FACULTAD DE CIENCIAS
UNIVERSIDAD DE ZARAGOZA

Las matemáticas de las criptomonedas: el caso de Bitcoin

Trabajo de fin de grado de Matemáticas

Autor:
Luis Palazón Simón

Directores del trabajo:
Paz Jiménez Seral
Ricardo Julio Rodríguez Fernández
Antonio Juano Ayllon

Septiembre de 2023

Abstract

The concept of money currently in use follows the model of *fiat* money [23] (based on the trust of the community) whose value comes from its endorsement by the Central Bank of each country, as well as by international agreements and conventions. To make a transfer from one bank account to another at a different bank, the order is placed, the bank checks whether the originator's account has funds, subtracts the transfer money from its balance, and informs the other bank that it must add that amount to the balance of the recipient's account. This does not move physical money from one side to the other, it is simply a change in account balances. The banks are the intermediaries and guarantors of the transactions, keeping all the information in their databases. In order to dispense with banking entities as intermediaries in transactions and achieve their anonymity, *cryptocurrencies* were created, an alternative model of money in the digital environment. This model has become very popular in recent years [4, 15] due to its independence from banks and its privacy due to the use of pseudonyms in transactions.

This document describes in a precise and detailed way the mathematics behind the cryptocurrencies, analysing the particular case of Bitcoin. It consists of five chapters and three appendices. First of all, the concept of cryptocurrency is introduced in Chapter 1 by defining its origin and characteristics and implicitly, elaborating on the concept of *blockchain*, used by cryptocurrencies as a distributing record of transactions. Along with storing the transactions, other data is kept in each block of the chain with the purpose of optimising and making its performance increasingly secure, such as the summary of the previous block, the root of the *Merkle tree* generated from the block's transactions, or the digital signatures of the transaction. These summaries are computed by means of *hash functions*, whereas the signature and validation of the transactions is performed through asymmetric cryptography (more precisely, applying the *Elliptic Curve Digital Signature Algorithm*). The mathematical preliminaries necessary in the development of this work are explained in Chapter 2. Chapter 3 details the cryptographic tools used to provide security to this digital money model, guaranteeing the integrity of the transaction information and the authenticity of the transactions. Chapter 4 describes the particular case of Bitcoin and its mathematics, as does Mikel Lezaun in his article [14]. Despite the fact that each cryptocurrency has its own particularities, the specifications and overall functioning of Bitcoin will be detailed because it is the first and most popular cryptocurrency nowadays, which laid the foundations of subsequent creations of cryptocurrencies. As for the appendices, Appendix A details the hash functions used in Bitcoin. Finally, in Appendix B a simple simulation is carried out with the aim of illustrating how a blockchain works.

Keywords: cryptocurrency, Bitcoin, cryptography, blockchain, digital signature, elliptic curve, hash function, Merkle tree.

Índice general

Capítulo 1

Conceptos previos: Criptomonedas

Según el Banco Central Europeo [9, p. 25], una ***criptomoneda*** puede definirse como una representación digital de valor, no emitida por un banco central, una entidad de crédito o una entidad de dinero electrónico, que, en algunas circunstancias, puede utilizarse como alternativa al dinero tradicional. Las criptomonedas hacen uso de un sistema criptográfico para garantizar su titularidad y asegurar la integridad de las transacciones entre sus usuarios. Además, cuentan con una característica diferencial respecto a los sistemas tradicionales, ya que no están reguladas ni controladas por ninguna institución.

Una *moneda* es una pieza de oro, plata, cobre u otro metal, regularmente en forma de disco y acuñada con los distintivos elegidos por la autoridad emisora para acreditar su legitimidad y valor. Las monedas tienen que cumplir tres funciones: i) permitir almacenar valor, ii) permitir realizar intercambios y transacciones, y iii) servir como referencia de valor.

Según esto, a día de hoy no se puede afirmar que las criptomonedas sean monedas, ya que aunque en ciertos casos podría considerarse su papel como almacenamiento de valor, actualmente no está generalizado su uso como medio de intercambio y, debido a su mayoritaria volatilidad, tampoco sirven como referencia.

Las principales características de las criptomonedas son las siguientes: utilizan criptografía para anonimizar y firmar las transacciones; utilizan una estructura *blockchain* a modo de registro contable distribuido; no son controladas por ninguna institución; no hay posibilidad de falsificación o duplicidad; no hay intermediarios en las transacciones; las transacciones son irreversibles, es decir, una vez se efectúe el pago no hay posibilidad de cancelación; y se pueden intercambiar por otras divisas.

Para operar con criptomonedas se necesita (al menos) un monedero o *wallet*. Un *wallet* es una aplicación informática que permite transferir, recibir y almacenar criptomoneda. Una vez esté en funcionamiento, genera tantos pares de clave pública y privada como sean necesarios. La clave pública servirá para recibir criptomoneda y la privada para transferirla.

1.1. Origen y antecedentes

David Chaum ideó en 1983 un sistema monetario digital llamado *eCash*, el cual contaba con propiedades de privacidad [3]. Este sistema, a pesar de contar con una entidad bancaria como tercera parte en las transacciones monetarias, estableció las bases de las criptomonedas actuales usando primitivas criptográficas con el fin de anonimizar dichas transacciones. Sin embargo, este sistema no llegó a arraigarse y desapareció en 1999.

Al comienzo de la década de 1980 surgió un colectivo conocido como los *cypherpunk*, cuyo objetivo era aplicar los principios de la criptografía al ámbito político. Una de sus metas era crear monedas digitales descentralizadas que no dependieran de terceros. Este colectivo lanzó iniciativas como B-Money (propuesta por Wei Day en 1998 [5]) y Bit Gold (propuesta por Nick Szabo en 2005 [28]). Ninguna de estas iniciativas logró prosperar debido a diversos problemas que quedaron sin resolver, como el llamado *problema del doble gasto*, que consiste en saber si una moneda ya ha sido utilizada para evitar que se pueda emplear de nuevo.

En 2007 se produjo un periodo de inestabilidad económica debido a la dinámica especulativa llevada a cabo en el sector inmobiliario a lo largo de la primera década del siglo XXI. Esto llevó a una parte de la población a cuestionar el sistema financiero del momento y a que en ciertos sectores sociales se fraguara un clima de desconfianza respecto a los bancos centrales de los distintos países o regiones, de forma que algunos colectivos reaccionaron planteando modelos financieros alternativos orientados a eliminar la dependencia respecto al dinero *fiat*.

En este contexto, el 1 de noviembre 2008, en el apartado de temática criptográfica de una lista de correo electrónico llamada *metzdown.com* y asociada con el movimiento *cypherpunk*, Satoshi Nakamoto publicó un mensaje donde se proponía *Bitcoin*, "*un nuevo sistema de dinero electrónico que funciona completamente entre iguales* (no existen jerarquías), *sin una tercera parte confiable*" mediante un documento teórico [19]. Bitcoin fue la primera criptomoneda que se asentó y logró resolver problemas que habían tenido otras propuestas de dinero digital anteriores, como el problema del doble gasto anteriormente mencionado.

1.2. Registro contable distribuido. Concepto de *Blockchain*

Una ***blockchain*** o **cadena de bloques** es una base de datos estructurada en bloques ordenados secuencialmente, donde cada bloque cuenta con una referencia del bloque anterior.

La referencia del bloque $i-1$ que figura en el bloque i se calcula en base al contenido del bloque $i-1$, es decir, en base a sus datos y a la referencia del bloque $i-2$. De esta forma, la modificación de cualquier dato del bloque i implica que la referencia de dicho bloque contenida en el bloque $i+1$ sea alterada, así como las referencias de los bloques posteriores.

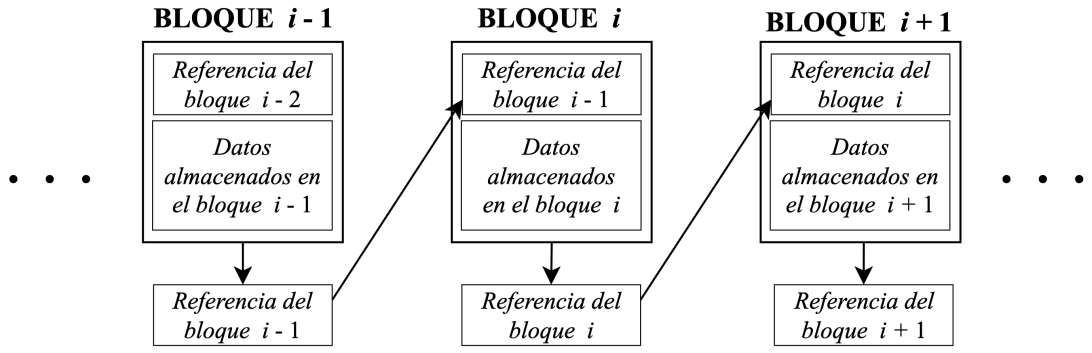


Figura 1.1: Esquema básico de una *blockchain*.

Estas referencias funcionan de tal forma que cualquier modificación del contenido del bloque, por mínima que sea, produce que dicha referencia se altere sustancialmente. Esto hace que sea muy sencillo preservar la integridad de los datos almacenados, ya que cualquier modificación de los mismos será rápidamente detectada, garantizando la fiabilidad de las posibles réplicas de la cadena. Para generar estas referencias se utiliza un herramienta clave en este tipo de estructuras, una función hash.

Las funciones hash generan una referencia numérica de longitud fija llamada *hash* o resumen a partir de un dato de entrada de cualquier longitud. Así, las referencias antes mencionadas son simplemente los resúmenes resultantes de aplicar el contenido del bloque como entrada de la función hash. Su uso posibilita validar de forma sencilla y eficiente que la información almacenada en la cadena de bloques no ha sido modificada. Más adelante se profundizará sobre este tipo de funciones, las cuales son el corazón de las criptomonedas y del concepto de blockchain, además de ser parte fundamental de la firma digital. A modo de ejemplo, la Figura 1.2 muestra tres posibles bloques de una *blockchain* con sus respectivos resúmenes (calculados mediante la función hash SHA-256 [27]).

Por otra parte, en el ámbito de las criptomonedas, destaca el uso de mecanismos basados en criptografía asimétrica, los cuales permite firmar electrónicamente las transacciones que se almacenan en la *blockchain* (explicados detalladamente en el Capítulo 3).

Figura 1.2: Bloques 3, 4 y 5 de una *blockchain* sencilla. Generada mediante [2].

Generalmente, este sistema de almacenamiento se implementa de forma distribuida en una red de computadores (también llamados *nodos*) que cuentan cada uno con una réplica de la cadena de bloques. Nótese que la naturaleza de esta estructura hace que sea sencillo y rápido actualizar las réplicas, ya que cuando hay que actualizar estas, sólo es necesario copiar los últimos bloques, y no toda la cadena.

Las estructuras *blockchain* ofrecen características que las hacen mejores que las bases de datos tradicionales en situaciones en las que se necesita almacenar un registro de cómo se encuentra un sistema en determinados momentos, existe una gran cantidad de usuarios con permisos de escritura que no son confiables, y no existe una autoridad confiable siempre disponible.

El usar una cadena de bloques como base de datos ha permitido la implementación de un registro contable en constante crecimiento en el que las transacciones se agrupan en bloques, que se van encadenando uno detrás de otro de manera que una pequeña modificación en un bloque de la base de datos requiere modificar todos los bloques sucesores, lo que técnicamente es irrealizable. Además, un conjunto de nodos posee una copia de la cadena, que sólo es actualizada si se alcanza consenso respecto a dicha actualización entre los nodos implicados, sin necesidad de recurrir a una entidad central de confianza. Esto confiere seguridad a la base de datos, seguridad que reposa en la propia red, sin depender de terceros. En el Apéndice B se exponen varios ejemplos de uso de esta estructura como registro contable.

Este sistema ha evolucionado desde su aparición, incorporando incluso mecanismos para automatizar la ejecución de programas informáticos en la propia *blockchain* (caso de Ethereum [8]). Dicha evolución ha permitido que sus usos se extiendan más allá de las criptomonedas y el intercambio de valor.

1.3. Criptomonedas en la actualidad

El origen y asentamiento de Bitcoin abrió el camino a la creación de un gran número de nuevas criptomonedas con distintas características. De hecho, según Forbes [12], se estimaba la existencia de casi 23.000 criptomonedas en marzo de 2023. Existen criptomonedas especializadas en el ámbito de la privacidad (p. ej. ZCash [32] o Monero [17]), o en el ámbito empresarial (p. ej. Ripple [22]), que son de interés a la hora de optimizar la consolidación y transferencia de información en consorcios y organismos.

En los últimos años también se ha popularizado un tipo específico de criptomonedas: las *stablecoins* o criptomonedas estables, que son criptomonedas cuyo valor se encuentra vinculado a activos externos, tales como monedas nacionales o metales preciosos. Algunos ejemplos de estas criptomonedas son Tether [29] y USD Coin [30].

Capítulo 2

Preliminares matemáticos

A lo largo de este capítulo se van a exponer los conceptos matemáticos que servirán como base de los mecanismos criptográficos utilizados por las criptomonedas que se detallan en el Capítulo 3.

2.1. Sistemas de numeración

2.1.1. Expresar en binario un número real en base decimal

Para un número real x, si

$$x = d_n \cdot 2^n + \cdots + d_2 \cdot 2^2 + d_1 \cdot 2 + d_0 + d_{-1} \cdot 2^{-1} + d_{-2} \cdot 2^{-2} + d_{-3} \cdot 2^{-3} + \cdots$$

con $d_i \in \{0,1\}$, se dice que $d_n \dots d_1 d_0, d_{-1} d_{-2} \dots$ es su expresión en binario. Para expresar en binario un número real en decimal, se trabaja independientemente sobre su parte entera y su parte fraccionaria. La parte entera de x coincide con los sumandos con subíndices no negativos y se transforma con el método habitual. Por otro lado, denótese a la parte fraccionaria (sumandos con subíndices negativos) por n tal que $0 \leq n < 1$. La expresión binaria de n es $d_{-1} d_{-2} d_{-3} \dots$ cumpliendo que

$$n = d_{-1} \cdot 2^{-1} + d_{-2} \cdot 2^{-2} + d_{-3} \cdot 2^{-3} + \cdots .$$

Para calcular cada d_{-i} nótese que

$$n \cdot 2^i = d_{-1} \cdot 2^{i-1} + d_{-2} \cdot 2^{i-2} + \cdots + d_{-i} + d_{-i-1} \cdot 2^{-1} + \cdots ,$$

y tomando partes enteras se tiene que

$$\lfloor n \cdot 2^i \rfloor = d_{-1} \cdot 2^{i-1} + d_{-2} \cdot 2^{i-2} + \cdots + d_{-i}.$$

Así, considerando módulo 2 se obtiene

$$d_{-i} = \lfloor n \cdot 2^i \rfloor \pmod 2.$$

2.2. Estructuras algebraicas

2.2.1. Grupos

Definición. Sea G un conjunto que cuenta con una operación binaria interna asociativa $*$, un elemento neutro y un inverso para cada elemento. En tal caso, a $(G,*)$ se le llama *grupo*. Si además, dicha operación binaria es conmutativa, el grupo se dice *abeliano*.

Ejemplo 1. Sea n un entero positivo, $(\mathbb{Z}_n, +)$ (clases de restos módulo n) y $\left(\{e^{\frac{2\pi ik}{n}} \mid 0 \leq k < n\}, \cdot\right)$ (raíces en $\mathbb{C}$ de $x^n - 1$) son grupos.

Notación. La operación en un grupo abeliano puede denotarse de manera aditiva o multiplicativa. En tales casos, se denota

$$kx = x + \overset{(k)}{\cdots} + x \quad \text{(Notación aditiva)}$$
$$x^k = x \overset{(k)}{\cdots} x \quad \text{(Notación multiplicativa)}$$

En la notación aditiva el elemento neutro se denota por 0 y en la notación multiplicativa por 1. Usualmente se utiliza esta última notación.

Definición. Si $(G,\cdot)$ es un grupo y G tiene un número finito de elementos, se dice que $(G,\cdot)$ es un *grupo finito*. En tal caso, se denomina *orden* de G al número de elementos de G y se denota por $|G|$.

Definición. Sea H un subconjunto de G tal que $(H,\cdot|_H)$ es un grupo, se dice que $(H,\cdot|_H)$ es un *subgrupo* de $(G,\cdot)$, o simplemente que H es *subgrupo* de G si no da lugar a confusión.

Definición. Sea H un subgrupo de un grupo G y $g \in G$. Al conjunto

$$gH := \{x \in G \mid x = gh \text{ para algún } h \in H\}.$$

se le llama *coclase a izquierda de g módulo H*. Además, al número de coclases a izquierda módulo H en G se le llama *índice de H en G* y se denota por $|G:H|$.

Teorema 2.1. ***(Teorema de Lagrange)*** *Sea G un grupo finito y sea H un* subgrupo *de G, se tiene que*

$$|G| = |H||G:H|.$$

Definición. Sea G un grupo, si $x \in G$, se llama *subgrupo generado por x* a la intersección de los subgrupos de G que contienen a x y se denota por $\langle x \rangle$. Nótese que este subgrupo será el menor subgrupo de G que contiene a x.

Definición. Sea $(G,\cdot)$ un grupo y sea $x \in G$. Si existe un entero positivo n tal que $x^n = 1$, al menor n cumpliendo dicha condición se le llama *orden de x* y se le denota por $o(x)$. Si no existe tal n, se dice que x no tiene orden finito.

Observación. Si $o(x) = n$, entonces:

$$x^{-1} = x^{n-1}$$

y

$$\langle x \rangle = \{1, x, \ldots, x^{n-1}\}.$$

Además, nótese que si G es finito, $o(x)$ divide a $|G|$ como consecuencia del Teorema 2.1.

Definición. Un grupo G se dice *cíclico* si $G = \langle x \rangle$ para algún $x \in G$. En tal caso, se dice que x es un *generador* de G.

