntos

suma = p1 + p2
resta = p1 - p2

# Producto por enteros

cinco_veces_punto_uno = 5*p1
cinco_veces_punto_uno = p1*5
```

Figura A.2: Ejemplo implementación curva elíptica

Anexo B

Código de los experimentos

B.1. Estudio de validación

```
# Librerias

from numpy.random import randint
import numpy as np
import pandas as pd
from math import log
import plotly.express as px
from EllipticCUrVa.Arithmetic.EllipticCurves.CurvePoint
    import CurvePoint
from EllipticCUrVa.Arithmetic.EllipticCurves.EllipticCurve
    import EllipticCurve
from EllipticCUrVa.Attack.PollardRho
    import PollarRhoAttackNoFloyd
from copy import deepcopy
from tqdm import tqdm
from typing import Generator
```

Figura B.1: Carga de librerías

```
### FUNCIONES AUXILIARES

# Funcion auxiliar para buscar el soporte de la curva
def curve_support(curve : EllipticCurve) -> list[CurvePoint]:

    def _point_generation(curve : EllipticCurve) ->
        Generator[tuple[int,int], None, None]:

        x : int = 0
        y : int = 0

        while y < curve.Characteristic:
            yield x,y
            x += 1

            if x == curve.Characteristic:
                x = 0
                y += 1

    candidates = [CurvePoint(curve, x, y)
                  for x, y in _point_generation(curve)
                  if curve.is_in_coordinates(x,y)]

    return candidates

# Funcion auxiliar para seleccionar un generador de la curva
def choose_generator(curve_support : list[CurvePoint]) ->
    tuple[CurvePoint, int]:

    curve_order : int = len(curve_support) + 1
    max_order_founded : int = 0
    generator_proposed : CurvePoint = None
    aux_counter : int = 0

    for point in curve_support:

        aux_point = deepcopy(point)
        aux_counter = 1

        while not aux_point.isInfinite:
            aux_point = aux_point + point
            aux_counter += 1

        if aux_counter > max_order_founded:
            max_order_founded = aux_counter
            generator_proposed = deepcopy(point)

        if max_order_founded == curve_order:
            break

    return generator_proposed, max_order_founded
```

Figura B.2: Implementación de funciones auxiliares (Parte 1)

```
# Funcion auxiliar que genera un muestreo aleatorio de las
# iteraciones que tarda el algoritmo Rho de Pollard en romper
# un cifrado
@np.vectorize
def sample_pollard_rho(private_key : int,
                       generator_point : CurvePoint,
                       generator_order : int,
                       max_iter_algorithm : int) -> int:

    public_key : CurvePoint = private_key * generator_point

    pra = PollarRhoAttackNoFloyd(max_iter_algorithm,
        generator_point, public_key, generator_order)
    result = pra.run()
    return -1 if result is None else result

# Encapsulacion del metodo de Montecarlo en una funcion
def montecarlo_3A(generator_point : CurvePoint,
                  generator_order : int,
                  max_iter_algorithm : int,
                  max_allowed_error : float = 0.01,
                  confidence_level : float = 0.05) -> list[float]:

    serie : list[int] = list()
    repetitions = int(-log(confidence_level / 2) / 2 /
        (max_allowed_error ** 2) + 1)

    elements = randint(1, generator_order - 1, repetitions)
    serie = sample_pollard_rho(elements, generator_point,
    generator_order, max_iter_algorithm)

    serie : pd.Series[int] = pd.Series(serie)
    serie = serie[serie != -1]
    serie = serie.groupby(serie).count().cumsum() / serie.count()
    serie_aux = pd.Series(0, index = range(serie.index.max() + 1))
    serie_aux += serie
    serie_aux.ffill(inplace=True)
    serie_aux.fillna(0, inplace=True)

    return serie_aux.to_list()

# Divergencia de Kullback-Leiber
def kl_divergence(density_x : np.ndarray, density_y : np.ndarray) ->
    float:

    positions = (density_y != 0) & (density_x != 0)
    density_x = density_x[positions]
    density_y = density_y[positions]

    return np.sum(density_x * np.log(density_x / density_y))
```

Figura B.3: Implementación de funciones auxiliares (Parte 2)

```
# DEFINICION DE PARAMETROS

# Se trabajara con la curva Y^2 = X^3 + 3X + 5 sobre los cuerpos de
# 31, 101, 153, 1009, 2003 y 10007 elementos

characteristics = [31, 101, 503, 1009, 2003]#, 10007]
A = 3
B = 5


print("Calculando curvas:")
curves = [EllipticCurve(x, A, B) for x in characteristics]
print("Curvas calculadas.\n")

print("Calculando el soporte de las curvas:")
curves_support : list[list[CurvePoint]] = [curve_support(curve)
    for curve in tqdm(curves)]
print("Soporte de las curvas calculado.\n")

print("Calulando ordenes de la curvas:")
curves_orders : list[int] = [len(support) + 1 for support
    in tqdm(curves_support)]
print("Ordenes calculados: ", curves_orders, ".\n")

print("Calculando generadores de la curva y sus ordenes:")
curves_generators : list[tuple[CurvePoint, int]] =
    [choose_generator(support) for support in tqdm(curves_support)]
print("Generadores calculados.\n")
```

Figura B.4: Definición de parámetros

```
# PARAMETROS DEL ESTUDIO

print("Curva: ", curves[0])
print("Generador: ", curves_generators[0][0])
print("Orden de la curva: ", curves_orders[0])
print("Orden del generador: ", curves_generators[0][1])

# COMIENZO DEL ESTUDIO
aaa_f31 = montecarlo_3A(generator_point=curves_generators[0][0],
                        generator_order=curves_generators[0][1],
                        max_iter_algorithm=curves_generators[0][1])
```

Figura B.5: Estudio curva F_{31}

```
# PARAMETROS DEL ESTUDIO

print("Curva:", curves[1])
print("Generador:", curves_generators[1][0])
print("Orden de la curva:", curves_orders[1])
print("Orden del generador:", curves_generators[1][1])

