# Índice general

# Introducción y objetivos.

El blockchain es un protocolo que permite mantener un registro distribuido e inmutable de las transacciones o comunicaciones entre los participantes de una determinada red. Para garantizar que en todo momento la veracidad de la información almacenada, el blockchain posee también un mecanismo de consenso que no necesita de ningún tipo de centralización para funcionar.

El origen del blockchain se remonta al año 2008, con la publicación del artículo *Bitcoin: A Peer-to-Peer Electronic Cash System* [1]. Se desconoce la verdadera identidad de la persona u organización oculta bajo el seudónimo de Satoshi Nakamoto, autor de este documento. En el año 2009, el propio Nakamoto publicó el código, escrito en C++, de la criptomoneda Bitcoin. Posteriormente fueron creadas nuevas monedas digitales basadas en la idea original del Bitcoin y a partir del año 2014 se comenzaron a estudiar aplicaciones del Blockchain distintas de las criptomonedas. Una aspecto importante de las primeras criptomonedas es su carácter completamente público, cualquier usuario de internet podía unirse a la red y participar activamente sin necesidad de ninguna autorización o verificación.

Todas estas implementaciones, independientemente de las variaciones en el protocolo, tienen en común el carácter distribuido del algoritmo y la inmutabilidad del registro (cadena de bloques). Sin embargo, se pueden establecer 3 distinciones generales en base a como se trate el aspecto público y el nivel de descentralización:

- *Blockchain público:* Cualquiera puede unirse a la red, todos los nodos pueden leer el registro, realizar transacciones y participar en el consenso.
- *Blockchain de consorcio:* En el algoritmo de consenso solo intervienen una parte de los nodos. El acceso a la red y las consultas al registro en principio están abiertas a todos. Este sería el caso de una red pública y centralizada.
- *Blockchain privado:* El acceso a la red no es público y además tanto la lectura del registro como la participión en el algoritmo de consenso pueden encontrarse limitados. Estamos ante una red privada que puede estar en mayor o menor medida centralizada.

Dado que es la variante más conocida y utilizada, y es a la que pertenece la primera criptomoneda, el Bitcoin, en este trabajo en general se estudiará el blockchain de tipo público.

Al igual que no existe una única alternativa en la elección de los niveles de centralización y privacidad, en diferentes implementaciones del blockchain se han establecido distintos

algoritmos de consenso. Sin embargo, para explicar sus diferencias hay que entender su funcionamiento y esto último es imposible sin antes haber definido y desarrollado una serie de cuestiones sobre criptografía.

El objetivo de este trabajo será en primer lugar dar las nociones matemáticas básicas que fundamentan el blockchain y una vez sea posible justificar el marco teórico de esta tecnología el siguiente paso será explicar su funcionamiento. En este punto será importante abstraer lo suficiente la especificación para intentar cubrir la mayor parte de las implementaciones existentes. Hay que tener en cuenta que el desarrollo de este protocolo ha estado ligado a sus implementaciones (o más bien al éxito de estas) y no a un trabajo teórico sistemático. El siguiente paso consistirá en estudiar el blockchain dentro del contexto específico de la computación distribuida y de la resolución de uno de los problemas fundamentales de este campo. Por último, se realizará una implementación del blockchain en la que se podrá comprobar que es posible llevar a la práctica de forma más o menos sencilla, las ideas desarrolladas en este trabajo.

# Criptografía y Blockchain

Garantizar la autenticidad de las transacciones entre los nodos y la integridad del registro es la tarea de la criptografía dentro del blockchain. La criptografía de clave pública, donde no es necesario la existencia de un canal seguro para transmitir la información, es el método criptográfico utilizado en el Blockchain, pues justamente en este entorno público toda la información transmitida es vista por los demás participantes. Este método, desarrollado en la década del 70 del siglo XX, solucionó algunas limitaciones de la criptografía de clave simétrica. En un entorno donde se use una misma clave tanto para cifrar como para descrifrar mensajes se debe disponer un canal seguro a través del cual esta clave sea distribuida. Existen situaciones en las que no se puede garantizar la existencia de tal canal y por tanto el uso de claves simétricas es inviable. La criptografía de clave pública se fundamentan en la generación de una dupla $(p, q)$ donde $p$ es una clave que se transmitirá a todos los demás participantes mientras que $q$ solo será conocida por quien haya creado la dupla. A partir de $p$ es computacionalmente infactible deducir $q$ y por tanto cualquiera que haya recibido la clave pública puede usarla para cifrar un mensaje pero solo quien conozca $q$ puede determinar el mensaje original. Dentro del blockchain sin embargo el objetivo del cifrado no es ocultar información sino verificar la identidad de cada participante. Quien controle la clave privada puede usarla para firmar sus mensajes y la autenticidad de esta firma podrá ser verificada por todos aquellos que hayan recibido la clave pública.

