Curvas de Edwards para firma digital: EdDSA

José Cristóbal Sendín Martín

Curvas de Edwards para firma digital: EdDSA

José Cristóbal Sendín Martín

Memoria presentada como parte de los requisitos para la obtención de los títulos de Grado en Ingeniería Informática - Tecnologías Infomáticas y Grado en Matemáticas por la Universidad de Sevilla.

Tutorizada por

Don Manuel Jesús Soto Prieto
Departamento de Álgebra
Facultad de Matemáticas

Doña María del Mar Martínez Ballesteros
Departamento de Lenguajes y Sistemas Informáticos
Escuela Técnica Superior de Ingeniería Informática

Índice general

Índice de figuras

Abstract

EdDSA is the digital signature used in the Signal protocol. The development of this protocol was a milestone for instant messaging applications. On top of that, digital signatures are an undeniably useful cryptographic primitive nowadays, which is why we will focus on the protocol's signature.

To understand it, we must not only have a clear comprehension of the basic concepts of cryptography and digital signatures. EdDSA is based on Edwards Curves. The research on these curves enabled the development of the protocol, which is why we will study them extensively, focusing on the three papers that introduced them. Additionally, we will cover the basic concepts of elliptic curves and the Signal protocol itself.

To conclude, we will implement the signature to ensure our comprehension of the algorithm and to analyze the development process.

Resumen

EdDSA es la □rma digital usada en el protocolo Signal. El desarrollo de este protocolo supuso un hito para las aplicaciones de mensajería instántanea. Además, la □rma digital es una primitiva criptógra□ca de innegable utilidad en la actualidad por lo que centraremos este trabajo en la □rma del protocolo.

Para comprender la misma, no solo debemos tener claro los conceptos básicos de criptografía y de □rma digital. EdDSA está basada en Curvas de Edwards. La investigación respecto a estas curvas permitió el desarrollo del protocolo, por lo que las estudiaremos profundamente, centrándonos en las tres publicaciones que las introdujeron. Asimismo, veremos los conceptos básicos de curvas elípticas y del propio protocolo Signal.

Para □nalizar, implementaremos la □rma para cerciorarnos de la comprensión del algoritmo y haremos un análisis del proceso de desarrollo.

1 Introducción

La □rma digital desempeña un papel fundamental en la criptografía moderna. En nuestro mundo cada vez más conectado, la comunicación y transmisión de datos rara vez es personal debido a la rapidez y comodidad que ofrecen las vías electrónicas. Por ello, veri□car que la información es íntegra y que corresponde a nuestro interlocutor se ha convertido en un requisito esencial.

Sumado a esto, destaca la utilidad de la □rma digital para el desarrollo de aplicaciones de mensajería instantánea. Cada día millones de personas usan aplicaciones de este tipo para comunicarse. La □rma digital es un elemento subyacente a estas aplicaciones, por tanto, es indudable que investigar métodos e□cientes y seguros de □rma es un hecho de interés y crucial.

La inspiración inicial de este trabajo ha sido el protocolo Signal que discutiremos más adelante. Este protocolo de intercambio de mensajes es el usado por la app homónima. Supuso una auténtica revolución al elevar la exigencia en seguridad y con□dencialidad. Tanto es así, que aplicaciones como Whatsapp, con una base de usuarios del orden de mil millones, optaron por incluirlo y así mejorar la calidad de su servicio.

Asimismo, el protocolo Signal y el campo en el que se ubica, la criptografía, encajan perfectamente en un Trabajo Fin de Grado del Doble Grado en Ingeniería Informática - Tecnologías Informáticas y en Matemáticas. El protocolo consta de tres herramientas fundamentales. La primera es X3DH que se utiliza para el intercambio de claves. La segunda es el Double Ratchet que resuelve el cifrado de los mensajes. La tercera y última es la □rma que se usa a lo largo del protocolo, EdDSA. Esta última se basa en un tipo especial de curvas, las Curvas de Edwards, en lugar de de en las curvas elípticas.

Este protocolo jamás hubiese sido posible sin los trabajos realizados por Bernstein et al. en varias publicaciones, destacando tres: [10], [7] y [8]. En ellas, introdujeron

las curvas de Edwards con su ley de suma, una variación de las mismas y la propia rma que utilizaría el protocolo.

De este modo, el presente trabajo se basa en un estudio en profundidad de estas tres publicaciones. Hemos intentado que sea lo más autocontenido posible. A su vez, podríamos decir que consta de tres partes:

La primera sería una presentación matemática formal de los conceptos que permiten que la rma funcione. Seguidamente, se tendría una parte de interés mixto: la explicación a alto nivel de la rma digital y de EdDSA, nuestro caso de rma concreto. Por último, la parte de interés informático sería la implementación del algoritmo, que además demostraría la comprensión adecuada de los conceptos anteriormente introducidos.

En el Capítulo 2 introduciremos la rma digital para entender el contexto de trabajo. Continuaríamos en el Capítulo 3 con una breve introducción de las curvas elípticas y de un algoritmo de rma asociado a ellas. Posteriormente, estableceremos una equivalencia entre las últimas y las curvas de Edwards, por lo que es necesario entenderlas bien.

En el Capítulo 4 trabajaremos sobre los contenidos más relevantes de las dos primeras publicaciones en las que nos centramos. En la primera sección del mismo, introduciremos las Curvas de Edwards y su ley de suma. Además, estableceremos una equivalencia birracional de las mismas con las curvas elípticas. Como veremos, las curvas de Edwards son más rápidas en el cálculo que las elípticas, por lo que al tener esta equivalencia, podemos aprovechar todo el potencial de la criptografía de curvas elípticas pero de forma más veloz. En la segunda sección de este capítulo principal, tratamos las curvas de Edwards torcidas. Estas suponen una mejora en ciertos casos sobre las curvas de Edwards normales, por lo que los resultados que conseguiremos serán aún mejores.

Posteriormente, trataremos el objeto que da título al trabajo: La rma EdDSA. En el Capítulo 5 la desgranamos, describiendo por partes los procesos de generación de claves, rma y veri cación. Aparte de la descripción formal, introduciremos diagramas y pseudocódigo para su mayor comprensión.

En el siguiente Capítulo, el 6, describiremos la implementación en lenguaje de programación Python del algoritmo de rma. La implementación original del protocolo Signal está realizada en el lenguaje de programación en Rust y existen implementaciones de la rma en lenguaje C, por lo que por motivos didácticos hemos considerado

esta implementación como adecuada. Además, describiremos los problemas encontrados al pasar de la descripción formal a la implementación real.

Más adelante, cerramos el contenido del trabajo con los Capítulos 7 y 8. En el primero, explicamos a nivel formal el protocolo que ha servido de inspiración. No solo por su propio interés, sino para ilustrar la utilidad de la □rma explicada en el contexto del protocolo. En el siguiente capítulo haremos el análisis temporal y de costes de todo el trabajo.

Por último, hacemos una breve conclusión en el Capítulo 9. En ella, discutiremos los objetivos aquí planteados y cómo el desarrollo de este Trabajo Fin de Grado ha sido fundamental y el punto □nal de estos cinco años de Doble Grado.

2 Firma Digital

Dado que el objetivo fundamental de este trabajo es el estudio de la firma EdDSA junto con las herramientas necesarias para construirla, comenzaremos recordando los conceptos básicos de criptografía que se usarán a lo largo del documento (Sección 2.1) y describiendo brevemente en qué consiste una firma digital (Sección 2.2). Ejemplificaremos el concepto introduciendo en la Sección 2.3 la firma DSA, que será fundamental para el desarrollo de ECDSA (Sección 3.4 del Capítulo 3). Por último, veremos brevemente los conceptos de Certificado Digital e Infraestructura de Clave Pública. No nos adentraremos mucho en ellos pues escapan del objetivo del trabajo. No obstante, sí que es interesante describirlos para contextualizar una de las utilidades de la firma digital (Sección 2.4).

2.1 Nociones básicas

Tomamos [4] como referencia principal de la presente sección.

2.1.1 Definiciones

Definimos informalmente las siguientes nociones comunes en criptografía:

- **Confidencialidad**: consiste en aplicar un cifrado a un mensaje, de modo que solo los agentes que participan en la comunicación (Alice y Bob) puedan descifrarlo.
- **Integridad**: tenemos datos íntegros si podemos garantizar que no han sido modificados.

- **Autenticidad**: consiste en asegurar la identidad del agente o el origen de los datos.
- **No repudio**: garantiza que un agente no pueda negar a posteriori haber realizado cierta acción.
- **Ataque** *Man in the middle* (MITM): Suponemos comunicación entre Alice y Bob a través de cierto canal usando criptografía de clave pública. Tenemos un ataque MITM si un agente externo (Mallory) acuerda una clave con Alice haciéndose pasar por Bob y viceversa. De esta forma, Alice y Bob pensarían que su conversación es legítima, sin embargo, esta está siendo escuchada, y posiblemente modi□cada, por Mallory.

2.1.2 Funciones Hash

Una vez recordado a que aluden los términos que iremos utilizando, necesitamos repasar las funciones hash. Nos limitaremos a verlas conceptualmente, sin entrar en profundidad en su implementación.

Intuitivamente, decimos que f es un función trampilla si $O.f/$ es polinomial y $O.f^{*1}/$ es al menos exponencial.

Definición 2.1. *Dado* A *un alfabeto, i.e., un conjunto nito de símbolos utilizado en sistemas criptográ cos, de nimos una función hash como una función trampilla* $H : A^{Ø} \to A^{t}$ *que cumple las siguientes propiedades criptográ cas:*

1. *Resistencia a preimagen: Dado* $s \in A^{t}$*, es computacionalmente infactible calcular* $m \in A^{Ø}$ *tal que* $H.m/ = s$*.*
2. *Resistencia a segunda preimagen: Dado* $s = H.m/$*, es computacionalmente infactible encontrar* $m' \neq m$ *tal que* $H.m'/ = s$*.*
3. *Resistencia a colisiones: Es computacionalmente infactible encontrar* m *y* m' *distintos tales que* $H.m/ = H.m'/$*.*

Además, se le pide:

4. *Determinismo: Dada una entrada, la salida de* H *es predecible.*
5. *Efecto avalancha: Si* m' *se obtiene cambiando uno o pocos bits de* m*, entonces* $H.m'/$ *di ere en muchos bits (aproximadamente la mitad) de* $H.m/$*.*

2.2 Firma digital

La aparición de ataques MITM supuso un problema importante. Efectivamente, ¿cómo saber si un mensaje ha sido modi□cado, o más aún, cómo saber si nuestro interlocutor es verdaderamente quien a□rma ser? Entre las diversas soluciones al primer problema, aparece la □rma digital. Respecto al segundo, veremos que solo hay que añadir un capa de complejidad a la solución del primero.

A muy alto nivel, la idea es sencilla. Si Alice quiere enviar un mensaje a Bob debe calcular el hash del mismo, cifrarlo y enviarlo junto con el mensaje. Así, cuando Bob descifre, podrá calcular el hash del mensaje recibido usando la función hash acordada previamente y comparar los resultados. Si coinciden, el mensaje no habría sido modi□cado. De esta forma, se garantizaría la integridad de los datos. La con□dencialidad no es objeto de interés en □rma digital, ya que de manera análoga a las □rmas convencionales, no queremos cifrar los mensajes sino que queremos que cualquiera compruebe que el creador del documento aprueba la información presente en el mismo.

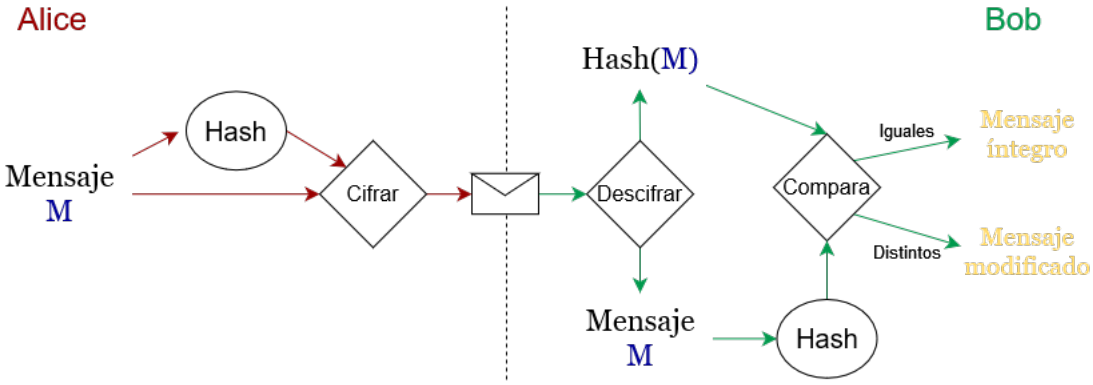

Figura 2.1: Esquema general de intercambio usando una □rma digital

A continuación introducimos un modelo de □rma digital bien conocido.

2.3 Firma DSA

La □rma DSA (Digital Signature Algorithm) es el estándar de □rma digital en Estados Unidos (también se la conoce como □rma DSS, Digital Signature Standard). Fue introducida por el NIST (National Institute of Standards and Technology) en 1991 y

es un sistema de firma asimétrico parecido a los descritos por Schnorr y por ElGamal. Veamos su funcionamiento.

El enfoque que usaremos será recurrente a lo largo del trabajo. Primeramente introducimos los parámetros del algoritmo en la Sección 2.3.1. Le sigue la Sección 2.3.2 donde se explica como obtener las claves. Finalizamos con la firma en la Sección 2.3.3 y la verificación en la Sección 2.3.4.

2.3.1 Parámetros

Los parámetros son los siguientes:

- H función hash.
- primo q del orden de 2^{160}.
- primo p del orden de 2^{1024} y que verifique $p \equiv 1 \pmod{q}$. Observamos que el grupo $F_p^{\bullet}$ tiene un único subgrupo cíclico de orden q.
- entero aleatorio g. Calculamos $G = g^{(p-1)/q} \pmod{p}$ y comprobamos que sea distinto a 1, de modo G que sea un generador del subgrupo cíclico de orden q.

2.3.2 Generación de claves

Tomamos un entero x tal que $0 < x < q$ que actuará como clave privada. A continuación, calculamos $E = G^x \pmod{p}$ y lo tomamos como clave pública.

2.3.3 Firma

La firma de un mensaje m (entero) con la clave privada x se define como sigue:

1. Tomamos un entero aleatorio e tal que $0 < e < q$
2. Calculamos $X \equiv G^e \pmod{p}$ y $r \equiv X \pmod{q}$. Si $r = 0$ volvemos al paso 1.
3. Calculamos el inverso de e módulo q.
4. Calculamos $s \equiv e^{-1}.H(m) + xr) \pmod{q}$. Si $s = 0$ volvemos al paso 1.
5. Devolvemos la firma $(r; s)$.

2.3.4 Verificación

1. Verificamos que $r; s \in .0; n/$ si no, rechazamos la firma.
2. Calculamos $w \leftarrow s^{*1} .\mathrm{mod}\ q/$
3. Calculamos $u_1 \leftarrow H.m/w .\mathrm{mod}\ q/$ y $u_2 \leftarrow rw .\mathrm{mod}\ q/$.
4. Calculamos $X = G^{u_1} E^{u_2} .\mathrm{mod}\ p/$ y $v \leftarrow X .\mathrm{mod}\ q/$.
5. Aceptamos la firma si $v = r$.

En la descripción de DSA hemos trabajado con el grupo $.F_p^{\bullet}; </.$. Posteriormente, introduciremos ECDSA que es la versión sobre curvas elípticas y la que inspira el algoritmo principal del trabajo, EdDSA. La referencia principal ha sido [14].

2.4 Certificado Digital

Observamos que con el procedimiento abstracto de firma digital descrito (y la descripción de DSA) solo podemos garantizar la integridad del mensaje. Sabemos que el mensaje no ha sido modificado pero no podemos asegurar que quien lo ha firmado sea verdaderamente quien dice ser. Para resolver esto, necesitaríamos algún medio que garantizase que la clave pública corresponde a cierta identidad.

Es aquí donde entra la conocida como **Infraestructura de Clave Pública** (PKI por sus siglas en inglés). Aunque no la estudiaremos a fondo, es conveniente introducirla para tener una visión más general de la utilidad de la firma digital.

El componente principal de una PKI es el **Certificado Digital**. Consiste en una identificación entre una identidad y una clave pública. Evidentemente, no cualquiera podría crear este certificado pues encontraríamos los mismos problemas que antes. Es por ello que son emitidos y firmados por una autoridad de confianza para todas las partes, la conocida como **Autoridad de Certificación**. De este modo, conseguimos autenticación y no repudio, pues cualquiera podría confirmar la identidad con el certificado. La única forma de suplantar esta identidad sería obteniendo la clave privada asociada a la pública del certificado. Para controlar estas problemáticas existe la denominada **Lista de Revocación de Certificados** (CRL por sus siglas en inglés) que indica si un certificado ya no es válido. La gestiona la Autoridad de Certificación. Además, para realizar el registro tendríamos la **Autoridad de Registro** que verificaría, generalmente presencialmente, la identidad del solicitante del certificado. Por

último, tendríamos las denominadas **Autoridades de Validación** que comprobarían la validez de los certi□cados. Notemos que una misma entidad podría jugar el papel de varias de las autoridades mencionadas. En el caso de España la principal autoridad de certi□cación es la Fábrica Nacional de Moneda y Timbre (FNMT).

El siguiente diagrama ejempli□ca la versión de la PKI expuesta. Esta es una versión simple, pudiendo ganar complejidad con diferentes autoridades más como la **Autoridad de sellado en el tiempo**.

Como se observa en la □gura, Bob tendría la certeza de que el documento no ha sido modi□cado y que ha sido □rmado por Alice. Además, Alice no podría negar en un futuro haberlo □rmado. Es decir, tenemos integridad, autenticidad y no repudio.