Teorema 2.2. *Todo subgrupo de un grupo cíclico es cíclico.*

Teorema 2.3. *Todo grupo cíclico es abeliano.*

Proposición 2.4. *Si $G = \langle x \rangle$ es un grupo cíclico de orden n, los generadores de G son los x^r tal que $1 \leq r \leq n$ con $mcd(r,n) = 1$. Por lo tanto, G tiene $\phi(n)$ generadores, siendo ϕ la función de Euler.*

Proposición 2.5. *Si G es un grupo cíclico de orden n y d un entero positivo que divide a n, entonces G posee un único subgrupo de orden d, que es también cíclico.*

Ejemplo 2. Sea n un entero positivo y $\mathbb{Z}_n$ el conjunto de enteros módulo n, el grupo $(\mathbb{Z}_n,+)$ es cíclico.

Proposición 2.6. *Sea G un grupo cuyo orden es un número primo. Entonces, G es un grupo cíclico y cualquier elemento no nulo del mismo lo genera.*

2.2.2. Cuerpos

Definición. Sea $\mathbb{F}$ un conjunto con dos operaciones binarias, suma (denotada por $+$) y multiplicación (denotada por $\cdot$). Se dice que $(\mathbb{F},+,\cdot)$ es un *cuerpo* si cumple las siguientes propiedades:

1. $(\mathbb{F},+)$ es un grupo abeliano.
2. $(\mathbb{F}^*,\cdot)$ es un grupo abeliano, siendo $\mathbb{F}^* := \mathbb{F}\setminus\{0\}$ el grupo de las unidades de $\mathbb{F}$.
3. (Distributividad) $\forall x,y,z\in\mathbb{F}$ se tiene que $x\cdot(y+z)=(x\cdot y)+(x\cdot z)$.

Definición. Sea K un cuerpo, se llama *característica* de K y se denota por *char(K)* al menor número de veces que debemos sumar la identidad multiplicativa para conseguir la identidad aditiva. Si esta igualdad nunca se verifica para ningún valor de n, decimos que $char(K)=0$.

Definición. Si un cuerpo tiene un número finito de elementos, se dice que es un *cuerpo finito*. El *orden* del cuerpo es el número de elementos que contiene.

Ejemplo 3. Sea p un número primo, los enteros módulo p forman un cuerpo de p elementos, denotado por $\mathbb{F}_p$. Dicho cuerpo es de característica p y se le llama *cuerpo primo de característica p*.

Teorema 2.7. *La característica de cualquier cuerpo es cero o un número primo p. Además, todo cuerpo de característica p contiene un cuerpo isomorfo a* $\mathbb{F}_p$.

Teorema 2.8. *Sea K un cuerpo finito, entonces* $|K|=p^n$ *para algún primo p y algún entero n. Recíprocamente, para todo primo p y entero positivo n, existe un único cuerpo, salvo isomorfismos, de* p^n *elementos. Además, dicho cuerpo está formado por las raíces del polinomio* $x^{p^n}-x\in\mathbb{F}_p[x]$.

Teorema 2.9. *Sea* $(\mathbb{F},+,\cdot)$ *un cuerpo finito. El grupo multiplicativo de las unidades de* $\mathbb{F}$, $(\mathbb{F}^*,\cdot)$, *es un grupo cíclico.*

2.3. Curvas elípticas

Dado que este concepto se expone como preliminar para una posterior aplicación criptográfica, este documento se centra en curvas definidas sobre el cuerpo de p elementos, $\mathbb{F}_p$, donde p será un primo grande y, desde luego, $p>3$. Es por ello que se va a reducir la definición de curva elíptica al ámbito de los cuerpos de característica distinta de 2 y de 3, con el objetivo de simplificar la misma.

Definición. Sea un cuerpo $\mathbb{F}$, se define la *ecuación de Weierstrass* en coordenadas no homogéneas mediante la siguiente expresión:

$$y^2+a_1xy+a_3y=x^3+a_2x^2+a_4x+a_6, \tag{2.1}$$

con $a_1,a_2,a_3,a_4,a_5,a_6\in\mathbb{F}$.

Proposición 2.10. *Si la característica del cuerpo* $\mathbb{F}$ *anterior es distinta de 2 y de 3, el cambio de variable*

$$(x,y)\longrightarrow\left(\frac{x-3a_1^2-12a_2}{36},\frac{y-3a_1x}{216}-\frac{a_1^3+4a_1a_2-12a_3}{24}\right)$$

transforma la ecuación (2.1) en

$$y^2=x^3+ax+b. \tag{2.2}$$

A la ecuación (2.2) se le denomina *ecuación de Weierstrass reducida*.

Definición. Sea $\mathbb{F}$ un cuerpo de característica distinta de 2 y de 3, y $a,b\in\mathbb{F}$ tal que x^3+ax+b no tenga raíces múltiples. Una *curva elíptica* sobre $\mathbb{F}$ es el conjunto de soluciones de la ecuación de Weierstrass reducida (2.2), es decir, los $(x,y)\in\mathbb{F}^2$ que satisfacen dicha ecuación, y un elemento adicional, $\mathcal{O}$, llamado *punto del infinito*. Usualmente, se denotará a una curva elíptica sobre $\mathbb{F}$ por $E(\mathbb{F})$.

Proposición 2.11. *Que x^3+ax+b no tenga raíces múltiples equivale a que $4a^3+27b^2 \neq 0$.*

Demostración. Un polinomio $p(x)$ con derivada no nula tiene una raíz múltiple si y sólo si dicha raíz lo es también de su derivada $p'(x)$. En este caso, $p(x) = x^3+ax+b$ y entonces $p'(x) = 3x^2+a$, que será no nulo siempre que la característica sea distinta de 3.

Si $a = 0$, $p(x)$ y $p'(x)$ tienen una raíz común ($x = 0$) si y sólo si $b = 0$, así que se puede asumir que $a \neq 0$ de ahora en adelante.

Ahora, si existe una raíz común entre $p(x)$ y $p'(x)$ que llamamos r, se verifica

$$r(r^2+a)+b = 0 \qquad \text{y} \qquad r^2 = -a/3.$$

Así, se tiene que

$$r\,\frac{2a}{3} = -b$$

y elevando al cuadrado

$$\frac{-a}{3}\,\frac{4a^2}{9} = b^2,$$

lo que prueba la primera implicación.

Para el recíproco, ya se ha observado en la primera igualdad que, de existir $r = \frac{-3b}{2a}$ y suponiendo que $4a^3+27b^2 = 0$, se tiene que r es raíz común. □

Ejemplo 4. A continuación, se muestran en la Figura 2.1 algunos ejemplos de curvas elípticas representados sobre el plano real.

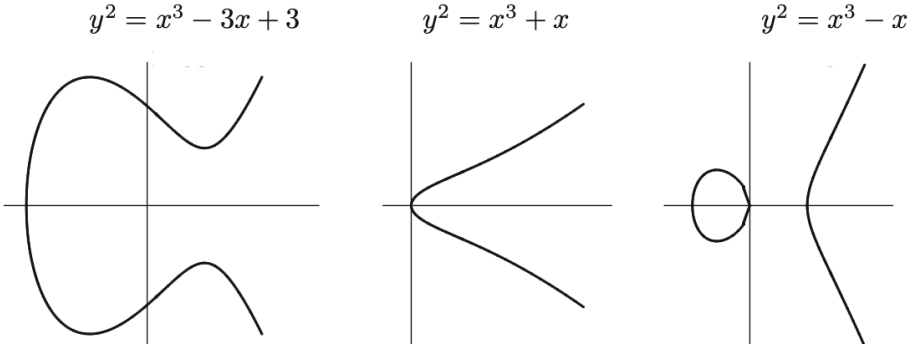


Figura 2.1: Representación geométrica en $\mathbb{R}^2$ de algunas curvas elípticas. Fuente: [24, p. 43]

2.3.1. Estructura de grupo abeliano

En el conjunto de los puntos de una curva elíptica, se define una operación binaria interna con la que se tendrá un grupo abeliano. Para definir dicha operación, se va a utilizar notación aditiva y se van a ir exponiendo ejemplos sobre la curva elíptica sobre $\mathbb{F}_7$ definida por las soluciones de la ecuación $y^2 = x^3 - x + 1$.

Esta operación se observa mejor geométricamente, cuando se aplica a una curva elíptica sobre el plano real.

Definición. Sea E una curva elíptica. Para definir la operación, hay que tener en cuenta que al hacer la intersección de una recta que pasa por dos puntos de la curva con la propia curva, se tendrá una ecuación de tercer grado. Así, dicha intersección puede tener como máximo 3 puntos distintos.

En primer lugar, el punto del infinito actúa como elemento neutro: $P+\mathcal{O} = \mathcal{O}+P = P \quad \forall P \in E$. Además, si $P = (x,y) \in E$, se tiene que $-P = (x,-y)$. Observar que la recta que pasa por (x,y) e $(x,-y)$ corta a la curva únicamente en estos dos puntos.

Sean $P, Q \in E$ tal que $P \neq \pm Q$, considérense las dos siguientes situaciones:

1. Supóngase que la recta que une P y Q no es tangente a E, es decir, dicha recta corta a E en un tercer punto R (véase la Figura 2.2a). En tal caso se tiene que

$$P+Q=-R,$$

o equivalentemente,

$$P+Q+R=\mathcal{O}.$$

Ejemplo 5. Sea $E(\mathbb{F}_7)$ una curva elíptica definida por las soluciones de la ecuación $y^2=x^3-x+1$. Sean $P=(0,1)$ y $Q=(1,6)$ puntos de la curva. Hallemos $P+Q$.

Calculemos la recta que pasa por ambos puntos. Para ello, nótese que las rectas que pasan por P son de la forma $y=mx+1$, luego para que también Q pertenezca a una recta de este tipo, necesariamente $6=m+1$, es decir, $m=5$ y entonces la recta que pasa por P y Q es

$$y=5x+1.$$

Cortando dicha recta con la curva tenemos que:

$$x^3-x+1=(5x+1)^2=25x^2+10x+1=4x^2+3x+1$$

luego

$$x^3-x=4x^2+3x$$
$$x^3+3x^2+3x=0$$
$$x(x^2+3x+3)=0$$

Como $char(\mathbb{F})\neq 2$, se puede utilizar la fórmula habitual para hallar las soluciones de una ecuación de segundo grado para obtener las soluciones de $x^2+3x+3=0$:

$$x=\frac{-3\pm\sqrt{-3}}{2}=\frac{-3\pm\sqrt{4}}{2}=\begin{cases}-\frac{1}{2}=-1\cdot 4=-4=3\\ -\frac{5}{2}=-5\cdot 4=-20=1\end{cases}$$

Así, las soluciones de la ecuación $x(x^2+3x+3)=0$ son 0, 1 y 3.

Excluyendo las soluciones que corresponden a P y Q, tenemos que la intersección de la recta con la curva en el tercer punto se produce en $x=3$.

Ahora, si se sustituye en la recta que pasa por P y Q esta coordenada, se obtiene que

$$y=15+1=16=2,$$

luego el tercer punto de corte es $R=(3,2)$.

Por lo tanto,

$$P+Q=-R=(3,-2)=(3,5).$$

2. Supóngase que la recta que une P y Q es tangente a E en P, y por tanto, dicha recta no corta a E en ningún otro punto (véase la Figura 2.2b). En tal caso se tiene que P es un punto doble y

$$P+Q=-P.$$

Ejemplo 6. Sea $E(\mathbb{F}_7)$ una curva elíptica definida por las soluciones de la ecuación $y^2=x^3-x+1$. Sean $P=(0,1)$ y $Q=(2,0)$ puntos de la curva. Hallemos $P+Q$.

Calculemos la recta que pasa por ambos puntos. Para ello, nótese que las rectas que pasan por P son de la forma $y=mx+1$, luego para que también Q pertenezca a una recta de este tipo, necesariamente $0=2m+1$, es decir, $m=-\frac{1}{2}=-4=3$ y entonces la recta que pasa por P y Q es

$$y=3x+1.$$

Cortando dicha recta con la curva tenemos que

$$x^3 - x + 1 = (3x+1)^2 = 9x^2 + 6x + 1 = 2x^2 + 6x + 1,$$

luego

$$x^3 - x = 2x^2 + 6x$$
$$x^3 - 2x^2 = 0$$
$$x^2(x-2) = 0$$

Las soluciones de esta ecuación son 0 (solución doble) y 2. Por lo tanto,

$$P + Q = -P = (0,-1) = (0,6).$$

Sea P un punto de la curva, se distinguen de nuevo dos posibles situaciones para calcular $P+P$:

1. Supóngase que la recta tangente a la curva en P corta a la curva en un punto distinto Q. En tal caso, se tienen dos puntos distintos P y Q y la recta que los une corta a la curva sólo en P y Q, siendo P un punto doble. Entonces por el caso 2 de la distinción anterior, $P+Q=-P$, luego $P+P=-Q$.

2. Supóngase que la recta tangente a E en P no corta a ningún otro punto de la curva (véase la Figura 2.2c).

   En tal caso se tiene que

   $$P + P = -P.$$

   Ejemplo 7. Sea $E(\mathbb{F}_7)$ una curva elíptica definida por las soluciones de la ecuación $y^2 = x^3 - x + 1$. Sea $P = (3,2) \in E(\mathbb{F}_7)$. Hallemos $2P$.

   Se deriva para hallar la pendiente de la recta tangente a la curva:

   $$2yy' = 3x^2 - 1$$

   y se sustituyen los valores de P:

   $$4y' = 5 \Longrightarrow y' = \frac{5}{4} = 5 \cdot 2 = 10 = 3$$

   Por tanto, la pendiente de la recta tangente a la curva en P es 3, luego la dicha recta es de la forma

   $$y = 3x + n,$$

   y si se sustituyen los valores de P se obtiene que $n = 0$ y la recta tangente a la curva en P es

   $$y = 3x.$$

   Cortando dicha recta con la curva tenemos que (nótese que forzosamente 3 será solución doble):

   $$x^3 - x + 1 = (3x)^2 = 9x^2 = 2x^2$$

   luego

   $$x^3 - 2x^2 - x + 1 = 0$$

   y como 3 es solución doble se puede factorizar de forma que:

   $$x^3 - 2x^2 - x + 1 = (x-3)^2(x-3) = (x-3)^3 = 0$$

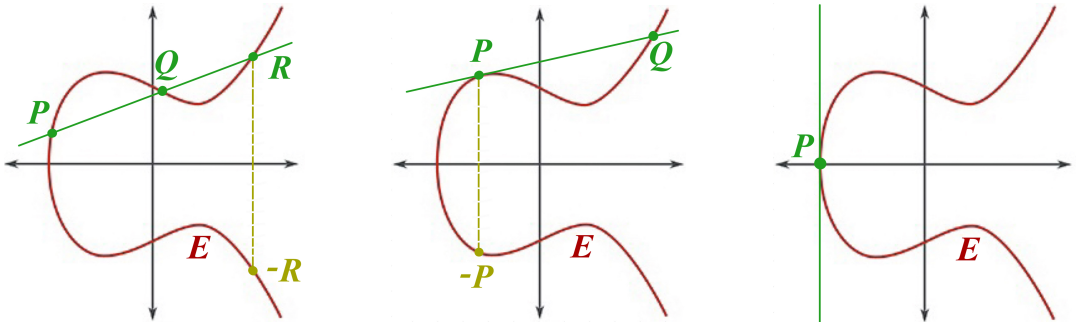
   Claramente, la solución triple de esta ecuación es 3.

   Así, tenemos que la intersección de la recta con la curva en el tercer punto se produce en $x = 3$.

   Ahora, si se sustituye en la recta tangente esta coordenada, se obtiene que $y = 9 = 2$, luego el tercer punto de corte es de nuevo $P = (3,2)$.

   Por lo tanto,

   $$P + P = -P = (3,-2) = (3,5).$$



(a) Primer caso (b) Segundo caso (c) Tercer caso

Figura 2.2: Casos tratados en la definición de la operación en curva elíptica representados en $\mathbb{R}^2$.

Proposición 2.12. *Sea E una curva elíptica cualquiera y denotemos por $+$ la operación detallada en la definición anterior. Entonces $(E,+)$ es un grupo abeliano con el punto del infinito como elemento neutro.*

Demostración. Veamos que se verifican las propiedades que debe satisfacer un grupo abeliano:

- La conmutatividad es obvia ya que para cualesquiera P y Q puntos de la curva, la recta que une P y Q es la misma que une Q y P.
- La existencia de elemento neutro y elemento inverso viene dada por definición.
- Probar la asociatividad no es trivial. Esta demostración puede encontrarse en [31, sec. 2.4].

□

2.3.2. Curvas elípticas sobre cuerpos finitos

Obsérvese que como consecuencia del Teorema 2.8, la existencia de un cuerpo finito de orden q, denotado como $\mathbb{F}_q$, está ligada a la condición de que $q = p^m$ para cierto número primo p y entero positivo m. En general, en los criptosistemas basados en curvas elípticas se utiliza un cuerpo finito primo $\mathbb{F}_p$.

Orden de una curva elíptica sobre un cuerpo finito

Nótese que si el cuerpo sobre el que está definido la curva es finito, el orden de la curva será finito. Se denota al orden de la curva elíptica E definida sobre el cuerpo finito $\mathbb{F}_q$ por $\#E(\mathbb{F}_q)$.

Ejemplo 8. Volvamos a la curva elíptica $E(\mathbb{F}_7)$ utilizada en los ejemplos anteriores, es decir, la definida por las soluciones de la ecuación $y^2 = x^3 - x + 1$, y hallemos su orden.

Para ello, se va a seguir el procedimiento descrito en la Tabla 2.1. En él, se evalúa cada miembro de la ecuación por separado para todos los valores posibles de cada coordenada.

x	x^3-x+1
0	1
1	1
2	0
3	4
4	5
5	2
6	1

y	y^2
0	0
1	1
2	4
3	2
4	2
5	4
6	1

Tabla 2.1: Evaluación de cada miembro de la ecuación $y^2 = x^3 - x + 1$.

Así, se pueden extraer los puntos (0,1), (0,6), (1,1), (1,6), (2,0), (3,2), (3,5), (5,3), (5,4), (6,1) y (6,6), los cuales sumados al punto del infinito hacen un total de 12 puntos. Por lo tanto, $\#E(\mathbb{F}_7) = 12$.