# COMIENZO DEL ESTUDIO
aaa_f101 = montecarlo_3A(generator_point=curves_generators[1][0],
                         generator_order=curves_generators[1][1],
                         max_iter_algorithm=curves_generators[1][1])
```

Figura B.6: Estudio curva F_{101}

```
# PARAMETROS DEL ESTUDIO

print("Curva:", curves[2])
print("Generador:", curves_generators[2][0])
print("Orden de la curva:", curves_orders[2])
print("Orden del generador:", curves_generators[2][1])

# COMIENZO DEL ESTUDIO
aaa_f503 = montecarlo_3A(generator_point=curves_generators[2][0],
                         generator_order=curves_generators[2][1],
                         max_iter_algorithm=curves_generators[2][1])
```

Figura B.7: Estudio curva F_{503}

```
# PARAMETROS DEL ESTUDIO

print("Curva:", curves[3])
print("Generador:", curves_generators[3][0])
print("Orden de la curva:", curves_orders[3])
print("Orden del generador:", curves_generators[3][1])

# COMIENZO DEL ESTUDIO
aaa_f1009 = montecarlo_3A(generator_point=curves_generators[3][0],
                          generator_order=curves_generators[3][1],
                          max_iter_algorithm=curves_generators[3][1])
```

Figura B.8: Estudio curva F_{1009}

```
# PARAMETROS DEL ESTUDIO

print("Curva:", curves[4])
print("Generador:", curves_generators[4][0])
print("Orden de la curva:", curves_orders[4])
print("Orden del generador:", curves_generators[4][1])

# COMIENZO DEL ESTUDIO
aaa_f2003 = montecarlo_3A(generator_point=curves_generators[4][0],
                          generator_order=curves_generators[4][1],
                          max_iter_algorithm=curves_generators[4][1])
```

Figura B.9: Estudio curva F_{2003}

```
# Conformacion del data_frame para representacion

x = np.array(range(len(aaa_f31)))

aaa_f31_DF = pd.DataFrame({
    "Iterations" : x.tolist() + x.tolist(),
    "Probability" : aaa_f31 + (1-np.exp(-x * (x - 1) / 2 /
        curves_orders[0])).tolist(),
    "Legend" : ["Empiric" for _ in x] + ["Theoric" for _ in x]
})

aaa_f31_DF.to_csv("aaa_f31_DF.csv", index=False)

# Representacion grafica

aaa_f31_DF = pd.read_csv("aaa_f31_DF.csv")
px.line(aaa_f31_DF, x = "Iterations", y = "Probability",
    color = "Legend").update_layout(showlegend = False)
```

Figura B.10: Representación de F_{31}

```
# Conformacion del data_frame para representacion

x = np.array(range(len(aaa_f31)))

aaa_f101_DF = pd.DataFrame({
    "Iterations" : x.tolist() + x.tolist(),
    "Probability" : aaa_f31 + (1-np.exp(-x * (x - 1) / 2 /
        curves_orders[1])).tolist(),
    "Legend" : ["Empiric" for _ in x] + ["Theoric" for _ in x]
})

aaa_f101_DF.to_csv("aaa_f101_DF.csv", index=False)

# Representacion grafica

aaa_f101_DF = pd.read_csv("aaa_f101_DF.csv")
px.line(aaa_f101_DF, x = "Iterations", y = "Probability",
    color = "Legend").update_layout(showlegend = False)
```

Figura B.11: Representación de F_{101}

```
# Conformacion del data_frame para representacion

x = np.array(range(len(aaa_f503)))

aaa_f503_DF = pd.DataFrame({
    "Iterations" : x.tolist() + x.tolist(),
    "Probability" : aaa_f503 + (1-np.exp(-x * (x - 1) / 2 /
        curves_orders[2])).tolist(),
    "Legend" : ["Empiric" for _ in x] + ["Theoric" for _ in x]
})

aaa_f503_DF.to_csv("aaa_f503_DF.csv", index=False)

# Representacion grafica

aaa_f503_DF = pd.read_csv("aaa_f503_DF.csv")
px.line(aaa_f503_DF, x = "Iterations", y = "Probability",
    color = "Legend").update_layout(showlegend = False)
```

Figura B.12: Representación de F_{503}

```
# Conformacion del data_frame para representacion

x = np.array(range(len(aaa_f1009)))

aaa_f1009_DF = pd.DataFrame({
    "Iterations" : x.tolist() + x.tolist(),
    "Probability" : aaa_f31 + (1-np.exp(-x * (x - 1) / 2 /
        curves_orders[3])).tolist(),
    "Legend" : ["Empiric" for _ in x] + ["Theoric" for _ in x]
})

aaa_f1009_DF.to_csv("aaa_f1009_DF.csv", index=False)

# Representacion grafica

aaa_f1009_DF = pd.read_csv("aaa_f1009_DF.csv")
px.line(aaa_f1009_DF, x = "Iterations", y = "Probability",
    color = "Legend").update_layout(showlegend = False)
```

Figura B.13: Representación de F_{1009}

```
# Conformacion del DF

x = np.array(range(len(aaa_f2003)))

aaa_f2003_DF = pd.DataFrame({
    "Iterations" : x.tolist() + x.tolist(),
    "Probability" : aaa_f1009 + (1-np.exp(-x * (x - 1) / 2 /
        curves_orders[4])).tolist(),
    "Legend" : ["Empiric" for _ in x] + ["Theoric" for _ in x]
})

aaa_f2003_DF.to_csv("aaa_f2003_DF.csv", index=False)

# Representacion grafica

aaa_f2003_DF = pd.read_csv("aaa_f2003_DF.csv")
px.line(aaa_f2003_DF, x = "Iterations", y = "Probability",
    color = "Legend").update_layout(showlegend = False)
```

Figura B.14: Representación de F_{2003}