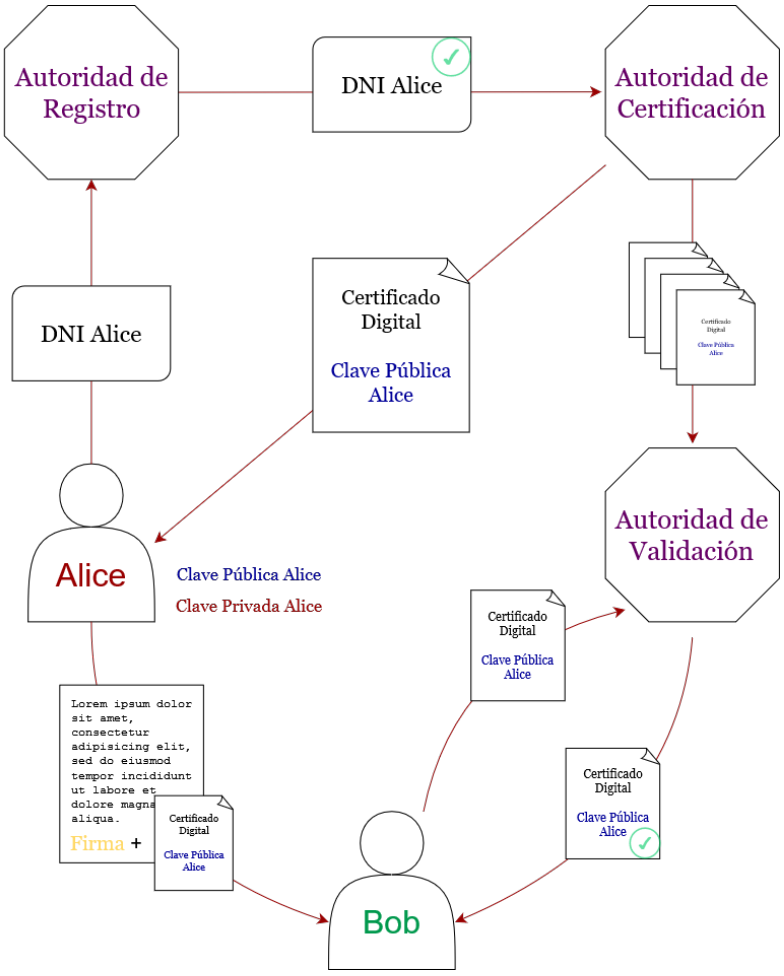

Figura 2.2: Esquema general de infraestructura de clave pública

3 Curvas Elípticas

En este capítulo haremos una introducción a las curvas elípticas (Sección 3.1), muy relevantes en Criptografía. Nos centraremos en estudiar la fórmula de suma y en comprobar la estructura de grupo en la Sección 3.2. Asimismo, veremos qué estructura tiene el grupo cuando estamos trabajando con cuerpos □nitos (Sección 3.3). Todo esto nos servirá para introducir la □rma DSA sobre curvas elípticas: ECDSA (Sección 3.4).

En el capítulo siguiente introduciremos las curvas de Edwards y estableceremos una equivalencia birracional entre estas últimas y las curvas elítpticas. Ahora empezamos con las siguientes de□niciones:

3.1 Curvas Elípticas: Definiciones

Definición 3.1. *Sea* K *un cuerpo. Sea* F Ë K[X; Y; Z] *un polinomio homogéneo de grado 3 de la forma:*

$$F.X;Y;Z/ = Y^2Z + a_1XYZ + a_3YZ^2 * X^3 * a_2X^2Z * a_4X^2Z * a_4XZ^2 * A_6Z^3$$

donde a_i Ë K *para todo* i Ë ^1; :::; 6`. *De nimos una ecuación de Weierstrass como una ecuación de la forma* F = 0.

Definición 3.2. *De nimos una curva elíptica sobre un cuerpo* K *como una curva proyectiva, algebraica, plana y no singular, dada por una ecuación de Weierstrass. Denotamos al conjunto de puntos de la curva como* E.K/.

Observación 3.1. Hemos dado la forma general de las curvas elípticas, sin embargo, si la característica del cuerpo es distinta a 2 y 3 existe un cambio de sistema de referencia respecto del cual la ecuación queda de la forma:

$$F.X;Y;Z/ = Y^2Z * X^3 * aXZ^2 * bZ^3$$

con a; b Ë K. Considerando la parte afín (Z = 1) tenemos la ecuación afín:

$$y^2 = x^3 + Ax + B$$

con A; B Ë K. [19] Denominamos a esta forma de describir la curva como forma (o ecuación) de Weierstrass corta.

En la práctica identi□caremos las curvas elípticas con su parte afín. En lo que sigue denotaremos al punto del in□nito como O.

3.2 Ley de grupo en Curvas Elípticas

De□niremos la ley de grupo usando un enfoque geométrico.

Definición 3.3 (Ley de composición). *Sea* E *una curva elíptica dada por la ecuación de Weierstrass. Sean* P *y* Q *puntos de la curva. Denominamos* L *a la recta que pasa por* P *y* Q *(si* P = Q, L *es la tangente a* E *en* P*), y sea* R *el tercer punto de intersección de* L *con* E*.*

Llamamos L¨ *a la recta que pasa por* R *y* O*. Entonces,* L¨ *interseca a la curva* E *en* R, O *y un tercer punto que denotamos* P + Q*.*

Teorema 3.1. *La ley de composición tiene las siguientes propiedades:*

1. *Si* L *interseca a* E *en los puntos (no necesariamente distintos)* P ,Q *y* R*, entonces*

$$.P + Q/ + R = O$$

2. P + O = P *para todo* P Ë E*.*
3. P + Q = Q + P *para todo* P ; Q Ë E*.*
4. *Dado* P Ë E *existe otro punto de* E*, denotado* *P *que satisface*

$$P + .*P/ = O$$

5. *Sean* P ; Q; R Ë E*. Entonces*

$$.P + Q/ + R = P + .Q + R/$$

Por tanto, la ley de composición dota a E *de estructura de grupo abeliano con elemento neutro* O*. Más aún,*

6. *Suponiendo* E *de nida sobre el cuerpo* k. *Entonces,*

$$E.k/ = \hat{}.x; y/ \ddot{E}\ k^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6` \ddot{a}\ \hat{}O`$$

es un subgrupo de E.

Demostración. La prueba puede consultarse en [18, Capítulo 3, Proposición 2.2]. ∎

Hemos comprobado que la ley de composición actúa como la suma en curvas elípticas. Veamos grá□camente sobre la curva $y^2 = x^3 * x + 1$ ejemplos de los anteriores casos.

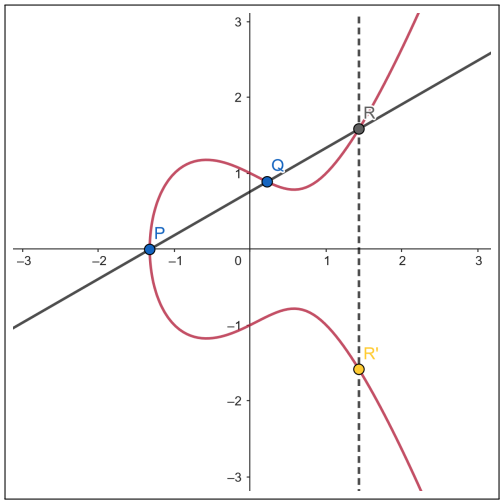

Figura 3.1: Caso general suma curvas elípticas: P + Q = R¨

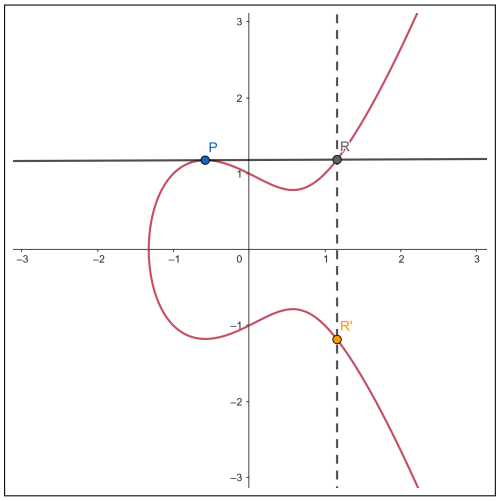

Figura 3.2: Caso suma curvas elípticas: P = Q

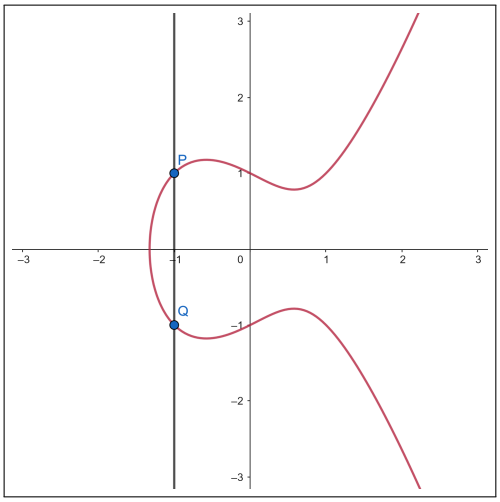

Figura 3.3: Caso suma curvas elípticas: r paralela al eje OY

3.3 Curvas Elípticas sobre Cuerpos Finitos

En esta sección nos centraremos en estudiar el grupo $.E.F_q/; +/$ donde F_q es un cuerpo □nito con $q = p^r$ y $p \neq 2; 3$ primo. Dado que $E.F/ \subset P^2.F_q/$, siendo este último un conjunto □nito, $E.F_q/$ debe serlo también. Además, se tiene la siguiente cota.

Teorema 3.2 (de Hasse). *Sea* $E.F_q/$ *una curva elíptica, entonces*

$$\delta E.F_q/ * .q + 1/\delta \; f \; 2 \; \grave{u} \; \overline{q}$$

Demostración. Se puede consultar la prueba en [19]

No obstante, no solo es interesante tener identificada una cota sino también tener clara la estructura del grupo abeliano. Ese resultado nos lo da el siguiente teorema.

Teorema 3.3. *Sea* $E.F_q/$*, entonces se dan una de estas dos situaciones:*

1. $E.F_q/ \ddot{o} \; Z\_Z_n$ *para algún entero* $n \; g \; 1$.
2. $E.F_q/ \ddot{o} \; .Z\_Z_{n_1}/ \bullet .Z\_Z_{n_2}/$ *para dos enteros* $n_1; n_2$ *con* $n_1 \; \acute{Y} \; n_2$.

Demostración. La prueba podemos encontrarla en [19].

Es decir, el grupo de puntos de la curva $E.F_q/$ es cíclico o producto de dos grupos cíclicos.

3.4 Firma digital ECDSA

Una vez hemos visto que es una firma digital y la teoría básica de curvas elípticas podemos construir una firma usando estas curvas. Al igual que hicimos con DSA introducimos el algoritmo ECDSA (*Elliptic Curve Digital Sign Algorithm*) del siguiente modo. Nos basamos en la descripción del algoritmo de [14].

3.4.1 Parámetros del algoritmo

Los parámetros son los siguientes:

- H función Hash.
- E curva elíptica definida sobre el cuerpo finito F_q con $q = p$ primo impar o $q = 2^m$.
- dos elementos $a; b \; \ddot{E} \; F_q$ que definen la ecuación de la curva E sobre F_q. Esto es:
  - $y^2 = x^3 + ax + b$ si $p > 3$
  - $y^2 = x^3 + ax^2pb$ si $p = 2$

– $G \in E(F_q)$ punto base de orden n primo y que cumpla $n > 2^{160}$ y $n > \sqrt[4]{q}$.
– $h = \frac{\#E(F_q)}{n}$ llamado el cofactor.

3.4.2 Generación de claves

Partimos de un entero aleatorio d perteneciente al intervalo $(0, n)$. Calculamos

$$Q = dG$$

Nuestra clave pública sería Q y la privada asociada, d.

3.4.3 Firma

Partiendo del mensaje a firmar m (entero) y del par de claves (Q, d) hacemos lo siguiente:

1. Tomamos un entero aleatorio $k \in (0, n)$.
2. Calculamos $kG = (x, y)$ tomando x como entero.
3. Calculamos $r = x \pmod n$. En el caso que $r = 0$ volvemos al paso 1.
4. Calculamos el inverso de k módulo n.
5. Calculamos $s = k^{-1}(H(m) + dr) \pmod n$. También comprobamos que $s \neq 0$. Si no, volvemos al paso 1.
6. Devolvemos la firma del mensaje, (r, s).

3.4.4 Verificación

Partimos del mensaje m y su firma (r, s).

1. Verificamos que $r, s \in (0, n)$. Si no, rechazamos la firma.
2. Calculamos $w = s^{-1} \pmod n$.
3. Calculamos $u_1 = H(m)w \pmod n$ y $u_2 = rw \pmod n$:
4. Calculamos $X = u_1G + u_2Q$.
5. Si $X = O$ rechazamos la firma. En caso contrario, tomamos la primera componente de X como un entero x y calculamos $v = x \pmod n$.

6. Aceptamos la firma si y solo si $v = r$.

Podemos encontrar los cálculos de la verificación en [14].

Observación 3.2 (Comparación de DSA y ECDSA). A nivel conceptual, hemos obtenido ECDSA a partir de DSA al reemplazar el subgrupo de orden q de F_p^x generado por g por el subgrupo de puntos de la curva elíptica generado por G. La única diferencia significativa entre ambos la encontramos en el miembro derecho de la firma r. Mientras que en DSA se obtiene al reducir módulo q el elemento X resultando en un entero en el intervalo $.0; q/$, en ECDSA generamos r en el intervalo $.0; n/$ tomando la primera coordenada del punto aleatorio kG y reduciéndola módulo n.

4 Curvas de Edwards

El propósito de este capítulo es introducir una clase de curvas, las curvas de Edwards, que nos permiten usar todo el potencial de la rma digital de curvas elípticas con una mayor velocidad de ejecución. Se introducirá la ley de suma de las mismas y cómo podemos pasar de curvas elípticas clásicas a curvas de Edwards. Además, se proponen las Curvas de Edwards torcidas como una mejora de las anteriores, con resultados aún más rápidos para ciertos casos. La referencias fundamentales de este capítulo son dos de las publicaciones principales que mencionamos en la Introducción, [10] para la Sección 4.1 y [7] para la Sección 4.2.

4.1 Curvas de Edwards y método de suma

En [12], Harold M. Edwards prueba que toda curva elíptica sobre un cuerpo de números algebraicos es birracionalmente equivalente a una curva de la forma $x^2+y^2 = c^2.1 + x^2y^2/$ con $.0; c/$ como elemento neutro ($c \neq 0$) y con la siguiente fórmula de suma:

$$.x_1; y_1/; .x_2; y_2/ \mapsto \left(\frac{x_1y_2 + y_1x_2}{c.1 + x_1x_2y_1y_2/}; \frac{y_1y_2 * x_1x_2}{c.1 * x_1x_2y_1y_2/} \right)$$

Lo que nos lleva a la siguiente de nición:

Definición 4.1. *Sea k un cuerpo con característica distinta a 2. De nimos una curva de Edwards como una curva de la forma*

$$E_c : x^2 + y^2 = c^2.1 + x^2y^2/$$

donde $c \neq k$ *es no nulo.*

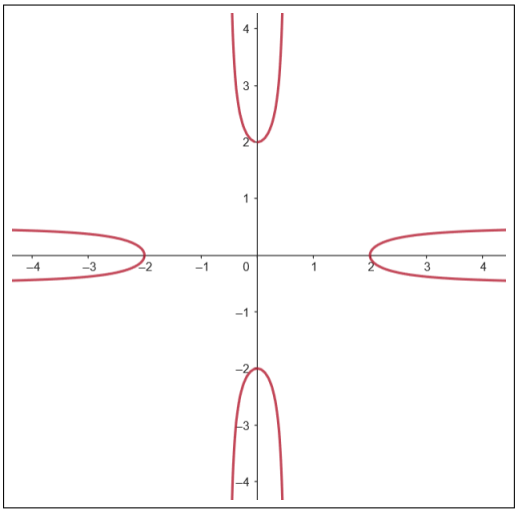

Figura 4.1: Curva de Edwards con c = 2

A continuación probaremos que, dadas ciertas condiciones, estas curvas son birracionalmente equivalentes a algún tipo de twist de curva elíptica. Recordamos el concepto de twist cuadrático.

Definición 4.2. *Sea* k *un cuerpo con característica distinta a* 2. *Suponemos* E.k/ *curva elíptica de la forma*

$$y^2 = x^3 + a_2x^2 + a_4x + a_6$$

Sea d ‘ 0 *un no cuadrado en* k. *De nimos el twist cuadrático de* E *como*

$$E_d : dy^2 = x^3 + a_2x^2 + a_4x + a_6$$

Observación 4.1. Notemos que E y E_d no son isomorfas sobre k. Sí lo son sobre la extensión k. $\sqrt{d}$/.

Debemos hacer una consideración más. En la introducción mencionamos la equivalencia birracional pero no la hemos de nido todavía. Si bien la igualdad entre dos curvas es la isomorfía, podemos establecer una noción de "casi iguales". Esta noción intuitiva es la que representa la equivalencia birracional. De modo riguroso:

Definición 4.3. *Sean* E_1 *y* E_2 *dos curvas. Decimos que son birracionalmente equivalentes si existe una función racional* : E_1 ™ E_2 *de nida en todo punto de* E_1 *salvo*

un número nito de excepciones, y si su inversa $^{*1} : E_2 \to E_1$ *también es racional y está de nida en todo punto de* E_2 *salvo número nito de excepciones. La función se conoce como equivalencia birracional y es de la forma:*

$$.x; y/ = \left(\frac{p_1.x; y/}{p_2.x; y/}; \frac{q_1.x; y/}{q_2.x; y/}\right)$$

donde $p_1; p_2; q_1; q_2$ *son polinomios.*

Una vez recordadas ambas nociones, pasamos al primer teorema en el que establecemos la equivalencia birracional entre curvas de Edwards y curvas elípticas.

Teorema 4.1. *Sea* k *un cuerpo con característica distinta de* 2. *Sea* E *una curva elíptica sobre* k *tal que el grupo* E.k/ *tiene un elemento de orden* 4. *Se tiene que:*

1. *Existe* $d \in k \setminus \{0; 1\}$ *tal que la curva* $x^2 + y^2 = 1 + dx^2y^2$ *es birracionalmente equivalente sobre* k *a un twist cuadrático de* E.
2. *Si* E.k/ *tiene un único elemento de orden* 2, *entonces existe un no cuadrado* $d \in k$ *tal que la curva* $x^2 + y^2 = 1 + dx^2y^2$ *es birracionalmente equivalente sobre* k *a un twist cuadrático de* E.
3. *Si* k *es nito y* E.k/ *tiene un único elemento de orden* 2, *entonces existe un no cuadrado* $d \in k$ *tal que la curva* $x^2 + y^2 = 1 + dx^2y^2$ *es birracionalmente equivalente sobre* k *a* E.

