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* [11]. 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

Definición 2.1 (Curva elíptica)**.** *Una curva elíptica sobre un cuerpo* $\mathbb{K}$ *queda 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

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

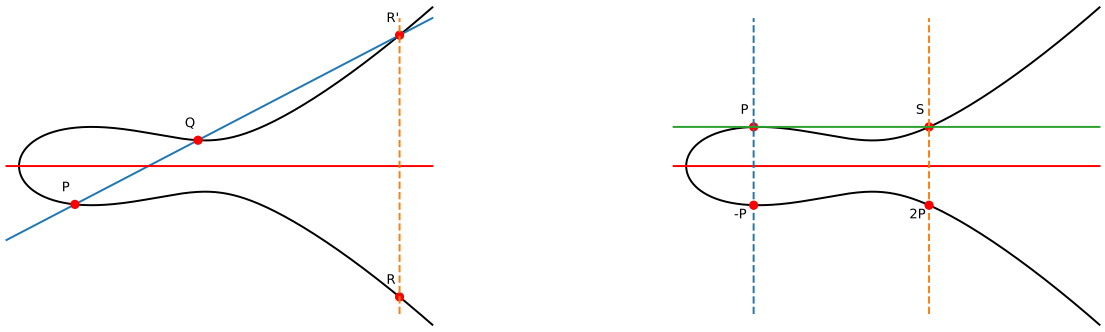
Y para que el discriminante sea distinto de cero los coeficientes a y b deben cumplir $4a^3 + 27b^2 \not\equiv 0 (\text{mod } p)$ En lo adelante cuando se hable de una curva elíptica nos estaremos refiriendo a 2.1 sobre un cuerpo primo de característica distinta de 2 o 3, aunque para visualizar algunas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

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 2.2 (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 [**?**], 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 2.2 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 [13], 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)**(author?)** [5], 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 2.3 (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 [10]. 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 $w = s^{-1}(modn)$
4: Se calcula $u_1 = ew(modn)$, $u_2 = rw(modn)$
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 $v = x_1(modn)$
7: Si $v = r$ **se acepta la firma**, en otro caso **se rechaza la firma**

Especificación del protocolo blockchain

Una vez se han visto las dos nociones criptográficas fundamentales en las que se fundamenta el blockchain (algoritmo ECDSA y funciones Hash) es posible explicar tanto el funcionamiento de este protocolo como justificar su corrección. Aunque las ideas fundamentales que serán expuestas están basadas en el artículo original del Bitcoin [11] en los puntos en los que existan divergencias entre distintas implementaciones del protocolo se intentarán señalar las diferentes alternativas. Siendo esta una tecnología que se ha desarrollado más con la práctica a través de diversas implementaciones que con trabajos teóricos resulta complicado abarcar en un solo trabajo todas sus variantes.

3.1. Objetivos del Blockchain

El Bitcoin surgió con el objetivo de por un lado eliminar la necesidad de una entidad de confianza que actúe de intermediario en las transacciones económicas digitales y por otro lado de resolver el problema derivado de la ausencia de este ente. En un intercambio basado en artículos físicos a los que se le asigna cierto valor (el papel moneda por ejemplo) los únicos problemas de los que las partes deben preocuparse son aquellos derivados de la estafa o la falsificación. Cuando estas transacciones se realizan sin intervenir medios físicos no se puede garantizar que la parte encargada de entregarle la retribución monetaria a la otra posea los medios económicos que dice tener. Más aún, alguien aún disponiendo de la capacidad de realizar cierta transacción puede enviarle a diferentes individuos cantidades que por separado no superen sus fondos, pero que en conjunto sí lo hagan, este es el llamado problema del doble gasto. Los intermediarios cumplen entonces la función de evitar que esto ocurre al llevar un registro actualizado de las transacciones de cada cliente y por tanto de los fondos de los que dispone.