```
# F_31
density_x = aaa_f31_DF[aaa_f31_DF["Legend"] == "Theoric"]
    ["Probability"].diff().dropna()
density_y = aaa_f31_DF[aaa_f31_DF["Legend"] == "Empiric"]
    ["Probability"].diff().dropna()

print("F_31_KL:", kl_divergence(density_x.to_numpy(),
    density_y.to_numpy()))

# F_101
density_x = aaa_f101_DF[aaa_f101_DF["Legend"] == "Theoric"]
    ["Probability"].diff().dropna()
density_y = aaa_f101_DF[aaa_f101_DF["Legend"] == "Empiric"]
    ["Probability"].diff().dropna()

print("F_101_KL:", kl_divergence(density_x.to_numpy(),
    density_y.to_numpy()))

# F_503
density_x = aaa_f503_DF[aaa_f503_DF["Legend"] == "Theoric"]
    ["Probability"].diff().dropna()
density_y = aaa_f503_DF[aaa_f503_DF["Legend"] == "Empiric"]
    ["Probability"].diff().dropna()

print("F_503_KL:", kl_divergence(density_x.to_numpy(),
    density_y.to_numpy()))

# F_1009
density_x = aaa_f1009_DF[aaa_f1009_DF["Legend"] == "Theoric"]
    ["Probability"].diff().dropna()
density_y = aaa_f1009_DF[aaa_f1009_DF["Legend"] == "Empiric"]
    ["Probability"].diff().dropna()

print("F_1009_KL:", kl_divergence(density_x.to_numpy(),
    density_y.to_numpy()))

# F_2003
density_x = aaa_f2003_DF[aaa_f2003_DF["Legend"] == "Theoric"]
    ["Probability"].diff().dropna()
density_y = aaa_f2003_DF[aaa_f2003_DF["Legend"] == "Empiric"]
    ["Probability"].diff().dropna()

print("F_2003_KL:", kl_divergence(density_x.to_numpy()
    , density_y.to_numpy()))
```

Figura B.15: Cálculo de la KL-divergencia

B.2. Análisis de sistemas reales

```
from EllipticCUrVa.Arithmetic.EllipticCurves.CurvePoint
    import CurvePoint
from EllipticCUrVa.Arithmetic.EllipticCurves.EllipticCurve
    import EllipticCurve


Curve384 = EllipticCurve(
    2 ** 384 - 2 ** 128 - 2 ** 96 + 2 ** 32 - 1,
    -3,
    275801935599597058778490118403890480 9 /
    305690585636156852142870730198868924 1 /
    309860865136260764883745107765439761 2 /
    30575
)

Curve25519 = EllipticCurve(
    2**255 - 19,
    192986815395526992372618308347813179 7 /
    554499744427342733990959733457324163 9 /
    236,
    557517466698189089076452890782571408 1 /
    824110372790101231529440083795672935 8 /
    436
)

Generator384 = CurvePoint(
    Curve384,
   262470350957996892686231567445669818 91 /
   852923491109213387815615900925518854 73 /
   805008902238805397571978665087247673 20 /
   87,
   832571096148902998554675128952010817 92 /
   878530488613155947092059024805031998 84 /
   419224438643760392947333078086511627 87 /
   1)

Generator25519 = CurvePoint(
    Curve25519,
    192986815395526992372618308347813179 7 /
    554499744427342733990959733465218843 5 /
    546,
    431144251710685529207648989359339670 3 /
    937038619820380673076391016620097858 2 /
    548)

GeneratorOrder384 = 39402006196394479212 2 /
790401001436138050797392704654466679469 05 /
279627659399113263569398956308152294913 55 /
4433653942643

GeneratorOrder25519 = 72370055773322622 13 /
973186563042994240857116359379907606001 95 /
0938285454250989
```

Figura B.16: Definición de los parámetros de dominio de la curva P384 y Z25519

```
import plotly.express as px
from math import log2, ceil, sqrt, log, floor
import pandas as pd

# Calculus for the curve P384

aux_y = [x / 100 for x in range(1,100)]
aux_x = [log2(1 + sqrt(1 - 8 * GeneratorOrder384 * log(1-x)) / 2)
         for x in aux_y]
aux_x0 = list(range(floor(aux_x[0])))
aux_y0 = [0 for _ in aux_x0]
aux_xf = list(range(ceil(aux_x[-1]),
              ceil(log2(GeneratorOrder384))))
aux_yf = [1 for _ in aux_xf]

X_Curve384 = aux_x0 + aux_x + aux_xf
Y_Curve384 = aux_y0 + aux_y + aux_yf
Category384 = ["P384" for _ in X_Curve384]

# Calculus for the curve Z25519

aux_y = [x / 100 for x in range(1,100)]
aux_x = [log2(1 + sqrt(1 - 8 * GeneratorOrder25519 * log(1-x)) / 2)
         for x in aux_y]
aux_x0 = list(range(floor(aux_x[0])))
aux_y0 = [0 for _ in aux_x0]
aux_xf = list(range(ceil(aux_x[-1]),
              ceil(log2(GeneratorOrder384))))
aux_yf = [1 for _ in aux_xf]

X_Curve25519 = aux_x0 + aux_x + aux_xf
Y_Curve25519 = aux_y0 + aux_y + aux_yf
Category25519 = ["Z25519" for _ in X_Curve25519]

df = pd.DataFrame(
    {"X" : X_Curve384 + X_Curve25519,
     "Y" : Y_Curve384 + Y_Curve25519,
     "Curve" : Category384 + Category25519})