En este ejemplo, ha sido posible obtener con facilidad el orden de la curva elíptica debido a que el orden del cuerpo finito no era demasiado grande. La complejidad de esta tarea aumenta conforme lo hace el orden del cuerpo finito sobre el que está definida la curva. Sin embargo, el *teorema de Hasse* proporciona una cota para el orden de una curva elíptica sobre un cuerpo finito.

Teorema 2.13. ***(Hasse)*** *Sea $E(\mathbb{F}_q)$ una curva elíptica. Se tiene la siguiente cota:*

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

De esta forma, el número de puntos de $E(\mathbb{F}_q)$ puede acotarse de la siguiente manera:

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

Demostración. La demostración de este teorema se puede encontrar en [31, sec. 4.2]. □

Orden de un punto de una curva elíptica sobre un cuerpo finito

Observación. Como en todos los grupos finitos, en las curvas definidas sobre cuerpos finitos el orden de un punto cualquiera P siempre será un divisor del orden de la curva.

Cabe mencionar que el orden de la curva puede ser un número primo o compuesto, por ello tiene sentido introducir la siguiente definición:

Definición. Sea E una curva elíptica. Al cociente del orden de E entre su mayor factor primo se le denomina *cofactor* de E.

Ejemplo 9. En el ejemplo anterior, se había obtenido que el orden de la curva elíptica $E(\mathbb{F}_7)$ definida por las soluciones de la ecuación $y^2 = x^3 - x + 1$ tenía orden 12. Como $12 = 2^2 \cdot 3$, entonces el cofactor de E es

$$h = \frac{12}{3} = 4.$$

En criptografía es conveniente utilizar curvas cuyo orden sea un número primo o que, como segunda opción, sea el producto de un número primo y un cofactor pequeño (típicamente 2, 3 o 4) [11, p. 54], de esta forma, la probabilidad de que un punto cualquiera no nulo tenga orden elevado es muy alta.

Además, nótese que las curvas elípticas cuyo número de puntos es un número primo ofrecen la ventaja de que la propia curva es un grupo cíclico y que cualquier elemento no nulo lo genera, como se ha mencionado en la Proposición 2.6.

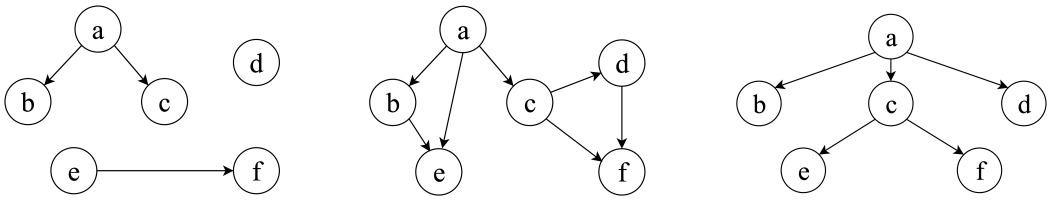
Parámetros de una curva elíptica definida sobre un cuerpo finito primo $\mathbb{F}_p$

Según el estándar SEC 1 (*Standards for Efficient Cryptography 1*) [25, p. 15], los parámetros que definen una curva elíptica E sobre $\mathbb{F}_p$ componen la tupla

$$T = (p, a, b, P, n, h), \tag{2.3}$$

donde:

- p es el número primo que define el cuerpo finito $\mathbb{F}_p$;
- $a, b \in \mathbb{F}_p$ son los coeficientes de la ecuación de Weierstrass reducida (2.2);
- $P \in E$ es un punto base de orden primo del grupo de puntos de la curva;
- n es el número primo que tiene que como orden el subgrupo cíclico generado por P; y
- h es el cofactor de E.



(a) Grafo del Ejemplo 10. (b) Grafo del Ejemplo 11. (c) Grafo del Ejemplo 12.

Figura 2.3: Ejemplos de los grafos tratados.

2.4. Árboles

Definición. Un *grafo dirigido G* se compone de un conjunto finito de vértices V, y un conjunto de aristas dirigidas $A \subseteq \{(x,y) \in V^2 : x \neq y\}$, donde cada arista dirigida va del primer vértice dentro del par (x) al segundo vértice (y).

Definición. Dos vértices $x, y \in V$ de un grafo dirigido $G = (V,A)$ son *adyacentes* si $(x,y) \in A$ o $(y,x) \in A$.

Definición. Sea $G = (V,A)$ un grafo dirigido y k un entero positivo. Un *camino* en un grafo G es una secuencia de vértices $v_1, v_2, \ldots, v_k \in V$ tal que v_i y v_{i+1} son adyacentes para todo $1 \leq i \leq k-1$. Si la secuencia está compuesta por vértices distintos entre sí, al camino se le llama *camino simple*.

Sea $G = (V,A)$ un grafo dirigido y $x, y \in V$, se denotará por $x \sim y$ cuando los vértices x e y puedan ser unidos mediante un camino simple en G, es decir, si existe una secuencia $v_1, v_2, \ldots, v_k \in V$ para cierto entero positivo k tal que $v_1 = x$ y $v_k = y$. Es sencillo comprobar que $\sim$ es una relación de equivalencia en V que hace dicho conjunto esté particionado en clases de equivalencia disjuntas.

Definición. Sea $G = (V,A)$ un grafo dirigido, V se puede particionar respecto a la relación de equivalencia $\sim$ de forma que

$$V = \bigcup_{i=1}^{r} V_i$$

para cierto entero positivo r. Se denota por A_i $(1 \leq i \leq r)$ al subconjunto de A que comprende aquellas aristas cuyos vértices están ambos en V_i. Entonces, a los grafos $G_i = (V_i, A_i)$ se les llama *componentes* de G. Si G tiene solamente una componente, se dice que G es *conexo*.

Ejemplo 10. En la Figura 2.3a se muestra un grafo dirigido $G = \big(\{a,b,c,d,e,f\}, \{(a,b),(a,c),(e,f)\}\big)$ con 3 componentes.

Definición. Sea $G = (V,A)$ un grafo dirigido y k un entero positivo. Un camino $v_1, v_2, \ldots, v_k$ en G tal que todos sus vértices son distintos a excepción de $v_1 = v_k$ recibe el nombre de *ciclo*.

Ejemplo 11. En la Figura 2.3b se muestra un grafo dirigido conexo con 2 ciclos: $\{a,b,e\}$ y $\{c,d,f\}$.

Definición. Un *árbol* es un grafo dirigido conexo que no contiene ciclos. Sus vértices son habitualmente llamados *nodos*. En este contexto, se llama *nodo raíz* al nodo $x \in V$ tal que $\nexists y \in V : (y,x) \in A$. Por otro lado, sea $(x,y) \in A$, se dice que x es el *nodo padre* de y, o equivalentemente, que y es el *nodo hijo* de x. Sea $z \in V$ un nodo que no tiene nodos hijos (tal que $\nexists y \in V : (z,y) \in A$), se dice que z es un *nodo hoja*.

Ejemplo 12. En la Figura 2.3c se muestra un árbol dirigido $T = (V,A)$ con $V = \{a,b,c,d,e,f\}$ y $A = \{(a,b),(a,c),(a,d),(c,e),(c,f)\}$. En dicho árbol, se tiene que el nodo raíz de T es el nodo a. Los nodos b, c y d son los nodos hijos del nodo a, o equivalentemente, el nodo a es el nodo padre de los nodos b, c y d. Además, los nodos e y f son los nodos hijos del nodo c, o equivalentemente, el nodo c es el nodo padre de los nodos e y f. Los nodos hoja de T son los nodos b, d, e y f.

Capítulo 3

Herramientas criptográficas

En este capítulo se explican los mecanismos criptográficos fundamentales utilizados por las criptomonedas. En primer lugar se definirán las funciones hash. Después, se explicarán los árboles de Merkle. Por último, se expondrá en detalle el algoritmo de firma digital con curva elíptica.

3.1. Funciones hash

Definición. Una función $f : X \longrightarrow Y$ es *unidireccional* si $\forall x \in X$, calcular $f(x) = y$ es poco costoso computacionalmente y, por otro lado, sea $y \in f(X)$, es muy costoso computacionalmente encontrar $x \in X$ tal que $f(x) = y$. A esta última característica se le llama *resistencia a la preimagen*.

Definición. Una función $f : X \longrightarrow Y$ es *resistente a las colisiones* si es computacionalmente difícil encontrar dos elementos de X, x_1 y x_2, tal que $x_1 \neq x_2$ y $f(x_1) = f(x_2)$.

Se denomina *función hash* o *función resumen* a una función unidireccional $\mathscr{H}$ resistente a las colisiones y muy sensible a las modificaciones, capaz de transformar una información o mensaje de longitud arbitraria en otro que tiene un tamaño fijado de antemano. Es decir, en términos de bits, se puede definir $\mathscr{H}$ como una función

$$\mathscr{H} : \{\{0,1\}^j | j \in \mathbb{N}\} \longrightarrow \{0,1\}^n, \text{ para algún } n \in \mathbb{N} \text{ fijo},$$

que cumple con las características anteriormente mencionadas. Al resultado de tal transformación se le denomina resumen o *hash*.

Una función hash se considera muy sensible a las modificaciones si al calcular el resumen de un mensaje y el resumen de ese mismo mensaje cambiando un bit, este último difiera del anterior en, aproximadamente, la mitad de sus bits [1, p. 23]. Esta característica se conoce como *efecto avalancha*.

En el Apéndice A se explican en detalle las dos funciones hash que son utilizadas en Bitcoin: SHA-256 y RIPEMD-160.

3.2. Árboles de Merkle

Definición. Un *árbol de Merkle* asociado a un conjunto de datos es un árbol en el cual cada nodo hoja contiene el resumen de un dato. En cada nodo que no sea hoja se almacena el resumen de los resúmenes concatenados que aparecen en sus nodos hijos, o el resumen del resumen concatenado consigo mismo del de su nodo hijo, en caso de tener un único hijo. Este árbol está construido de tal forma que cada dos hojas existe un nodo que será el padre de las dos, y si el número de hojas fuera impar existiría un nodo que sería el padre de la hoja desparejada. De esta forma, si se tienen n_0 hojas, existirán $n_1 := \lceil \frac{n_0}{2} \rceil$ nodos padre de las mismas, siendo $\lceil x \rceil := \text{mín}\{z \in \mathbb{Z} : x \leq z\}$. Del mismo modo, existirán $n_2 := \lceil \frac{n_1}{2} \rceil$ nodos padres de los n_1 anteriores, y así sucesivamente hasta la raíz.

Este tipo de estructura fue ideada por Ralph Merkle [16] en 1979 y permite que una gran cantidad de datos puedan ser ligados a un único valor hash, almacenado en el nodo raíz del árbol y conocido como *raíz de Merkle*, que posibilita verificar la integridad de grandes cantidades de información de forma eficiente.

Ejemplo 13. Se denota por "$\|$" al operador de concatenación. Esto es, si $h_1 =$ "abc" y $h_2 =$ "def", entonces $h_1 \| h_2 =$ "abcdef". Si se quiere generar un árbol de Merkle asociado a tres mensajes m_1, m_2, m_3 mediante una función hash H, este tendría la forma que se muestra en la Figura 3.1.

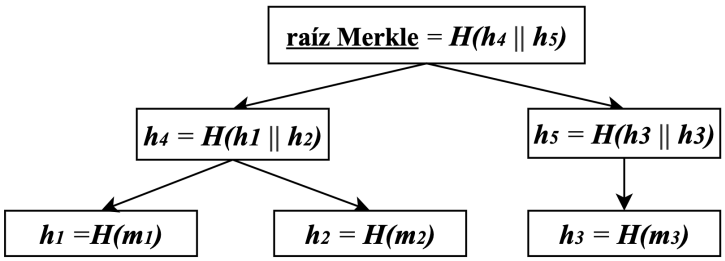


Figura 3.1: Esquema de árbol de Merkle asociado a tres mensajes.

3.3. Firma digital con curva elíptica

3.3.1. Criptografía asimétrica y firma digital

Siguiendo los criterios de Amparo Fuster y otros en [10, p. 221], un esquema de *firma digital* es un concepto criptográfico basado en determinadas propiedades matemáticas que permiten demostrar la autenticidad de un documento digital o un mensaje; mientras que una firma electrónica es un concepto de naturaleza legal, que se refiere a un medio que indica que una persona es responsable del contenido de un documento electrónico y que puede realizarse a través de medios no criptográficos.

La firma digital hace uso de *criptografía asimétrica* o *de clave pública*, la cual se caracteriza por el uso de dos claves, distintas pero relacionadas entre sí, para el cifrado y descifrado de información. Este hecho la distingue de la criptografía simétrica, en la que se usa una misma clave tanto para cifrar como para descifrar. El par de claves está compuesto por una clave pública, que se puede dar a conocer a los distintos usuarios con los que se quiera establecer comunicación, y una clave privada, la cual debería permanecer únicamente en conocimiento de su poseedor.

Supóngase que Alice quiere firmar digitalmente un documento. Para ello, Alice calcula el valor hash del documento, cifra dicho valor con su clave privada, y se lo envía a Bob junto con el documento. Por su parte, Bob descifra el valor hash usando la clave pública de Alice, la cual conoce. Bob aplica la función hash al mensaje original de Alice. Si el resultado no es el mismo que el valor que ha enviado Alice, Bob sabe que el mensaje ha sido alterado. De esta forma se consigue verificar la integridad del documento. Además, se asegura la autenticidad de la firma y el no repudio, ya que sólo Alice ha podido realizar el cifrado al haberse realizado con su clave privada.

En la actualidad, los criptosistemas asimétricos están basados en la dificultad de resolución de determinados problemas matemáticos, como el problema del logaritmo discreto.

3.3.2. El problema del logaritmo discreto y criptosistemas relacionados

Definición. Sea G un grupo y $\alpha \in G$ de orden finito. Sea $x \in \langle \alpha \rangle$, empleando notación multiplicativa se tiene que $x = \alpha^a$ para cierto entero no negativo a, o equivalentemente $x = a\alpha$ utilizando notación aditiva. A este a se le llama *logaritmo discreto* de x módulo α en el grupo que genera α.

Aunque todos los grupos cíclicos son isomorfos, en algunos conocido x y α es fácil calcular a, mientras que en otros es muy difícil. Para criptografía se necesita que sea irrealizable computacionalmente, y en ese caso se dice que el grupo tiene el *problema del logaritmo discreto*.

Ejemplo 14. Por muy grande que sea un primo p, $(Z_p,+)$ no tiene problema del logaritmo discreto. Sin embargo, una curva elíptica con p elementos (o n grande) sí lo tiene.

En 1976, Whitfield Diffie y Martin Hellman [6] publicaron el diseño del primer sistema asimétrico, conocido como el intercambio de clave Diffie-Hellman. Este sistema permite el intercambio de una clave simétrica de sesión para comunicarse posteriormente por un sistema simétrico como AES u otros.

Taher Elgamal [7] propuso en 1985 un sistema de cifrado que puede verse como una extensión del algoritmo de intercambio de clave Diffie-Hellman. Dicho sistema se puede implantar con el grupo de las unidades de $\mathbb{Z}_p$ (como en el caso de anterior), con las unidades de otro cuerpo, como por ejemplo el grupo de las unidades de $GL(2^m)$ (matrices regulares en el cuerpo de 2^m elementos). Nótese que se deben elegir grupos que tengan el problema del logaritmo discreto, como los anteriormente mencionados. A diferencia del protocolo definido por Diffie-Hellman, la elección del grupo y el generador sólo depende del receptor del mensaje. De esta forma, se evita que estos parámetros puedan conocerse por un tercero. Además de ser uno de los criptosistemas de clave asimétrica más extendidos, dio también lugar a los criptosistemas basados en curvas elípticas.

3.3.3. El problema del logaritmo discreto en curva elíptica

Nótese que los criptosistemas anteriores se consideran seguros si los tamaños de las claves son grandes, es decir, de unos 2048 bits. No obstante, este tamaño hace que sea complicado implementar estos sistemas de cifrado en dispositivos con poca capacidad de cálculo o con poco espacio de almacenamiento, como es el caso de tarjetas inteligentes (tarjetas bancarias, de identificación, telefónicas, etc.). Por este motivo, se proponen grupos alternativos a Z_p^* para usar como base e implementar con ellos sistemas de cifrado que requieran menor capacidad de cómputo, pero que ofrezcan niveles de seguridad equivalentes.

La idea de emplear como grupo alternativo un grupo aditivo de curva elíptica fue propuesta en 1985 por Neal Koblitz [13] y Victor Miller [18] de forma independiente. Considerando la estructura definida por los puntos de una curva de este tipo, se define un sistema de cifrado asimétrico similar al sistema de ElGamal mencionado en la Sección 3.3.2. No obstante, existen un par de diferencias importantes entre ambos tipos de sistemas: la longitud de las claves de un sistema basado en curvas elípticas es mucho menor y sin embargo se alcanza un nivel de seguridad equivalente; y la operación de suma que se lleva a cabo con los puntos de una curva elíptica tiene una mayor complejidad que en el caso del producto en el sistema de ElGamal.

En el ámbito de las curvas elípticas, el problema del logaritmo discreto se traduce en, sea E el grupo de puntos de una curva elíptica sobre un cuerpo finito generado por el punto P y sea $Q \in E$, hallar el entero positivo k que cumple que $Q = kP$.

Para elegir una curva elíptica E sobre $\mathbb{F}_p$ para uso criptográfico, hay que seleccionar un número primo p grande tal que el orden de la curva sea un número primo o, en su defecto, el producto de un primo por un entero pequeño. Además, hay que descartar los casos en los que la curva sea anómala, o supersingular, es decir, aquellos en los que $\#E(\mathbb{F}_p) = p+1$ o $\#E(\mathbb{F}_p) = p$, respectivamente.

3.3.4. Algoritmo de firma digital con curva elíptica (ECDSA)

El algoritmo de firma digital con curva elíptica o ECDSA (*Elliptic Curve Digital Signature Algorithm*) utiliza una curva elíptica E sobre un cuerpo finito (usualmente primo) $\mathbb{F}_p$ y un punto base $P \in E$, el cual genera un subgrupo cíclico de orden n. Además, será necesario un par de claves privada y pública del firmante (u,U), siendo u un entero positivo (clave privada) y $U = uP$ un punto de la curva E (clave pública), y una función hash $\mathscr{H}$, conocida y acordada de antemano. Nótese que como tanto P como U pertenecen a $\langle P \rangle$, se verifica que $nP = nU = \mathscr{O}$.