Pero, ¿qué sentido tiene entonces eliminar el intermediario si con ello solo generamos problemas que debemos resolver? Un sistema descentralizado tolera mucho mejor las particiones que uno centralizado: basta aislar al servidor para dejar inoperativa toda la red. Además, desde el punto de vista de la seguridad cualquier ataque con éxito al servidor supone a su vez la caída de todo el sistema. Incluso suponiendo que se dispone de un intermediario seguro y distribuido en diferentes nodos de forma que la caída de alguno de ellos no suponga mayor problema para la red, la pregunta que debemos hacernos es cuanto podemos llegar a confiar en esta entidad. En la medida en que controla nuestro dinero puede llegar a actuar de forma

más o menos arbitraria con él ya sea por interés propio o al servicio de algún poder externo. Ejemplos en la vida real hay muchos.

El objetivo será por tanto proporcionar un protocolo seguro, que aún estando completamente descentralizado pueda resolver en particular el problema del doble gasto y en general problemas de consenso de otra clase. Por tanto, la corrección de este protocolo vendrá dada por la comprobación de que justamente se han solucionado estas cuestiones.

3.2. Configuración de la red

El protocolo blockchain se define e implementa sobre una red basada en la arquitectura peer-to-peer (red entre pares). El término peer-to-peer (en lo adelante P2P) viene a decir que al contrario que en un sistema cliente/servidor aquí todos los nodos realizan potencialmente las mismas funciones. Este punto puede no ser del todo cierto en determinadas implementaciones como por ejemplo en Blockchain de consorcio o privados, pero en estos casos es el protocolo elegido y no la estructura de la red quien restringe las funciones de algunos nodos. Estas restricciones no limitan en ningún caso la capacidad de enviar o recibir información directamente entre los participantes, sino que está controlada que clase de información determinados nodos pueden enviar y como los demás interpretan estos mensajes. El Bitcoin es el ejemplo más importante de red blockchain donde no hay ninguna clase de jerarquía

Aunque todos los nodos teóricamente puedan realizar las mismas funciones en la práctica suelen elegir que tareas realizar y cuales ignorar. Así, algunas máquinas pueden dedicarse a ejecutar las labores más complejas del algoritmo de consenso (más adelante veremos en que consiste este algoritmo), otras almacenan parte (o la totalidad) de la cadena de bloque, mientras que otros nodos ejecutan tareas de verificación. Existen otras especializaciones (o combinaciones de especializaciones) que pueden variar entre diferentes implementaciones del blockchain.

Un problema importante en las redes es p2p es como los participantes se encuentran unos con otros para establecer una conexión. Los nodos mantienen un registro privado de las direcciones ip con las que se han comunicado, y pueden compartir este registro con otros miembros que lo soliciten, esta sería la forma estándar de ampliar la agenda de contactos. La primera vez que un nodo se une a la red dispone de una lista de direcciones ip prefijada, en ciertos protocolos blockchain que han alcanzado un número de participantes considerable existe también la posibilidad de conectarse a varias DNS Seeds donde recibe una lista generada aleatoriamente de otros nodos que han estado activos en la red en un rango de tiempo determinado. Este punto, dado que es el único en el que existe algo de centralización, entraña ciertos riesgos, pues alguno de estos DNS Seed puede caer bajo el control de una entidad maliciosa que le proporcione a los nuevos participantes solo direcciones ip de nodos bajo su control, de esta forma se podría crear una red independiente de la principal donde existe una cadena de bloques generada de forma arbitraria. Por ello se mantiene cierto control de forma pública y abierta sobre estas DNS Seed y se puede igualmente usar direcciones ip recibidas a través de cualquier otra vía.

3.3. Cadena de bloques

El componente fundamental del blockchain es la estructura de datos donde se almacena toda la información que los nodos que participan en el protocolo han acordado que es válida. En las criptomonedas este registro cumple la función de libro de cuentas, sin embargo, se pueden guardar todo tipo de datos.