Demostración. Tomamos la curva

$$E : s^2 + a_1rs + a_3s = r^3 + a_2r^2 + a_4r + a_6$$

Sin pérdida de generalidad podemos asumir que $a_1 = 0$ y $a_3 = 0$. Si no, para el caso general consideramos $\bar{s} = s + \frac{a_1r + a_3}{2}$.

De nimos P como el punto de orden 4 en E. Asumimos también sin pérdida de generalidad que $2P = .0; 0/$, lo que implica que $a_6 = 0$. Del mismo modo, para el caso general podemos de nir $\bar{r} = r * r_2$ donde $2P = .r_2; s_2/$.

Establecido esto, nos queda que la curva E tiene la forma

$$s^2 = r^3 + a_2r^2 + a_4r$$

Sea $P = .r_1; s_1/$. Vamos a intentar expresar a_2 y a_4 en términos de r_1 y s_1.

Notemos que $s_1 \neq 0$, si no P tendría orden 2. Por consiguiente, obtenemos también que $r_1 \neq 0$. La ecuación $2P = .0; 0/$ signi□ca que la tangente de E en el punto P pasa por $.0; 0/$, esto es,

$$s_1 * 0 = .r_1 * 0/$$

donde $= \dfrac{3r_1^2 + 2a_2r_1 + a_4}{2s_1}$ es la pendiente de la tangente en el punto P. Entonces,

$$3r_1^3 + 2a_2r_1^2 + a_4r_1 = 2s_1^2$$

Además, como $P \in E$

$$2s_1^2 = 2r_1^3 + 2a_2r_1^2 + 2a_4r_1$$

Restando una de las ecuaciones a la otra, obtenemos $r_1^3 = a_4r_1$, es decir,

$$a_4 = r_1^2$$

Adicionalmente, $a_2 = \dfrac{s_1^2 * r_1^3 * a_4r_1}{r_1^2} = \dfrac{s_1^2}{r_1^2} * 2r_1$. Tomando $d = 1 * 4\dfrac{r_1^3}{s_1^2}$ obtenemos que

$$a_2 = 2\frac{1 + d}{1 * d}r_1$$

Notemos que $d \neq 1$ pues $r_1 \neq 0$. Notemos también que $d \neq 0$. Si no, tendríamos una contradicción pues $r^3 + a_2r^2 + a_4r = r^3 + 2r_1r^2 + r_1^2r = r.r + r_1/^2$ lo que contradiría la hipótesis de que E sea curva elíptica.

Consideremos dos twists cuadráticos de E,

$$E'' : \frac{r_1}{1 * d}s^2 = r^3 + a_2r^2 + a_4r$$

$$E'''' : \frac{dr_1}{1 * d}s^2 = r^3 + a_2r^2 + a_4r$$

Notemos por la Observación 4.1 que ambas curvas no son isomorfas pero sí birracionalmente equivalentes. Veremos que $x^2 + y^2 = 1 + dx^2y^2$ es birracionalmente equivalente a alguno de los dos twists.

Si tomamos $u = \frac{r}{r_1}$ y $v = \frac{s}{r_1}$ y sustituimos en los twists cuadráticos vemos que $E^{¨}$ es isomorfa a

$$C^{¨} : \frac{1}{1 * d}v^2 = u^3 + 2\frac{1 + d}{1 * d}u^2 + u$$

y $E^{····}$ es isomorfa a

$$C^{····} : \frac{d}{1 * d}v^2 = u^3 + 2\frac{1 + d}{1 * d}u^2 + u$$

Veamos ahora que la curva $x^2 + y^2 = 1 + dx^2y^2$ es birracionalmente equivalente a $C^{¨}$ y por consiguiente lo es con $E^{¨}$. Presentamos la siguiente función

$$.u; v/ ;™ \quad 2\frac{u}{v}; \frac{u * 1}{u + 1}$$

con inversa,

$$.x:y/ ;™ \quad \begin{matrix} 0 \\ \frac{1 + y}{1 * y}; 2\frac{1 + y}{1 * y}x \end{matrix}^1$$

Solo hay un número □nito de puntos excepcionales tales que $v.u + 1/ = 0$ y $.1 * y/x = 0$ respectivamente. En [9] encontramos el cálculo directo que muestra que la imagen de la inversa satisface $C^{¨}$. Por tanto, existe una equivalencia birracional entre $x^2 + y^2 = 1 + dx^2y^2$ y $E^{¨}$, con lo que tendríamos el primer resultado.

Notemos también que si d es un cuadrado, existe otro punto de orden 2 en $E.k/$, concretamente $r_1 \begin{matrix} H \\ ù \end{matrix} \frac{u}{d} \begin{matrix} \_ \\ \_ \end{matrix} \frac{d + 1}{d * 1}; 0 \begin{matrix} I \\ \_ \end{matrix}$. Por tanto, también tenemos el punto 2, pues al existir un solo punto de orden 2 necesariamente d tiene que ser no cuadrado.

Concluimos probando el tercer resultado. Si k es □nito y d no es un cuadrado entonces, o $\frac{r_1}{1 * d}$ o $\frac{dr_1}{1 * d}$ es un cuadrado en k. Por tanto, E es isomorfa a $E^{¨}$ o $E^{····}$. Vimos para el primer punto la equivalencia birracional entre $E^{¨}$ y $x^2 + y^2 = 1 + dx^2y^2$. Veamos que también se tiene entre esta última y $E^{····}$.

Si tomamos $C^{¨}$ y sustituimos 1_d por d y $*u$ por u vemos que $x^2 + y^2 = 1 + dx^2y^2$ es birracionalmente equivalente a

$$\frac{1}{1 * 1\_d}v^2 = .*u/^3 + 2\frac{1 + 1\_d}{1 * 1\_d}.*u/^2 + .*u/$$

esto es, es birracionalmente equivalente a $C^{····}$ y por tanto a $E^{····}$.

Entonces, como E es isomorfa a E'' o E''' y $x^2+y^2 = 1+dx^2y^2$ es birracionalmente equivalente a ambas, se sigue que $x^2 + y^2 = 1 + dx^2y^2$ es birracionalmente a E sobre k, con lo que queda probado el tercer punto.

Observación 4.2. Si tomamos $d = dc^4$, $x = cx$ e $y = cy$, entonces tenemos que la curva $x^2 + y^2 = c^2.1 + dx^2y^2/$ es isomorfa a $x^2 + y^2 = 1 + dx^2y^2$. Las condiciones sobre c y d que nos quedarían serían $c; d \neq 0$ y $dc^4 \neq 1$.

Este teorema es fundamental, pues nos da una equivalencia clara entre curvas de Edwards y curvas elípticas. Esto será necesario para nuestro objetivo de usar criptografía de curvas elípticas con Curvas de Edwards.

A partir de ahora, para considerar un mayor número de curvas elípticas, ampliamos la noción de curva de Edwards a las curvas de la forma

$$E_{c;d} : x^2 + y^2 = c^2.1 + dx^2y^2/$$

con $c; d \neq 0$ y $dc^4 \neq 1$ y con la siguiente fórmula de suma:

$$.x_1; y_1/; .x_2; y_2/ \mapsto \left(\frac{x_1y_2 + y_1x_2}{c.1 + dx_1x_2y_1y_2/}; \frac{y_1y_2 * x_1x_2}{c.1 * dx_1x_2y_1y_2/} \right) \tag{4.1}$$

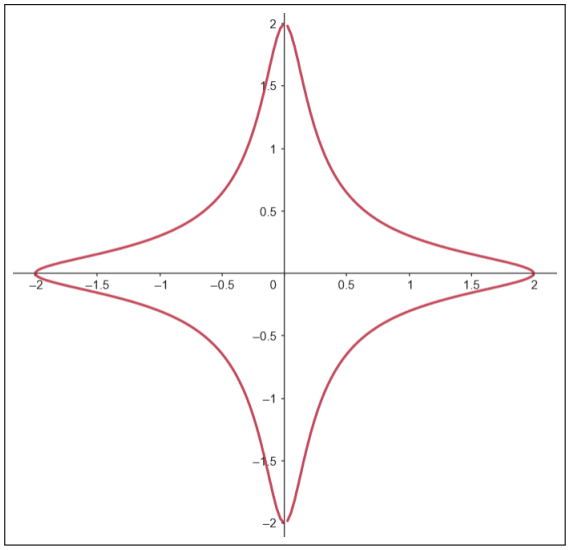

Figura 4.2: Curva de Edwards con $c = 2$ y $d = *8$

Una vez establecidas las curvas con las que trabajaremos, veamos que ocurre con la fórmula de suma introducida. En efecto, comprobaremos que efectivamente es una

fórmula de suma y bajo qué condiciones no tendremos puntos especiales. Es necesario hacer este estudio para ver si los puntos de las Curvas de Edwards con la fórmula 4.1 forman un grupo aditivo.

Comenzaremos con el siguiente teorema, donde veremos en qué condiciones la fórmula de suma de nida anteriormente es una operación interna.

Teorema 4.2. *Sea* k *un cuerpo con característica distinta de* 2. *Sea* $E_{c;d}$ *una curva de Edwards. Dados* $.x_1; y_1/; .x_2; y_2/ \ddot{E} E_{c;d}$ *y asumiendo* $\eth dx_1x_2y_1y_2\eth \text{ ‘ } 1$, *de nimos* $.x_3; y_3/$ *como el resultado de sumar* $.x_1; y_1/$ *y* $.x_2; y_2/$ *con la suma de nida por la fórmula (4.1). Entonces se tiene que:*

$$x_3^2 + y_3^2 = c^2.1 + dx_3^2y_3^2/$$

Es decir, $.x_3; y_3/ \ddot{E} E_{c;d}$.

Demostración. De nimos

$$T = .x_1y_2+y_1x_2/^2.1*dx_1x_2y_1y_2/^2+.y_1y_2*x_1x_2/^2.1+dx_1x_2y_1y_2/^2*d.x_1y_2+y_1x_2/^2.y_1y_2*x_1x_2/^2$$

Manipulando la ecuación llegamos a que

$$T = .x_1^2 + y_1^2 * .x_2^2 + y_2^2/dx_1^2y_1^2/.x_2^2 + y_2^2 * .x_1^2 + y_1^2/dx_2^2y_2^2/$$

Tomamos los puntos $.x_1; y_1/$ y $.x_2; y_2/$ de la hipótesis. Sustrayendo

$$.x_2^2 + y_2^2/dx_1^2y_1^2 = c^2.1 + dx_2^2y_2^2/dx_1^2y_1^2$$

de la ecuación

$$x_1^2 + y_1^2 = c^2.1 + dx_1^2y_1^2/$$

vemos que

$$x_1^2 + y_1^2 * .x_2^2 + y_2^2/dx_1^2y_1^2 = c^2.1 * d^2x_1^2x_2^2y_1^2y_2^2/$$

Análogamente,

$$x_2^2 + y_2^2 * .x_1^2 + y_1^2/dx_2^2y_2^2 = c^2.1 * d^2x_1^2x_2^2y_1^2y_2^2/$$

Por tanto, $T = c^4.1 * d^2x_1^2x_2^2y_1^2y_2^2/^2$.

Para nalizar con la prueba, aplicando la fórmula de suma obtenemos $.x_3; y_3/$ en términos de $x_1; x_2; y_1; y_2$. Esto es,

$$x_3^2 + y_3^2 * c^2dx_3y_3 =$$

$$\frac{.x_1y_2 + y_1x_2/^2}{c^2.1 + dx_1x_2y_1y_2/^2} + \frac{.y_1y_2 * x_1x_2/^2}{c^2.1 * dx_1x_2y_1y_2/^2} * \frac{c^2d.x_1y_2 + y_1x_2/^2.y_1y_2 * x_1x_2/^2}{c^4.1 + dx_1x_2y_1y_2/^2.1 * dx_1x_2y_1y_2/^2} =$$

$$\frac{T}{c^2.1 + dx_1x_2y_1y_2/^2.1 * dx_1x_2y_1y_2/^2} =$$

$$\frac{T}{c^2.1 * d^2x_1^2x_2^2y_1^2y_2^2/^2} = c^2$$

Por consiguiente $x_3^2 + y_3^2 = c^2.1 + dx_3y_3/^2$, llegando al resultado.

Una vez visto que la fórmula es interna, veamos en las mismas condiciones cómo podemos trasladar los puntos de una curva de Edwards a puntos de una curva elíptica. Concretamente, veremos que los puntos trasladados pertenecen a una curva elíptica y preservan la fórmula de suma estándar para curvas elípticas de□nida en la sección anterior.

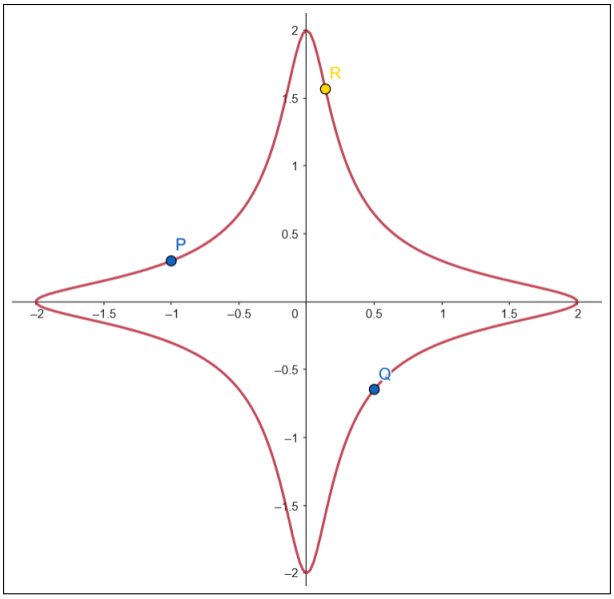

Figura 4.3: Suma de Edwards P + Q = R en la curva con c = 2 y d = *8

Teorema 4.3. *En la situación del Teorema 4.2, sea* $e = 1 * dc^4$ *y sea la curva elíptica* $E: \frac{1}{e} v^2 = u^3 + \frac{4}{e} * 2 u^2 + u$. *Para cada* i Ë $1; 2; 3$ *se de ne* P_i *de la siguiente forma:*

$$P_i = \begin{cases} O & si\ \ .x_i; y_i/ = .0; c/ \\ .0; 0/ & si\ \ .x_i; y_i/ = .0; *c/ \\ \left(\dfrac{c + y_i}{c * y_i}; \dfrac{2c.c + y_i/}{x_i.c * y_i/}\right) & si\ \ x_i \ ‘\ 0 \end{cases}$$

Entonces P_i Ë $E.k/$ *y* $P_1 + P_2 = P_3$.

Demostración. Comencemos viendo que los P_i Ë $E.k/$.

Si $.x_i; y_i/ = .0; c/$ entonces se tiene que $P_i = O$ Ë $E.k/$. También si $.x_i; y_i/ = .0; *c/$ obtenemos que $P_i = .0; 0/$ Ë $E.k/$. El último caso es un cálculo análogo al hecho en el Teorema 4.1

Continuamos viendo que $P_1 + P_2 = P_3$. Veamos los diferentes casos de la fórmula de suma estándar. A lo largo de la demostración, la fórmula de suma estándar y la fórmula de Edwards se referirán a la fórmula de suma de la De nición 3.3 y a la fórmula 4.1 respectivamente.

A su vez, a lo largo de todos los casos denotaremos $P_i = .u_i; v_i/$ con

$$u_i = \frac{c + y_i}{c * y_i} \quad y \quad v_i = 2c\frac{u_i}{x_i}$$

para $i = 1; 2; 3$.

Caso 1. Si $.x_1; y_1/ = .0; c/$ entonces de la fórmula de suma de Edwards resulta que $.x_3; y_3/ = .x_2; y_2/$. Ahora, usando la de nición de los P_i del enunciado, P_1 es el punto del in nito y $P_2 = P_3$. En consecuencia, se tiene que

$$P_1 + P_2 = O + P_2 = P_2 = P_3$$

El razonamiento es análogo para $.x_2; y_2/ = .0; c/$.

Visto este caso, asumiremos a partir de ahora que $.x_1; y_1/; .x_2; y_2/ \ ‘\ .0; c/$

Caso 2. Suponemos $.x_3; y_3/ = .0; c/$. De nuevo obtenemos por la fórmula de Edwards que $.x_2; y_2/ = .*x_1; y_1/$. Tenemos entonces dos casos para el primer punto. O bien, $.x_1; y_1/ = .0; *c/$ o bien $x_1 \ ‘\ 0$.

Caso 2.1 Si $.x_1; y_1/ = .0; *c/$ entonces también $.x_2; y_2/ = .0; *c/$ y trasladando según el enunciado se tiene que $P_1 = .0; 0/ = P_2$.

Caso 2.2 Si $x_1 \neq 0$ entonces $x_2 \neq 0$. Entonces, la primera componente de P_2 queda

$$u_2 = u_1$$

Respecto a la segunda componente obtenemos

$$v_2 = 2c\frac{u_2}{x_2} = *2c\frac{u_1}{x_1} = *v_1$$

lo que implica que $P_1 = *P_2$.

En ambos casos usando la fórmula de suma estándar de curvas elípticas llegamos a que $P_1 + P_2 = O = P_3$.

Asumimos a partir de ahora que $.x_3; y_3/ \neq .0; c/$.

Caso 3. Supongamos que $.x_1; y_1/ = .0; *c/$. En ese caso concluimos que

$$.x_3; y_3/ = .*x_2; *y_2/$$

Ahora, como $.x_2; y_2/ \neq .0; *c/$ y $.x_2; y_2/ \neq .0; c/$ debe darse que $x_2 \neq 0$.

