¿QUÉ SABEMOS DE?

Blockchain

David Arroyo Guardeño,
Jesús Díaz Vico
y Luis Hernández Encinas

CSIC
CATARATA

Blockchain

David Arroyo Guardeño, Jesús Díaz Vico
y Luis Hernández Encinas

CSIC

CATARATA

Colección ¿Qué sabemos de?

Catálogo general de publicaciones oficiales
http://publicacionesoficiales.boe.es

Diseño gráfico de cubierta: Carlos Del Giudice

ISBN (CSIC): 978-84-00-10478-8
ISBN electrónico (CSIC): 978-84-00-10479-5
ISBN (CATARATA): 978-84-9097-684-5
ISBN electrónico (CATARATA): 978-84-9097-685-2
NIPO: 694-19-065-2
NIPO electrónico: 694-19-066-8
Depósito legal: M-15.055-2019
IBIC: PDZ/PDR/UBW

Índice

Introducción

El problema de la confianza en los sistemas de información y comunicación

Este libro pretende ser una introducción a un concepto de gran resonancia en el momento actual, la tecnología *blockchain*. Esta tecnología se está convirtiendo en una de las grandes protagonistas de la segunda década del siglo XXI, de forma que se la suele considerar como el gran motor de cambio de la esfera digital. Este protagonismo viene derivado del peso central que tiene todo lo relacionado con la gestión de la información en nuestro tiempo, y por el hecho de que la blockchain ha venido a cambiar (para mejor, claro está) dicha gestión. Ahora bien, esta consideración se realiza en no pocas ocasiones sin especificar qué se quiere cambiar y cómo habría de cambiarse. Y lo que es más importante, sin llegar a plantear si no existe ya alguna otra tecnología que solucione ese problema de manera más efectiva y eficiente.

Como se puede desprender de lo anterior, el enfoque que adoptaremos a lo largo de este libro es ambivalente, pues pretende dar cuenta del entusiasmo que ha concitado el fenómeno blockchain, a la vez que intenta no dejarse atrapar por un exceso de confianza en la tecnología. Siendo conscientes de que existe un abuso de simplificación, podemos considerar que en el núcleo de los problemas resueltos en el ecosistema blockchain se encuentra, de hecho, la confianza.

La definición del concepto de *confianza* no es fácil. Es cierto que la irrupción de la blockchain vino de la mano de *bitcoin* (moneda virtual sobre la que volveremos más adelante) y el cuestionamiento del papel que juega el sistema financiero. No obstante, en este punto vamos a intentar resumir lo que pretendemos denotar y connotar con el término confianza en el contexto de los sistemas de información y comunicación. Usaremos este término en el sentido de la acción por la cual una entidad (persona, organización o máquina) otorga credibilidad a otra entidad a la hora de realizar una operación sobre un conjunto de datos. Esa credibilidad es una suerte de externalización en la generación y custodia de la información, y es el pilar sobre el que se han ido construyendo las diversas Tecnologías de la Información y de la Comunicación (TIC). Consecuentemente, la confianza sobre el origen y el contenido de los datos es capital en el correcto funcionamiento de los sistemas de información y comunicación.

Tal y como veremos en el capítulo 1, la criptografía proporciona una serie de mecanismos matemáticos que hacen factible identificar el origen de una fuente de información y comprobar si la información ha sido modificada. En la raíz de este par de familias de soluciones criptográficas se encuentra la necesidad de los sistemas de información de contar con mecanismos de atribución. Esto es, las TIC deben contar con procedimientos que hagan viable saber qué entidad pretende procesar información y qué operaciones puede realizar una entidad sobre tal información. El primer tipo de procedimientos se denomina *autenticación*, mientras que el segundo corresponde a la *autorización*.