Cada uno de los bloques que conforman esta cadena además de la propia información que pueda contener mantiene una referencia al bloque anterior. La excepción a esto es el bloque inicial, que se genera de forma diferente al resto. Esta estructura enlazada no es suficiente para garantizar la integridad de la cadena de bloques, pues un adversario podría modificar la información almacenada en un bloque dejando intacta la referencia al bloque anterior. Aquí es donde se utilizan por primera vez las funciones hash definidas en 2.2: además de la información almacenada y la referencia al bloque anterior guardaremos también el resultado de aplicar cierta función hash al conjunto de los datos del bloque (incluido quien es el bloque anterior). Si la función elegida es resistente frente a segundas preimágenes se puede verificar que el bloque es correcto simplemente aplicándole la función Hash a sus datos y comparando con la almacenada. Y la referencia al bloque anterior en lugar de ser únicamente un índice incluirá también el valor hash de ese bloque.

Sin embargo, aún con este añadido seguimos sin tener garantizado que un adversario pueda hacer modificaciones en la cadena. Es cierto que ahora para hacer cambios en un único bloque tendría que generar nuevamente toda la cadena a partir de ese bloque, pero el coste de operar con funciones hash es lineal, así que es factible hacer estas modificaciones en un tiempo razonable. Además de la referencia al bloque anterior y la información del bloque actual en los datos que sobre los que se aplica la función hash se incluirá un número (o una cadena) generados de forma aleatoria de tal forma que la cadena hash resultante tenga ciertas propiedades definidas en el protocolo. Por ejemplo, en el Bitcoin a la cadena hash se le pide que comience por cierto número de ceros. El cálculo de este valor aleatorio, denominado nonce (number used once), es la tarea principal del algoritmo de prueba de trabajo que se verá en el siguiente apartado.

Otra información que se incluye en los bloques, asociada con la función hash, es una marca temporal (timestamp) del momento en que fue creado el bloque. De esta forma se puede probar que la información existía cuando se calculó el valor hash. Mantener un registro temporal también servirá en el algoritmo de consenso para rechazar directamente aquellos bloques con marcas de tiempo superiores al momento actual.

Lo anterior se puede considerar más o menos estándar en todas las implementaciones del blockchain, pero se pueden incluir otros elementos dentro de cada bloque.

3.4. Algoritmo de prueba de trabajo

La estructura en cadena de bloques enlazados usando funciones hash no resuelve por si sola el problema del doble gasto visto en 3.1, para eso hay que establecer un algoritmo que permita a los participantes decidir que información se puede incluir en la cadena de bloques,

validando de esta forma las transacciones correctas. En el Bitcoin se estableció como tal sistema el algoritmo de prueba de trabajo (Proof of Work), en posteriores implementaciones del blockchain, salvo ciertas modificaciones se ha mantenido en general este mismo algoritmo. La idea de la prueba de trabajo se originó para resolver el problema del uso abusivo de ciertos recursos como el correo electrónico mediante el envío masivo de mensajes de spam. Hashcash [3], una de las primeras implementaciones de un mecanismo para solucionar este problema, se basaba en usar una función de coste para generar una ficha que permitiría disponer de cierto recurso. Esta ficha probaba que el usuario había invertido determinado tiempo de CPU en resolver el problema planteado por la función de coste y de esta forma el uso del recurso en cuestión estaba condicionado por la inversión en procesamiento del usuario. Con el Bitcoin este planteamiento basado en un sistema cliente-servidor fue extendido a un contexto descentralizado.

El resultado de ejecutar la función de coste (ficha) debe poder ser verificado fácilmente por cualquiera, mientras que el cálculo de esta función debe ser una tarea moderadamente complicada. La funciones hash cumplen estos requisitos y son usadas como función de coste. Un participante que quiera validar un bloque, es decir, que quiera afirmar que cierto conjunto de transacciones o datos son válidos y que por tanto se pueden añadir al registro toma el bloque tal y como se definió en 3.3 le añade cierto valor en bits cuya posible longitud y posición dentro del bloque están determinados en el protocolo y analiza el resultado de aplicarle cierta función hash acordada. Si este resultado cumple ciertos requisitos (por ejemplo la cadena hash devuelta empieza por $n0's$) entonces se dice que se ha resuelto el problema del algoritmo de prueba de trabajo (llamado puzzle en algunas fuentes) y se comparte con los demás nodos este resultado. Puede encontrarse también en ciertos artículos y especificaciones de criptomonedas que el puzzle que se debe resolver para validar cierto bloque consiste en encontrar una cadena hash menor que cierta longitud. Esto no es más que buscar cadenas hash que comiencen con ciertos número de ceros y considerarlas como de tipo numérico donde esos ceros desaparecen dejando una cadena n posiciones más corta que la longitud hash de la función del algoritmo.