Trasladando, llegamos a $P_1 = .0; 0/$ y $P_2 = .u_2; v_2/$. La fórmula de suma estándar establece que $.0; 0/ + .u_2; v_2/ = .r_3; s_3/$ donde

$$r_3 = \frac{1}{e}\left(\frac{v_2}{u_2}\right)^{0}\left(\frac{4}{e} * 2\right)^{1} * u_2 = \frac{1}{u_2}$$

y

$$s_3 = *\frac{v_2}{u_2}r_3 = *\frac{v_2}{u_2^2}$$

Por otro lado, $P_3 = .u_3; v_3/$. Veamos detenidamente cada componente. Por un lado,

$$u_3 = \frac{c + y_3}{c * y_3} = \frac{c * y_2}{c + y_2} = \frac{1}{u_2} = r_3$$

Por otro lado,

$$v_3 = 2c\frac{u_3}{x_3} = *2c\frac{x_2}{u_2} = *\frac{v_2}{u_2^2} = s_3$$

Es decir, $P_1 + P_2 = P_3$.El razonamiento es análogo para $.x_2; y_2/ = .0; *c/$.

A partir de ahora, asumiremos que $x_1; x_2 \neq 0$.

Caso 4. En este caso suponemos que $.x_3; y_3/ = .0; *c/$. Una vez más, tenemos por la fórmula de Edwards que $.x_1; y_1/ = .x_2; *y_2/$ por lo que

$$u_1 = \frac{c + y_1}{c * y_1} = \frac{c * y_2}{c + y_2} = \frac{1}{u_2}$$

y

$$v_1 = 2c\frac{u_1}{x_1} = \frac{v_2}{u_2^2}$$

Además, $P_3 = .0; 0/$, por lo que usando la fórmula de suma estándar obtenemos que

$$*P_3 + P_2 = .0; 0/ + P_2 = \left(\frac{1}{u_2}; *\frac{v_2}{u_2^2}\right) = .u_1; *v_1/ = *P_1$$

Es decir, hemos obtenido de nuevo que $P_1 + P_2 = P_3$.

A partir de ahora asumimos que $x_3 \neq 0$.

Caso 5. Suponemos que $P_2 = *P_1$. Por la fórmula de Edwards,

$$u_2 = u_1 \quad y \quad v_2 = *v_1$$

por lo que

$$x_2 = *x_1$$

y

$$y_2 = c\frac{u_2 * 1}{u_2 + 1} = c\frac{u_1 * 1}{u_1 + 1} = y_1$$

En consecuencia $.x_3; y_3/ = .0; c/$ que ya ha sido discutido.

Por tanto, a partir de ahora asumimos que $P_2 \neq *P_1$.

Caso 6. Si $u_2 = u_1$ y $v_2 \neq *v_1$ entonces por la fórmula de suma estándar obtenemos que $.u_1; v_1/ + .u_2; v_2/ = .r_3; s_3/$ donde

$$= \frac{3u_1^2 + 2\frac{4}{e*2}u_1 + 1}{\frac{2}{e}v_1}$$

$$r_3 = \frac{1}{e}^{2} * .\frac{4}{e} * 2/ * 2u_1$$
$$s_3 = .u_1 * r_3/ * v_1$$

En [9] se tiene el cálculo que muestra que $.r_3; s_3/ = .u_3; v_3/$.

Caso 7. Si $u_2 \text{ ‘ } u_1$. Entonces por la fórmula de suma estándar obtenemos que $.u_1; v_1/ + .u_2; v_2/ = .r_3; s_3/$ donde

$$= \frac{v_2 * v_1}{u_2 * u_1}$$
$$r_3 = \frac{1}{e}^{2} * .\frac{4}{e} * 2/ * u_1 * u_2$$
$$s_3 = .u_1 * r_3/ * v_1$$

De nuevo, en [9] se tiene el cálculo que muestra que $.r_3; s_3/ = .u_3; v_3/$

Con esto quedan cubiertos todos los casos, por tanto se tiene que

$$P_1 + P_2 = P_3$$

Observación 4.3. El teorema anterior muestra que la imagen de la fórmula de suma de Edwards corresponde con la imagen de la fórmula de suma estándar en una curva elíptica E birracionalmente equivalente. Por consiguiente, podemos hacer operaciones de grupo en E haciendo las operaciones de grupo correspondientes en la curva de Edwards e invirtiendo la correspondencia.

Observación 4.4. Notemos que $x_i \text{ ‘ } 0$ implica que $y_i \text{ ‘ } c$.

A pesar de la correspondencia encontrada, las operaciones de grupo podrían encontrar puntos excepcionales donde la fórmula de suma de Edwards no estuviese de-nida. Veamos en el siguiente teorema como añadiendo la condición de que d no sea un cuadrado tenemos que la fórmula de suma de Edwards es **completa**, esto es, que está bien de-nida para todo par de puntos.

Teorema 4.4. *Sea* k *un cuerpo con característica distinta de* 2. *Sea* $E_{c;d}$ *una curva de Edwards y sean* $.x_1; y_1/; .x_2; y_2/ \text{ Ë } E_{c;d}$. *Si* d *no es un cuadrado en* k, *entonces* $ðdx_1x_2y_1y_2ð \text{ ‘ } 1$.

Demostración. Razonaremos por reducción al absurdo. Sea = $dx_1x_2y_1y_2$. Supongamos que Ë ^*1;1`. Entonces $x_1; x_2; y_1; y_2$ ‘ 0. Además, como .$x_2; y_2$/ Ë $E_{c;d}$ se tiene que

$$x_2^2 + y_2^2 = c^2.1 + dx_2^2y_2^2/$$

luego, multiplicando por $dx_1^2y_1^2$,

$$\begin{aligned} dx_1^2y_1^2.x_2^2 + y_2^2/ &= c^2.dx_1^2y_1^2 + d^2x_1^2y_1^2x_2^2y_2^2/ \\ &= c^2.dx_1^2y_1^2 + \ ^2/ \\ &= c^2.1 + dx_1^2y_1^2/ \\ &= x_1^2 + y_1^2 \end{aligned}$$

por lo que

$$\begin{aligned} .x_1 + \ y_1/^2 = x_1^2 + y_1^2 + 2\ x_1y_1 &= dx_1^2y_1^2.x_2^2 + y_2^2/ + 2x_1y_1dx_1x_2y_1y_2 \\ &= dx_1^2y_1^2.x_2^2 + 2x_2y_2 + y_2^2/ \\ &= dx_1^2y_1^2.x_2 + y_2/^2 \end{aligned}$$

Si $x_2 + y_2$ ‘ 0 entonces

$$d = \left(\frac{x_1 + \ y_1}{x_1y_1.x_2 + y_2/}\right)^2$$

y por lo tanto d sería un cuadrado, lo cual es una contradicción.

Obtenemos la misma contradicción si consideramos $x_2 * y_2$ ‘ 0. En efecto, esto implica de forma análoga que $d = \left(\frac{x_1 * \ y_1}{x_1y_1.x_2 * y_2/}\right)^2$ y por lo tanto d también sería un cuadrado, lo cual es una contradicción.

Dado que tanto $x_2 + y_2$ como $x_2 * y_2$ son 0, concluimos $x_2 = 0$ e $y_2 = 0$ pero esto es otra contradicción.

Por tanto, Ì ^*1;1`, esto es, ð$dx_1x_2y_1y_2$ð ‘ 1. ∎

Observación 4.5. Notamos que lo que hemos probado es la hipótesis ð$dx_1x_2y_1y_2$ð ‘ 1 del teorema 4.2 por lo que, trivialmente, que d no sea cuadrado en k es la condición de completitud que buscábamos en la ley de suma.

Con estos resultados las Curvas de Edwards y su fórmula de suma quedan claramente de□nidas. Esta familia de curvas ofrecen multitud de posibilidades como muestran Bernstein y Lange en [7]. Usando estas curvas se precisan de muchas menos operaciones que los algoritmos tradicionales de curvas elípticas. Solo con esto tendríamos

una mejora considerable, sin embargo, veremos en la siguiente sección que podemos hacerles un twist y obtener resultados aún mejores.

4.2 Curvas de Edwards torcidas

A continuación introduciremos la noción de curvas de Edwards torcidas. Estas no dejan de ser un twist cuadrático de las curvas de Edwards. El teorema 4.5 que probaremos más adelante es la conclusión fundamental de este breve estudio de curvas de Edwards. Estableceremos una equivalencia birracional entre las curvas de Edwards torcidas y las curvas de Montgomery.

A lo largo de [10] y [7] Bernstein et al. hacen un estudio exhaustivo del número de operaciones necesarias para ambas leyes de suma de Edwards, la usual y la torcida (que veremos más adelante). No solo muestran que las últimas son más rápidas, sino que comparan empíricamente con varios ejemplos. Es por ello que resulta interesante considerar este tipo de curvas. Comenzamos con unas de□niciones.

Definición 4.4. *Sea* k *un cuerpo con característica distinta de* 2. *Fijamos* a; d Ë k *distintos y no nulos. La curva de Edwards torcida con coe cientes* a *y* d *es la curva:*

$$E_{E;a;d} : ax^2 + y^2 = 1 + dx^2y^2$$

Observación 4.6. Una curva de Edwards es una curva de Edwards torcida con a = 1.

De□nimos a su vez las curvas de Montgomery.

Definición 4.5. *Sea* k *un cuerpo con característica distinta de* 2. *Sean* A Ë k ä ^*2; 2` *y* B Ë k ä ^0`. *La curva de Montgomery con coe cientes* A *y* B *es la curva elíptica:*

$$E_{M;A;B} : Bv^2 = u^3 + Au^2 + u:$$

El teorema que presentamos a continuación es el teorema fundamental de este trabajo. Hemos introducido las curvas de Montgomery porque son más rápidas que las curvas elípticas normales [16]. Asimismo, partiendo de que las curvas de Edwards

torcidas suponen una mejora signi□cativa respecto de las curvas de Edwards usuales, daremos una equivalencia birracional entre las curvas de Montgomery y las curvas de Edwards torcidas. Al establecer esta equivalencia birracional, tenemos garantizado que los resultados de criptografía que ya se tenían siguen funcionando con las nuevas curvas pero de forma considerablemente más rápida.

Teorema 4.5. *Sea* k *un cuerpo con característica distinta de* 2.

1. *Toda curva de Edwards torcida sobre* k *es birracionalmente equivalente sobre* k *a una curva de Montgomery.*

   Especí camente, jando a; d Ë k *distintos y no nulos, la curva de Edwards* $E_{E;a;d}$ *es birracionalmente equivalente a la curva de Montgomery* $E_{M;A;B}$ *donde*

   $$A = \frac{2.a + d/}{a * d} \text{ y } B = \frac{4}{a * d}$$

   La función

   $$: .x; y/ ;™ .u; v/ = \left(\frac{1 + y}{1 * y}; \frac{1 + y}{.1 * y/x}\right)$$

   es una equivalencia birracional de $E_{E;a;d}$ *a* $E_{M;A;B}$*, con inversa*

   $$^{*1} : .u; v/ ;™ .x; y/ = \left(\frac{u}{v}; \frac{u * 1}{u + 1}\right)$$

2. *Del mismo modo, toda curva de Montgomery sobre* k *es birracionalmente equivalente sobre* k *a una curva de Edwards torcida.*

   Especí camente, jando A Ë k ä ^*2; 2` *y* B Ë k ä ^0`*, la curva de Montgomery* $E_{M;A;B}$ *es birracionalmente equivalente a la curva de Edwards torcida* $E_{E;a;d}$*, donde*

   $$a = \frac{A + 2}{B} \text{ y } d = \frac{A * 2}{B}$$

Demostración. Comencemos por *1*.

Partiendo de

$$A = \frac{2.a + d/}{a * d},\ B = \frac{4}{a * d},\ u = \frac{1 + y}{1 * y} \text{ y } v = \frac{1 + y}{.1 * y/x}$$

Notamos que A y B están definidos ya que a ‘ d. Además, A Ë k ä ^*2; 2` y B Ë k ä ^0`. En efecto, por reducción al absurdo, si A = 2 entonces se tendría que a + d = a * d lo que implicaría d = 0, que es una contradicción. Igualmente, si A = *2 se obtiene la contradicción a = 0. De este modo, $E_{M;A;B}$ es curva de Montgomery. Sustituyendo ahora en $E_{M;A;B}$ obtenemos que

$$\frac{4}{a * d}\frac{.1 + y/^2}{.1 * y/^2x^2} = \frac{.1 + y/^3}{.1 * y/^3} + \frac{2.a + d/}{a * d}\frac{.1 + y/^2}{.1 * y/^2} + \frac{1 + y}{1 * y}$$

Observamos que para que esté bien definido, x ‘ 0 y y ‘ 1. Multiplicando por .1 * y/^{3} y dejando los términos en un mismo miembro, obtenemos

$$.1 + y/^4\ .1 + y/^2 + .1 * y/^2 + .1 + y/.1 * y/^0\ \frac{2.a + d/}{a * d} * \frac{4}{.a * d/x^2}^{15} = 0$$

Suponiendo y ‘ *1, tenemos que el segundo factor es igual a 0. Simplificando nos queda

$$.1 + y/^2 + .1 * y/^2 + 2.1 * y^2/^0\ \frac{.a + d/x^2 * 2}{.a * d/x^2}^1 = 0$$

Manipulando la ecuación,

$$.1 + y/^2 + .1 * y/^2 + 2.1 * y^2/^4\ 1 + \frac{.a + d/x^2 * 2}{.a * d/x^2}^0 * 1^{15} = 0$$

$$[.1 + y/ + .1 * y/]^2 + 2.1 * y^2/^0\ \frac{.a + d/x^2 * 2}{.a * d/x^2}^1 * 1 = 0$$

$$2 + .1 * y^2/^0\ \frac{2dx^2 * 2}{.a * d/x^2}^1 = 0$$

Multiplicando por .a * d/x^2 y simplificando nos queda:

$$.a * d/x^2 + .1 * y^2/.dx^2 * 1/ = 0$$

Por lo que nos queda:

$$ax^2 + y^2 = 1 + dx^2y^2$$

que es $E_{E;a;d}$. Los puntos excepcionales ocurren un número finito de veces, por lo que tenemos una equivalencia birracional y se obtiene el resultado.

Demostremos *2*.

Por un lado, partimos de que A ‘ *2 y A ‘ 2, a ‘ 0 y d ‘ 0. Además, a ‘ b. Por tanto, $E_{E;a;b}$ es curva de Edwards torcida.

Debemos ver que dado $a = \frac{A+2}{B}$ y $d = \frac{A*2}{B}$ se tiene que $A = \frac{2.a+d/}{a*d}$ y $B = \frac{4}{a*d}$. En efecto,

$$2\frac{a+d}{a*d} = 2\frac{\frac{A+2}{B}+\frac{A*2}{B}}{\frac{A+2}{B}*\frac{A*2}{B}} = A;$$

$$\frac{4}{a*d} = \frac{4}{\frac{A+2}{B}*\frac{A*2}{B}} = B$$

Por lo que se tiene que $E_{E;a;d}$ es birracionalmente equivalente a $E_{M;A;B}$ por *1*.

Para concluir la demostración estudiamos los puntos donde la inversa de la equivalencia birracional no está de nida. Estos puntos se corresponden a v = 0 y u = *1.

El punto .0; 0/ Ë $E_{M;A;B}$ corresponde al punto afín .0; *1/ Ë $E_{E;a;d}$ de orden 2. Este punto junto con .0; 1/ son los únicos puntos excepcionales de (Se tendría .0; 1/ = O).

Si .A + 2/.A * 2/ es un cuadrado existen dos puntos excepcionales más. Estos serían

$$H\ *A,\ \frac{ù\sqrt{.A+2/.A*2/}}{2};0\ I$$

ya que se tendría v = 0. Ambos tienen orden 2 en el in nito de la desingularización de $E_{E;a;d}$.

En el caso en que $\frac{A*2}{B}$ sea un cuadrado existen dos puntos excepcionales con u = *1. Vienen dados por

$$H\ *1;,\ u\sqrt{\frac{A*2}{B}}\ I$$

que corresponden con dos puntos de orden 4 en el in nito en la desingularización de $E_{E;a;d}$.

Hemos establecido una equivalencia birracional entre curvas de Montgomery y curvas de Edwards torcidas, sin embargo existen ciertas situaciones donde el twist no es necesario. Las exponemos en las siguientes proposiciones.

Proposición 4.1. Sea E una curva elíptica sobre un cuerpo k de característica distinta de 2. Entonces, el grupo E.k/ tiene un elemento de orden 4 si y solo si E es birracionalmente equivalente sobre k a una curva de Edwards.

Demostración. Supongamos que E es birracionalmente equivalente a $E_{E;1;d}$. Como el punto .1; 0/ en $E_{E;1;d}$ tiene orden 4, se tiene que E debe tener uno del mismo orden.

Por otro lado, suponemos que E tiene un punto $.u_4; v_4/$ de orden 4 (con $u_4; v_4 \neq 0$ como en la demostración del Teorema 4.3). Sin pérdida de generalidad suponemos que E tiene la forma

$$v^2 = u^3 + \left(\frac{v_4^2}{u_4^2} * 2u_4\right) u^2 + u_4^2 u$$

De□nimos $d = 1 * 4\frac{u_4^3}{v_4^3}$ y observamos que $d \neq \{0; 1\}$.

Puede comprobarse que tanto $x = \frac{v_4 u}{u_4 v}$ e $y = \frac{u * u_4}{u + u_4}$ satisfacen $x^2 + y^2 = 1 + dx^2y^2$ en el cuerpo de funciones de E. Los casos excepcionales $u_4 v = 0$ y $u = * u_4$ ocurren en un número □nito de puntos de E.

Por otro lado, tomando $u = u_4\frac{1 + y}{1 * y}$ y $v = v_4\frac{1 + y}{1 * y}x$ tenemos también que los casos excepcionales y = 1 y x = 0 ocurren en número □nito de puntos de $E_{E;1;d}$.

En conclusión, la función racional

$$.u; v/ \mapsto .x; y/ = \left(\frac{v_4 u}{u_4 v}; \frac{u * u_4}{u + u_4}\right)$$

con inversa

$$.x; y/ \mapsto .u; v/ = \left(u_4\frac{1 + y}{1 * y}; v_4\frac{1 + y}{1 * y}x\right)$$

es una equivalencia birracional de E a la curva de Edwards $E_{E;1;d}$.