px.line(df,
        x="X",
        y="Y",
        color="Curve",
        labels={
          "X" : "Pollard-Rho's-Algorithm-Iterations-(log-scaled)",
          "Y" : "Probability"
        })
```

Figura B.17: Comparación de los sistemas inducidos por Z25519 y P384

Anexo C

Demostraciones del capítulo tercero

Proposición C.0.1 (Proposición 4.1.1)**.** *Para cada $n \in \mathbb{N}$ fijo, la función $3A(\cdot,\cdot,\cdot,n)$ induce una* ***relación de orden parcial*** *(véase [Dugundji, 1966]) entre los distintos elementos de EllipBase.*

Demostración. Sean $\mathcal{E}_1, \mathcal{E}_2 \in EllipBase$ sin pérdida de generalidad y fíjese $n \in \mathbb{N}$. Se define la siguiente relación:

$$\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_2 \iff \begin{cases} 3A(\mathcal{E}_1, n) > 3A(\mathcal{E}_2, n) \\ \mathcal{E}_1 = \mathcal{E}_2 \end{cases}$$

Se probará que efectivamente es una relación de orden parcial:

- **Propiedad Reflexiva:** Por caracterización todo par de elementos sobre *EllipBase* se encuentra relacionado consigo mismo.
- **Propiedad Antisimétrica:** Se toman $\mathcal{E}_1, \mathcal{E}_2 \in EllipBase$ tales que $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_2$ y $\mathcal{E}_2 \mathcal{R}_n \mathcal{E}_1$ y se pretende ver que $\mathcal{E}_1 = \mathcal{E}_2$.

  Por caracterización, como $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_2$ entonces se debe cumplir una de dos cosas, o $3A(\mathcal{E}_1, n) > 3A(\mathcal{E}_2, n)$ o $\mathcal{E}_1 = \mathcal{E}_2$. Si $3A(\mathcal{E}_1, n) > 3A(\mathcal{E}_2, n)$ entonces, por caracterización, $\mathcal{E}_2$ no se encontraría relacionado con $\mathcal{E}_1$ y como por hipótesis esto si que se tiene sólo queda que $\mathcal{E}_2 = \mathcal{E}_1$.
- **Propiedad Transitiva**: Se toman $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3 \in EllipBase$ tales que $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_2$ y $\mathcal{E}_2 \mathcal{R}_n \mathcal{E}_3$ y se pretende probar $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_3$. Claramente, si $\mathcal{E}_1 = \mathcal{E}_2$ entonces $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_3$. Si $3A(\mathcal{E}_1, n) > 3A(\mathcal{E}_2, n)$ y $\mathcal{E}_2 = \mathcal{E}_3$ también se sigue directamente que $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_3$ puesto que $3A(\mathcal{E}_1, n) > 3A(\mathcal{E}_2, n) = 3A(\mathcal{E}_3, n)$. Finalmente, si $3A(\mathcal{E}_1, n) > 3A(\mathcal{E}_2, n)$ y $3A(\mathcal{E}_2, n) > 3A(\mathcal{E}_3, n)$ es claro que $3A(\mathcal{E}_1, n) > 3A(\mathcal{E}_2, n) > 3A(\mathcal{E}_3, n)$ ergo $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_3$.

Por último, como elementos distintos con mismos índices $3A$ no son comparables a través de la relación $\mathcal{R}_n$, se prueba que esta es una relación de orden parcial sobre los elementos de *EllipBase*. □

Proposición C.0.2 (Proposición 4.1.2)**.** *Fijados un cuerpo K, una curva $\mathbb{E}$ sobre K y un punto $P \in \mathbb{E}$, la función $3A(K, \mathbb{E}, P, \cdot)$ es monótona creciente.*

Demostración. La demostración sigue la definición de distribución de probabilidad (véase [Billingsley, 1995]). □

Corolario C.0.1 (Corolario 4.1.1)**.** *Dados $\mathcal{E}_1, \mathcal{E}_2 \in EllipBase$ y dos enteros positvos $n > m$. Si $3A(\mathcal{E}_1, n) = 3A(\mathcal{E}_2, m)$ entonces $\mathcal{E}_1 \mathcal{R}_n \mathcal{E}_2$ o no se pueden comparar a través del índice $3A$ para n iteraciones.*

Demostración. La demostración de este resultado es directa tras aplicar la Proposición 4.1.1 y la Proposición 4.1.2. □

Anexo D