En el algoritmo 4 se presenta una posible especificación (muy simplificada) del método de prueba de trabajo para calcular un nonce asociado a cierto bloque. En esta especificación se supone que quien la ejecuta ha verificado antes que todas los datos que se pretenden incluir en la cadena son correctos. Este algoritmo tal y como está planteado puede no terminar, pues es posible que no exista ningún nonce de longitud menor o igual que m que resuelva el puzzle. En la práctica si el bloque b está formado por cadenas de la forma $b = s+p+k+t_1+\ldots+t_q$ donde s es el timestamp, p la referencia al valor hash del bloque anterior, k es la clave pública del nodo que está calculando el nonce y t_i el conjunto de datos que almacena el bloque, se suele proceder reordenando los t_i a la vez que se modifica el nonce y comprobando el resultado de estas combinaciones. También se suelen hacer pequeñas modificaciones en el timestamp s, pues entre la hora marcada por el bloque anterior y lo que se tarda en generar el siguiente bloque hay un rango de tiempo en el que moverse. Este proceso es lo que se conoce en las criptomonedas como minado. Las implementaciones de este algoritmo deben incluir, una condición de parada extra, pues no tiene sentido seguir operando si alguien ya ha resuelto el puzzle, así, se debe mantener un proceso que en caso de recibir información

sobre la resolución (correcta) del problema de prueba de trabajo para el bloque en el que se está trabajando detenga el algoritmo.

Algoritmo 4 Prueba de trabajo (Cálculo)

Entrada:

H función hash del protocolo de longitud de cadena k.
l <k longitud (dificultad) del puzzle.
m longitud máxima del nonce.
b bloque que se quiere validar.

1: Generar $nonce$ de forma aleatoria tal que $|nonce| \leq m$
2: **si** $|H(nonce + b)| \leq l$ **devolver** $(nonce, b)$
3: **en otro caso**Volver a 1

Se incluye también el algoritmo de verificación que deben utilizar los nodos para comprobar el par $(nonce, b)$. Si acepta el bloque la forma de comunicarlo al resto de la red es incluir ese bloque en la cadena y empezar a trabajar en uno nuevo que lo tenga como predecesor. Puede ocurrir que diferentes nodos acepten distintos bloques en determinado punto, esto se denomina bifurcación y se trata en la siguiente sección.

Algoritmo 5 Verificación de la prueba de trabajo

Entrada:

H función hash del protocolo de longitud de cadena k.
l <k longitud (dificultad) del puzzle.
m longitud máxima del nonce.
$(nonce, b)$

1: Se comprueba que b está formado por datos válidos (bloque previo, timestamp y transacciones). Si falla cualquiera de estas comprobaciones **rechazar**
2: **si** $|H(nonce + b)| \leq l$ **devolver aceptar el bloque**
3: **en otro caso rechazar**

La dificultad del puzzle, es decir, la condición que se le pide al valor hash del bloque para ser aceptado, en general no se mantiene fija. Cada determinado número de bloques (fijado en el protocolo) se vuelve a calcular este valor usando cierta función dependiente de la dificultad actual del puzzle. De esta forma según crece la cadena es necesario invertir más recursos en añadir un nuevo bloque.