Para especificar el principal algoritmo de clave pública usado en el blockchain (el ECDSA Elliptic Curves Digital Signature Algorithm) y explicar con más detalle el funcionamiento del proceso de autenticación hay que definir una serie de conceptos criptográficos y algebraicos.

## 2.1. Curvas elípticas y ley de grupo

Una curva elíptica sobre un cuerpo $\mathbb{K}$ está definida por la ecuación

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

con $a_i$ elementos del cuerpo tales que el discriminante de la curva sea distinto de cero. Para los algoritmos que vamos a estudiar nos restringiremos a los casos en que $\mathbb{K}$ sea un cuerpo finito de característica $p$, con $p$ un número primo distinto de 2 o 3. Bajo estas condiciones, y haciendo un cambio de variable, la ecuación general de una curva elíptica se puede reescribir

como

$$y^2 = x^3 + ax + b \tag{2.1}$$

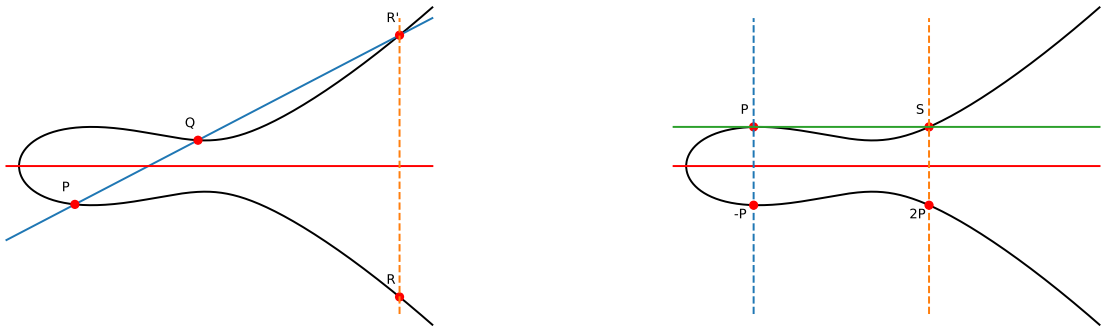
Y para que el discriminante sea distinto de cero los coeficientes $a$ y $b$ deben cumplir $4a^3 + 27b^2 \not\equiv 0 (\text{mod } p)$ En lo adelante cuando se hable de una curva elíptica nos estaremos refiriendo a 2.1 sobre un cuerpo primo de característica distinta de 2 o 3, aunque para visualizar algunas operaciones debamos suponer la curva definida sobre $\mathbb{R}$

A los pares $(x, y)$ que cumplen 2.1 se les añade el punto del infinito y sobre esta estructura proyectiva se define la operación de suma y calculo del inverso como sigue:

**Suma** Dados dos puntos $P, Q$ de la curva, la recta (proyectiva) que pasa por ellos corta a la curva en otro punto (si se trata de una recta vertical se dice que corta a la curva en el punto del infinito) la reflexión de este otro punto de corte respecto del eje de las abscisas será la suma de P y Q Un caso especial de la suma es cuando se quiere calcular P+P, en esta situación si $P = (x, y)$ con $y \neq 0$ se toma como $2P$ la reflexión respecto del eje x del punto de la corte de la tangente de P con la curva. Si $y = 0$ se toma como $2P$ el punto del infinito.

**Inverso** El inverso de un punto $P$ es el resultado de su reflexión respecto del eje de las abscisas, así, si $P$ tiene por coordenadas $(x, y)$ entonces $-P$ tendrá coordenadas $(x, -y)$

Esta definición geométrica de las operaciones se aprecia mejor trabajando sobre los números reales, y así es como se suele representar en gráficos. En cualquier caso tal y como se hizo con el inverso, se puede dar una definición algebraica de la suma.