Desarrollo de las cuentas en el experimento dos

En este apéndice se muestra el desarrollo matemático para resolver la Ecuación 5.2:

$$\int_0^\infty x \cdot \frac{\partial}{\partial x}\left(3A(Z22519,x) - 3A(P384,x)\right)dx =$$

$$= \left| x \cdot \Big(3A(Z22519,x) - 3A(P384,x)\Big)\right|_0^\infty - \int_0^\infty 3A(Z22519,x) - 3A(P384,x)dx =$$

$$= \underbrace{\left| \frac{x}{\left(\exp\left(\frac{-x(x-1)}{2\cdot ord(Z25519)}\right) - \exp\left(\frac{-x(x-1)}{2\cdot ord(P384)}\right)\right)^{-1}}\right|_0^\infty}_{\text{Parte 1}} \underbrace{- \int_0^\infty 3A(Z25519,x)dx}_{\text{Parte 2}} + \underbrace{\int_0^\infty 3A(P384,x)dx}_{\text{Parte 3}} =$$

Se procede a resolver cada sumando en la expresión anterior por separado:

Parte 1: Para $x = 0$, la expresión $x \cdot \Big(3A(Z22519,x) - 3A(P384,x)\Big)$ es claramente cero. Por otro lado, cuando $x \to \infty$ la expresión se convierte en una indeterminación del tipo $0 \cdot \infty$ que puede ser resuelta aplicando la **Regla de L'Hôpital** (véase [Spivak, 2006]):

$$\lim_{x\to\infty} \frac{x}{\left(\exp\left(\frac{-x(x-1)}{2\cdot ord(Z25519)}\right) - \exp\left(\frac{-x(x-1)}{2\cdot ord(P384)}\right)\right)^{-1}} =$$

$$= \lim_{x\to\infty} \frac{\left(\exp\left(\frac{-x(x-1)}{2ord(Z25519)}\right) - \exp\left(\frac{-x(x-1)}{2ord(P384)}\right)\right)}{\left(\frac{2x-1}{2}\right)\left(\frac{1}{ord(Z25519)} - \frac{1}{ord(P398)}\right)} = 0$$

Concluyendo así que el termino primero de la expresión superior vale cero.

Parte 2: Las integrales en los términos dos y tres tienen truco, pues se pueden reinterpretar como la función de Gauss multiplicada por un factor constante:

$$\int_0^\infty e^{\frac{-x(x-1)}{2\cdot ord(Z25519)}}dx = \int_0^\infty e^{\frac{-(x-1/2)^2}{2\cdot ord(Z25519)}} \cdot e^{\frac{1}{8\cdot ord(Z25519)}}dx =$$

$$= \sqrt{2\pi \cdot ord(Z25519)} \cdot e^{\frac{1}{8 \cdot ord(Z25519)}} \int_0^{\infty} \frac{1}{\sqrt{2\pi \cdot ord(Z25519)}} \cdot e^{-\frac{(x-1/2)^2}{2 \cdot ord(Z25519)}} dx =$$

$$= (1 - F_N(0; 1/2, ord(Z25519))) \cdot \sqrt{2\pi \cdot ord(Z25519)} \cdot e^{\frac{1}{8 \cdot ord(Z25519)}} \approx$$

$$\sqrt{\frac{\pi \cdot ord(P25519)}{2}}$$

Donde $F_N(\cdot; \mu, \sigma^2)$ representa la función de distribución de una variable normal con media μ y varianza σ^2.

Parte 3: Aplicando un razonamiento simétrico al aplicado en el término dos se obtiene que el valor del tercer término es $\sqrt{\frac{\pi \cdot ord(P384)}{2}}$.

Juntando todos los los términos, se llega a un resultado aproximado de $\sqrt{\frac{\pi \cdot ord(P384)}{2}} - \sqrt{\frac{\pi \cdot ord(Z25519)}{2}} \approx 2^{192,33}$.

Anexo E

Facturas, nóminas y contratos

simyo

Eduardo Quintana Adeva
[redacted]

501042500494376

Nº CLIENTE: [redacted]

Nombre titular:	EDUARDO QUINTANA ADEVA
NIF, NIE, CIF:	[redacted]
Modalidad pago:	CONTRATO
Periodo de consumo:	20/03/2025 - 19/04/2025
Fecha de emisión:	25/04/2025
Método de pago:	[redacted]
Número de cuenta:	[redacted]
Fecha de vencimiento:	28/04/2025
Importe:	27,99€

RESUMEN DE MI FACTURA Nº 5010425-00494376

Tarifa base fibra
FIBRA 600 estándar
IUA J40064AAA508

Cuota Mensual	23,14€
Base imponible	23,13€
IVA 21%	4,86€
Importe total	27,99€
IMPORTE TOTAL A PAGAR	**27,99€**

* En 'Mis consumos sin bonos' se reflejan los consumos de llamadas, datos, mensajes, roaming, que no estén incluidos dentro de tus bonos o en el caso de que no tengas bonos contratados los consumos tarificados al precio de tu tarifa

Puedes consultar los precios aplicados en www.simyo.es/tarifas-movil.html

Si no quieres recibir tu factura en papel te recomendamos que puedas recibirla en PDF seleccionando esta opción en tu área personal de simyo.es