3.5. Bifurcaciones

Cuando se envían por la red varias versiones correctas del siguiente bloque los nodos pueden recibirlas en diferente orden. Por protocolo incluirán la primera que han recibido, así que puede ocurrir que la cadena de bloques deje de ser única. Para resolver este problema cada nodo almacena las otras versiones del último bloque que ha recibido, aunque solo

considere como buena una de ellas. Con el cálculo de un nuevo bloque alguna versión se hará mas extensa que las otras (tendrá un bloque nuevo) y en este caso todos los nodos que estuvieran en una rama distinta de la cadena cambiarán a la versión más larga.

Hay que señalar que el término bifurcación (fork) es usado también en la literatura sobre el blockchain para designar el proceso en el que se produce una modificación del protocolo acordada (o no) por los participantes. Cuando esto ocurre quienes siguen trabajando con una versión anterior dejan de reconocer los mensajes que se producen en el nuevo protocolo. Los que participan en este cambio pueden o bien dejar de reconocer también al protocolo antiguo y se dice entonces que estamos ante una bifurcación dura (hard fork) o seguir aceptando estos mensajes y se dice entonces que se ha producido una bifurcación suave (soft fork). Con esto se busca corregir errores en el código, prevenir ataques a la cadena de bloque o simplemente introducir mejoras.

3.6. Alternativas a la prueba de trabajo

El método de prueba de trabajo explicado en 3.4 al estar basado en el uso de CPU tiene ciertos inconvenientes. Por un lado, entidades con el poder suficiente para controlar un gran número de máquinas pueden intentar alterar el protocolo de consenso en la red. En general la filosofía del blockchain supone que alguien que tiene interés en participar en el protocolo y lo demuestra invirtiendo su tiempo de procesamiento estará también interesado en que el consenso se mantenga y por tanto no validará operaciones falsas. Igualmente, se pueden utilizar las bifurcaciones explicadas en 3.5 para modificar el protocolo si es necesario contrarrestar algún tipo de hardware específico diseñado para hacer más efectivo el proceso de minado.

Otra de las consecuencias negativas del protocolo de prueba de trabajo es de índole ecológica. El coste computacional de una red grande, como la de las criptomonedas Bitcoin y Ethereum por ejemplo, se refleja en un gasto energético extraordinario. Hay que tener en cuenta que en el proceso de validar un conjunto de operaciones solo se tiene en cuenta el resultado del primero que resuelve el puzzle, y por tanto el trabajo de todos los que han estado trabajando en ese mismo bloque ha sido en vano. Se estima que la red del Bitcoin a finales de 2017 ya gastaba más energía eléctrica que 159 países [1]. Teniendo en cuenta que las principales fuentes energéticas continúan siendo los combustibles fósiles es comprensible la preocupación que esto despierta.

A la vista de lo anterior se han propuesto varias alternativas entre las que se destacan las siguientes:

- *Prueba de participación (Proof of Stake)*: Para validar las transacciones se hace necesario disponer de cierta cantidad del recurso en que se basa la red (generalmente criptomonedas). En base a la cantidad de este recurso que posea, cada participante tiene asignado un peso que representa el valor de sus decisiones (a mayor cantidad de recursos mayor poder de decisión). Para proponer y elegir el siguiente bloque de la cadena se tienen en cuenta los votos y propuestas de cada participante de acuerdo a

su peso. La filosofía detrás de esta idea es que quienes posean un mayor número del recurso de la red estarán más comprometidos con ella y por tanto serán los mejores garantes del consenso. Este método se puede combinar con la prueba de trabajo para de esta forma reducir la complejidad del puzzle que hay que resolver, o implementarlo de forma independiente. En ambos casos representa un notable ahorro energético respecto a la prueba de trabajo.