Como todos los protocolos de firma, el ECDSA está formado por dos algoritmos diferentes, el de elaboración de firma y el de verificación de la misma. El procedimiento que lleva a cabo el firmante del mensaje se muestra en el Algoritmo 1.

El verificador obtiene el mensaje a comprobar m y la firma (r,s). Para validar dicha firma, sigue los pasos del Algoritmo 2.

Algoritmo 1 Elaboración de firma ECDSA en pseudocódigo

$\tilde{m} \leftarrow \mathscr{H}(m)$ ▷ Se calcula el resumen del mensaje
hacer
  $k \leftarrow enteroAleatorio(1, n-1)$ ▷ Se toma un entero k tal que $1 \leq k \leq n-1$ (clave de sesión)
  $(x_0, y_0) \leftarrow kP$
  $r \leftarrow x_0 \pmod{n}$
mientras $r = 0$
$s \leftarrow k^{-1}(\tilde{m} + ur) \pmod{n}$
devuelve (r,s) ▷ La firma es el par (r,s)

Algoritmo 2 Verificación de firma ECDSA en pseudocódigo

$\tilde{m} \leftarrow \mathscr{H}(m)$ ▷ Se calcula el resumen del mensaje
si $r \notin [1, n-1] \cap \mathbb{Z}$ **o** $s \notin [1, n-1] \cap \mathbb{Z}$
  devuelve inválida
$z_1 \leftarrow \tilde{m}s^{-1} \pmod{n}$
$z_2 \leftarrow rs^{-1} \pmod{n}$
$(x_1, y_1) \leftarrow z_1 G + z_2 U$
si $r \equiv x_1 \pmod{n}$
  devuelve válida
si no
  devuelve inválida

Proposición 3.1. *El proceso de verificación descrito en el Algoritmo 2 es correcto para validar una firma generada siguiendo el Algoritmo 1.*

Demostración. Si se denota por $(Q)_x$ a la primera coordenada de un punto $Q \in E$, se tiene que

$$\begin{aligned} x_1 &= (z_1 P + z_2 U)_x = (\tilde{m}s^{-1}P + rs^{-1}U)_x = (\tilde{m}s^{-1}P + rs^{-1}uP)_x = \left((\tilde{m} + ru)s^{-1}P\right)_x \\ &= \left((\tilde{m} + ru)k(\tilde{m} + ur)^{-1}P\right)_x = (kP)_x = x_0 \equiv r \pmod{n}. \end{aligned}$$

□

Ejemplo 15. Sea E una curva elíptica definida por las soluciones de la ecuación $y^2 = x^3 + 3x + 43$ en $\mathbb{F}_{101}$. El orden de dicha curva es 97, que es un número primo, luego la propia curva es un grupo cíclico y cualquier punto distinto de $\mathscr{O}$ lo genera. Tomemos como generador al punto $P = (77, 61)$, y por lo anterior tenemos que $n = 97$.

Pongamos que la clave privada del firmante es $u = 33$ y, por tanto, su clave pública es

$$U = uP = 33(77, 61) = (29, 73).$$

Supóngase que el mensaje m a firmar tiene como resumen $\tilde{m} = \mathscr{H}(m) = 42$. El firmante elige $k = 11$ de manera aleatoria y calcula

$$kP = 11(77, 61) = (73, 5),$$

luego $r = 73$.

Posteriormente, el firmante calcula

$$k^{-1} = 11^{-1} = 53 \pmod{97}$$

$$s = k^{-1}(\tilde{m} + ur) = 53(42 + 33 \cdot 73) = 20 \pmod{97}$$

y entonces la firma del mensaje m es el par $(r,s) = (73, 20)$.

Para que otra persona verifique la firma (r,s) anterior, debe contar con la clave pública del firmante, $U = (29,73)$. En primer lugar, calcula el resumen del mensaje, $\tilde{m} = \mathscr{H}(m) = 42$. Tras comprobar que $r = 73$ y $s = 20$ pertenecen a $[1,96] \cap \mathbb{Z}$, calcula

$$z_1 = \tilde{m}s^{-1} = 42 \cdot 20^{-1} = 42 \cdot 34 = 70 \pmod{97}$$

$$z_2 = rs^{-1} = 73 \cdot 20^{-1} = 73 \cdot 34 = 57 \pmod{97}$$

Por último, se computa

$$z_1 P + z_2 U = 70(77,61) + 57(29,73) = (20,86) + (24,1) = (73,5),$$

y como la primera coordenada de $z_1 P + z_2 U$ es igual a r, se verifica la corrección de la firma.

Capítulo 4

Bitcoin

Como se ha comentado en la Sección 1.1, el artículo de Bitcoin fue publicado en 2008 bajo el seudónimo de Satoshi Nakamoto. En él, se definió Bitcoin como una red *peer-to-peer* o P2P (todos los nodos que la componen son iguales), abierta (cualquiera que lo desee puede participar sin restricciones ni justificaciones), descentralizada (no hay un nodo central que gestiona la red) y pública (tanto el protocolo de Bitcoin como el acceso a su registro de transacciones está disponible para todo el mundo).

Durante este capítulo, se usa el término *Bitcoin* para referirse a la red y el término *bitcoin* para referirse a la criptomoneda. A continuación, se explica en más detalle tanto la red como la criptomoneda.

4.1. Especificaciones

En Bitcoin, se utilizan las funciones hash SHA-256 y RIPEMD-160 descritas en el Apéndice A. La primera se usa para obtener los resúmenes de los bloques que conforman su blockchain, para los árboles de Merkle y para el algoritmo de firma digital, mientras que ambas se emplean para la generación de direcciones.

Para la autentificación de las transacciones, Bitcoin utiliza un sistema criptográfico asimétrico de firma digital basado en curva elíptica.

Estructura de un bloque de la cadena de Bitcoin

La información de un bloque de la blockchain de Bitcoin se divide en una cabecera y un listado de transacciones junto con su árbol de Merkle asociado. La cabecera contiene:

- El hash de la cabecera del bloque previo para mantener el bloque encadenado a los anteriores.
- La raíz del árbol de Merkle de las transacciones del bloque.
- Un sello temporal o *timestamp* con la hora y la fecha de creación del bloque para imposibilitar su duplicación posterior en la cadena.
- La dificultad de minado.
- Un número aleatorio llamado número de un solo uso o *nonce* (del inglés, *number used once*).

El problema del doble gasto y la prueba de trabajo

Uno de los grandes logros de Bitcoin fue resolver uno de los problemas que tuvieron las primeras criptomonedas: el problema del doble gasto. La solución de Bitcoin al problema del doble gasto en una red descentralizada viene dada por la prueba de trabajo o PoW (*Proof of Work*). El origen de este concepto se remonta a 1997, cuando Adam Back presentó el algoritmo *hashcash* de prueba de trabajo, que tenía como objetivo combatir los correos electrónicos spam haciendo que su envío tuviera un coste que no

lo hiciera rentable. Consistía en una prueba de computación costosa de efectuar, pero fácil de verificar, cuya realización requería tiempo de CPU y, en consecuencia, acarreaba un coste ligado al consumo de energía. Así, un uso individual podía ser asumido, pero el coste de un uso masivo no era despreciable. La prueba de trabajo de Bitcoin se inspira en el *hashcash* introducido por Adam Back.

La prueba de trabajo de Bitcoin consiste en la resolución de un problema basado en obtener un valor aleatorio que, al escribirlo en la cabecera del bloque, haga que el resumen de dicha cabecera sea menor que un número de 256 bits denominado *target* (denotado por t en la Figura 4.1). Este valor aleatorio se denomina como número de un solo uso o *nonce*. La solución a este problema es exigida para la incorporación de un nuevo bloque a la cadena con la finalidad de regular el ritmo de la formación de los bloques y evitar comportamientos indeseados.

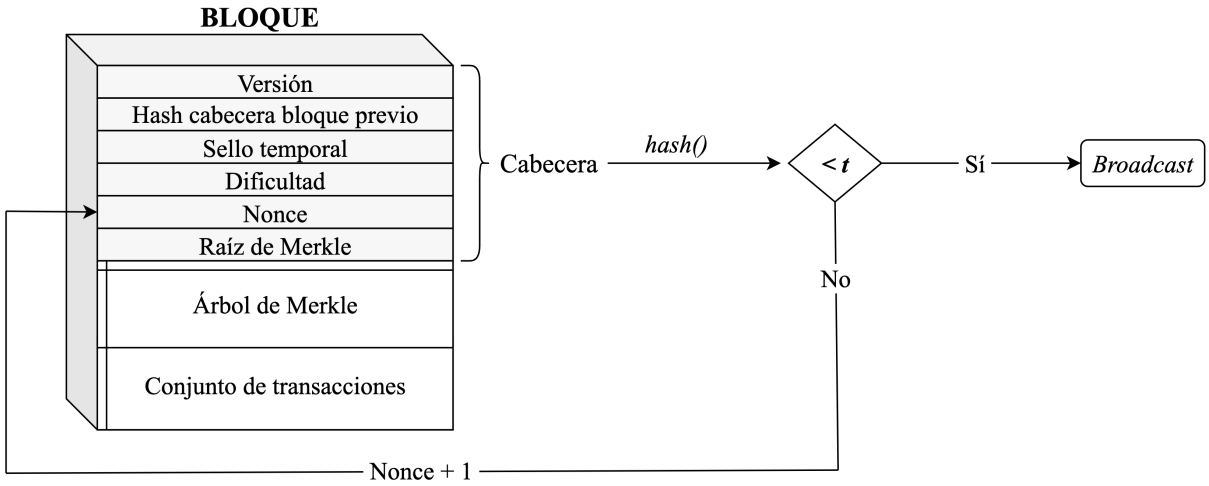


Figura 4.1: Esquema básico de la prueba de trabajo.

Al procedimiento de resolución de la prueba de trabajo se le conoce también como *minado*, y los encargados de realizarlo son los llamados *nodos mineros*. Cuando un minero encuentra un valor de *nonce* que satisface las condiciones de la prueba de trabajo, lo anexa al bloque que está construyendo y lo difunde a la red (fase de *broadcast*). Los demás mineros verifican la validez de las transacciones de ese bloque y del *nonce* fácilmente y, si todo es correcto, lo aceptan, deshacen sus bloques candidatos y comienzan a trabajar en el siguiente bloque.

Controlando la dificultad del minado, se consigue mantener un ritmo de creación de un bloque cada diez minutos aproximadamente, y el primer minero que supera la prueba de trabajo recibe una recompensa en bitcoin de nueva creación. La dificultad del minado es una medida de la cantidad de recursos necesarios para minar Bitcoin, que sube o baja dependiendo de la cantidad de potencia computacional disponible en la red.

En Bitcoin, la dificultad, que se puede denotar por *dfct*, se ajusta automáticamente cada 2016 bloques (dos semanas aproximadamente). La nueva dificultad, *dfct*$'$, se calcula de la siguiente forma

$$C = \text{máx}\left(\text{mín}\left(\frac{2016T}{T_A}, 4\right), \frac{1}{4}\right)$$

$$\text{dfct}' = \text{dfct} \cdot C$$

donde T es el tiempo ideal entre dos bloques, que en Bitcoin es de 10 minutos, y T_A el tiempo utilizado para minar los últimos 2016 bloques. Esto hace que $C \in [\frac{1}{4}, 4]$ para que los cambios de dificultad no sean demasiado bruscos. En la Figura 4.2 se muestra una gráfica con la variación de la dificultad del minado de Bitcoin en el tiempo medida en trillones (T).

La relación entre la dificultad de minado y el *target* de la prueba de trabajo de un bloque viene dada por la siguiente fórmula:

$$\text{target del bloque} = \frac{\text{target de referencia}}{\text{dfct}},$$

Figura 4.2: Variación de la dificultad del minado de Bitcoin en el tiempo. Fuente: `https://btc.com/`

y se recalcula cuando se obtiene un nuevo valor de dificultad.

En Bitcoin, el target de referencia es el del primer bloque minado, cuyo valor es 26959535291011309493156476344723991336010898738574164086137773096960.

Este ritmo regulado de la creación de bloques permite que finalmente se alcance consenso entre los nodos de la red sobre una cadena de bloques válida, en lugar de una distinta por cada nodo de la red, lo que a su vez permite impide el doble gasto.

Por cada bloque minado, el minero que consigue resolver en primer lugar su prueba de trabajo recibe una recompensa en forma de nuevos bitcoin, además de las comisiones de las transacciones del bloque. De hecho, la totalidad de los bitcoin de nueva emisión se dedican a estas recompensas de minado.

Halving

El *halving* es un proceso automatizado que reduce a la mitad los bitcoins recibidos por los mineros como recompensa por la creación de un bloque. La primera criptomoneda en poner en práctica este proceso fue Bitcoin, ya que, como está establecido en el protocolo de su red, la cantidad de bitcoin que puede llegar a existir es finita. Concretamente, de 21 millones de bitcoins, lo que al ritmo de minado actual significaría el fin de emisión en el año 2140.

Este evento se produce cada 210.000 bloques minados, lo que lleva unos cuatro años a ritmo de unos diez minutos por bloque. El reducir la recompensa de bloque para los mineros a la mitad significa que se emite menos moneda. Este proceso está diseñado para controlar la oferta de bitcoin y evitar la inflación, ya que se limita la cantidad de nuevas monedas que se pueden producir. Se estima que el próximo halving de Bitcoin ocurrirá en el año 2024.

Tipos de nodos de la red de Bitcoin

Dentro de la red Bitcoin existen diferentes tipos de nodos:

- *Nodos completos*, que son aquellos que almacenan una copia completa y actualizada de la cadena de bloques y verifican el cumplimiento del protocolo de Bitcoin.
- *Nodos podados*, que son como los nodos completos pero sólo tienen una copia parcial de la blockchain.
- *Nodos públicos*, que son nodos completos que sirven inentirrumpidamente como punto de comunicación e interconexión con otros nodos de la red para transmitir datos e información.
- *Nodos mineros*, que son nodos completos encargados de recopilar y agrupar en un bloque las transacciones que están pendientes en la red, resolviendo las pruebas de trabajo con hardware

especializado para ello. Nótese que los mineros priorizan incluir los bloques cuyas transacciones incluyen una mayor comisión.

- *Nodos ligeros*, que son nodos también conocidos como clientes SPV (*Simplified Payment Verification*) y tienen la capacidad de solicitar y recibir de los nodos públicos el estado de transacciones concretas, sin tener que descargar la base de datos de la cadena completa.

Firma digital de Bitcoin

Bitcoin utiliza el algoritmo de firma digital con curva elíptica explicado en la Sección 3.3, en el que toma como función hash $\mathscr{H}$ la función SHA-256. La curva elíptica que emplea para dicho algoritmo es la *curva de Koblitz secp256k1*, la cual está definida en el estándar SEC 2 (*Standards for Efficient Cryptography 2*) [26, p. 9] y tiene los siguientes parámetros $T = (p, a, b, P, n, h)$ definidos en (2.3):

- $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
- $a = 0$ y $b = 7$, luego esta curva tiene como ecuación de Weierstrass reducida (2.2) a

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

- $P = (x_P, y_P)$ cuyas coordenadas en hexadecimal son:

  $x_P =$ `79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798`

  $y_P =$ `483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8`

- $n = 1157920892373161954235709850086879078528375642790749043826051631415181614944337$
- $h = 1$

Esta curva ofrece ventajas como la facilidad de realizar operaciones aritméticas y la proximidad de n a p, siendo n primo. La primalidad de n y p se puede comprobar fácilmente mediante SageMath.

Uso de árboles de Merkle ligados a las transacciones

En Bitcoin se genera un árbol de Merkle (definido en la Sección 3.2) asociado a las transacciones de cada bloque con el objetivo de verificar eficientemente la integridad de las mismas. Como se ha comentado, la raíz de este árbol figurará en la cabecera del bloque. Para calcular los resúmenes de dicho árbol, Bitcoin aplica la función hash SHA-256 dos veces, esto es $H(m) = \text{SHA-256}(\text{SHA-256}(m))$, para todo elemento a resumir m.

Ejemplo 16. Para este ejemplo se hará uso de unas transacciones simplificadas ya que posteriormente se detallará su forma real. Supóngase que se quiere elaborar un bloque con las transacciones t_1, t_2, t_3:

$t_1 =$ ‘‘`Alice envía 5 bitcoins a Bob`’’

$t_2 =$ ‘‘`Bob envía 2 bitcoins a Charlie`’’

$t_3 =$ ‘‘`Charlie envía 1 bitcoin a Alice`’’

Entonces, el minero necesita generar el árbol de Merkle (véase la Figura 4.3) para incluirlo en el bloque. Nótese que para comprobar posteriormente que una transacción está ligada al árbol no es necesario recalcular ni tener acceso a todos los resúmenes. Simplemente basta rehacer los resúmenes del camino que va desde la hoja que almacena el resumen de la transacción a comprobar hasta la raíz de Merkle. Así, si se quiere comprobar si la transacción t_2 está ligada al árbol a partir de su raíz de Merkle, se calcula el resumen de t_2, denotado como h_2, y a través de los resúmenes h_1 y h_5 se recalculará ágilmente la raíz de Merkle, comprobando si el valor coincide con el que se tenía previamente. De esta forma, será necesario calcular solamente tres resúmenes: h_2, h_4 y la raíz de Merkle del árbol.