Proposición 4.2. Si k es un cuerpo □nito con $\}k \neq 3 .mod4/$ entonces toda curva de Montgomery sobre k es birracionalmente equivalente sobre k a una curva de Edwards.

Demostración. Fijamos $A \ddot{E} k \ddot{a} \hat{} * 2; 2 \grave{}$ y $B \ddot{E} k \ddot{a} \hat{} 0 \grave{}$. Consideramos los siguientes casos:

Caso 1. Suponemos que $\frac{A+2}{B}$ es un cuadrado. Entonces, $E_{M;A;B}$ tendría el punto

$$\left(1; \sqrt{\frac{A+2}{B}}\right)$$

de orden 4.

Caso 2. Suponemos que $\frac{A+2}{B}$ no es un cuadrado pero $\frac{A*2}{B}$ sí lo es. Siguiendo el mismo razonamiento del caso anterior, $E_{M;A;B}$ tiene el punto

$$\left(*1; \sqrt{\frac{A*2}{B}}\right)$$

de orden 4.

Caso 3. Supongamos ahora que ni $\frac{A+2}{B}$ ni $\frac{A*2}{B}$ son cuadrados.

Como k es □nito, debe tenerse que $.A+2/.A*2/$ es un cuadrado. Esto implicaría que la curva de Montgomery $E_{M;A;A+2}$ tendría tres puntos:

$$.0; 0/ \text{ y } \left(*A, \sqrt{.A+2/.A*2/}; 0\right)$$

de orden 2, y un punto $.1; 1/$ de orden 4. Entonces,

$$}E_{M;A;A+2}.k/ \text{ ’ } 0 \ .\text{mod } 8/$$

Además, $E_{M;A;B}$ es un twist cuadrático de $E_{M;A;A+2}$, por lo que

$$}E_{M;A;B} + }E_{M;A;A+2} = 2.}k/ + 2 \text{ ’ } 0 \ .\text{mod } 8/$$

Por lo tanto,

$$}E_{M;A;B}.k/ \text{ ’ } 0 \ .\text{mod } 8/$$

Como la curva $E_{M;A;B}$ no puede tener más de tres puntos de orden 2, debe tener uno de orden 4.

En cualquier caso $E_{M;A;B}$ tendría un punto de orden 4. Usando la proposición anterior tenemos que $E_{M;A;B}$ es birracionalmente equivalente a una curva de Edwards.

Proposición 4.3. Sea k un cuerpo nito con }k ’ 1 .mod4/. Sea $E_{M;A;B}$ una curva de Montgomery tal que .A + 2/.A * 2/ sea un cuadrado. Tomamos un Ë k no cuadrado. Entonces, o bien $E_{M;A;B}$ o bien su torsión cuadrática no trivial $E_{M;A;\ B}$ es birracionalmente equivalente a una curva de Edwards.

En particular, $E_{M;A;A+2}$ es birracionalmente equivalente a una curva de Edwards.

Demostración. Como .A + 2/.A * 2/ es un cuadrado tanto $E_{M;A;B}$ como $E_{M;A;\ B}$ contienen un subgrupo isomorfo a Z_2Z • Z_2Z.

Este subgrupo supone un factor de 4 en el orden del grupo. Además,

$$\}E_{M;A;B}.k/ + \}E_{M;A;\ B}.k/ = 2.\}k/ + 2 \text{ ’ } 4 \text{ .mod } 8/$$

Por consiguiente, solo uno de los sumandos, $\}E_{M;A;B}.k/$ o $\}E_{M;A;\ B}.k/$, puede ser divisible por 4 y no por 8. De aquí sigue que solo una curva puede tener un punto de orden 4.

Usando este hecho con la proposición 4.1 obtenemos el primer resultado.

Para obtener el segundo resultado solo tenemos que hacer una pequeña consideración. Dado que el punto .1; 1/ tiene orden 4 en la curva $E_{M;A;A+2}$, usando de nuevo la proposición 4.1 concluimos la demostración.

Con todos los resultados tenemos una imagen general de las curvas de Edwards, no obstante, para tener una imagen más completa de las curvas de Edwards torcidas introducimos en la siguiente observación la fórmula de suma.

Observación 4.7. La suma de curvas de Edwards torcidas $E_{E;a;d}$ viene dada por:

$$+ : .x_1; y_1/; .x_2; y_2/ \text{ ;™ } \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}; \frac{y_1y_2 * ax_1x_2}{1 * dx_1x_2y_1y_2}\right)$$

El elemento neutro es el .0; 1/ y el opuesto de .x; y/ es . * x; y/.

Observación 4.8. Dado que la fórmula de suma coincide con la fórmula de suma de Edwards dada por la fórmula 4.1 para $x^2 + y^2 = 1 + .d\_a/x^2y^2$ con $x = \frac{u}{a}x$, se tiene las mismas características probadas anteriormente para fórmula de suma de Edwards.

5 La firma del protocolo: EdDSA

Una vez establecidos los fundamentos teóricos comenzaremos describiendo en profundidad el algoritmo de □rma digital introducido por Bernstein, Duif, Lange, Schwabe y Yang en [8], nuestra tercera publicación principal.

Este algoritmo es una variación del algoritmo clásico introducido por ElGamal en [13] cuya diferencia principal es el uso de curvas de Edwards, de ahí el nombre EdDSA (*Edwards-curve Digital Signature Algorithm*).

Iremos desgranando el algoritmo en diferentes partes. Comenzaremos enumerando los parámetros del mismo en la Sección 5.1. Posteriormente veremos cómo obtener la clave pública a partir de la privada en la Sección 5.2. Seguiremos en las Secciones 5.3 y 5.4 con los procesos de □rma y veri□cación respectivamente. Para concluir, mostraremos en pseudocódigo las principales componentes del algoritmo (Sección 5.5).

5.1 Parámetros de EdDSA

Los parámetros son los siete siguientes:

- entero $b \geq 10$.
- función hash H con salida de $2b$ bits.
- primo q de la forma $2^{255} - 19$ o $2^{448} - 2^{224} - 1$.
- una codi□cación de elementos del cuerpo □nito F_q de b bits.
- $d \in F_q$ no cuadrado.
- elemento $B \neq (0, 1)$ del conjunto $E = \{(x, y) \in F_q \times F_q : -x^2 + y^2 = 1 + dx^2y^2\}$
- primo l entre 2^{b-1} y 2^{b-3} tal que $lB = 0$.

Codificación: Un elemento de la curva (x, y) queda codi□cado con una cadena de

b bits, donde .b * 1/ codi□can y y el bit restante es el de signo de x (1 si x es negativa). De este modo, x queda determinado por $x = \pm\sqrt{\frac{y^2 * 1}{dy^2 + 1}}$. Usaremos el subrayado para denotar la codi□cación de un punto. Por ejemplo, si A = .x; y/ punto de la curva, $\underline{A}$ denotaría la cadena de b bits que codi□ca .x; y/.

5.2 Claves privada y pública

La **clave privada** de EdDSA es una cadena k de b bits. El hash H .k/ = .h_0; h_1; :::; h_{2b*1}/ determina un entero

$$a = 2^{b*2} + \sum_{3fifb*3} 2^i h_i \quad Ë ^2^{b*2}; 2^{b*2} + 8; :::; 2^{b*1} * 8`$$

que a su vez determina el múltiplo A = aB. La correspondiente **clave pública** es A.

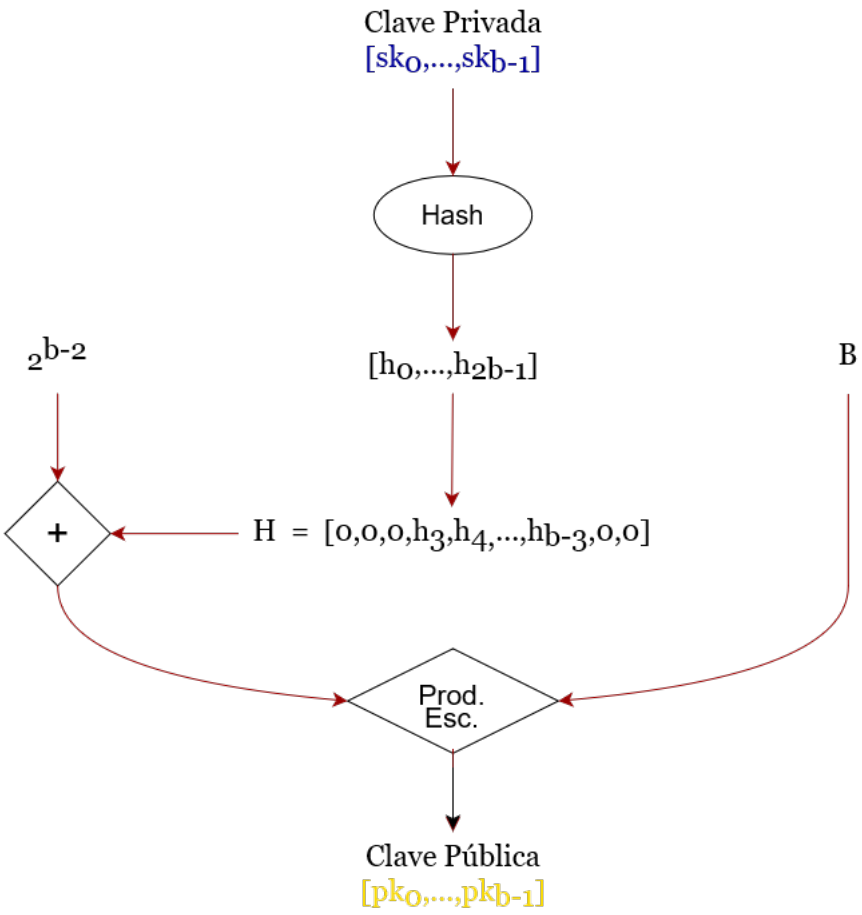

Figura 5.1: Diagrama clave pública EdDSA

5.3 Firma

La □rma de un mensaje M con la clave privada k se de□ne como sigue:

1. De□nimos r = H .h_b; :::; h_{2b*1}; M / Ë ^0; 1; :::; 2^{2b} * 1`. Interpretamos las cadenas de 2b bits (*little-endian*) como enteros entre 0 y 2^{2b} * 1.
2. De□nimos R = rB.
3. De□nimos S = .r + H .R; A; M /a/ mod l.
4. Tomamos la cadena de 2b bits .R; S/ como la □rma de M usando k.

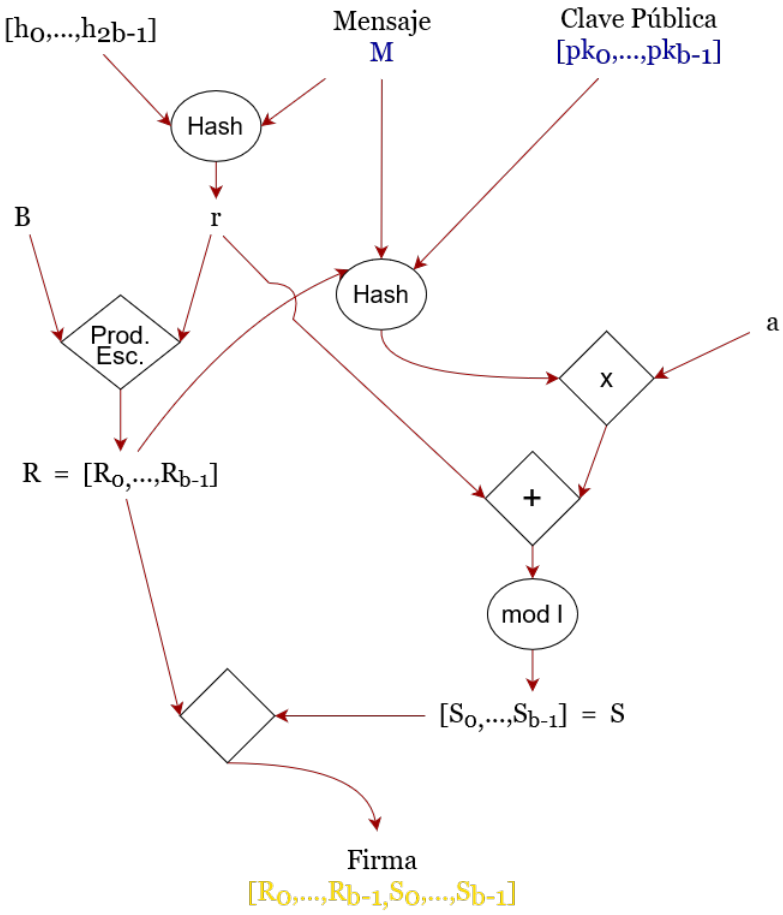

Figura 5.2: Diagrama □rma EdDSA

5.4 Verificación

Para veri□car nuestro mensaje M con la clave pública A hacemos lo siguiente:

1. Tomamos la clave pública como un punto de la curva E, i.e., A Ë E.
2. De la □rma .R; S/ tomamos R Ë E y S Ë ^0; 1; :::; l * 1`.
3. Calculamos H .R; A; M /.
4. Comprobamos en la ecuación SB = R + H .R; A; M /A en E. Rechazamos si no se da la igualdad.

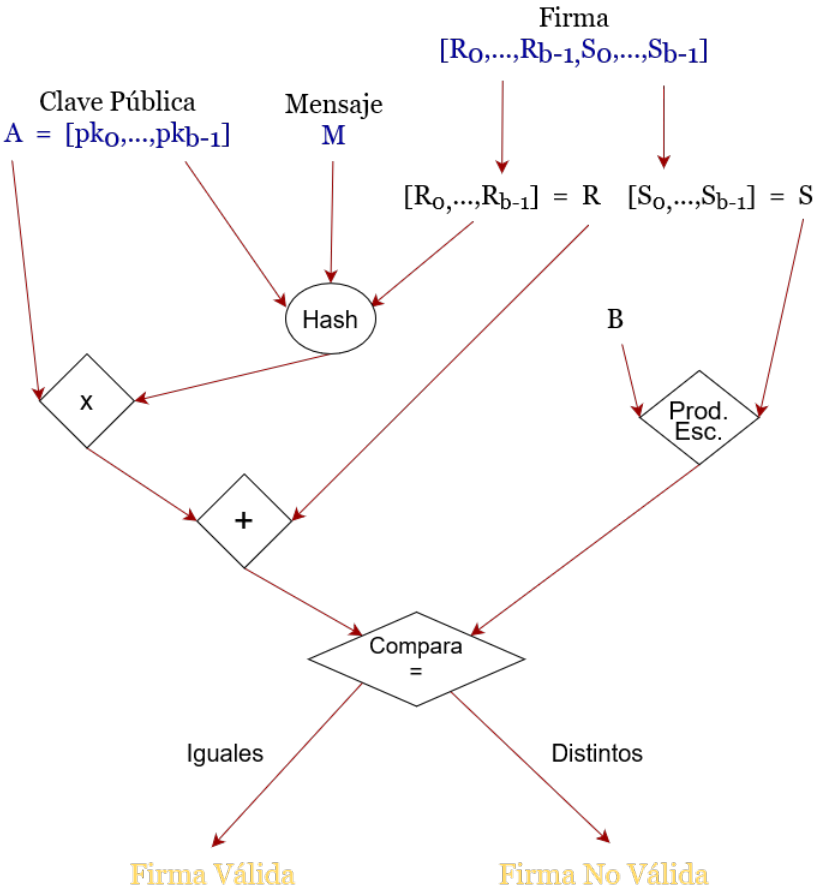

Figura 5.3: Diagrama veri□cación EdDSA

5.5 Pseudocódigo

A continuación, usamos pseudocódigo para aclarar la descripción formal de EdDSA. Los algoritmos principales serían el de firma (Algoritmo 3) y el de verificación (Algoritmo 4). Comenzamos con la definición de la Suma de Edwards y la obtención de la clave pública.

Algoritmo 1: Suma De Edwards (*SumaDeEdwards*)

Datos: P y Q puntos de la curva
Resultado: R resultado de sumar P y Q
$(x_1, y_1) \leftarrow P$;
$(x_2, y_2) \leftarrow Q$;
$R \leftarrow \left(\frac{x_1 y_2 + y_1 x_2}{c(1 + d x_1 x_2 y_1 y_2)}, \frac{y_1 y_2 * x_1 x_2}{c(1 * d x_1 x_2 y_1 y_2)} \right)$;

Algoritmo 2: Clave Pública

Datos: sk clave privada, b tamaño bits, B punto de la curva
Resultado: pk clave pública
$h \leftarrow Hash(k)$;
$a \leftarrow 2^{b*2}$;
para $i \leftarrow 3$ **a** $b * 3$ **hacer**
  $a \leftarrow a + 2^i h_i$;
fin
$A \leftarrow aB$;
$pk \leftarrow Codifica(A)$;

Seguidamente, vemos como firmar.

Algoritmo 3: Firma EdDSA

Datos: M mensaje, sk clave privada, pk clave pública, b tamaño bits, B punto de la curva, l primo
Resultado: sign firma digital
h } Hash.sk/;
a } 2^{b*2};
para i } 3 **a** b * 3 **hacer**
  a } a + $2^i h_i$;
fin
r } Hash.h_b; :::; h_{2b*1}; M/;
R } rB;
H } Hash.Codifica.R/; pk; M/;
S } .r + Ha/ mod l;
sign } Concatenar.Codifica.R/; Codifica.S//;

Finalmente, vemos como verificar la firma.

Algoritmo 4: Verificación EdDSA

Datos: M mensaje, sign firma, pk clave pública, b tamaño bits, B punto de la curva
Resultado: val validez de la firma
R } b primeros bits de sign;
S } b últimos bits de sign;
H } Hash.R; pk; M/;
ls } Decodifica.S/ < B;
rs } SumaDeEdwards.Decodifica.R/; H < Decodifica.A//;
si ls = rs **entonces**
  val } FIRMA VÁLIDA
en otro caso
  val } FIRMA NO VÁLIDA
fin

6 Una implementación: Ed25519

En el capítulo anterior hemos introducido a nivel formal el algoritmo de firma digital EdDSA. Se han estudiado los diferentes parámetros y los requisitos que deben cumplir. Sin embargo, el interés práctico también es importante por lo que vamos a implementar una versión del mismo. La correcta implementación no solo nos permitirá usar la firma, sino que demostrará el entendimiento de los conceptos hasta ahora discutidos.