Oficina de Atención al Usuario de Telecomunicaciones de la Secretaría de Estado para la Sociedad de la Información y la Agenda Digital. 911814045 (precio llamada a número nacional según tarifa). 901336699 (precio 18,15 cts.est. llamada, 36,3 cts./min). http://www.usuariosteleco.gob.es.

Si bien la información de consumos se muestra con 4 decimales, el cálculo de la base imponible se realiza utilizando el valor con 6 decimales. de ahí que pueda haber discrepancias entre la suma aquí mostrada y la indicada en la página 2.

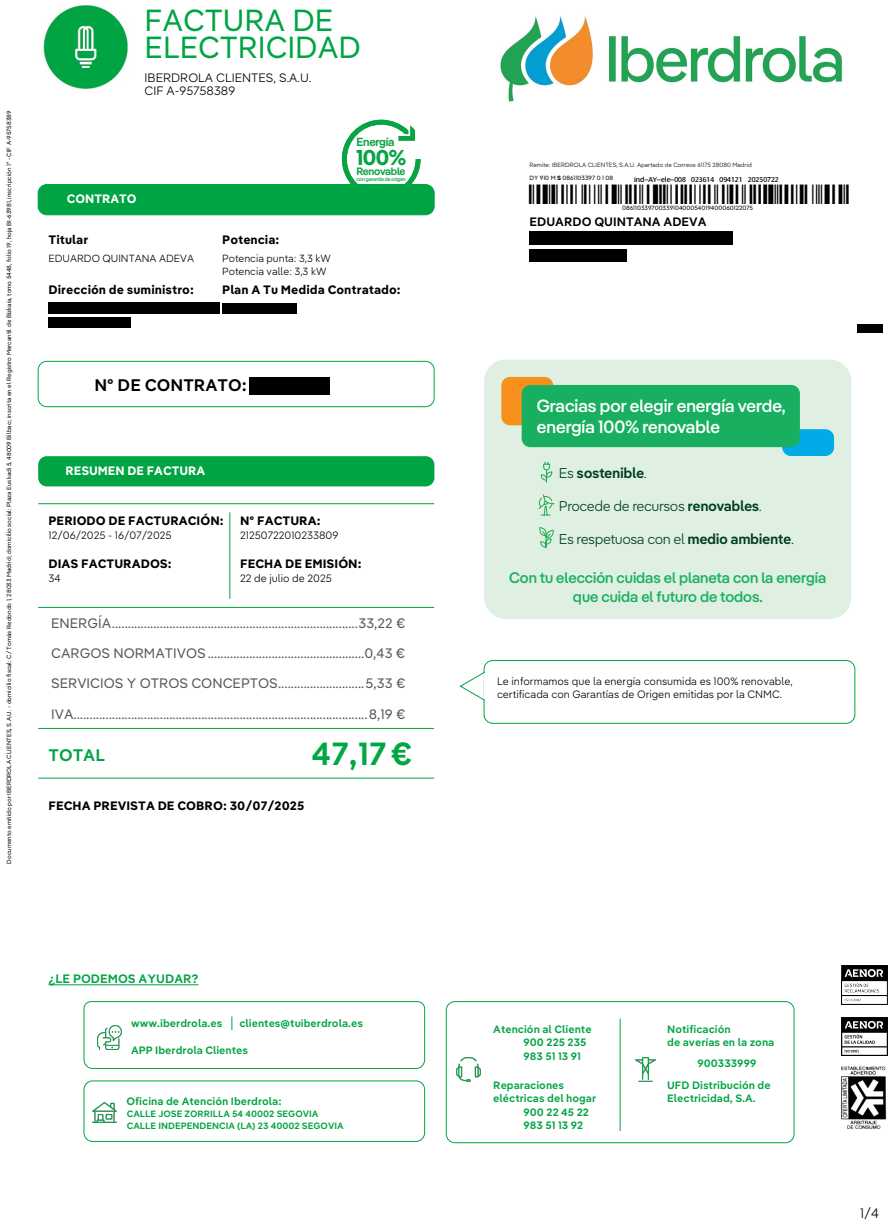
¡PSSST! TODAVÍA PUEDES PAGAR MENOS
INVITA A TUS AMIGOS Y DISFRUTA DE MÁS VENTAJAS
infórmate en simyo.es o en tu APP

Más info: 121 gratis desde tu móvil simyo.

simyo.es

Orange España Virtual S.L.U., con sede social en Paseo del Club Deportivo, 1, Parque Empresarial la Finca, Edificio 8, 28223 Pozuelo de Alarcón (Madrid). Inscrita en el registro mercantil de madrid tomo 24.069 folio 19 sección 8 hoja M-432374 inscripción 1 CIF: B-85057974

Figura E.1: Factura de Internet



FACTURA DE ELECTRICIDAD

IBERDROLA CLIENTES, S.A.U.
CIF A-95758389

Iberdrola

Energía 100% Renovable con garantía de origen

Remite: IBERDROLA CLIENTES, S.A.U. Apartado de Correos 6075 28080 Madrid

EDUARDO QUINTANA ADEVA
[redacted]

CONTRATO

Titular
EDUARDO QUINTANA ADEVA

Potencia:
Potencia punta: 3,3 kW
Potencia valle: 3,3 kW

Dirección de suministro:
[redacted]

Plan A Tu Medida Contratado:
[redacted]

N° DE CONTRATO: [redacted]

Gracias por elegir energía verde, energía 100% renovable

- Es **sostenible**.
- Procede de recursos **renovables**.
- Es respetuosa con el **medio ambiente**.

Con tu elección cuidas el planeta con la energía que cuida el futuro de todos.

RESUMEN DE FACTURA

PERIODO DE FACTURACIÓN:
12/06/2025 - 16/07/2025

N° FACTURA:
2125072201023809

DIAS FACTURADOS:
34

FECHA DE EMISIÓN:
22 de julio de 2025

ENERGÍA	33,22 €
CARGOS NORMATIVOS	0,43 €
SERVICIOS Y OTROS CONCEPTOS	5,33 €
IVA	8,19 €
TOTAL	**47,17 €**

Le informamos que la energía consumida es 100% renovable, certificada con Garantías de Origen emitidas por la CNMC.

FECHA PREVISTA DE COBRO: 30/07/2025

¿LE PODEMOS AYUDAR?

www.iberdrola.es | clientes@tuiberdrola.es
APP Iberdrola Clientes

Oficina de Atención Iberdrola:
CALLE JOSE ZORRILLA 54 40002 SEGOVIA
CALLE INDEPENDENCIA (LA) 23 40002 SEGOVIA

Atención al Cliente
900 225 235
983 51 13 91

Reparaciones eléctricas del hogar
900 22 45 22
983 51 13 92

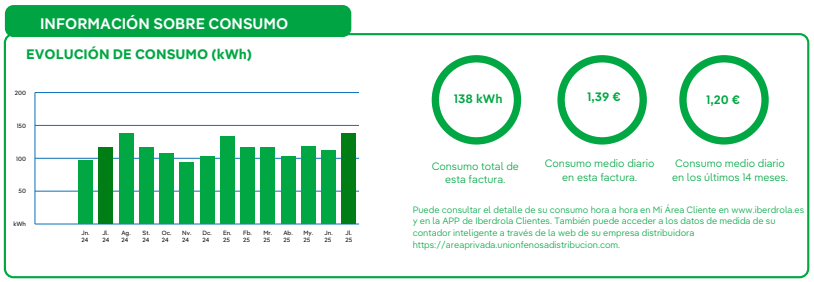
Notificación de averías en la zona
900333999

UFD Distribución de Electricidad, S.A.

AENOR

1/4

Figura E.2: Factura del consumo eléctrico (Página 1)



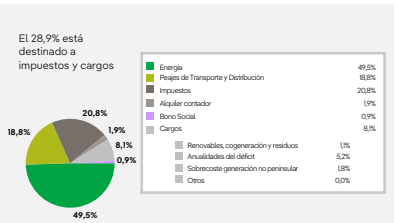
DETALLE DE FACTURA

ENERGÍA		
Potencia facturada	Punta 3,3 kW x 34 días x 0,091996 €/kW día	10,32 €
	Valle 3,3 kW x 34 días x 0,002757 €/kW día	0,31 €
Total importe potencia hasta 16/07/2025		**10,63 €**
Energía consumida	138 kWh x 0,151803 €/kWh	20,95 €
CARGOS NORMATIVOS		
Financiación bono social fijo	34 días x 0,012742 €/día	0,43 €
Impuesto sobre electricidad	5,11269632 % s/32,01 €	1,64 €
TOTAL ENERGÍA		33,65 €
SERVICIOS Y OTROS CONCEPTOS		
Alquiler equipos medida	34 días x 0,02663014 €/día	0,91 €
Protección Electrodomésticos	1,12 meses x 3,95 €/mes	4,42 €
TOTAL SERVICIOS Y OTROS CONCEPTOS		5,33 €
IMPORTE TOTAL		**38,98 €**
IVA	**21 % s/38,98 €**	**8,19 €**
TOTAL IMPORTE FACTURA		47,17 €

INFORMACION ADICIONAL

NIF titular del contrato:	[redacted]
Nº contador:	[redacted]
Peaje de acceso a la red (ATR):	2.0TD
Mercado:	Libre
Segmento tarifario:	I
Precios de peajes de acceso:	B.O.E. del 16/12/2024