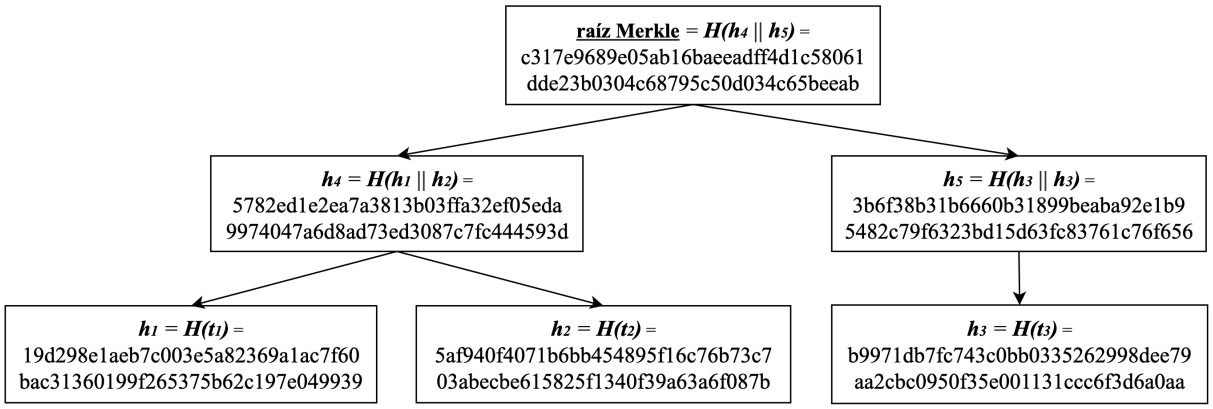


Figura 4.3: Árbol de Merkle asociado a t_1, t_2, t_3 con los resúmenes en hexadecimal.

Ejemplo 17. Obsérvese que por el efecto avalancha de las funciones hash, cualquier leve cambio en una transacción originaría una raíz de Merkle totalmente diferente, haciendo muy sencilla su detección. Supóngase que se tienen las transacciones del ejemplo anterior, pero con un pequeño cambio en t_2:

$$t_2' = \text{``Bob envía 3 bitcoins a Charlie''}$$

El árbol de Merkle asociado a estas transacciones se muestra en la Figura 4.4 y, como se puede observar, su raíz ha cambiado sustancialmente respecto a la de la Figura 4.3.

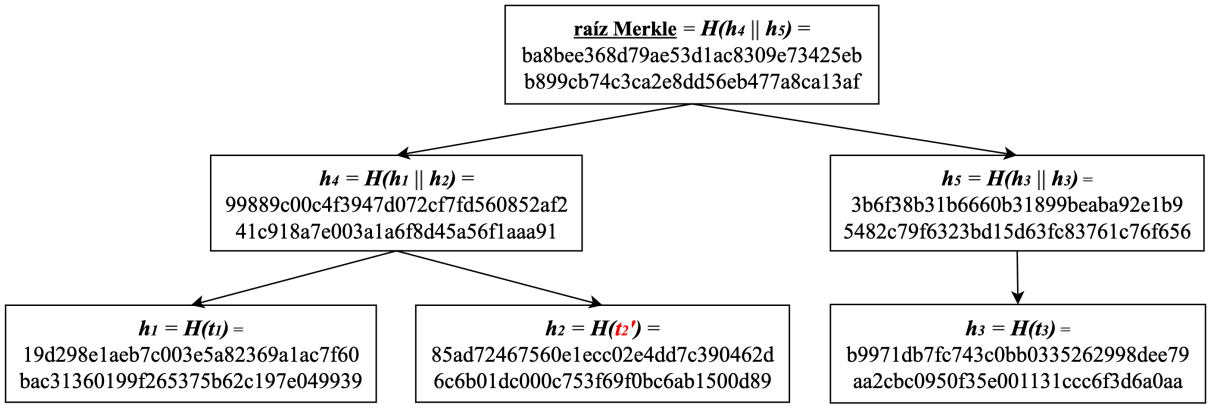


Figura 4.4: Árbol de Merkle asociado a t_1, t_2', t_3 con los resúmenes en hexadecimal.

4.2. Funcionamiento general

Cuando un usuario efectúa una transacción, esta se transmite a todos los nodos mineros de la red. Cada nodo minero forma un bloque agrupando transacciones tras verificar la validez de sus firmas y revisar que se cuenta con la suficiente liquidez comprobando los bloques previos, e intenta resolver la prueba de trabajo de dicho bloque. Cuando un nodo resuelve esta prueba, transmite el bloque a todos los nodos, que verifican el resultado de la prueba de trabajo y aceptan el bloque si todas las transacciones que figuran en él son válidas. Los nodos mineros expresan su aceptación del bloque usando el hash de su cabecera como hash previo en la creación de su siguiente bloque.

Los nodos siempre consideran correcta a la cadena más larga y se mantendrán trabajando para extenderla. Si dos mineros transmiten simultáneamente diferentes versiones del siguiente bloque, algunos nodos recibirán una antes que la otra. En ese caso, trabajarán sobre la primera que hayan recibido, pero guardarán la otra ramificación por si acaso se convierte en la más larga. El empate se romperá cuando se resuelva la siguiente prueba de trabajo y una ramificación se convierta en la más larga. Entonces, los nodos que trabajaban en la otra ramificación cambiarán automáticamente a la más larga.

Direcciones

Como se mencionó en el Capítulo 1, para operar con criptomonedas como bitcoins, será necesario un monedero o *wallet* para transferir, recibir y almacenar dichas criptomonedas y, mientras que la clave pública sirve para recibir bitcoins, la privada sirve para transferirlos.

Una dirección o *wallet address* es la referencia a la que se asocia una determinada cantidad de criptomonedas. Toda persona que quiera enviar o recibir criptomonedas necesitará siempre una dirección. El motivo de usar una dirección en lugar de la clave pública es dotar de capa extra de seguridad al criptosistema, manteniendo su robustez en el caso de que se encontrara alguna vulnerabilidad en el protocolo de firma digital. El monedero genera de forma determinista su dirección de 200 bits a partir de una clave pública k_{pub}, realizando el siguiente proceso en el que intervienen las funciones hash $\mathscr{S}$ =SHA-256 y $\mathscr{R}$ =RIPEMD-160:

1. Se calcula el resumen SHA-256 de la clave pública, $\mathscr{S}(k_{pub}) = a$.
2. Se calcula el resumen RIPEMD-160 del resumen anterior, $\mathscr{R}(a) = b$.
3. A b se le añaden 8 bits con valor 0 delante, denotemos a este nuevo número por $c := 00000000 \,\|\, b$.
4. Se calcula el resumen SHA-256 del número anterior, $\mathscr{S}(c) = d$.
5. Se calcula el resumen SHA-256 del resumen anterior, $\mathscr{S}(d) = e$.
6. Se seleccionan los 32 primeros bits de e, que se pueden denotar por $[e_{\,255:224}]$ y se añaden a c. Es decir, se toma $f := c \,\|\, [e_{\,255:224}]$, que será la dirección Bitcoin.

Transacciones

Cada transacción consta de entradas, salidas y una cabecera donde figura su resumen SHA-256 y el número de entradas y salidas. Las entradas contienen la información de las transacciones de las que se obtienen las cantidades a transferir, mientras que las salidas contienen la dirección de destino de la transacción junto con la clave pública del destinatario y la cantidad a transferir.

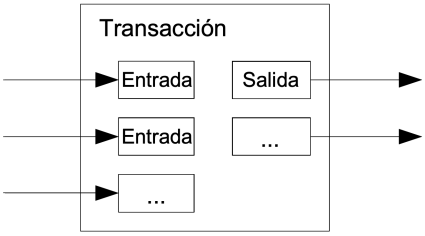


Figura 4.5: Esquema básico de una transacción en Bitcoin. Obtenida de [19, p. 5].

Se denomina UTXO (*Unspent Transaction Output*) a aquella salida que no ha sido gastada. Las UTXO se almacenan en el monedero de la dirección que tienen asociada y sólo pueden ser utilizadas por el poseedor de la clave privada ligada a la clave pública que consta en su salida.

Una UTXO sólo se puede utilizar una vez, ya que a partir de entonces dejará de ser UTXO. En la elaboración de una transacción, la suma de las cantidades de las UTXO que se usan como entrada debe ser igual a la suma de las cantidades de las salidas. Para que esto sea así, debe haber un mínimo de 3 salidas en una transacción: una salida para el principal beneficiario, otra salida para la comisión del minero y el sobrante para el propio remitente.

Para demostrar que quien crea una transacción es el propietario de la UTXO que lleva en la entrada se utiliza criptografía asimétrica (concretamente, el algoritmo de firma digital con curva elíptica). Mediante este algoritmo, se autentifica la transacción con la clave privada asociada a la clave pública de la UTXO que se utiliza en la entrada. De esta forma, la propia red garantiza la validez de las transacciones, como se muestra en la Figura 4.6.

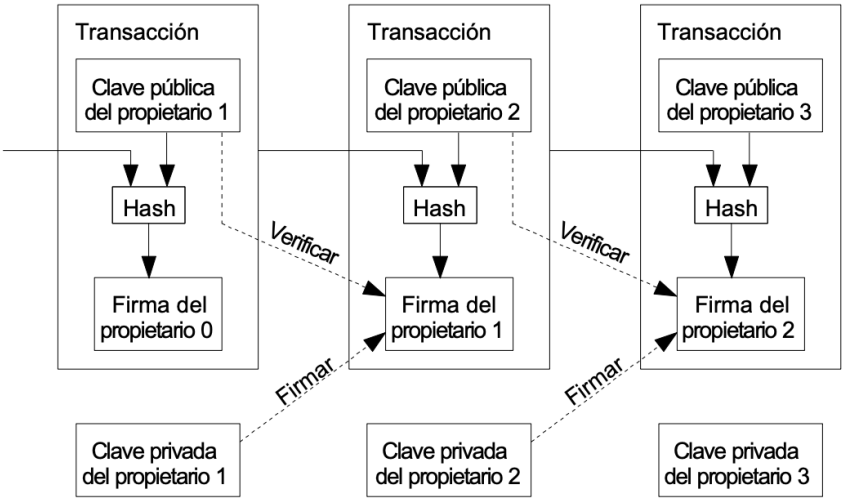


Figura 4.6: Sistema de firma digital aplicado a las transacciones. Obtenida de [19, p. 2].

Un tipo especial de transacciones son las llamadas *transacciones coinbase*. Una transacción coinbase es la primera transacción de un bloque y contiene los nuevos bitcoin acuñados para pagar el minado. Esta transacción no tiene entrada y su salida está destinada a una dirección del minero.

La transmisión de nuevas transacciones no precisa alcanzar todos los nodos. Con alcanzar a la mayoría de los nodos, entrarán en un bloque en poco tiempo. Las transmisiones de nodos también toleran mensajes perdidos. Si un nodo no recibe un bloque, lo reclamará cuando reciba el siguiente bloque y se dé cuenta de que falta uno.

Formación de la cadena de bloques

Una vez realizadas, las transacciones se transmiten entre los nodos y quedan a la espera de ser confirmadas. Los nodos completos reciben las transacciones y, una vez que verifican su validez, las retransmiten. Los mineros agrupan las transacciones en un bloque en construcción. Una vez que se mina y confirma ese bloque, cada bloque posterior que se mine sobre él le aporta un mayor nivel de seguridad. Se acepta como *suficientemente seguro* un bloque que tenga confirmados sus seis bloques siguientes.

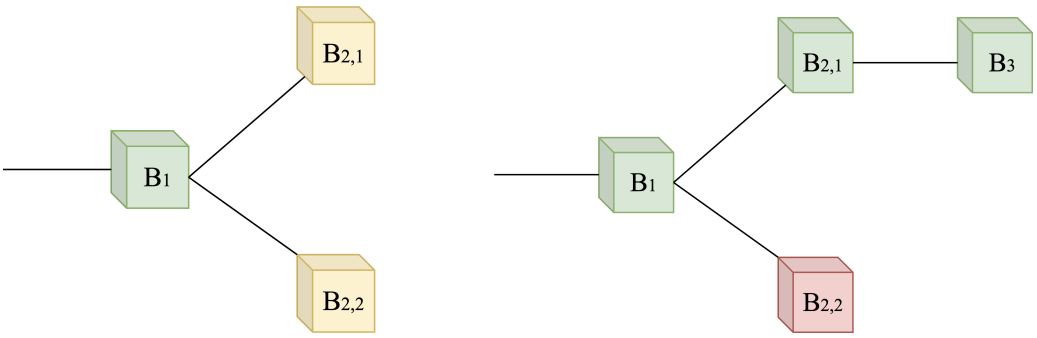
Cada minero selecciona las transacciones que desea incluir y construye su propio bloque. Si existen transacciones ya confirmadas e incluidas en bloques anteriores, se eliminan del bloque en construcción. El tamaño de un bloque tiene que ser inferior a un megabyte.

El sello temporal del bloque es incluido por el minero y comprobado por el resto de nodos. Para que se acepte un bloque, su sello temporal debe estar comprendido entre la mediana de los sellos temporales de los once últimos bloques y dos horas después de la hora actual de la red.

Ramificación de la cadena de bloques

En Bitcoin, es frecuente que en la cadena de bloques se produzca alguna ramificación (*fork*) temporalmente. Supóngase el siguiente caso: se denota B_1 al último bloque de la cadena de la cual tienen copia los mineros M_1 y M_2. Supóngase además que M_1 y M_2 forman un bloque, $B_{2,1}$ y $B_{2,2}$ respectivamente, y resuelven la PoW prácticamente al mismo tiempo. En ese caso, ambos mineros transmitirán su bloque al resto de nodos de la red (véase la Figura 4.7a), pero solamente uno de ellos acabará prosperando en la estructura. Cada nodo trabajará sobre la primera ramificación recibida, pero se guardará si le llega otra ramificación por si acaso se convierte en la más larga, ya que el criterio de consenso para decidir la rama de la bifurcación que se toma como válida es elegir la que cuente con mayor trabajo por parte de los mineros. Nótese que se puede suponer, sin perder generalidad, que un minero que considera $B_{2,1}$ como válido enlaza otro bloque minado B_3. Entonces, los mineros que tenían como válido el bloque $B_{2,2}$ verán que B_3 es incompatible, por lo que retrocederán en su cadena hasta llegar al bloque común B_1 y

tomarán como válida la rama con mayor trabajo por parte de los mineros, es decir, $B_1 - B_{2,1} - B_3$ (véase la Figura 4.7b). En este caso, la otra ramificación se descartará (se perderá el bloque $B_{2,2}$). Estos bloques descartados reciben el nombre de *bloques huérfanos*.



(a) Primera fase (b) Segunda fase

Figura 4.7: Ejemplo de ramificación de la blockchain.

Las transacciones contenidas en un bloque que queda huérfano quedan a la espera de ser incluidas en un nuevo bloque, mientras que el minero que había realizado la prueba de trabajo de dicho bloque no recibirá su recompensa.

Optimización del espacio de almacenamiento

Para evitar que la base de datos global crezca en exceso es necesario deshacerse de información prescindible. Así, se suprimen de bloques previos las transacciones cuyas salidas ya han sido empleadas como entradas de otras transacciones, y por tanto ya no son necesarias para las verificaciones. Por este motivo se incluye únicamente la raíz de Merkle en la cabecera de los bloques. De esta forma, se eliminan estas transacciones y las ramas del árbol innecesarias preservando la solidez de la cadena a la vez que se optimiza el espacio en memoria (véase Figura 4.8). De esta forma y tal y como se ha visto en el apartado de la Sección 4.1 en el que se trata el uso de árboles de Merkle en Bitcoin, es posible realizar las validaciones pertinentes pese a haber prescindido de algunas transacciones (y ramas del árbol) de la cadena.

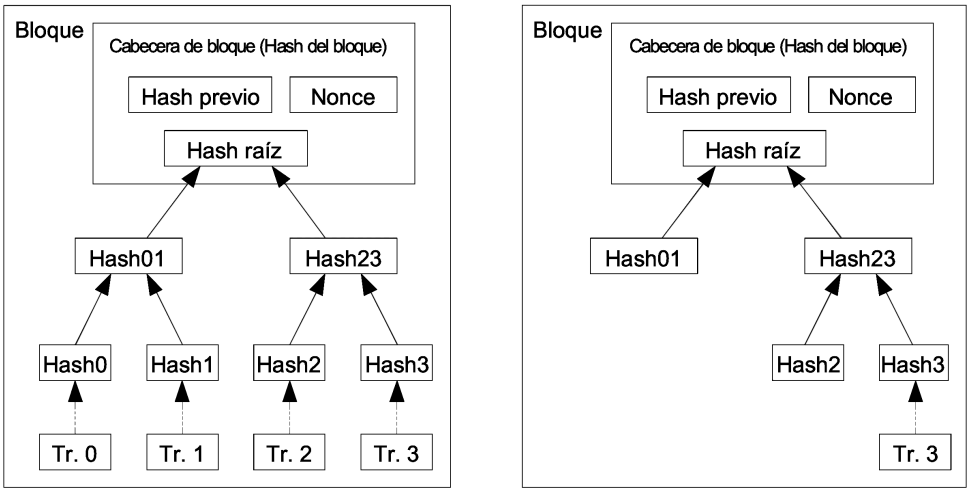


Figura 4.8: Antes y después de eliminar las transacciones 0, 1 y 2 del bloque. Fuente: [19, p. 4]

Bibliografía

[1] D. ARROYO, J. DÍAZ, L. HERNÁNDEZ, *Blockchain (Qué Sabemos de)*, Catarata, 2019.

[2] ANDERS BROWNWORTH, *Blockchain Demo, Publick Key / Private Key Demo*, https://github.com/anders94, 2016.

[3] DAVID CHAUM, *Blind Signatures for Untraceable Payments*, https://chaum.com/wp-content/uploads/2022/01/Chaum-blind-signatures.pdf, 1982.

[4] LAURA CUESTA, *El auge de las criptomonedas en la era digital*, https://www.lavanguardia.com/vida/junior-report/20220927/8518224/auge-criptomonedas-digital.html, 2022.

[5] WEI DAI, *b-money*, http://www.weidai.com/bmoney.txt, 1998.

[6] W. DIFFIE, M. HELLMAN, *New directions in cryptography*, IEEE Transactions on Information Theory, Vol. 22, Núm. 6, 1976, Págs. 644–654.

[7] TAHER ELGAMAL, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, Núm. 4, 1985, Págs. 469–472.

[8] ETHEREUM, *Introduction to smart contracts*, https://ethereum.org/en/smart-contracts/, 2023.