(a) Cálculo de $P + Q = R$ (b) Cálculo de $2P$ y $-P$

Figura 2.1: Curva elíptica $y^2 = x^3 - 3x + 5$

A partir de la definición anterior de las operaciones no es difícil ver que el punto del infinito es el elemento neutro de la suma, que esta operación de suma es conmutativa y que existe el inverso de todo elemento. Sin embargo que la suma está cerrada con esta definición

y que se cumple la propiedad asociativa no son resultados inmediatos. Una prueba de esto se puede consultar en [3] y en [4]

Al cumplirse las cuatro propiedades anteriores tenemos que la suma así definida induce un grupo sobre los puntos de la curva elíptica. Sea $(G, +)$ este grupo y $n = |G|$ su orden. Sabemos que $n$ no es infinito pues la curva está definida sobre $\mathbb{K}$ un cuerpo finito, y por tanto sus puntos vendrán dados por $\{(x, y)|y^2 \equiv x^3 + ax + b(modp)\}$. Tenemos por tanto un grupo finito en el que podemos encontrar algún subgrupo cíclico. Un subgrupo cíclico tiene orden $q$ siendo $q$ un primo tal que que $q|n$. Si $n$ es primo decimos entonces que $|G|$ es un grupo cíclico. Será sobre este subgrupo cíclico sobre el que se definan los protocolos criptográficos que veremos.

## 2.2. Funciones Hash

Antes de ver el algoritmo ECDSA hay que definir el concepto de función hash y ver alguna de sus propiedades.

**Definición 2.1** (Función Hash)**.** *Una función hash es una aplicación $H$ entre cadenas tal que su conjunto imagen está formado por cadenas de longitud fija, siendo esta magnitud la longitud hash de la función, mientras que su preimagen está formada por cadenas de longitud arbitraria. Se escribe entonces $H : \{0, 1\}^* \to \{0, 1\}^n$ si hablamos de cadenas de bits y denotamos por $n$ a la longitud hash de $H$.*

Aunque se hable de longitudes arbitrarias pueden existir limitaciones técnicas en cuanto a la longitud de la cadena de entrada. Por ejemplo la función hash SHA-256 (SHA son las siglas de Secure Hash Algorithm) usada ampliamente en diferentes implementaciones del protocolo blockchain no puede recibir entradas de más de $(2^{64} - 1)$ bits [6], pero teniendo en cuenta que este valor equivale a más de 2 millones de terabytes esto no suponer un verdadero problema en la práctica.

Una primera consecuencia de la definición 2.1 es que la preimagen de una función hash tiene cardinal estrictamente mayor que su imagen. Por tanto, por el principio del palomar, tendremos que si $A$ es el conjunto preimagen de cierta función hash $H$ y $B$ es su imagen para todo $b \in B$ $\exists\{a_1, a_2\}$ con $a_i \in A$ distintos tales que $H^{-1}(b) = a_i$. Este fenómeno se llama colisión y se define en la resistencia a colisiones de una función hash como la probabilidad de encontrar un par $(a_0, a_1)$ con $a_0 \neq a_1$ tales que se cumpla $H(a_0) = H(a_1)$. A toda función hash $H$, que usemos en el protocolo blockchain o en el algoritmo ECDSA, le pediremos que ningún algoritmo pueda encontrar una colisión para $H$ en tiempo polinómico, en este caso diremos que $H$ es resistente a colisiones. Otras dos propiedades importantes de las funciones hash serán la resistencia a preimágenes y a segundas preimágenes. La primera de estas propiedades viene a decir que debe ser *díficil* dado un $b$ en el espacio de llegada encontrar un $a$ en el espacio de partida tal que $H(a) = b$. La segunda propiedad es similar a la definición de resistencia a colisiones, dado un $a_0$ elegido de forma aleatoria debe ser *díficil* encontrar un $a_1$ distinto de $a_0$ tal que $H(a_0) = H(a_1)$. Esta noción de *difícil* viene dada por la imposibilidad de resolver en tiempo polinómico cualquiera de estos problemas.

La propiedad de resistencia a segundas preimágenes es más fuerte que la resistencia a colisiones como consecuencia de la paradoja del cumpleaños. En efecto, si suponemos cadenas binarias y una longitud hash de $n$ nuestra función devolverá exactamente $2^n$ cadenas distintas. Dadas $l$ cadenas elegidas al azar la probabilidad de no tener una colisión viene dada por:

$$NC(n,l) = \frac{2^n}{2^n}\frac{2^n-1}{2^n}\ldots\frac{2^n-(l-2)}{2^n}\frac{2^n-(l-1)}{2^n} = \prod_{i=0}^{l-1}(1-\frac{i}{2^n}) \tag{2.2}$$

Así que la probabilidad de tener una colisión $C(2^n,l)$ será su complementario:

$$C(n,l) = 1 - \prod_{i=0}^{l-1}(1-\frac{i}{2^n}) \tag{2.3}$$

Considerando que el cociente $i/2^n$ es cercano a cero, que es consecuencia de tomar $n$ suficientemente grande, y teniendo en cuenta que la expansión en Serie de Taylor de $e^x$ se puede truncar a partir del término cuadrático para $x \ll 1$. Tenemos que:

$$C(n,l) \approx 1 - \prod_{i=0}^{l-1}\exp\left(\frac{-i}{2^n}\right) = 1 - \exp\left(\sum_{i=0}^{l-1}\frac{-i}{2^n}\right) \tag{2.4}$$