Como hicimos a nivel formal, haremos una descripción inicial de los parámetros de EdDSA (Sección 6.1). Proseguiremos con la implementación en la Sección 6.2 de la funciones auxiliares, las funciones de codificación y decodificación además de la declaración de los parámetros descritos anteriormentes. Las implementación de las tres partes fundamentales del algoritmo, la generación de claves, la firma y la verificación se verán en las Secciones 6.3, 6.4 y 6.5 respectivamente.

Se conoce como Ed25519 a la implementación de EdDSA con una curva de Edwards torcida birracionalmente equivalente a *Curve25519*, curva introducida por Bernstein en [6]. Notemos que cualquier equivalencia birracional preserva la complejidad del problema del logaritmo discreto para curvas elípticas. Esta implementación es la que más éxito ha tenido. Una lista actualizada de las aplicaciones que ha tenido Ed25519 puede encontrarse en [2]. No solo tenemos de referencia [8] para la implementación sino también la librería *ed25519* [1] y la web de *ed25519* de Daniel J. Bernstein [5].

6.1 Parámetros

La elección de parámetros es la siguiente:

- entero b = 256
- función hash H = SHA * 512
- potencia prima q = 2^{255} * 19 ' 1 .mod4/
- ^0; 1; :::; 2^{255} * 20` como codi□cación de elementos del cuerpo □nito F_q de 255 bits (*little-endian*).
- d = * $\frac{121665}{121666}$ Ë F_q no cuadrado.
- único elemento B = .x; 4_5/ del conjunto E = ^.x; y/ Ë F_q • F_q : * $x^2 + y^2$ = 1 + d x^2y^2 ` para el que x es positiva.
- primo l = 2^{252} + 27742317777372353535851937790883648493 entre 2^{b*1} y 2^{b*3} tal que lB = 0.

Curve25519 es la curva de Montgomery sobre F_q de□nida por:

$$v^2 = u^3 + 486662u^2 + u$$

que es birracionalmente equivalente a la curva de Edwards torcida:

$$x^2 + y^2 = 1 + \frac{121665}{121666}x^2y^2$$

con equivalencia

$$.u; v/ ;™ . \grave{u}\sqrt{\frac{}{486664}}\frac{u}{v}; \frac{u * 1}{u + 1}/$$

Notemos que como *1 es un cuadrado en F_q la curva de Edwards torcida es isomorfa a

$$*x^2 + y^2 = 1 * \frac{121665}{121666}x^2y^2$$

6.2 Implementación

Una vez establecidos los parámetros que vamos a usar, pasamos a implementar el algoritmo. Iremos paso por paso, aclarando las técnicas utilizadas para la resolución de los problemas encontrados.

Vamos a implementarlo en Python 3.10.11. Se ha optado por una implementación en Notebook de Google Colab, dada la facilidad de creación de código en los mismos. Podemos encontrarlo en [3].

Empezamos discutiendo cuestiones sobre el desarrollo del código. Debido que estamos trabajando con bits usaremos la clase predefinida de Python *bytes* a lo largo de toda la implementación. Comenzamos definiendo los parámetros b, q y l. No definimos aún d y B ya que para la división usaremos la multiplicación por inverso en F_q. Definiremos una función que devolverá el inverso usando la siguiente propiedad algebraica de la curva: el inverso se obtiene elevando a q * 2 [6].

Obtener el módulo de una potencia en un cuerpo finito no es trivial. Para resolver este problema usaremos exponenciación modular rápida.

6.2.1 Exponenciación modular rápida

Este método de exponenciación modular es el más eficaz [11, Sección 11.3]. Supongamos que buscamos x tal que

$$x \equiv b^e \bmod p$$

Primeramente, pasamos nuestro exponente a base binaria. Sea n el número de bits necesarios para escribir e, es decir, $n = \lfloor \log_2 e \rfloor + 1$.

$$e = \sum_{i=0}^{n-1} a_i 2^i$$

donde $a_i \in \{0, 1\}$. Trivialmente $a_{n-1} = 1$. Con el exponente en notación binaria podemos escribir la exponenciación del siguiente modo

$$b^e = b^{\sum_{i=0}^{n-1} a_i 2^i} = \prod_{i=0}^{n-1} b^{2^i a_i}$$

por lo que tomando módulo obtenemos c

$$c \equiv \prod_{i=0}^{n-1} b^{2^i a_i}$$

Una vez aclarada la exponenciación modular rápida pasamos a implementarla. Usaremos un enfoque recursivo.

```
def ExpMod(base, exp, mod):
    '''Realiza la operacion base^exp (modulo mod)'''

    res = 1
    if exp == 0: #Caso base
        return res

    else: #Caso general
        res = ExpMod(base, exp//2, mod)**2 % mod
        if exp & 1:
            res = (res*base) % mod

    return res
```

Con esta función podemos de□nir la función inverso que necesitamos del siguiente modo

```
def Inv(x):
    '''Dado un elemento del grupo finito devuelve su inverso'''

    return ExpMod(x,q-2,q)
```

Ya podríamos de□nir d. Para de□nir B necesitamos una función que dado el valor de la componente y nos devuelva la componente x. Vimos una fórmula explícita en el capítulo anterior. Necesitaremos calcular una raíz cuadrada modular, veamos como hacerlo.

6.2.2 Raíz cuadrada modular

La resolución de una raíz cuadrada modular no es trivial. En nuestro caso, el primo q es congruente con 5 módulo 8 por lo que cualquier Ë F_q cuadrado satisface $^2 = ^4$ donde $= \frac{q+3}{8}$. Se tiene entonces que $^2 = ,$. Para obtener el resultado de la raíz únicamente tendríamos que comprobar ^{2}. Si $^2 = *$ solo tendríamos que multiplicar por $*1$. [8]

```
from math import sqrt

I = ExpMod(2,(q-1)//4,q)

def ObtenXdeY(y):
```

```
    '''Usando las propiedades algebraicas de la curva, obtiene el
    valor de x a partir de y'''

    #Calculamos la raiz como una potencia
    xcuad = (y*y - 1) * Inv(d*y*y + 1)
    x = ExpMod(xcuad, (q+3)//8, q)

    if (x*x - xcuad) % q != 0:
        x = (x*I) % q

    if x & 1 == 0:
        x = q - x

    return x
```

Usando la función ObtenXdeY podemos calcular B.

```
By = 4 * Inv(5)
Bx = ObtenXdeY(By)
B = [Bx % q,By % q]
```

Una vez hecho esto, de□niremos funciones relacionadas con la curva, una para comprobar si un punto pertenece a la misma y otra para realizar la suma de Edwards como vimos en el Capítulo 4.

```
def PerteneceCurva(P):
    '''Comprueba si el punto pertenece a la curva de Edwards'''

    x,y = P
    return (-x*x + y*y - 1 - d*x*x*y*y) % q == 0
```

```
def SumaEdwards(P,Q):
    '''Devuelve la suma de Edwards de dos puntos P y Q'''

    x1,y1 = P
    x2,y2 = Q
    x3 = (x1*y2+x2*y1) * Inv(1+d*x1*x2*y1*y2)
    y3 = (y1*y2+x1*x2) * Inv(1-d*x1*x2*y1*y2)

    return x3 % q, y3 % q
```

También necesitaremos una función que realice el producto escalar. Al igual que para la exponenciación modular usaremos un enfoque recursivo para implementar la

misma.

```
def MultEsc(e, P):
    '''Realiza la multiplicacion escalar eP'''

    Q = (0,1)
    if e == 0: #Caso base
        return Q

    else: #Caso general
        Q = MultEsc(e//2, P)
        Q = SumaEdwards(Q,Q)

        if e & 1:
            Q = SumaEdwards(P, Q)

    return Q
```

Para ayudarnos en la implementación del algoritmo vamos a definir funciones auxiliares que nos transformarán listas de bits en *bytes* o nos obtendrán el bit en una posición determinada en un objeto *bytes*.

```
def BitsABytes(bits):
    '''Pasa de una lista de bits a un objeto bytes'''

    return bytes(sum(2**i * bits[j*8 + i] for i in range(8)) for
   j in range(b//8))
```

```
def ObtenBitPos(bytes_obj, pos):
    '''Obtiene el bit en la posicion pos de una objeto bytes'''

    byte = bytes_obj[pos//8]

    return (byte >> (pos % 8)) & 1
```

```
from hashlib import sha512

def Hash(mensaje):
    '''Toma una mensaje y le hace el hash SHA-512'''

    return sha512(mensaje).digest()
```

Seguidamente definimos las funciones de codificación y decodificación. Recordamos que estamos interpretando las cadenas de bits como enteros.

```
def CodificaEntero(e):
    '''Dado un entero lo codifica como objeto bytes de tamanno b//8'''

    return  BitsABytes([(e >> i) & 1 for i in range(b)])
```

```
def CodificaPunto(P):
    '''Dado un punto devuelve la codificacion'''
    x,y = P
    bits =  [(y >> i) & 1 for i in range(b - 1)] + [x & 1] #vemos si x negativo

    return BitsABytes(bits)
```

```
def DecodificaEntero(bytes_obj):
    '''Devuelve un objeto byte interpretado como un entero'''

    return sum(256**i * bytes_obj[i] for i in range(len(bytes_obj)))
```

```
def DecodificaPunto(bytes_obj):
    '''Dado un objeto bytes devuelve el punto asociado'''

    y = sum(2**i * ObtenBitPos(bytes_obj,i) for i in range(b-1))
    x = ObtenXdeY(y)
    if x & 1 != ObtenBitPos(bytes_obj,b-1): #si no coinciden los signos, negamos
        x = q-x

    P = (x,y)

    if not PerteneceCurva(P):
        raise Exception("El punto decodificado no pertenece a la curva")

    return P
```

Con todo esto estamos preparados para implementar EdDSA.

6.3 Implementación de la Clave Pública

Siguiendo el esquema introducido en el anterior capítulo implementamos la siguiente función. Dado que posteriormente para la rma usaremos el hash y a, los

devolvemos junto a la clave pública para no tener que calcularlos de nuevo más adelante.

```
def ClavePublica(clave_priv):
    '''Dada una clave privada de nuestra eleccion, devuelve la clave publica asociada, el hash y el escalar a'''
    bytes_clave_priv = bytes(clave_priv, 'utf-8')

    #Calculamos el hash de la clave privada
    hash = Hash(bytes_clave_priv)

    #Calculamos el escalar asociado
    a = 2**(b-2)+ sum(2**i * ObtenBitPos(hash,i) for i in range(3,b-2))

    #Obtenemos la clave publica multiplicando a y B
    A = MultEsc(a,B)

    return CodificaPunto(A), hash, a
```

6.4 Implementación de la Firma

Del mismo modo, implementamos la función Firma siguiendo los pasos del algoritmo.

```
def Firma(mensaje,clave_priv):
    '''Firma el mensaje con la clave privada dada'''
    bytes_mensaje = bytes(mensaje,'utf-8')

    #0. Obtenemos la clave publica, el hash y el escalar a
    clave_publ, hash, a = ClavePublica(clave_priv)

    #1. Definimos r
    r = DecodificaEntero(Hash(bytes(ObtenBitPos(hash,i) for i in range(b,2*b)) + bytes_mensaje))

    #2. Definimos R
    R = MultEsc(r, B)

    R_cod = CodificaPunto(R)

    #3. Definimos S
    hash_aux = Hash(R_cod + clave_publ + bytes_mensaje)
```

```
    S = (r + DecodificaEntero(hash_aux) * a) % l

    S_cod = CodificaEntero(S)

    #4. Devolvemos la firma
    return R_cod + S_cod
```

6.5 Implementación de la Verificación

Para finalizar el algoritmo, necesitamos una función que verifique si la firma es válida. Análogamente, seguimos los pasos introducidos en el capítulo anterior.

```
def Verifica(mensaje, firma, clave_publ):
    '''Verifica que la firma del mensaje es correcta para la
    clave publica'''

    bytes_mensaje = bytes(mensaje,'utf-8')

    #1. Obtenemos A como punto de la curva
    A_cod = clave_publ
    A = DecodificaPunto(A_cod)

    #2. Tomamos R y S de la firma
    R_cod = firma[0:b//8] #los primeros b bits
    R = DecodificaPunto(R_cod)

    S = DecodificaEntero(firma[b//8:b//4]) #los ultimos b bits

    #3. Calculamos el Hash
    hash = DecodificaEntero(Hash(R_cod + A_cod + bytes_mensaje))

    #4. Verificamos la ecuacion
    if MultEsc(S,B) != SumaEdwards(R, MultEsc(hash, A)):
        return "!!!!! LA FIRMA NO ES VALIDA !!!!!"

    return "La firma es valida"
```

Tanto en la firma como en la verificación hemos introducido el constructor de *bytes* para que trabajemos con strings.

6.6 Pruebas

Hagamos unas pequeñas pruebas para comprobar que tenemos el funcionamiento esperado. La primera prueba será tomar una clave privada y un mensaje, y □rmarlos. Posteriormente, veri□camos que la □rma es correcta.

```
#Definimos nuestra clave privada
clave_privada = 'Esta es mi clave privada'

#Obtenemos la clave publica asociada
clave_publica,_,_ = ClavePublica(clave_privada)

#Definimos nuestro mensaje y lo firmamos con la clave privada
mensaje = 'Mensaje para firmar'
firma = Firma(mensaje,clave_privada)

#Verificamos
Verifica(mensaje, firma, clave_publica)
```

En efecto, nos indica que nuestra □rma es válida. Nuestra segunda prueba consiste en añadir al mensaje un carácter. Imaginemos que recibimos el mensaje alterado con la □rma anterior. A la hora de veri□car, veríamos que la □rma es válida.

```
mensaje2 = 'Mensaje para firmar '

Verifica(mensaje2, firma, clave_publica)
```

7 El protocolo Signal

El protocolo Signal sirve de inspiración para el desarrollo de este trabajo. La criptografía con curvas de Edwards es fundamental, sin embargo, son necesarias otras herramientas para tener el protocolo completo. Iremos desgranando cada una a lo largo del presente capítulo.

El objetivo de Signal es crear un protocolo lo más completo posible. El primer problema al que busca solución es a la comunicación asíncrona. No solo queremos que la transmisión de los mensajes sea segura, sino que necesitamos que no sea necesario que ambas partes estén presentes al mismo tiempo. Por ejemplo, Alice podría enviar un mensaje cuando Bob no estuviese disponible y Bob podría responder más adelante. Este comportamiento deseable supone, sin embargo, un reto en el intercambio de claves. Además, plantea otras necesidades como la resistencia de los mensajes tanto a pasado como a futuro, i.e., si desciframos un mensaje de la conversación no debemos ser capaces de obtener los mensajes anteriores o posteriores. Efectivamente, si Alice puede tardar un par de meses en contestar a Bob, cualquier potencial agente externo malicioso tendría su□ciente tiempo para descifra un mensaje, por lo que es necesario que aunque uno sea descifrado no lo sean los demás.

Estos retos no son de ningún modo triviales. A continuación, veremos qué herramientas emplea el protocolo Signal para obtener el resultado deseado. La primera de ellas, X3DH, resuelve el intercambio de claves inicial. La estudiaremos en profundidad en al Sección 7.1. La segunda herramienta es el algoritmo de Double Ratchet cuyo funcionamiento será descrito en la Sección 7.2.

7.1 X3DH

El primer problema que solucionaremos será el intercambio de claves usando el algoritmo X3DH (*Extended Triple Di e-Hellman*). Partimos del conocido algoritmo de intercambio de claves de Di□e-Hellman y realizamos algunas modi□caciones para adaptarlo a intercambios asíncronos.

En efecto, el protocolo Di□e-Hellman funciona bien para intercambiar un secreto sobre un canal inseguro, sin embargo su comportamiento es totalmente síncrono. Ambos agentes deben estar conectados en la comunicación.

Como hemos mencionado antes, queremos cambiar este hecho. Sería deseable que el secreto pudiese ser acordado sin ambas partes presentes. Asimismo, la presencia de un servidor central hace que debamos añadir una capa de seguridad para que tengamos con□dencialidad. No debemos con□ar en el servidor (posibles ataques MITM), y este no debería conocer el secreto entre Alice y Bob. En este servidor Alice publicará una información que Bob podrá usar más adelante para intercambiar el secreto.

Veamos como X3DH nos proporciona este intercambio de claves avanzado. Comenzamos describiendo los elementos del algoritmo.

7.1.1 Elementos del algoritmo

Imaginemos un intercambio entre Alice (A) y Bob (B). Las claves públicas que aparecen son las siguientes:

- Claves de identidad: ik_A, ik_B.
- Claves efímeras: ek_A,ek_B.
- Claves previas: sk_A, sk_B. Son las claves de identidad que se □rmarán usando EdDSA.
- Claves previas de un solo uso: ok^i_A, ok^i_B. Se usan cada vez que se quiera iniciar una conversación.

Todas las claves públicas tienen su clave privada asociada. Respecto a las funciones usadas, tenemos las siguientes:

- *DH(x,y)*: función Di□e-Hellman. A partir de dos claves, *x* (privada) e *y* (pública) nos da un secreto compartido. Notamos que al usar esta función el otro agente estaría realizando *DH*(y^{*1},x^{*1}).
- *KDF(x)* (*Key Derivation Function*): A partir de una clave *x*, obtiene otra nueva mediante el uso de funciones trampilla.
- *E(x,y)*, *D(x,y)*: funciones de cifrado y descifrado respectivamente. Cifran el mensaje x con la clave y.
- *Sig(x,y)*: función de □rma digital. En nuestro caso, EdDSA. Firma el mensaje x con la clave y.

7.1.2 Funcionamiento del algoritmo

El algoritmo tiene tres fases. Imaginemos que Alice y Bob van a entablar una conversación. Primeramente, Bob debe entregar sus claves al servidor. Concretamente entrega:

- Su clave de identidad pública ik_B.
- Su clave previa sk_B. Esta se actualizará cada cierto intervalo de tiempo previamente establecido..
- La □rma de la clave previa *Sig*(sk_B, ik_B^{*1}).
- Un conjunto de claves previas de un solo uso $[ok_B^1; ok_B^2; :::]$. También se actualizan cada cierto tiempo.