[9] EUROPEAN CENTRAL BANK, *Virtual currency schemes - a further analysis*, https://www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemesen.pdf, 2015.

[10] A. FUSTER, L. HERNÁNDEZ, A. MARTÍN, F. MONTOYA, J. MUÑOZ, *Criptografía, protección de datos y aplicaciones*, RA-MA Editorial, 2012.

[11] V. GAYOSO, L. HERNÁNDEZ, A. MARTÍN, *Criptografía con curvas elípticas*, Consejo Superior de Investigaciones Científicas, 2018.

[12] C. HICKS, M. ADAMS, *Different Types of Cryptocurrencies*, Forbes, https://www.forbes.com/advisor/investing/cryptocurrency/different-types-of-cryptocurrencies/, 2023.

[13] NEAL KOBLITZ, *Elliptic curve cryptosystems*, Mathematics of Computation, Vol. 48, Núm. 177, 1987, Págs. 203–209.

[14] MIKEL LEZAUN, *Blockchain. Bitcoin*, La Gaceta de la RSME, Vol. 24, Núm. 3, 2021, Págs. 533–558.

[15] MÓNICA MENA, *La adopción de las criptomonedas en el mundo*, https://es.statista.com/grafico/18425/adopcion-de-las-criptomonedas-en-el-mundo/, 2022.

[16] R.C. MERKLE, *Method of providing digital signatures*, US Patent n. 4309569A, 1982.

[17] *Monero*, https://www.getmonero.org/get-started/what-is-monero/.

[18] VICTOR S. MILLER, *Use of elliptic curves in cryptography*, Advances in Cryptology - CRYPTO, Vol. 85, Núm. 218, 1986, Págs. 417–426.

[19] SATOSHI NAKAMOTO, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.

[20] B. PRENEEL, H. DOBBERTIN, A. BOSSELAERS, *The Cryptographic Hash Function RIPEMD-160*, RSA Laboratories, 1997.

[21] *Online RIPEMD160 Hash Calculator*, `https://md5calc.com/hash/ripemd160`.

[22] *Ripple*, `https://ripple.com/company/`.

[23] LORENA RAMÍREZ, *Del dinero fiat a las criptomonedas*, `https://www.iebschool.com/blog/dinero-fiat-criptomonedas-evolucion-diferencias-finanzas/`, 2022.

[24] JOSEPH H. SILVERMAN, *The Arithmetic of Elliptic Curves*, Second Edition, Springer, 2009.

[25] STANDARDS FOR EFFICIENT CRYPTOGRAPHY, *SEC 1: Elliptic Curve Cryptography*, Certicom Research, 2009.

[26] STANDARDS FOR EFFICIENT CRYPTOGRAPHY, *SEC 2: Recommended Elliptic Curve Domain Parameters*, Certicom Research, 2010.

[27] *SHA256 online hash function*, `https://emn178.github.io/online-tools/sha256.html`, 2017.

[28] NICK SZABO, *Bit Gold: Towards Trust-Independent Digital Money*, `https://web.archive.org/web/20140406003811/http://szabo.best.vwh.net/bitgold.html`, 2005.

[29] *Tether*, `https://tether.to/es/`.

[30] *USD Coin*, `https://www.circle.com/en/usdc`.

[31] LAWRENCE C. WASHINGTON, *Elliptic Curves. Number Theory and Cryptography*, Chapman & Hall / CRC, 2008.

[32] *ZCash*, `https://z.cash/learn/what-is-zcash/`.

Apéndice A

Funciones hash SHA-256 y RIPEMD-160

Notación. Se denomina *palabra* a una secuencia de 32 bits.

Notación. Denotaremos por "$\|$" al operador de concatenación. Por ejemplo, si $x := 1001$ e $y := 10001$, entonces $x \| y = 100110001$.

SHA-256

El resumen de esta función tiene una longitud fija de 256 bits. Para calcular el resumen SHA-256 de una entrada de texto cualquiera m se siguen los siguientes pasos:

Paso 1: De texto plano a bloques de 512 bits

1. Se convierte el texto de entrada m a binario siguiendo el estándar UTF-8. Denotemos a este número por a.
2. Se le concatenan a dicho número los bits `1000`, denotemos a este número por $b := a \| \texttt{1000}$.
3. A continuación, se le concatenan al número anterior p ceros, siendo

$$p := \text{mín}\{j \in \mathbb{N} \mid \text{longitud}(b) + j \equiv 448 \pmod{512}\},$$

   siendo longitud(b) el número de cifras de b.

   Es decir, tomamos $c := b \| \texttt{0} \| \overset{(p)}{\cdots} \| \texttt{0}$. Se tiene así un último bloque de 448 bits y, según sea la longitud del texto de entrada, uno, varios o ningún bloque de 512 bits.
4. El último bloque de 448 bits se completa hasta 512 bits. Sea d el número de cifras de a representado como número binario de 64 cifras (es decir, añadiendo ceros a izquierda si fuera necesario), se toma $e := c \| d$.
5. Se divide e en bloques de 512 bits, y cada uno de ellos se subdivide en 16 palabras, las cuales denotaremos por $W_0, \ldots, W_{15}$.

Ejemplo 18. Pongamos que queremos calcular el resumen SHA-256 del texto plano "unizar23". En este primer paso, se realizarían las siguientes acciones:

1. Para convertir el texto de entrada $m := \texttt{unizar23}$ a binario, se va a asociar cada carácter con su correspondiente codificación en el estándar UTF-8 (8-bit Unicode Transformation Format), cuya correspondencia se puede encontrar en `https://www.charset.org/utf-8`. Así, por ejemplo el carácter 'u' se codifica con el número 117, que expresado en 8 bits es `01110101`, y el carácter '2' se codifica con el número 50, que expresado en 8 bits es `00110010`. Así, se obtiene

$$a = \texttt{01110101011011100110100101111010011000010111001000110010001100011},$$

ya que la correspondencia UTF-8 carácter a carácter es la siguiente:

01110101	01101110	01101001	01111010	01100001	01110010	00110010	00110011
u	*n*	*i*	*z*	*a*	*r*	*2*	*3*

2. Se le concatenan a dicho número los bits `1000`, tomando

$$b = \texttt{01110101011011100110100101111010011000010111001000110010001100111000}.$$

3. Como la longitud de b es de 68 bits, se concatenan 380 bits con valor `0`, obteniendo c.

4. Como la longitud de a es de 64 bits, tenemos que

$$d = \texttt{0000000000000000000000000000000000000000000000000000000001000000}$$

y se toma $e := c \,\|\, d$.

5. De esta manera, se obtiene un único bloque que se muestra en la Figura A.1

Bloque - 512 Bits

$W_0 =$	01110101	01101110	01101001	01111010	01100001	01110010	00110010	00110011	$= W_1$
$W_2 =$	10000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	$= W_3$
$W_4 =$	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	$= W_5$
$W_6 =$	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	$= W_7$
$W_8 =$	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	$= W_9$
$W_{10} =$	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	$= W_{11}$
$W_{12} =$	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	$= W_{13}$
$W_{14} =$	00000000	00000000	00000000	00000000	00000000	00000000	00000000	01000000	$= W_{15}$

Figura A.1: Bloque resultante de aplicar el paso 1 al texto plano “unizar23”.

Paso 2: Extensión de los bloques

Se realiza la extensión de los bloques de 512 bits a 2048 bits por recurrencia, hallando

$$W_i := \sigma_1(W_{i-2}) \boxplus W_{i-7} \boxplus \sigma_0(W_{i-15}) \boxplus W_{i-16}, \quad i = 16, 17, \ldots, 63$$

con

$$\sigma_0(W) := (W \ggg 7) \oplus (W \ggg 18) \oplus (W \gg 3),$$
$$\sigma_1(W) := (W \ggg 17) \oplus (W \ggg 19) \oplus (W \gg 10),$$

siendo

- $W \ggg k$ una rotación de k bits hacia la derecha aplicada a W,
- $W \gg k$ un desplazamiento de k bits hacia la derecha aplicado a W,
- $\oplus$ la suma bit a bit (XOR) que se detalla en la Tabla A.1, y
- $\boxplus$ la suma módulo 2^{32}, de forma que su resultado se puede expresar como una palabra.

Ejemplo 19. Veamos algunos ejemplos de las operaciones anteriores, que son aplicables sobre operandos de cualquier número de bits:

- $101\mathbf{001} \ggg 4 = \mathbf{001}10$
- $101\mathbf{001} \gg 4 = \mathbf{0000}10$
- $10101 \oplus 11001 = 01100$
- $10101 \boxplus_5 11001 = 01110$ (denotando por $\boxplus_5$ a la suma módulo 2^5)

Paso 3: Cálculo de resúmenes intermedios

Para este paso, vamos a definir las constantes K_i para $i = 0, \ldots, 63$, cuyos valores son los de los primeros 32 bits de la parte fraccionaria de la raíz cúbica de los 64 primeros números primos respectivamente.

Adicionalmente, se inicializan las variables $A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0$ de forma que tomen el valor de los primeros 32 bits de la parte fraccionaria de la raíz cuadrada de los 8 primeros números primos respectivamente.

Ejemplo 20. Hallemos A_0. Para ello, nótese que

$$\sqrt{2} \approx 1,41421356237309,$$

por lo que se buscará expresar $n = 0,41421356237309$ en binario.

Para ello, se seguirá el procedimiento expuesto en la Sección 2.1.1:

i	$n \cdot 2^i$	$\lfloor n \cdot 2^i \rfloor$	d_{-i}
1	0,82842712474619	0	0
2	1,65685424949238	1	1
3	3,31370849898476	3	1
4	6,627416997969521	6	0
5	13,254833995939042	13	1

De esta forma, n en binario se expresará como `0,01101...` y si se sigue con este método hasta conseguir 32 bits, se halla finalmente

$$A_0 = \texttt{01101010000010011110011001100111}.$$

De cada conjunto de 64 palabras se obtiene en un resumen intermedio realizando una iteración por cada palabra de dicho conjunto. En la iteración i se utilizarán los valores $A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i$, la palabra i (W_i) y la constante i (K_i) para obtener $A_{i+1}, B_{i+1}, C_{i+1}, D_{i+1}, E_{i+1}, F_{i+1}, G_{i+1}, H_{i+1}$. El proceso iterativo que se realiza se describe en la Figura A.2.

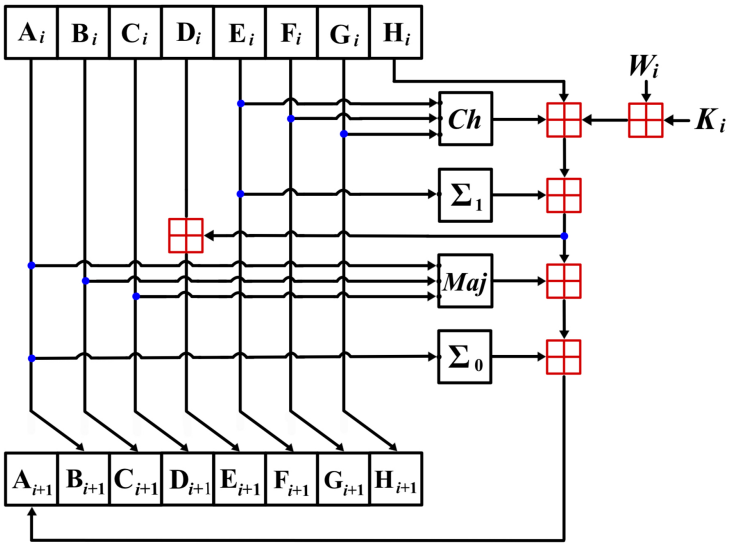


Figura A.2: Cálculo de una iteración para la obtención de los resúmenes intermedios en SHA-256.

Las operaciones utilizadas en el procedimiento descrito en la Figura A.2 son las siguientes:

$$\begin{aligned} \mathrm{Ch}(E,F,G) &= (E \wedge F) \oplus (\neg E \wedge G) \\ \mathrm{Maj}(A,B,C) &= (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C) \\ \Sigma_0(A) &= (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22) \\ \Sigma_1(E) &= (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25) \end{aligned}$$

y los operadores binarios en los que se basan se detallan en la Tabla A.1.

X	Y	$X \wedge Y$	$X \vee Y$	$X \oplus Y$	$\neg X$
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Tabla A.1: Definición de los operadores binarios AND, OR, XOR y negación.

Cuando finalice este proceso iterativo, se habrán calculado las variables A_{64}, B_{64}, C_{64}, D_{64}, E_{64}, F_{64}, G_{64}, H_{64}.

Paso 4: Últimos cálculos

Por último, se computa

$$A = A_0 \boxplus A_{64}, \quad B = B_0 \boxplus B_{64}, \quad C = C_0 \boxplus C_{64}, \quad D = D_0 \boxplus D_{64},$$
$$E = E_0 \boxplus E_{64}, \quad F = F_0 \boxplus F_{64}, \quad G = G_0 \boxplus G_{64}, \quad H = H_0 \boxplus H_{64}.$$

para finalmente obtener, concatenando estas variables, el resumen SHA-256 de m

$$\mathscr{S}(m) = A \,\|\, B \,\|\, C \,\|\, D \,\|\, E \,\|\, F \,\|\, G \,\|\, H.$$

Ejemplo 21. Veamos algunos ejemplos de resultados (representados en hexadecimal) de la función $\mathscr{S}$ = SHA-256 calculados mediante [27]:

$m =$ ‘‘Hola’’
$\mathscr{S}(m) =$ `e633f4fc79badea1dc5db970cf397c8248bac47cc3acf9915ba60b5d76b0e88f`
$m =$ ‘‘hola’’
$\mathscr{S}(m) =$ `b221d9dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79`
$m =$ ‘‘Hola, esto es un texto de ejemplo’’
$\mathscr{S}(m) =$ `c99cdc468e763e4f923ae3f8679386efbfc03e2d4cf4f4b3649689b8d467cea5`

RIPEMD-160

El resumen de esta función tiene una longitud fija de 160 bits. Para calcularlo, se hará uso de las operaciones primitivas siguientes:

- $W \lll k$, rotación de k bits hacia la izquierda aplicada a W,
- los operadores binarios definidos en la Tabla A.1, y
- $\boxplus$, la suma módulo 2^{32}, de forma que su resultado se puede expresar como una palabra.

Paso 1: De texto plano a palabras

Se realiza el Paso 1 de la sección A para dividir el texto de entrada en bloques de 512 bits, pudiendo dividir cada bloque en 16 palabras. Sin embargo, a diferencia de SHA-256, que utilizaba el sistema *big-endian*, esta función utiliza el sistema *little-endian*. Esto es, a la hora de almacenar los datos en memoria, en el sistema *big-endian* se almacenan los bytes en el orden “natural"(es decir, los bytes más significativos primero); mientras que en el sistema *little-endian*, los bytes menos significativos se almacenan primero.

Tras este paso, el mensaje se ha dividido en t conjuntos de 16 palabras, que denotaremos por X_i con $0 \leq i \leq t-1$. Para referirnos a la palabra j del bloque i, se usará la notación $X_{i,j}$ con $0 \leq j \leq 15$.

Paso 2: Compresión

Se inicializa un conjunto de cinco palabras cuyo valor en hexadecimal es el siguiente

$$h_0 := \texttt{0x67452301} \qquad h_1 := \texttt{0xefcdab89} \qquad h_2 := \texttt{0x98badcfe}$$
$$h_3 := \texttt{0x10325476} \qquad h_4 := \texttt{0xc3d2e1f0}$$

y se realiza lo que se conoce como proceso de compresión y se expone en el Algoritmo 3.