Por otro lado, se tiene el problema de como iniciar la cadena cuando ninguno de los participantes posee recurso alguno. Debido a esto se suele optar por un enfoque mixto entre ambos métodos. Ejemplos del uso de este método están en versiones más recientes de Ethereum y en la criptomoneda Peercoin [**?**]

- *Proof of burn:* En este caso para obtener el derecho de validar transacciones (crear bloques) hay que destruir cierto recurso como prueba de nuestro compromiso con la red. Generalmente el recurso que se destruye es otra criptomoneda y esto se consigue enviándola a determinada dirección desde donde es irrecuperable. Este método tiene el problema de requerir de la existencia de al menos una moneda con ciertas propiedades similares a la nuestra (si es fácil crearla por ejemplo no tiene valor práctico destruirla). Aunque este mecanismo no requiera de un gran consumo energético la moneda alternativa que estamos destruyendo puede que esté basada en la prueba de trabajo, así que estaríamos en una situación similar a la explicada al inicio de este sección. Por todo esto la aplicación de este método aunque interesante desde el punto de vista económico, resulta contraproducente respecto al problema del gasto energético.

  La criptomoneda Slimcoin es la primera implementación de este algoritmo de consenso [2]

3.7. Privacidad y seguridad

Al inicio de este capítulo se explicaron los objetivos del Bitcoin, sin embargo, no se hizo referencia a la privacidad. En los sistemas bancarios tradicionales se mantiene cierto nivel de confidencialidad, y por tanto la información sobre el balance y las operaciones de determinada cuenta no se comparten libremente. En un sistema basado en el blockchain surge el problema de como garantizar la privacidad en un entorno público donde las transacciones por definición deben ser vistas por todos los participantes. Esto se resuelve manteniendo en secreto la identidad de los poseedores de las claves públicas. Además si se usa una clave pública distinta en cada transacción incluso aunque se llegue a comprometer la privacidad de alguna de nuestras claves se siguen manteniendo ocultas las otras y en consecuencia es imposible que alguien llegue a conocer todas las operaciones que hayamos realizado y nuestro balance total. En este sentido los sistemas monetarios basados en blockchain guardan cierta semejanza con el dinero físico y se hace necesario disponer de algún sitio seguro donde guardar el conjunto de claves privadas que hayamos usado ya sea para recibir transferencias o recompensas por haber participado en el algoritmo de consenso. Estos son los llamados monederos (wallets), que existen tanto en forma de software como de hardware específico.

Estas cuestiones sobre privacidad aunque en las criptomonedas se consideran necesarias no son imprescindibles y en otras aplicaciones del blockchain se pueden obviar.

Otra cuestión de gran relevancia es como de segura es una red basada en blockchain. Por un lado, hay que distinguir aquellos ataques basados en intentar encontrar la clave (o claves) privada de un usuario y por otro los ataques al protocolo de consenso de la red. En el primer caso, la seguridad de la red depende en gran medida de la implementación del algoritmo visto en 2. Sobre el problema del logaritmo discreto para curvas elípticas (en el que se basa el ECDSA) no se conoce ningún algoritmo que lo resuelva en tiempo subexponencial [10]. Por tanto, se puede considerar que encontrar la clave privada a partir de la pública es infactible siempre que se haya implementado correctamente el algoritmo de cifrado. Evidentemente es igual de importante almacenar las claves privadas en un sitio seguro. Los ataques al protocolo de consenso se basan en intentar que sean incluidos en la cadena bloques con información falsa, en el caso de las criptomonedas esto es el problema del doble gasto mencionado al inicio de este capítulo. Por ser este un problema de gran interés dentro de la programación distribuida y que va más allá de la idea de blockchain se tratará en el siguiente capítulo con más detalle.

El problema del consenso

Uno de los problemas fundamentales dentro de la computación distribuida es el de mantener o alcanzar alguna clase de acuerdo entre los miembros de determinada red. Una de las definiciones posibles de un sistema distribuido nos dice que son colecciones de ordenadores que ante sus usuarios funcionan como una única máquina [14], así, el hecho de disponer de algún mecanismo de consenso es fundamental para que el sistema se comporte de forma adecuada. Algo más cerca de la idea de consenso que nos interesa de cara al blockchain están los acuerdos que se alcanzan en una base de datos entre todos los procesos antes de hacer un *commit*, donde deben decidir si abortar o no la transacción con la certeza de que la acción tomada será la misma para todos.