Supongamos que Alice desea iniciar la conversación. Para ello, pide al servidor un paquete de claves de Bob que consiste en las anteriores claves pero tomando solo una previa de un solo uso, por ejemplo, ok_B^2. Una vez obtenido, se obtiene la clave común del siguiente modo (el operador ð es la concatenación):

1. K_1 = *DH*(ik_A^{*1}, sk_B)
2. K_2 = *DH*(ek_A^{*1}, ik_B)
3. K_3 = *DH*(ek_A^{*1}, sk_B)
4. K_4 = *DH*(ek_A^{*1}, ok_B^2)
5. K = *KDF*(K_1ðK_2ðK_3ðK_4)

Tras esto Alice conoce el secreto K por lo que ya puede borrar los K_i con $i = 1; 2; 3; 4$ y la clave ek^{*1}.

Ahora, la última fase de X3DH es que Bob reciba de Alice su clave pública y efímera, y el indicador de la de una solo uso escogida. A su vez, Alice envía $E(ik_A \eth ik_B, K)$ de forma que Bob pueda comprobar una vez obtenga el secreto K que el intercambio ha sido exitoso.

Una vez Bob recibe la información enviada por Alice obtiene las K_i, $i = 1; 2; 3; 4$ de la siguiente forma:

1. $K_1 = DH(sk_B^{*1}, ik_A)$
2. $K_2 = DH(ik_B^{*1}, ek_A)$
3. $K_3 = DH(sk_B^{*1}, ek_A)$
4. $K_4 = DH(.ok_B^2 /^{*1}, ek_A)$
5. $K = KDF(K_1 \eth K_2 \eth K_3 \eth K_4)$

Para □nalizar Bob debe comprobar que el intercambio ha sido correcto. Sencillamente, debe descifrar la comprobación $E(ik_A \eth ik_B, K)$ recibida anteriormente. Si coinciden las claves de identidad el intercambio ha sido exitoso, si no, ha habido algún problema y se aborta el intercambio por seguridad. A lo largo de esta sección nos hemos basado en [15].

7.2 Double Ratchet

El algoritmo de Double Ratchet (Doble Trinquete) es la solución a la criptografía de los mensajes, esto es, es el encargado de obtener las claves durante la comunicación. El ya estudiado X3DH se encargaría únicamente del intercambio inicial.

Como el nombre de Double Ratchet indica consiste en dos "trinquetes", Symmetric Ratchet (Sección 7.2.1) y Di□e-Hellman Ratchet (Sección 7.2.2). En mecánica, un trinquete es un tipo de engranaje que solo permite la rotación en un sentido (se bloquea en el otro). Análogamente, el algoritmo tiene dos componentes que funcionan como trinquetes y permiten la gestión de las claves con las que se cifran los mensajes de forma segura.

7.2.1 Symmetric Ratchet

El primer trinquete es el Symmetric Ratchet (trinquete simétrico) compuesto por el Sending Ratchet y Receiving Ratchet. Este trinquete es el encargado del cifrado y descifrado de los mensajes. El Sending Ratchet o trinquete de envío, gestiona las claves de los mensajes enviados y el Receiving Ratchet o trinquete de recepción hace lo mismo con las de los mensajes recibidos.

Estos trinquetes son funciones de derivación de claves (*Key Derivation Function*) donde la clave salida se reúsa como entrada de la *KDF* obteniendo una nueva clave. A nivel práctico, los trinquetes simétricos de Alice y Bob están sincronizados. Partiendo de la clave inicial obtenida por X3DH, cada mensaje va cifrado con una clave distinta. Esa clave se pasa por la *KDF* y se obtiene la clave para el mensaje siguiente, y así sucesivamente. De este modo, se resuelve el problema de resistencia a pasado. Si un mensaje se intercepta y se obtiene la clave, no pueden obtenerse los mensajes anteriores ya que la *KDF* es una función trampilla. No obstante, sí seguiríamos con el problema de resistencia futura, pues una vez obtenida la clave pueden obtenerse las siguientes ya que la función de derivación de claves es conocida. Para solucionar esto, entra en juego el otro trinquete [17].

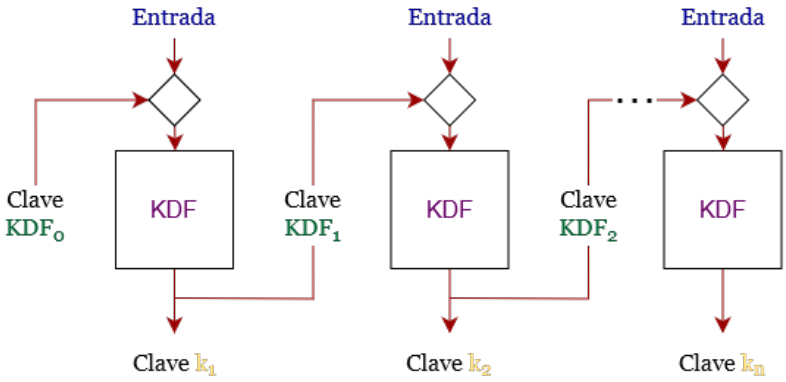

Figura 7.1: Esquema básico de una función de derivación de claves *KDF*

7.2.2 Diffie-Hellman Ratchet

El Di□e-Hellmann Ratchet o trinquete Di□e-Hellman consiste, como su nombre indica, en un intercambio de claves Di□e-Hellman. Cada cierto tiempo establecido (en la práctica, cada mensaje) se produce un intercambio de claves que reinicia el

trinquete simétrico. De este modo, se soluciona la resistencia futura, pues no hay forma de obtener a partir de un mensaje descifrado la clave que se obtendrá en el intercambio Di□e-Hellman y por tanto, no se puede obtener la salida de la *KDF*.

A continuación describiremos el algoritmo mediante diagramas para aclarar su funcionamiento.

Paso 0

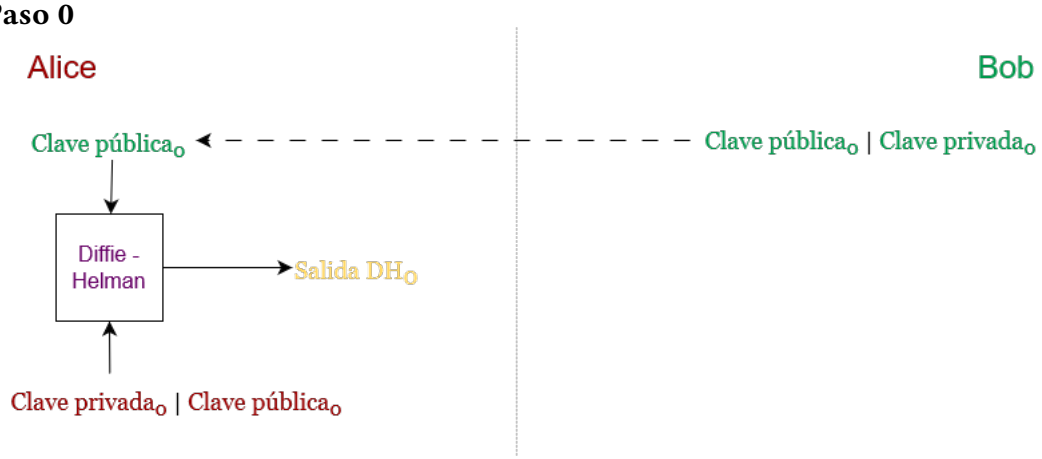

Figura 7.2: Bob envía su clave pública a Alice y desconoce la clave pública de Alice.

Paso 1

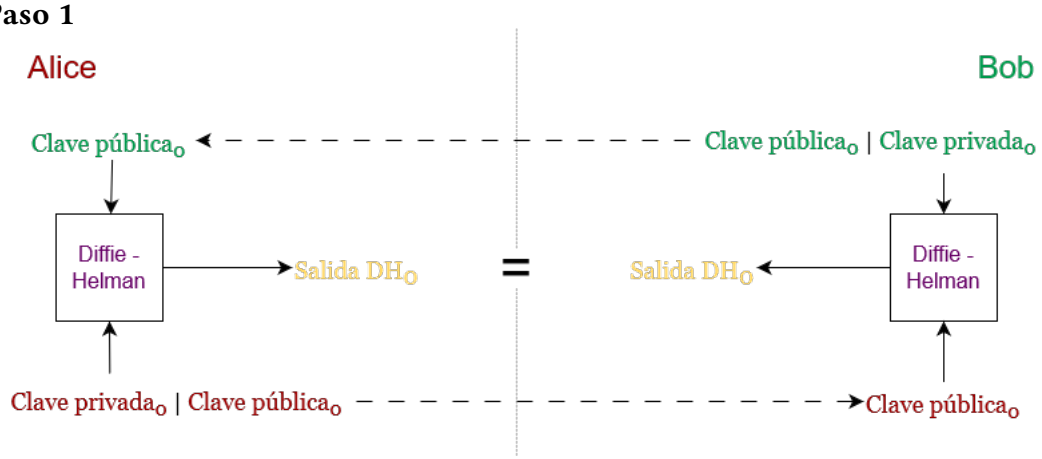

Figura 7.3: Bob recibe la clave pública de Alice y la usa para el intercambio DH.

Paso 2

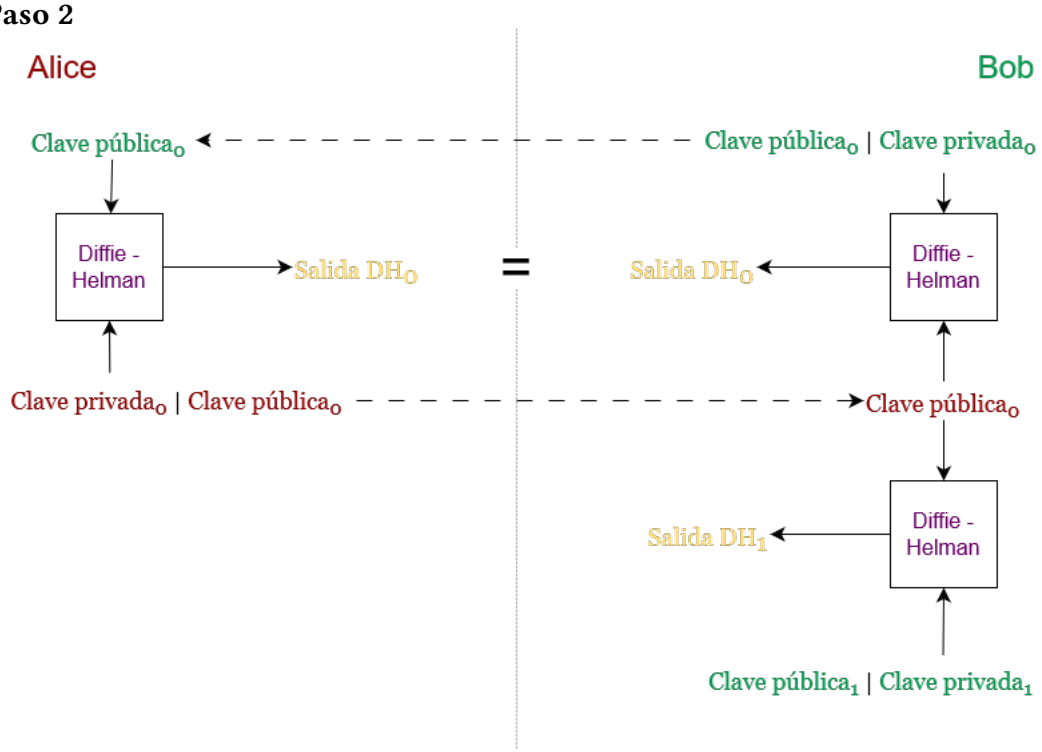

Figura 7.4: Bob toma un nuevo par de claves y calcula con la clave pública de Alice un nuevo DH.

Pasos sucesivos

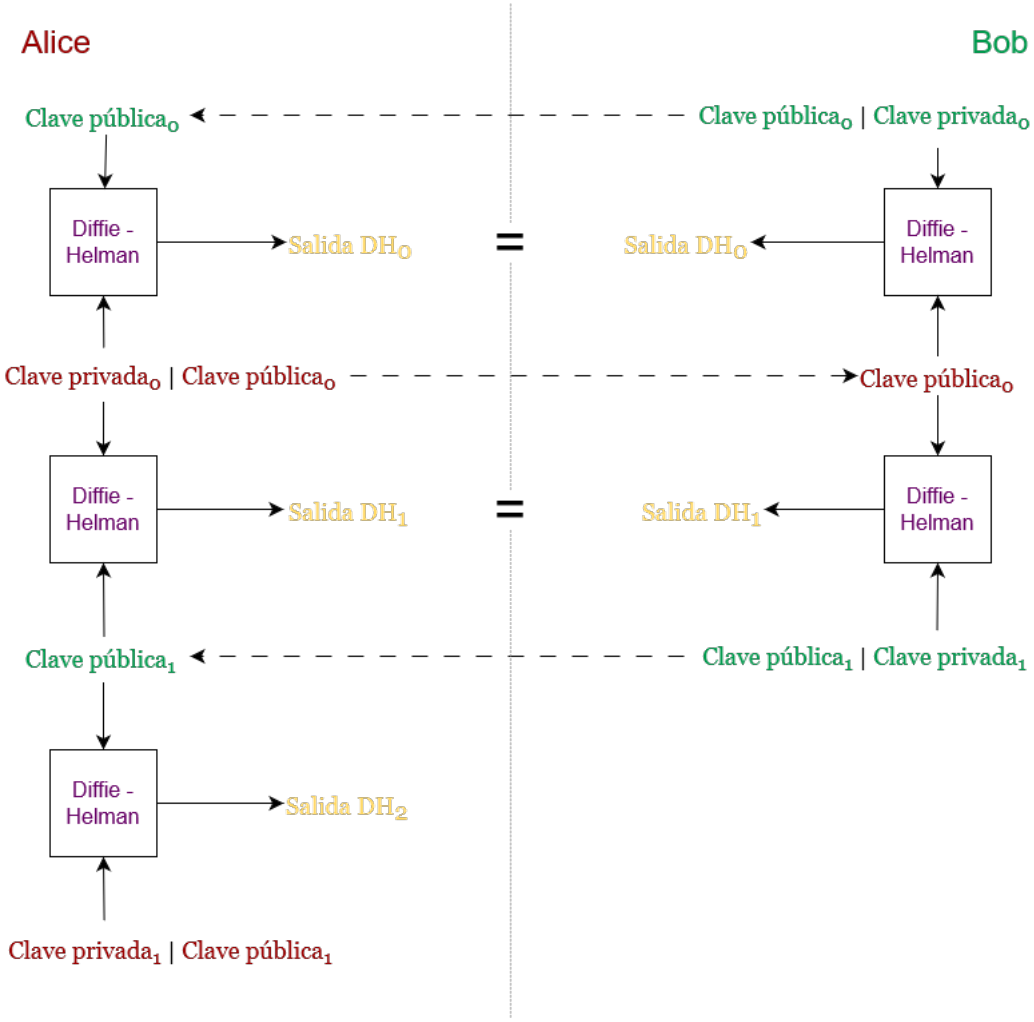

Figura 7.5: Alice genera un nuevo par de claves. Se van sucediendo los pasos anteriores.

Antes de concluir con el trinquete Di□e-Hellman debemos realizar una aclaración. En los diagramas encontramos como Alice y Bob se van turnando para obtener las mismas salidas de los intercambios Di□e-Hellman. No obstante, existe otra vuelta de tuerca. Tanto Alice como Bob pasarán las salidas DH a través de una cadena común de funciones de derivación de claves del siguiente modo:

KDF en DH Ratchet

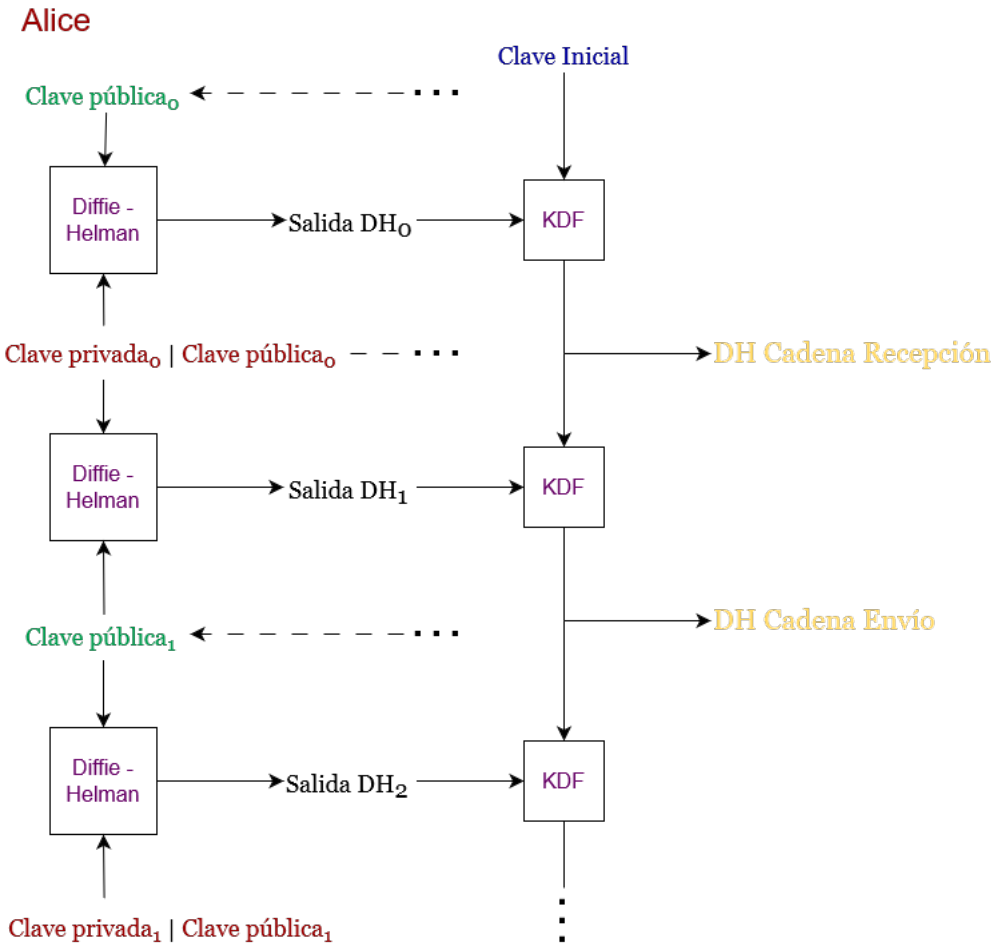

Figura 7.6: Cadena de KDF en el lado de Alice. Análoga para Bob

En el diagrama se observa como Alice obtendría sus claves para cadenas de envío y de recepción. Al partir del mismo secreto inicial y de las mismas salidas DH como vimos en las anteriores □guras, tanto Alice como Bob obtienen la misma clave. La clave para cadena de envío en un agente es la de recepción en el otro y viceversa.