Este último sumatorio se puede escribir como $-(l-1)l/2^{n+1}$ usando la fórmula de la suma de los $(l-1)$ primeros naturales. Y que para $l$ suficientemente grande se puede aproximar a $-l^2/2^{n+1}$ Tomando logaritmos y despejando se tiene que:

$$l \approx \sqrt{-2\log(1-C(n,l))}2^{n/2} \tag{2.5}$$

Si se considera una probabilidad $C(n,l)$ de 0.5 entonces esa raíz cuadrada es aproximadamente 1 y por tanto $l \approx 2^{n/2}$. Tomando $2n = 365$ se obtiene el problema del cumpleaños: para garantizar una colisión (dos individuos que cumplan años el mismo día) con una probabilidad del 50 % basta con reunir 22 personas, este resultado es mucho menor que el valor que podríamos dar intuitivamente, de ahí que se le llame paradoja aunque desde el punto de vista estrictamente lógico no lo sea.

Sin embargo, la probabilidad de encontrar una segunda preimagen comprobando solo $2^{n/2}$ cadenas es de $2^{n/2}/2^n = 1/2^{n/2}$, que para n = 64 es prácticamente cero. Para garantizar una probabilidad de al menos el 0.5 se tiene que:

$$k = \frac{1}{2}2^n = 2^{n-1} \approx 2^n \tag{2.6}$$

Donde k es el número de cadenas que hay que estudiar. En este caso el coste es prácticamente el mismo que el de comprobar todas las posibles cadenas que produce nuestra función hash.

Hay que señalar que los cálculos anteriores solo son válidos bajo ciertos supuestos teóricos que en la práctica no se dan, pues ninguna función hash lleva los elementos del espacio de partida al de llegada siguiendo patrones completamente aleatorios. Una vez han sido encontradas colisiones (o se ha probado que la probabilidad de encontrarlas es *alta*) la función hash deja de ser considerada segura. Un ejemplo de función hash en esta categoría es SHA-1, para la que se conocen colisiones desde 2017 [7], aunque era considerada insegura desde 2005. La función hash que usaremos (SHA-256) es considerada segura por el momento.

## 2.3. Algoritmo ECDSA

Ahora podemos especificar el algoritmo ECDSA (Elliptic Curve Digital Signature Algorithm)Hankerson [2], tanto el usado para la firma de mensajes como el correspondiente algoritmo de verificación. Este algoritmo se sustenta en la dificultad computacional de resolver el problema del logaritmo discreto para curvas elípticas (ECDLP).

**Definición 2.2** (Problema del logaritmo discreto para curvas elípticas)**.** *Sea* $E(\mathbb{K})$ *una curva elíptica sobre un cuerpo* $\mathbb{K}$*. Sea* $P \in E(\mathbb{K})$ *y* $Q \in < P >$*. Encontrar* $k$ *tal que* $Q = kP$*.*

La definición anterior tiene sentido pues como se vio en 2.1 los puntos de una curva elíptica definida sobre un cuerpo finito forman un grupo. No se conoce algoritmo que resuelva este problema en tiempo subexponencial [5]. Sin embargo, aunque aquí no serán especificados se necesitan algoritmos apropiados para crear y representar el cuerpo finito $\mathbb{F}_p$ de forma que se garantice la intratabilidad de este problema.

Antes de poder firmar un mensaje hay que generar las claves públicas y privada. Sea $P = (x_1, y_1)$ un punto de la curva elíptica $E(\mathbb{F}_p)$ que genera un subgrupo cíclico $< P >$ de orden primo $n$. El par $(d, Q)$ de claves privada y pública se genera usando el algoritmo 1. La elección aleatoria de $d$ es muy importante, en otro caso se pueden tener problemas de seguridad (un ejemplo de esto fue el hackeo del software de la PS3 de Sony por el grupo fail0verflow en 2010).

**Algoritmo 1** Generación de claves

1: Se elige $d \in [1, n-1]$ de forma aleatoria
2: Se calcula $Q = dP$
3: $Q$ es la clave pública y $d$ la clave privada.