En un sistema distribuido genérico suponiendo que todos los miembros actúan siempre de acuerdo a ciertas reglas comunes (no hay comportamientos maliciosos ni fallos), las máquinas de los participantes no se desconectan de la red de forma imprevista y que los canales de comunicación son estables, es decir, todo mensaje llega a su destinatario de forma íntegra y a lo sumo en tiempo t, el consenso se puede alcanzar siempre de forma sencilla con algún protocolo como el commit de 2 fases [7]. En la práctica no se suelen cumplir alguna o varias de estas condiciones, así que el interés está en crear algoritmos robustos que puedan funcionar bajo múltiples condiciones adversas.

El protocolo blockchain descrito en 3 es de hecho un algoritmo de consenso. Queda por ver las limitaciones de esta clase de consenso y como se podría aplicar a un contexto más general.

4.1. El problema de los generales bizantinos

El problema de los generales bizantinos [9] plantea un experimento mental en el que un grupo de divisiones de un ejército (del Imperio de Bizancio) se encuentran asediando una ciudad. Los generales al frente de cada división deben acordar, en una versión simplificada del problema, si atacan la ciudad o se retiran. Las líneas de comunicación no son seguras, así que algunos mensajes pueden no llegar a su destino. Igualmente, entre los generales existen traidores que pueden no tomar la mejor decisión o comunicar una acción y realizar otra diferente. El problema es por tanto alcanzar un acuerdo entre los generales leales sobre que decisión tomar. Para garantizar el éxito la decisión tomada debe ser la misma para todos ellos.

Este es claramente un problema de consenso como los definidos al inicio de este capítulo. En general, para alcanzar un acuerdo se necesitan al menos $3m + 1$ generales en total si tenemos m generales traidores [9], así por ejemplo con 3 generales si uno de ellos es traidor no existiría solución.

El algoritmo *Paxos*, creado por Leslie Lamport resuelve el problema anterior [8]. El algoritmo *Raft* se considera una simplificación de *Paxos* usando métodos e ideas similares [12]. Tanto *Paxos* como *Raft* para alcanzar el consenso requieren de la elección de un líder.

4.2. Solución desde el Blockchain

El problema anterior puede plantearse desde la perspectiva del protocolo blockchain. Los nodos o participantes de la red serán los generales y el acuerdo que buscan estará reflejado en la cadena de bloques. Algunos nodos toman una decisión (atacar o retirarse) y se la comunican a los otros participantes. Cada nodo *trabaja* en el primer mensaje que reciba y con el que esté de acuerdo: si por ejemplo considera que lo mejor es retirarse *trabajará* en el primer *retirarse* que reciba. Aquí trabajar es buscar un nonce del bloque que contiene las palabras *retirarse* o *atacar*, junto con información adicional como un índice y un timestamp por ejemplo, de longitud fijada por el protocolo. Cuando un nodo resuelve este problema comunica a los demás su solución y todos (los que estén de acuerdo con esta decisión) trabajan en un nuevo bloque que tenga como predecesor esta respuesta. Pasado cierto tiempo una de las dos cadenas (la de *atacar* o la de *retirarse*) habrá alcanzado cierta longitud fijada en el protocolo y por tanto se puede considerar que son mayoría quienes apoyan realizar la acción definida en esa cadena.

Puede darse el caso que la decisión a tomar no esté clara y por tanto los generales leales se encontrarán divididos en dos grupos de tamaño similar. En ese caso la decisión dependerá de los mensajes de los traidores, o incluso puede existir un empate en la longitud de ambas cadenas de bloques. En ese caso, como se plantea en [9] las posibles acciones se pueden considerar como igual de buenas así que el empate en la cadena de bloques se puede romper con algún mecanismo fijado en el protocolo, por ejemplo tomando la decisión dada por la cadena cuyo bloque inicial se haya creado antes (suponiendo que se incluye un timestamp)

Así, hemos conseguido una posible solución para el problema de los generales bizantinos donde no se requiere la elección de un líder. Más aún, el posible problema de suplantación de identidad (de un general traidor intentando modificar el mensaje de un general leal) se resuelve mediante el uso del algoritmo ECDSA como se dijo en 3.7.