8 Análisis temporal y de costes

En este capítulo estudiaremos las diversas tareas que se han realizado, así como los costes de las mismas y el tiempo dedicado.

Los roles necesarios han sido *Jefe de Proyecto*, *Analista*, *Desarrollador*, y *Tester*. A pesar de que todos han sido realizados por el alumno, se incluyen para poder ejecutar un análisis más preciso.

8.1 Roles

Roles
Jefe de Proyecto
Analista
Tester
Desarrollador

8.2 Tareas

Para la realización del proyecto se han realizado numerosas tareas de diferente complejidad. Se ha seguido un enfoque ordenado y constructivo para llegar al resultado final, delimitando claramente los pasos a dar.

1. **Iniciación del trabajo** (*Jefe de Proyecto*) En esta primera fase se planteó el tema a estudiar y los objetivos deseados. En concreto,

1.1. Se decidió que el tema estaría relacionado con el protocolo Signal como modo de integrar ambas disciplinas.
1.2. Se buscaba hacer un estudio teórico matemático que tuviese posibilidad de implementación.
1.3. Partiendo del protocolo Signal, se decidió centrarse en la □rma digital usada por el interés matemático del estudio de las curvas de Edwards.
1.4. Dado que el protocolo Signal está implementado en Rust y que la principal implementación hecha por los investigadores de la □rma está en C, se consideró interesante hacer una implementación en Python por interés didáctico.

2. **Estudio matemático previo** (*Analista*).
   2.1. Estudio de la □rma digital.
   2.2. Estudio de las Curvas Elípticas.
   2.3. Estudio de la □rma ECDSA.
3. **Estudio curvas de Edwards** (*Analista*).
   3.1. Estudio de las Curvas de Edwards (primer *paper*, [10]).
   3.2. Estudio de las Curvas de Edwards torcidas (segundo *paper*, [7]).
   3.3. Demostraciones de los teoremas expuestos.
4. **Estudio del algoritmo EdDSA** (*Analista*)
   4.1. Estudio del tercer *paper*, [8].
   4.2. Creación del pseudocódigo.
5. **Estudio del protocolo Signal** (*Analista*)
   5.1. Estudio de X3DH.
   5.2. Estudio de Double Ratchet.
   5.3. Creación de diagramas.
6. **Implementación** (*Desarrollador*)
   6.1. Estudio de los problemas encontrados al implementar en Python
   6.2. Implementación
   6.3. Documentación en el trabajo
7. **Pruebas** (*Tester*) Se realizaron algunas pruebas sobre el algoritmo para asegurar su comportamiento deseado.
8. **Elaboración del documento** (*Jefe de Proyecto*) Una vez identi□cadas las fuentes, los contenidos que era necesario añadir y se tenía el código para la implementación se procedió a la elaboración del presente documento.

9. **Cierre del proyecto** (*Jefe de Proyecto*) El proyecto queda cerrado con la defensa del mismo ante el tribunal.

8.3 Análisis temporal

Análisis temporal			
Tarea Nº	**Subtarea**	**Tiempo** (h)	**Tiempo total**(h)
1			16
2	2.1 2.2 2.3	20 60 20	100
3	3.1 3.2 3.3	35 25 60	120
4	4.1 4.2	26 4	30
5	5.1 5.2 5.3	20 20 10	50
6	6.1 6.2 6.3	10 30 10	50
7			4
8			70
9			10
Total			**450**

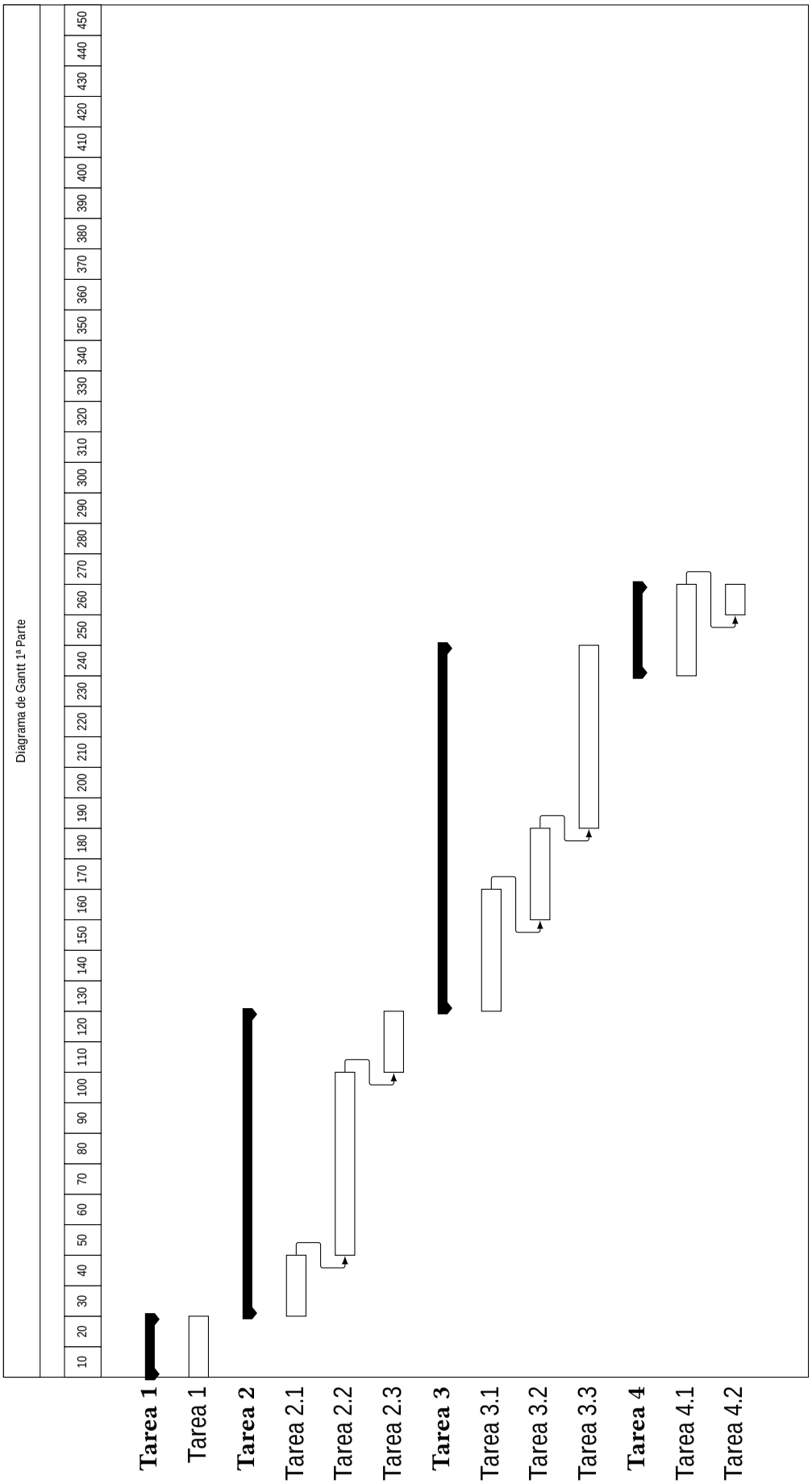
Diagrama de Gantt 1ª Parte
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450
Tarea 1
Tarea 1
Tarea 2
Tarea 2.1
Tarea 2.2
Tarea 2.3
Tarea 3
Tarea 3.1
Tarea 3.2
Tarea 3.3
Tarea 4
Tarea 4.1
Tarea 4.2

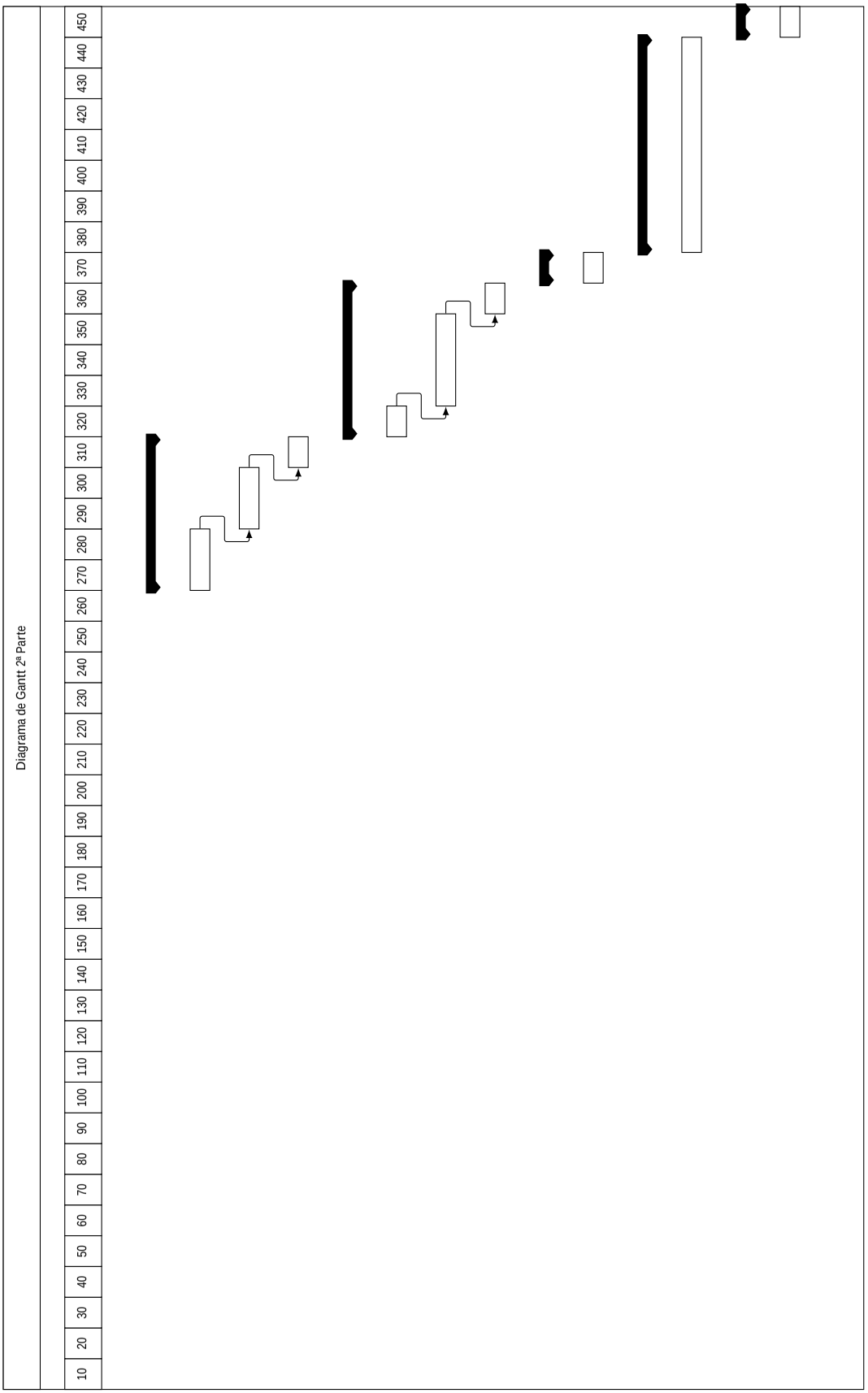
Diagrama de Gantt 2ª Parte
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450
Tarea 5
Tarea 3.1
Tarea 5.2
Tarea 5.3
Tarea 6
Tarea 6.1
Tarea 6.2
Tarea 6.3
Tarea 7
Tarea 7
Tarea 8
Tarea 8
Tarea 9
Tarea 9

8.4 Costes totales

La siguiente tabla expone los costes de cada rol por hora.

Precio rol/hora	
Rol	**Euros/hora**
Jefe de Proyecto	45
Analista	34
Tester	30
Desarrollador	30

Por tanto, nos quedan los siguientes costes de roles totales. Dado que la implementación puede hacerse en cualquier equipo con Python 11 suponemos unos costes de equipo y corriente nulos o prácticamente mínimos, por lo que no los consideramos en los costes totales.

Coste de Personal		
Jefe de Proyecto	96 h	4320 €
Analista	300 h	10200 €
Tester	4 h	120 €
Desarrollador	50 h	1500 €
Coste Total		**16140 €**

9 Conclusiones

En este trabajo hemos conseguido realizar una implementación exitosa del algoritmo Ed25519. Para realizarla, hemos tomado un enfoque constructivo tal y como lo presentamos en la Introducción, Capítulo 1.

Partiendo de los conceptos matemáticos y criptográ□cos previos, como la □rma digital y las curvas elípticas, hemos elaborado un capítulo cohesionado en el que se presentan las curvas de Edwards, se aclara el porqué de su elección y se demuestra que efectivamente puede establecerse una equivalencia birracional con las curvas elípticas.

Además, hemos descrito la □rma EdDSA tanto a nivel formal como a nivel práctico con la implementación en Python del algoritmo de □rma EdDSA basado en la curva *Curve25519*. Todo esto nos ha permitido comprender en profundidad una herramienta tan importante y usada como la □rma digital. Asimismo, hemos desvelado la magia detrás de la seguridad que ofrecen aplicaciones de mensajería instantánea usadas por millones de personas cada día, como Signal o Whatsapp, a lo largo del capítulo 7.

El desarrollo de este trabajo no habría sido posible sin las competencias adquiridas en ambos grados. Desde las más abstractas, como la resolución de problemas o la identi□cación de objetivos y requisitos, a las más aplicadas como la investigación o la elaboración de un escrito académico, por nombrar algunas. Sin duda, han jugado un papel importantísimo y la creación y exposición del trabajo lo demuestran.

Respecto a los contenidos presentados y retos futuros, a nivel académico queda claro que los avances han sido muy signi□cativos. Como han sido relativamente recientes, sigue siendo un campo abierto y fértil. Como desafío a largo plazo destaca una línea de investigación de uno de los autores más referenciados en este documento, Tanja Lange.

La investigación de Lange versa sobre la criptografía en el hipotético futuro post

cuántico, cuando los problemas que forman la base de la criptografía, como el problema del logaritmo discreto, puedan ser resueltos muy velozmente. Este problema afecta a muchos primitivos criptográ□cos, estando EdDSA en ese conjunto. Indudablemente, es una línea compleja pero interesantísima que in□uenciará enormemente la criptografía del futuro, y por consiguiente, la sociedad futura. Sin duda, el ingenio humano seguirá hallando soluciones exitosas y sorprendentes para los retos que vendrán.

Por último, concluimos este trabajo con un breve pero sincero agradecimiento. Innegablemente, la favorable consecución del Doble Grado no hubiese sido posible sin el apoyo de familiares, compañeros y personal de la Universidad de Sevilla. A su vez, la ayuda prestada por los tutores para la elaboración de este trabajo supone una especial mención. Este broche □nal del Doble Grado ha sido posible gracias a ellos.

Bibliografía

[1] Librería *ed25519*. https://github.com/warner/python-ed25519.

[2] Lista aplicaciones ed25519. https://ianix.com/pub/ed25519-deployment.html.

[3] Notebook collab con el código ed25519. https://colab.research.google.com/drive/1WYN7NZeMSbwmIaEwdTt97rbsTlNGPcyz?usp=sharing.

[4] Profesorado departamento Álgebra. teoría de códigos y criptografía (grado en matemáticas), 2022.

[5] Bernstein, D. J. Web *ed25519*. https://ed25519.cr.yp.to/index.html.

[6] Bernstein, D. J. Curve25519: new di□e-hellman speed records. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9* (2006), Springer, pp. 207–228.

[7] Bernstein, D. J., Birkner, P., Joye, M., Lange, T., and Peters, C. Twisted edwards curves. In *Progress in Cryptology–AFRICACRYPT 2008: First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings 1* (2008), Springer, pp. 389–405.

[8] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., and Yang, B.-Y. High-speed high-security signatures. *Journal of cryptographic engineering 2*, 2 (2012), 77–89.

[9] Bernstein, D. J., and Lange, T. Explicit formulas database. hyperelliptic.org/EFD, 2007.

[10] Bernstein, D. J., and Lange, T. Faster addition and doubling on elliptic curves. In *Advances in Cryptology–ASIACRYPT 2007: 13th International Conference*

on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007. Proceedings 13 (2007), Springer, pp. 29–50.

[11] Bruce, S. Applied cryptography: Protocols, algorthms, and source code in c.-2nd, 1996.

[12] Edwards, H. A normal form for elliptic curves. *Bulletin of the American mathematical society 44*, 3 (2007), 393–422.

[13] ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory 31*, 4 (1985), 469–472.

[14] Johnson, D., Menezes, A., and Vanstone, S. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security 1* (2001), 36–63.

[15] Marlinspike, M., and Perrin, T. The x3dh key agreement protocol. *Open Whisper Systems 283* (2016), 10.

[16] Montgomery, P. L. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation 48*, 177 (1987), 243–264.

[17] Perrin, T., and Marlinspike, M. The double ratchet algorithm. *GitHub wiki* (2016), 112.

[18] Silverman, J. H. *The arithmetic of elliptic curves*, vol. 106. Springer, 2009.

[19] Washington, L. C. *Elliptic curves: number theory and cryptography*. CRC press, 2008.