Fecha final del contrato:	02/12/2025
Permanencia:	No
Empresa distribuidora:	UFD Distribución de Electricidad, S.A.
Número de contrato de acceso:	[redacted]
Identificación punto de suministro (CUPS):	[redacted]
Forma de pago:	DOMICILIACION BANCARIA
IBAN:	[redacted]
BIC:	[redacted]
Código de mandato:	[redacted]

**** (Ocultos para su seguridad)



Las lecturas desagregadas según la tarifa de acceso, tomadas el 16/07/2025 son: punta: 6.860 kWh; llano: 5.724 kWh; valle 3.886 kWh, siendo estas lecturas reales. Sus consumos desagregados han sido punta: 33 kWh; llano: 32 kWh; valle 73 kWh.

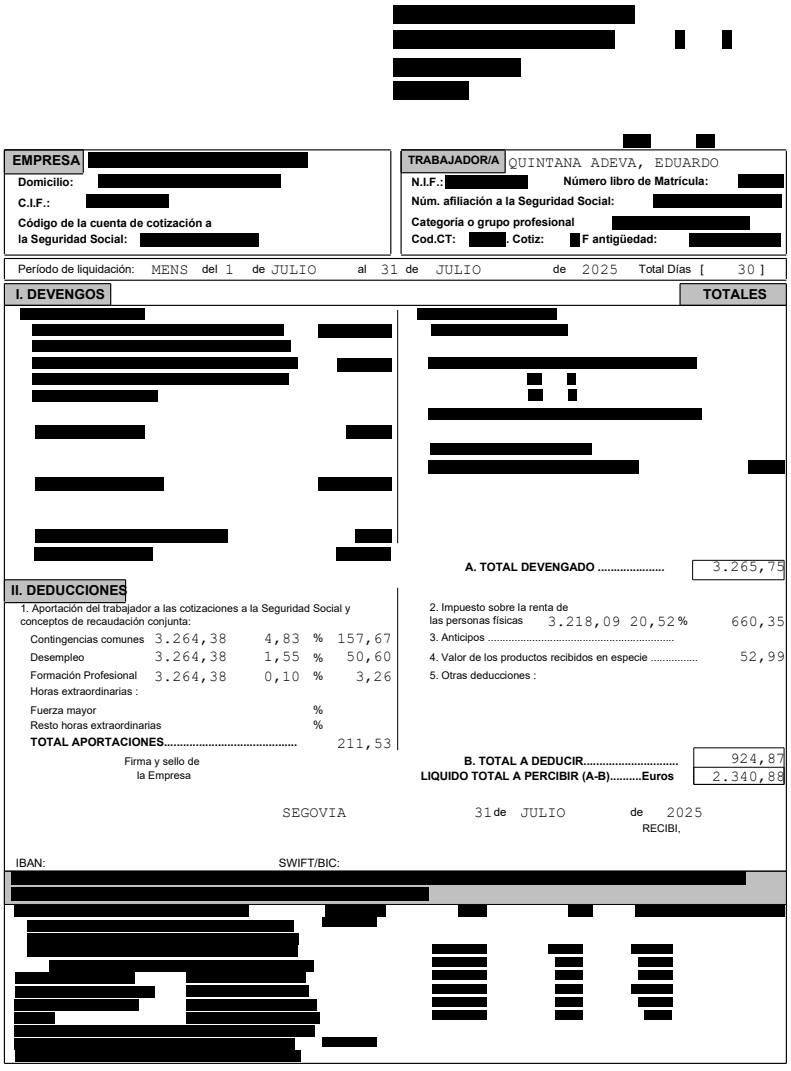
Las potencias máximas demandadas en el último año han sido 3,57 kW en P1 (punta) y 2,77 kW en P2 (valle).

El importe de los peajes de acceso a redes en esta factura, sin impuestos, es de 13,58 €, desglosado en: Potencia: 8,5 € , Energía: 4,17 €, Alquiler de contador: 0,91 €. Impuestos aplicables: Impuesto Eléctrico e IVA u otros impuestos indirectos. Estos importes están englobados en el importe total de la factura (no representan coste adicional).

Sin datos sobre el consumo medio de su código postal.

Para reclamaciones relacionadas con el contrato de suministro o la facturación puede dirigirse a cualquiera de los canales de atención indicados en esta factura, o en el Apartado de Correos 61090, 28080 de Madrid. Si su reclamación no ha sido resuelta o ha superado el plazo de 30 días, puede dirigirse a la Junta Arbitral de Consumo Autonómica de Castilla y León (teléfono de contacto: 983412595). Además, puede acudir a los órganos competentes en materia de Consumo y/o Energía de dicha Comunidad Autónoma.

Figura E.3: Factura del consumo eléctrico (Página 2)



EMPRESA [redacted]
Domicilio: [redacted]
C.I.F.: [redacted]
Código de la cuenta de cotización a la Seguridad Social: [redacted]

TRABAJADOR/A QUINTANA ADEVA, EDUARDO
N.I.F.: [redacted] Número libro de Matrícula: [redacted]
Núm. afiliación a la Seguridad Social: [redacted]
Categoría o grupo profesional [redacted]
Cod.CT: [redacted] Cotiz: [redacted] F antigüedad: [redacted]

Período de liquidación: MENS del 1 de JULIO al 31 de JULIO de 2025 Total Días [30]

I. DEVENGOS — TOTALES

[redacted]

A. TOTAL DEVENGADO 3.265,75

II. DEDUCCIONES

1. Aportación del trabajador a las cotizaciones a la Seguridad Social y conceptos de recaudación conjunta:

Contingencias comunes	3.264,38	4,83	%	157,67
Desempleo	3.264,38	1,55	%	50,60
Formación Profesional	3.264,38	0,10	%	3,26
Horas extraordinarias :				
Fuerza mayor			%	
Resto horas extraordinarias			%	
TOTAL APORTACIONES.....				211,53

2. Impuesto sobre la renta de las personas físicas 3.218,09 20,52 % 660,35
3. Anticipos .....
4. Valor de los productos recibidos en especie 52,99
5. Otras deducciones :

B. TOTAL A DEDUCIR..... 924,87
LIQUIDO TOTAL A PERCIBIR (A-B).....Euros 2.340,88

Firma y sello de la Empresa

SEGOVIA 31 de JULIO de 2025
RECIBI,

IBAN: SWIFT/BIC:

[redacted]

Figura E.4: Nómina del trabajador

Bibliografía

[GAP, 2024] (2024). *GAP – Groups, Algorithms, and Programming, Version 4.14.0*. The GAP Group.

[Apostol, 2020] Apostol, T. M. (2020). *Introducción a la teoría analítica de números*. Reverté.

[Beck, 2022] Beck, K. (2022). *Test driven development: By example*. Addison-Wesley Professional.

[Billingsley, 1995] Billingsley, P. (1995). *Probability and Measure*. John Wiley & Sons, New York, 3rd edition.

[Bos et al., 2014] Bos, J. W., Dudeanu, A., and Jetchev, D. (2014). Collision bounds for the additive pollard rho algorithm for solving discrete logarithms. *Journal of Mathematical Cryptology*, 8(1):71–92.