4.3. Limitaciones del Blockchain

Como se vió en 3.4 dada una función hash y cierta cadena de caracteres el algoritmo de prueba de trabajo está basado en encontrar cierto nonce que unido a la cadena de entrada y al aplicarle la función hash devuelva un resultado con determinadas propiedades fijadas en el protocolo. Esto es una medida del esfuerzo computacional puesto en resolver ese problema

y no es nada democrático, pues un único participante con una CPU avanzada podría igualar o superar al esfuerzo de varios nodos independientes. La idea detrás de usar este método en el blockchain, como se mencionó en 3 es suponer que el *valor intrínseco* de la red en su conjunto está relacionado directamente con el esfuerzo puesto en construir la cadena de bloques. Así, un ataque a una red, como la del Bitcoin por ejemplo, con un gran número de participantes sería muy difícil pues la potencia de cálculo combinada de sus nodos superaría a la de casi cualquier atacante. Y es justamente esta inversión en tiempo de CPU expresada en la cadena de bloques la que mide la importancia de la red.

Todo lo anterior supone que el bien en el que se basa la implementación del blockchain o el objetivo que persigue carece de valor por sí mismo. En el caso de las criptomonedas esto es cierto en general, pero otras situaciones, como la que plantea el problema de los generales bizantinos, puede no cumplirse. En estos casos, dado que puede ser interesante para alguna entidad interferir en el consenso desde un principio, la solución pasa por recurrir a un blockchain de consorcio o privado. Los participantes en este tipo de redes solo tienen en cuenta los bloques generados por ciertos nodos que definidos en una lista (consorcio) o llegan incluso a ignorar los mensajes que no provengan de esos nodos previamente (privado). El carácter público

El consenso que se alcanza en el el blockchain es probabilístico: con cada bloque que se añade aumenta la probabilidad de que los nodos lleguen a un acuerdo [6]. Como para los algoritmos *Paxos* y *Raft* hay que poner ciertas exigencias sobre el número de nodos maliciosos respecto al total de participantes para que sea probable llegar al consenso. Para el Bitcoin se suele aceptar que siempre que un atacante no posea más del 50 % del poder cálculo de la red será prácticamente imposible que pueda afectar al consenso. Aunque se llegue a un acuerdo sí puede ocurrir que un atacante consiga bloquear transacciones legítimas y alterar negativamente el funcionamiento de la red. Por ello en las diferentes implementaciones de criptomonedas se suelen adoptar medidas de seguridad adicionales en este sentido.

Bibliografía

[1] Bitcoin mining now consuming more electricity than 159 countries. https://powercompare.co.uk/bitcoin/. Accessed: 2018-11-13. (Archived by WebCite at http://www.webcitation.org/73uNlJOrn).

[2] Slimcoin a peer-to-peer crypto-currency with proof-of-burn. http://www.doc.ic.ac.uk/~ids/realdotdot/crypto_papers_etc_worth_reading/proof_of_burn/slimcoin_whitepaper.pdf. Accessed: 2019-02-19. (Archived by WebCite at http://www.webcitation.org/76JMlsixP).

[3] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[4] John William Scott Cassels and John William Scott Cassels. *LMSST: 24 Lectures on Elliptic Curves*, volume 24. Cambridge University Press, 1991.

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

[6] Vincent Gramoli. From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, 2017.

[7] Jim Gray and Leslie Lamport. Consensus on transaction commit. *ACM Transactions on Database Systems (TODS)*, 31(1):133–160, 2006.

[8] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[9] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[10] Kristin E Lauter and Katherine E Stange. The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences. In *International Workshop on Selected Areas in Cryptography*, pages 309–327. Springer, 2008.

[11] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[12] Diego Ongaro and John Ousterhout. Raft consensus algorithm. 2015.

[13] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full sha-1. In *Annual International Cryptology Conference*, pages 570–596. Springer, 2017.

[14] Andrew S Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.