Una vez se ha generado la clave privada y utilizando una función hash $H$ que devuelva cadenas de longitud menor o igual que $n$ se pueden firmar mensajes usando el algoritmo 2

**Algoritmo 2** Generación de firma

1: Se elige $k \in [1, n-1]$ de forma aleatoria
2: Se calcula $z_1 = kx_1$
3: Se calcula $r = z_1 (mod n)$ si $r = 0$ paso 1.
4: Se calcula $e = H(m)$
5: Se calcula $s = k^{-1}(e + dr)(mod n)$ si $s = 0$ paso 1. $\triangleright$ $k^{-1}$ es el inverso de k en el grupo $< P >$
6: Se devuelve $(r, s)$

Ahora, quien recibe un mensaje $m$ con la firma $(r, s)$ puede verificar su procedencia (suponiendo que ha recibido antes la clave pública $Q$) mediante el algoritmo 3

**Algoritmo 3** Validación de firma

1: Verificar que $r, s \in [1, n-1]$, en otro caso **se rechaza la firma**
2: Se calcula $e = H(m)$
3: Se calcula $w = s^{-1}(mod n)$
4: Se calcula $u_1 = ew(mod n)$, $u_2 = rw(mod n)$
5: Se calcula $X = u_1P + u_2Q$, si $X = \infty$ **se rechaza la firma** ▷ $X \in E(\mathbb{F}_p)$
6: Sea $X = (x_1, y_1)$, se calcula $v = x_1(mod n)$
7: Si $v = r$ **se acepta la firma**, en otro caso **se rechaza la firma**

# Bibliografía


[1] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2008, http://bitcoin.org/bitcoin.pdf

[2] D. Hankerson, A. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, New York, NY, 2nd edition, 2004.

[3] Cassels, J. (1991). *Non-singular cubics. The group law. In LMSST: 24 Lectures on Elliptic Curves* (London Mathematical Society Student Texts, pp. 27-31). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139172530.008

[4] H. Verrill, *Group Law for elliptic Curves*, Lecture Notes (Recurso Online http://www.math.ku.dk/~kiming/lecture_notes/2000-2001-elliptic_curves/grouplaw.pdf)

[5] K. Lauter, K. Stange, *The Elliptic Curve Discrete Logarithm Problem and Equivalent Hard Problems for Elliptic Divisibility Sequences. Selected Areas in Cryptography* pp. 309-327, Springer Berlin Heidelberg, isbn 978-3-642-04159-4

[6] *Secure Hash Standard (SHS)* Information Technology Laboratory National Institute of Standards and Technology, 2015, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[7] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, Y. Markov, *The first collision for full SHA-1*, Google Research, http://shattered.io/static/shattered.pdf

[8] A. Tanenbaum, M. van Steen. *Distributed Systems: Principles and Paradigms.* 2002.

[9] L. Lamport, J. Gray, Consensus on transaction commit, Technical Report MSR-TR-2003-96, Microsoft Research, January 2004. http://research.microsoft.com/research/pubs/view.aspx?trid=701.

[10] A. Back *Hashcash - A Denial of Service Counter-Measure* ,2002,http://www.hashcash.org/papers/hashcash.pdf

[11] R. Wattenhofer, *The Science of the Blockchain.*,2016, Inverted Forest Publishing.

[12] *Bitcoin Mining Now Consuming More Electricity Than 159 Countries Including Ireland & Most Countries In Africa* https://powercompare.co.uk/bitcoin/

[13] *Proof of Stake FAQ* https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ#what-is-proof-of-stake

[14] *Slimcoin A Peer-to-Peer Crypto-Currency with Proof-of-Burn*, 2014 http://www.doc.ic.ac.uk/~ids/realdotdot/crypto_papers_etc_worth_reading/proof_of_burn/slimcoin_whitepaper.pdf

[15] L. Lamport , R. Shostak , M. Pease, *The Byzantine Generals Problem*, 1982, ACM Transactions on Programming Languages and Systems, pp. 1-2 https://web.archive.org/web/20170205142845/http://lamport.azurewebsites.net/pubs/byz.pdf

[16] V. Gramoli, *From blockchain consensus back to byzantine consensus.* (2017). http://poseidon.it.usyd.edu.au/~gramoli/web/doc/pubs2/Blockchain2Byzantine.pdf

[17] L. Lamport, *Paxos Made Simple*, 2001, http://lamport.azurewebsites.net/pubs/paxos-simple.pdf

[18] *Raft consensus algorithm* https://raft.github.io/

[19] Documentación librería fastecdsa https://pypi.org/project/fastecdsa/

[20] Documentación librería hashlib de Python https://docs.python.org/2/library/hashlib.html

[21] Documentación librería multiprocessing https://docs.python.org/2/library/multiprocessing.html