[Chen et al., 2019] Chen, L., Moody, D., Regenscheid, A., and Randall, K. (2019). Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters. *NIST SP 800-186*.

[Dugundji, 1966] Dugundji, J. (1966). *Topology*. Allyn and Bacon series in advanced mathematics. Allyn and Bacon.

[Foundation, 2025] Foundation, P. S. (2025). The python standard library.

[Fulton, 2008] Fulton, W. (2008). Algebraic curves. *An Introduction to Algebraic Geom*, 54.

[Gentle, 1998] Gentle, J. (1998). *Random Number Generation and Monte Carlo Methods*. Statistics and computing. Springer.

[Hildebrand, 2005] Hildebrand, M. (2005). A survey of results on random random walks on finite groups. *Probability Surveys*.

[Judson, 2020] Judson, T. W. (2020). *Abstract algebra: theory and applications*.

[Karn et al., 1995] Karn, P. R., Simpson, W. A., and Metzger, P. E. (1995). The ESP Triple DES Transform. RFC 1851.

[Kelly et al., 2014] Kelly, J., Sadeghieh, T., and Adeli, K. (2014). Peer review in scientific publications: benefits, critiques, & a survival guide. *Ejifcc*, 25(3):227.

[Knuth, 2014] Knuth, D. E. (2014). *The Art of Computer Programming: Seminumerical Algorithms, Volume 2*. Addison-Wesley Professional.

[Koblitz, 1987] Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209.

[Koskela, 2007] Koskela, L. (2007). *Test driven: practical tdd and acceptance tdd for java developers*. Simon and Schuster.

[Kuhn and Struik, 2001] Kuhn, F. and Struik, R. (2001). Random walks revisited: Extensions of pollard's rho algorithm for computing multiple discrete logarithms. pages 212–229.

[Larman, 2004] Larman, C. (2004). *Agile and iterative development: a manager's guide*. Addison-Wesley Professional.

[Lenstra Jr, 1987] Lenstra Jr, H. W. (1987). Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673.

[Lynn, 2024] Lynn, B. (2024). Applied cryptography group — standford.

[Martín, 2024] Martín, J. I. F. (2024). *Matemática discreta: Lógica y estructuras discretas con Python*. Ediciones Universidad de Valladolid.

[Menezes et al., 2018] Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (2018). *Handbook of applied cryptography*. CRC press.

[NIST, 2001] NIST, U. (2001). Descriptions of sha-256, sha-384 and sha-512.

[Paar and Pelzl, 2010] Paar, C. and Pelzl, J. (2010). *Understanding cryptography*, volume 1. Springer.

[Popiński, 2022] Popiński, W. (2022). Bernstein polynomial distribution estimators and the dvoretzky–kiefer–wolfowitz inequality. *Applicationes Mathematicae*, 49(2):175–184.

[Quintana, 2022] Quintana, E. (2022). El teorema de cayley-bacharach y matemática discreta. *Trabajos Fin de Grado UVa*.

[Silverman, 2006] Silverman, J. H. (2006). An introduction to the theory of elliptic curves. *Brown University. June*, 19:2006.

[Silverman, 2009] Silverman, J. H. (2009). *The arithmetic of elliptic curves*, volume 106. Springer.

[Smith, 2013] Smith, J. (2013). *Introduction to Algebraic Geometry*.

[Spivak, 2006] Spivak, M. (2006). *Calculus*. Cambridge University Press.

[Tena, 2014] Tena, J. G. (2014). 25 años de criptografía con curvas elípticas.

[Tributaria, 2022] Tributaria, A. (2022). Orden hfp/1172/2022, de 29 de noviembre (boe de 1 de diciembre). `https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2023/c09-rendimientos-actividades-economicas-eo-fores/determinacion-rendimiento-neto/fase-2-determinacion-rendimiento-neto-minorado/amortizacion-inmovilizado-material-inmaterial/tabla-amortizacion.html`.

[Wiles, 1995] Wiles, A. (1995). Modular elliptic curves and fermat's last theorem. *Annals of mathematics*, 141(3):443–551.

[Yan, 2013] Yan, S. Y. (2013). *Computational number theory and modern cryptography*. John Wiley & Sons.