Algoritmo 3 Proceso de compresión de RIPEMD-160 en pseudocódigo (Fuente: [20, p. 10])

desde $i \leftarrow 0$ **hasta** $t-1$ **hacer**
- $A \leftarrow h_0$
- $B \leftarrow h_1$
- $C \leftarrow h_2$
- $D \leftarrow h_3$
- $E \leftarrow h_4$
- $A' \leftarrow h_0$
- $B' \leftarrow h_1$
- $C' \leftarrow h_2$
- $D' \leftarrow h_3$
- $E' \leftarrow h_4$
- **desde** $j \leftarrow 0$ **hasta** 79 **hacer**
  - $T \leftarrow \Big(\big(A \boxplus f(j,B,C,D) \boxplus X_{i,r(j)}\big) \boxplus K(j)\big) \lll s(j)\Big) \boxplus E$
  - $A \leftarrow E$
  - $E \leftarrow D$
  - $D \leftarrow C \lll 10$
  - $C \leftarrow B$
  - $B \leftarrow T$
  - $T \leftarrow \Big(\big(A' \boxplus f(79-j,B',C',D') \boxplus X_{i,r'(j)} \boxplus K'(j)\big) \lll s'(j)\Big) \boxplus E'$
  - $A' \leftarrow E'$
  - $E' \leftarrow D'$
  - $D' \leftarrow C' \lll 10$
  - $C' \leftarrow B'$
  - $B' \leftarrow T$
- **fin bucle**
- $T \leftarrow h_1 \boxplus C \boxplus D'$
- $h_1 \leftarrow h_2 \boxplus D \boxplus E'$
- $h_2 \leftarrow h_3 \boxplus E \boxplus A'$
- $h_3 \leftarrow h_4 \boxplus A \boxplus B'$
- $h_4 \leftarrow h_0 \boxplus B \boxplus C'$
- $h_0 \leftarrow T$

fin bucle

En dicho algoritmo se hace uso de:

- las operaciones bit a bit

$$\begin{aligned}
f(j,x,y,z) &= x \oplus y \oplus z && \forall j : 0 \leq j \leq 15 \\
f(j,x,y,z) &= (x \wedge y) \vee (\neg x \wedge z) && \forall j : 16 \leq j \leq 31 \\
f(j,x,y,z) &= (x \vee \neg y) \oplus z && \forall j : 32 \leq j \leq 47 \\
f(j,x,y,z) &= (x \wedge z) \vee (y \wedge \neg z) && \forall j : 48 \leq j \leq 63 \\
f(j,x,y,z) &= x \oplus (y \vee \neg z) && \forall j : 64 \leq j \leq 79
\end{aligned}$$

- las funciones r y r' definidas para seleccionar la palabra dentro del bloque

$$\begin{aligned} r(j) &= j \quad \forall j : 0 \leq j \leq 15 \\ r(16.,31) &= 7,4,13,1,10,6,15,3,12,0,9,5,2,14,11,8 \\ r(32.,47) &= 3,10,14,4,9,15,8,1,2,7,0,6,13,11,5,12 \\ r(48.,63) &= 1,9,11,10,0,8,12,4,13,3,7,15,14,5,6,2 \\ r(64.,79) &= 4,0,5,9,7,12,2,10,14,1,3,8,11,6,15,13 \\ r'(0.,15) &= 5,14,7,0,9,2,11,4,13,6,15,8,1,10,3,12 \\ r'(16.,31) &= 6,11,3,7,0,13,5,10,14,15,8,12,4,9,1,2 \\ r'(32.,47) &= 15,5,1,3,7,14,6,9,11,8,12,2,10,0,4,13 \\ r'(48.,63) &= 8,6,4,1,3,11,15,0,5,12,2,13,9,7,10,14 \\ r'(64.,79) &= 12,15,10,4,1,5,8,7,6,2,13,14,0,3,9,11 \end{aligned}$$

- las funciones s y s' definidas para alternar la cantidad del rotado hacia la izquierda

$$\begin{aligned} s(0..15) &= 11,14,15,12,5,8,7,9,11,13,14,15,6,7,9,8 \\ s(16..31) &= 7,6,8,13,11,9,7,15,7,12,15,9,11,7,13,12 \\ s(32..47) &= 11,13,6,7,14,9,13,15,14,8,13,6,5,12,7,5 \\ s(48..63) &= 11,12,14,15,14,15,9,8,9,14,5,6,8,6,5,12 \\ s(64..79) &= 9,15,5,11,6,8,13,12,5,12,13,14,11,8,5,6 \\ s'(0..15) &= 8,9,9,11,13,15,15,5,7,7,8,11,14,14,12,6 \\ s'(16..31) &= 9,13,15,7,12,8,9,11,7,7,12,7,6,15,13,11 \\ s'(32..47) &= 9,7,15,11,8,6,6,14,12,13,5,14,13,13,7,5 \\ s'(48..63) &= 15,5,8,11,14,14,6,14,6,9,12,9,12,5,15,8 \\ s'(64..79) &= 8,5,12,9,12,5,14,6,8,13,6,5,15,13,11,11 \end{aligned}$$

- las palabras constantes
  - $K_2,\ldots,K_5$, cuyo valor es el de la parte entera de la multiplicación de 2^{30} por la raíz cuadrada de los cuatro primeros números primos respectivamente.
  - $K'_1,\ldots,K'_4$, cuyo valor es el de la parte entera de la multiplicación de 2^{30} por la raíz cúbica de los cuatro primeros números primos respectivamente.
  - K_1 y K'_5, con valor nulo.

Paso 3: Obtención del resumen

En el paso anterior se han recalculado los valores de h_0,h_1,h_2,h_3,h_4. Ahora se obtiene, concatenando estas variables, el resumen RIPEMD-160 de m

$$\mathscr{R}(m) = h_0 \,\|\, h_1 \,\|\, h_2 \,\|\, h_3 \,\|\, h_4.$$

Ejemplo 22. Veamos algunos ejemplos de resultados (representados en hexadecimal) de la función $\mathscr{R}$ = RIPEMD-160 calculados mediante [21]:

$$\begin{aligned} m &= \text{``}\texttt{Hola}\text{''} \\ \mathscr{S}(m) &= \texttt{26e20144a449a535384453a2bc0b81a0cd39c4d6} \\ m &= \text{``}\texttt{hola}\text{''} \\ \mathscr{S}(m) &= \texttt{5b559f7d90a5cca70f7a3d5e2ca312bc7c1681df} \\ m &= \text{``}\texttt{Hola, esto es un texto de ejemplo}\text{''} \\ \mathscr{S}(m) &= \texttt{5afcfb4b58fe2c6f320fd91567677f4a72f086c9} \end{aligned}$$

Apéndice B

Simulación con una blockchain sencilla

En este apéndice se van a realizar ejemplos explicativos por partes sobre el software de simulación de una blockchain sencilla obtenido de [2]. En primer lugar, se van a repasar algunos conceptos. Como se ha expuesto en el Capítulo 4, la prueba de trabajo consiste en la resolución de un problema basado en encontrar colisiones de funciones hash por fuerza bruta, lo que se basa en obtener un valor aleatorio *nonce* que dé lugar a un resumen que sea menor que un número de 256 bits (*target*), es decir, que expresado en base hexadecimal, tenga un cierto número de ceros a la izquierda. Adicionalmente, conviene recordar que la información de un bloque se divide en una cabecera y un conjunto de transacciones. Las cabeceras de los bloques de esta simulación no contarán con todos los datos que se mencionaron en el Capítulo 4 por simplicidad, pero sí con algunos fundamentales, como son el número de bloque, el hash de la cabecera del bloque previo y el *nonce*.

En estos ejemplos, se considerará un *target* equivalente a obtener un resumen que comience por (al menos) cuatro ceros y se utilizará la función hash SHA-256. Además, un bloque minado se representará con fondo verde y un bloque sin minar con fondo rojo. En algunas figuras no se mostrarán los resúmenes completos con el objetivo de hacerlas más legibles.

Ejemplo 1: Minado de un bloque

Supóngase que se tiene una cadena formada por dos bloques (números de bloque 1 y 2) y se quiere añadir el bloque número 3 (Figura B.1).

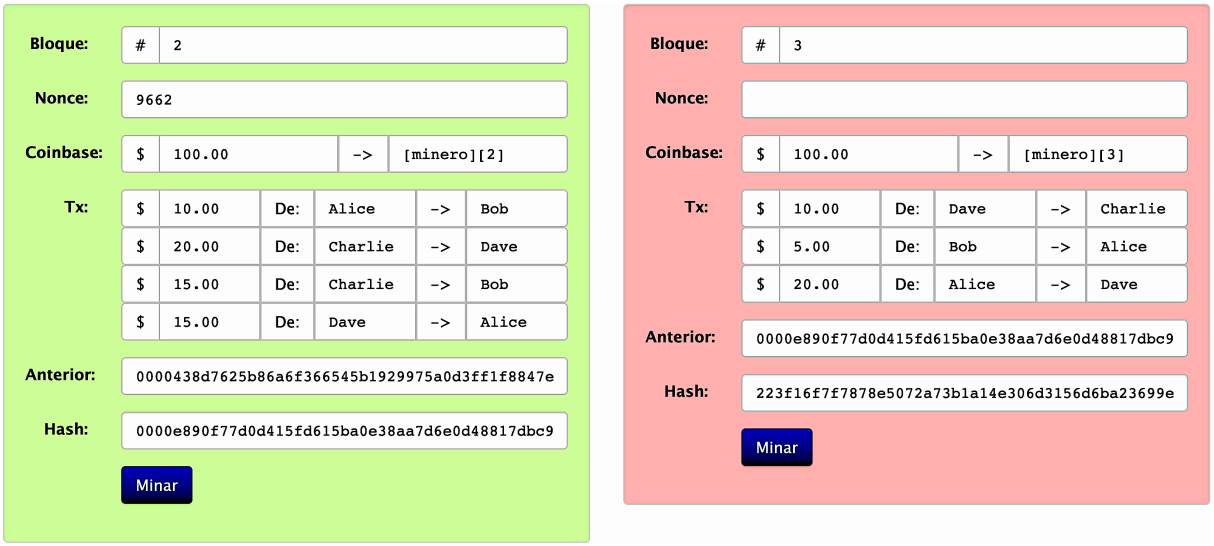


Figura B.1: Situación previa al minado de un nuevo bloque.

Entonces será necesario minar el bloque resolviendo la prueba de trabajo es decir, hallando un valor

de *nonce* que haga que el hash del bloque comience por cuatro ceros. Se recuerda que, como se muestra en la Figura B.2, el incrementar el *nonce* una unidad cambia completamente el resultado de la función hash sin un patrón que permita resolver de manera más ágil el problema que por fuerza bruta. Se va a

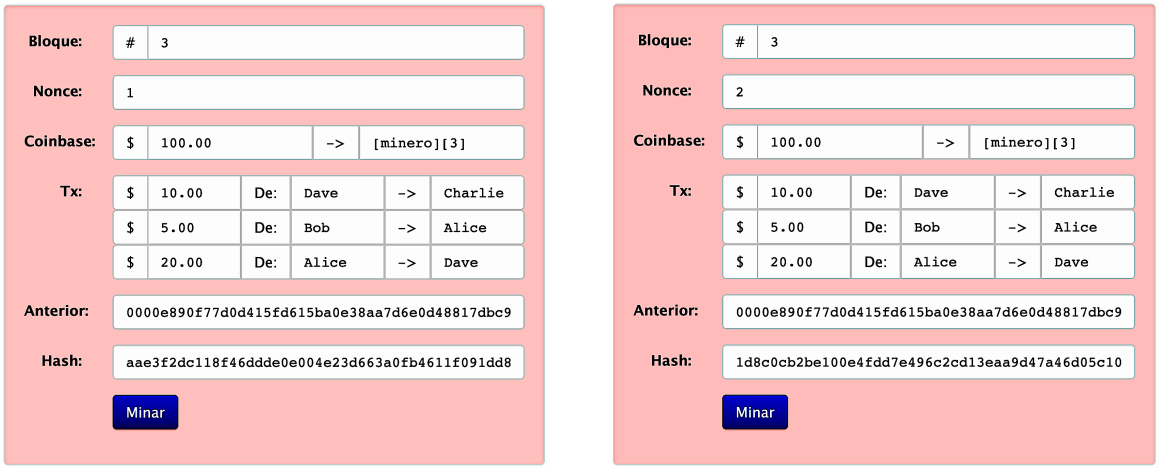


Figura B.2: Incremento del *nonce* del bloque 3.

hacer que el ordenador compute el *nonce* necesario para resolver la prueba de trabajo (Figura B.3).

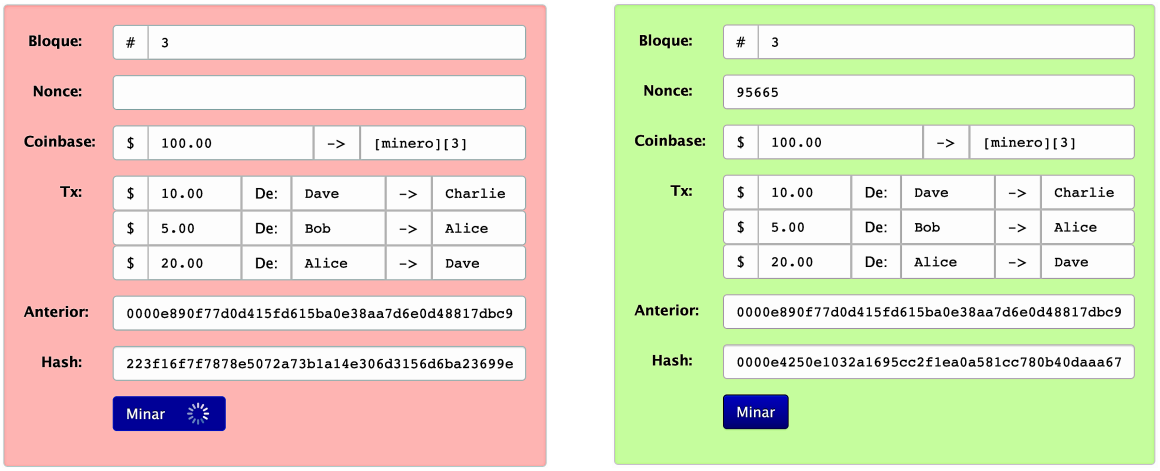


Figura B.3: Minado del bloque 3.

Así, cuando la prueba de trabajo del bloque 3 haya sido resuelta, el minero transmitirá al resto de nodos el resultado y tras realizar las correspondientes verificaciones estos lo añadirán a su cadena.

Ejemplo 2: Modificación de datos en un bloque

Se va a analizar qué ocurre al alterar algún dato del bloque una vez se encuentra en la cadena. Por ejemplo, se modifican sus datos para que la primera transacción no *coinbase* de Dave a Charlie sea de 10.01$ en vez de 10.00$ (Figura B.4). En tal caso, se puede observar que el resumen del bloque ya no cumple con el *target* establecido, por lo que se detecta rápidamente que el bloque es erróneo.

Para solventar esto, habría que volver a minar dicho bloque para que se reconociera como válido (Figura B.5).

La manipulación de datos en la cadena alcanza mayor complejidad cuando se produce en un bloque con un número considerable de bloques posteriores confirmados en la cadena. Supóngase que se parte de la situación en la que se había minado el bloque 3 (Figura B.6) y además ha pasado un periodo de tiempo desde que esto ocurrió, por lo que nuevos bloques se han ido añadiendo a la cadena, alcanzándose un total de 5 bloques en la misma.

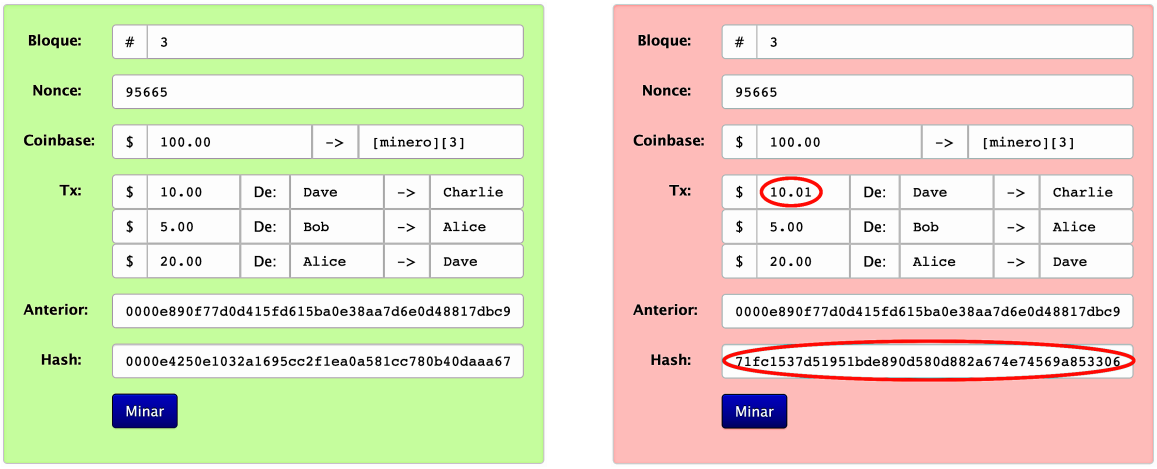


Figura B.4: Modificación de un dato del último bloque.

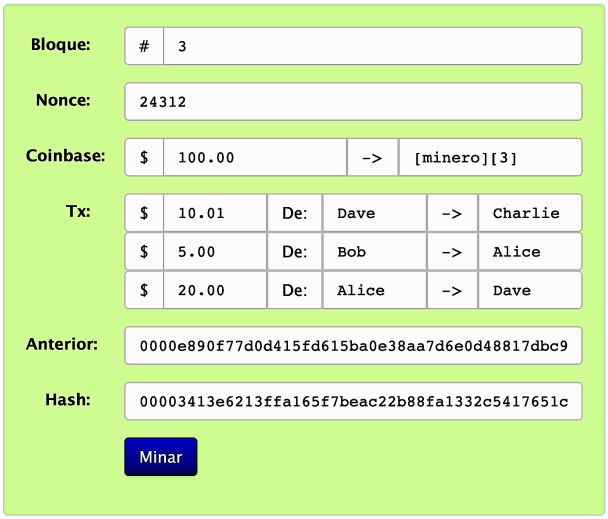


Figura B.5: Reminado del último bloque.

Si ahora se produce una modificación de algún dato, por ejemplo, que la segunda transacción no *coinbase* de Charlie a Dave sea de 30.00$ en vez de 20.00$ (Figura B.7), se tiene que se reconocen como inválidos el bloque 2 y todos sus sucesores. Esto es así ya que al modificarse el resumen del bloque 2, se modifica el resumen previo del bloque 3, lo que a su vez cambia su resumen. Esto ocurre sucesivamente hasta el último bloque de la cadena.

Además, es importante reseñar que el hecho de que múltiples nodos de la red P2P tienen la copia de la cadena (Figura B.8) es vital para la detección de estas modificaciones, ya que, aún en el caso de que se hubiera conseguido recalcular los *nonces* de los sucesivos bloques (cosa de por sí ya complicada si hay que reminar un número considerable), con una simple comprobación del hash del último bloque de la cadena se puede determinar por mayoría cuál es la cadena válida. En la Figura B.9 se muestra que en el caso de tener tres copias (pares A, B, C), se decide por mayoría que la copia de los pares B y C es la correcta.

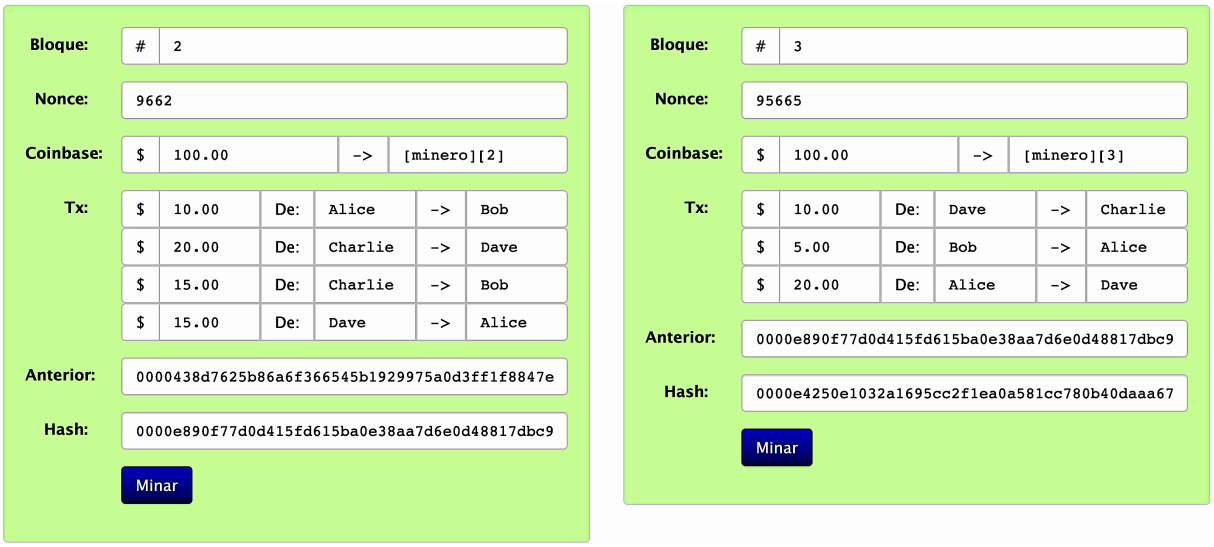


Figura B.6: Bloques 2 y 3 de la cadena.

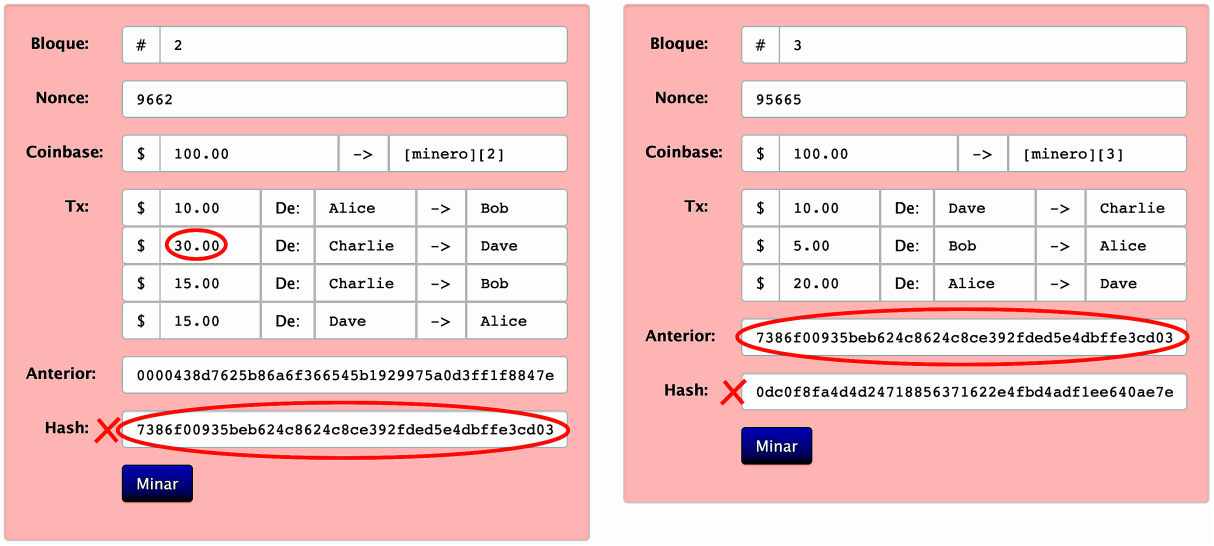


Figura B.7: Bloques 2 y 3 tras la modificación del bloque 2.

Figura B.8: Bloques 2 y 3 de los tres pares tras la modificación del par A.

Figura B.9: Último bloque de los tres pares tras el reminado del par A.

Ejemplo 3: Añadiendo el sistema de firma digital

En las situaciones anteriores, se ha obviado el problema que supondría añadir una transacción en la que figurara un usuario como emisario sin que este hubiera firmado la operación. Como se ha comentado a lo largo de este documento, la solución a este problema pasa por un criptosistema asimétrico de firma digital.

Figura B.10: Criptosistema asimétrico.

Recuérdese el funcionamiento de este tipo de sistemas. Supóngase que se cuenta con un par de claves asimétricas (Figura B.10) y se quiere firmar digitalmente el mensaje “unizar2023”. Entonces, se usará la clave privada para elaborar, a partir del mensaje, un resguardo de la firma (Figura B.11).

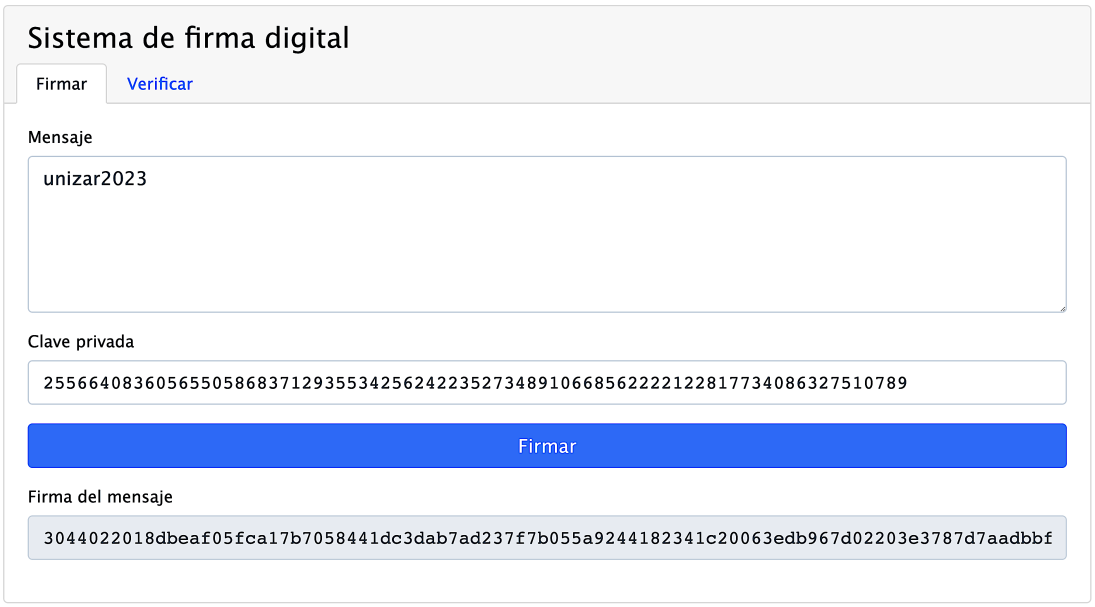


Figura B.11: Firma en un criptosistema asimétrico de firma digital.

El mensaje firmado digitalmente podrá ser verificado por otros usuarios a través de mi clave pública y el resguardo de la firma. Además, nótese que con cualquier inexactitud, tanto en el mensaje, como en la clave pública o en el resguardo, el proceso de verificación dará un resultado incorrecto, como se muestra en la Figura B.12.

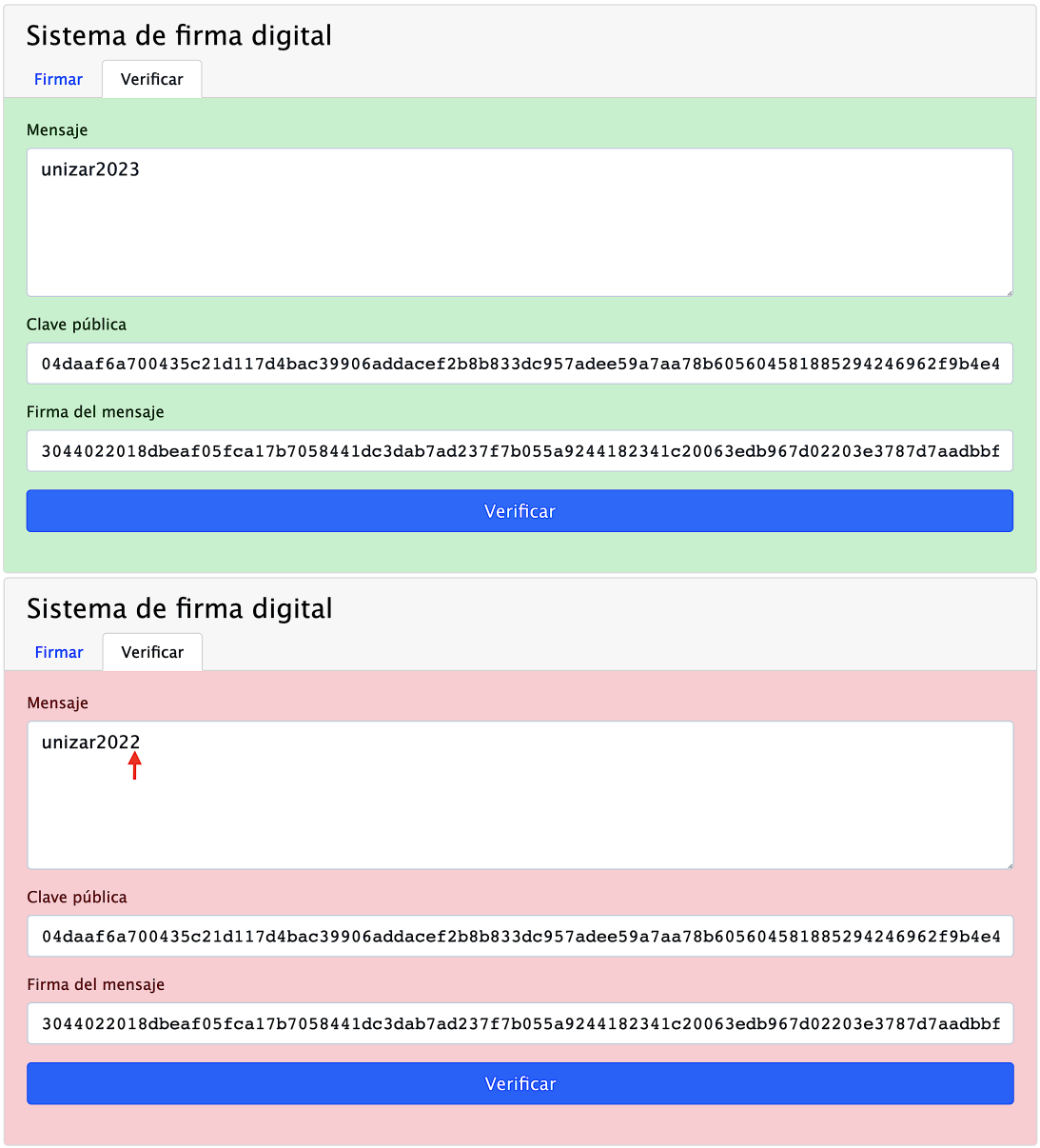


Figura B.12: Verificación en un criptosistema asimétrico de firma digital.

Una vez recordado lo anterior, se va a adaptar este sistema para trabajar con transacciones en vez de con mensajes de texto. A diferencia de los ejemplos anteriores, en los que los nombres de emisores y destinatarios de las transacciones eran visibles en la cadena, se va a dar un paso más en lo que respecta a un funcionamiento más realista de la blockchain haciendo que las transacciones sean seudónimas, es decir, que lo que sea visible sean las claves públicas de emisor y destinatario, como se puede observar en la Figura B.13.

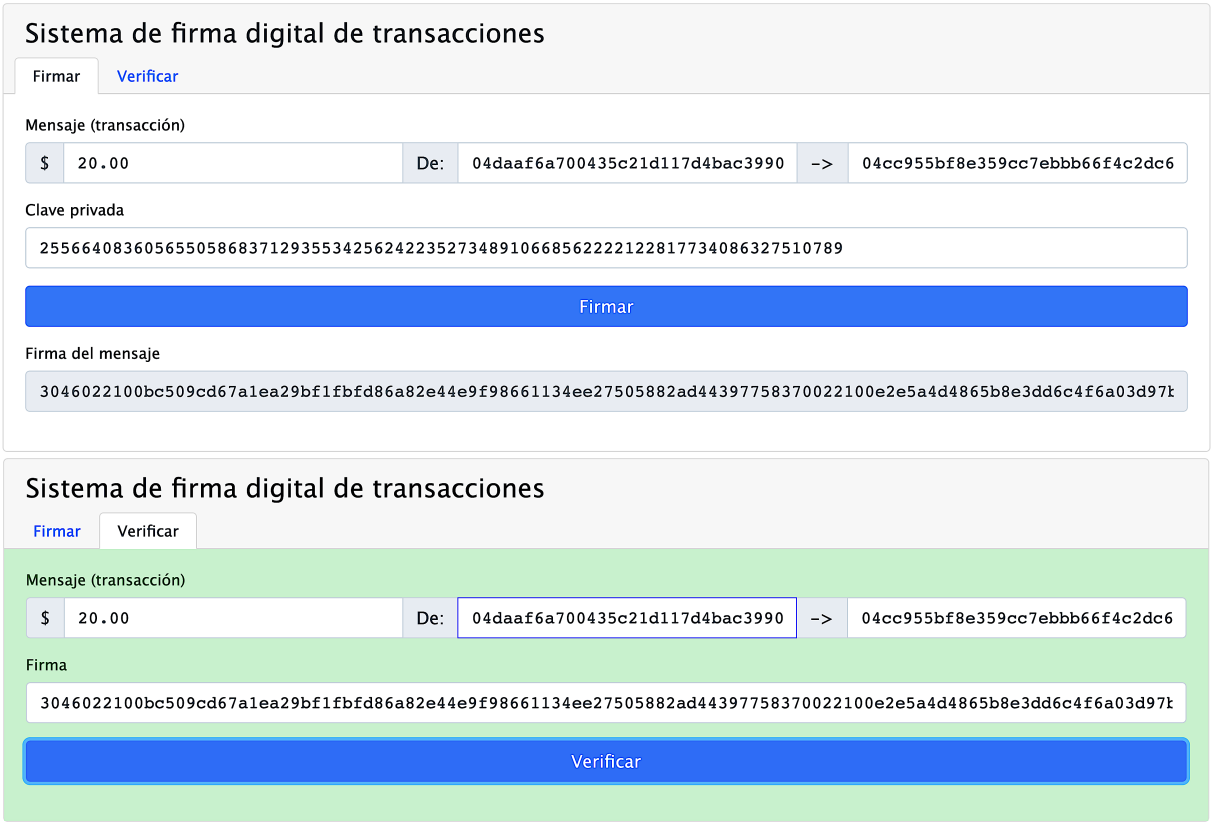


Figura B.13: Criptosistema asimétrico de firma digital de transacciones.

En este contexto, supóngase que se tiene una cadena cuyos bloques 3 y 4 se muestran en la Figura B.14. El sistema de firma incorpora un nivel de seguridad adicional, haciendo que el minero no pueda añadir a su bloque transacciones cuya firma no esté validada. Gracias al sistema de firma digital, un posible nodo malicioso no podría minar un bloque con una transacción creada por él sin la firma generada por la clave privada del propietario de los fondos. Por otro lado, si un nodo malicioso modificara un bloque minado de su copia de la cadena, además de las dificultades expuestas en los ejemplos anteriores, se podría identificar al instante que ha sido modificada y la transacción modificada en concreto, ya que la validación de la firma sería incorrecta aún recalculado el *nonce* (véase la Figura B.15).

Se recuerda al lector que estos ejemplos se realizan sobre un simulador simplificado, que debería ser adaptado para explicar cada cadena de bloques en particular. Por ejemplo, para simular el caso de Bitcoin, haría falta incluir en cada transacción la información asociada a las entradas y salidas, ya que esa cadena sigue el modelo transaccional detallado en la Sección 4.2. No obstante, el objetivo de este apéndice es exponer el funcionamiento general de una cadena de bloques.

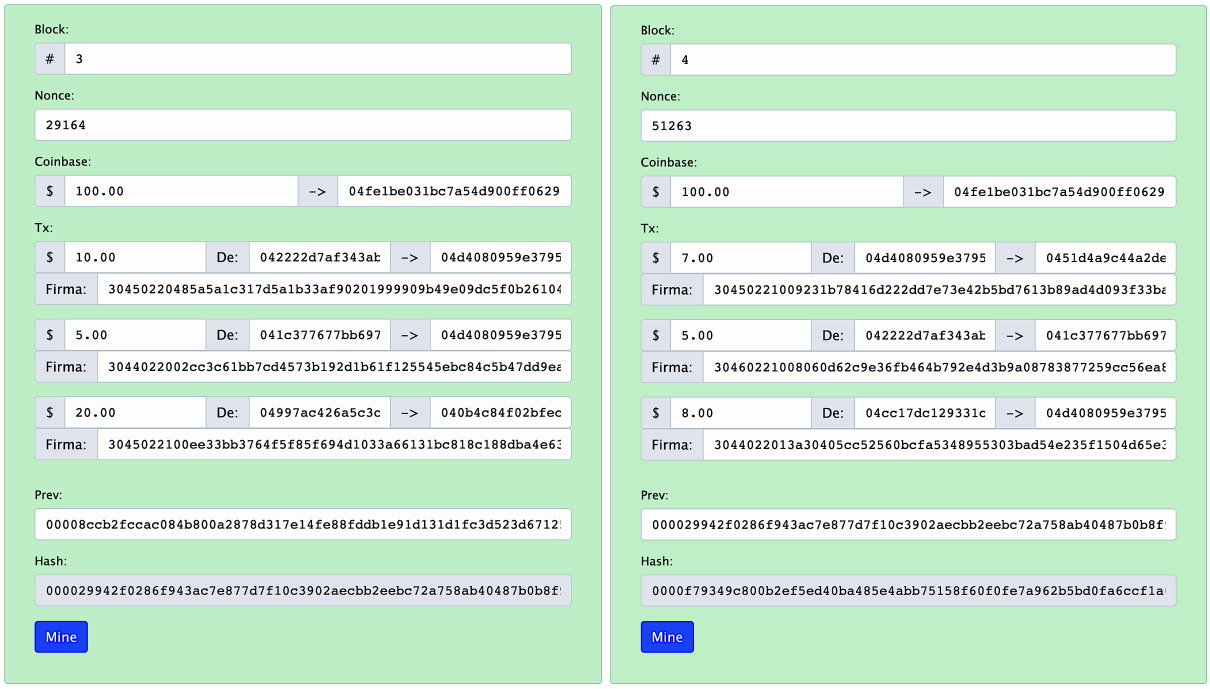


Figura B.14: Bloques 3 y 4 de la cadena del ejemplo 3.

Figura B.15: Bloque 4 modificado pre y postminado.