Cuando accedemos a nuestro correo electrónico o cuando intentamos recuperar algún fichero que tenemos almacenado en la nube, hemos de probar que somos una entidad que tiene acceso a esos recursos. Para ello, usamos de forma general un nombre de usuario y una contraseña. De hecho, el uso de contraseñas como proceso de autenticación es el mecanismo básico por el cual protegemos el acceso a nuestros servicios en la Internet. Tales servicios son proporcionados por un conjunto de operadores que almacenan nuestra información (mensajes de correo electrónico, documentos, fotografías, etc.), la protegen y la

ponen a nuestra disposición cuando la solicitamos de modo oportuno. Es más, los proveedores más populares de correo electrónico (Gmail, Hotmail, Yahoo) o de almacenamiento en la nube (Dropbox, Google Drive, Box) nos ofrecen esos servicios de forma gratuita, una vez hemos aceptado las condiciones de uso de su plataforma. En sentido estricto, tales proveedores de servicio centralizan nuestro acceso a la información, para lo cual habrán de almacenar de modo seguro nuestra contraseña. Además, dichas plataformas garantizan la confidencialidad de nuestros datos, de modo que solo nosotros accedemos a ellos. Dicho de otra forma, los proveedores de servicio con los que solemos gestionar nuestro día a día constituyen entidades confiables en las que delegamos la custodia de nuestra información (Sánchez-Gómez *et al.*, 2018).

Desafortunadamente, en los últimos años hemos sido testigos de una serie de eventos que podrían llevar a replantearnos la confianza que nos merecen algunos de los servicios que hemos reseñado. Así, no es difícil encontrar información sobre robo de contraseñas en algunas de las plataformas que solemos emplear para guardar nuestra información[1]. Este tipo de ataques pone en peligro el acceso confidencial a nuestros datos y, en caso de que utilicemos la misma contraseña para varios servicios, puede afectar a varios de nuestros perfiles de usuario en Internet. A todas luces, esta situación puede revertir en una quiebra de nuestra confianza en esta amalgama de servicios gratuitos que nos ayudan a realizar muchas de nuestras actividades y rutinas diarias, pero que también imponen un cierto esquema y modelo de dependencia.

Esta quiebra es todavía más relevante si tenemos en consideración todas las implicaciones de la custodia de nuestros datos por terceras partes. Esta cesión lleva implícita la creencia de que tales agentes no van a realizar operación alguna con nuestros

1. Aquí es necesario tener en cuenta que muchos atacantes suelen poner a disposición del público las contraseñas que obtienen tras sus ataques. Por tanto, es más que recomendable verificar si nuestra contraseña ha sido sustraída en algún momento debido a una vulnerabilidad en alguno de los servicios de Internet que empleamos. A tal efecto es de gran utilidad consultar plataformas como https://haveibeenpwned.com/, además de tener una buena política de generación y renovación de contraseñas.

datos sin contar con nuestro consentimiento expreso (Strandburg, 2014). Sin embargo, en la última década son demasiadas las ocasiones donde ha quedado patente que muchos proveedores de Internet, lejos de proporcionar sus servicios de forma gratuita, estaban obteniendo un beneficio explotando los datos de sus usuarios (Enserink y Chin, 2015).

Sucesos como la polémica de Cambridge[2] Analytica (Aoyagi y Adachi, 2018) llevan a cuestionar el papel central que empresas como Google, Facebook, Apple y Amazon juegan tanto en nuestro ciberespacio como en nuestro mundo físico. La desazón y las dudas que tales corporaciones puedan generar no deberían hacernos obviar que nuestro modelo de sociedad requiere el uso masivo de las TIC. Esto es, no podemos renunciar a la Internet, pero tampoco es aceptable que el pago por sistemas eficientes de comunicación sea la degradación de principios democráticos como el derecho a la privacidad. Es por ello que urge buscar alternativas frente a cualquier entrada en el mundo digital que exija un nivel de dependencia con escaso despliegue de políticas para la gestión transparente de los datos. En resumidas cuentas, la correcta y adecuada configuración de la nueva realidad ciberfísica parece demandar un nuevo escenario tecnológico donde la digitalización de nuestros intereses y la explotación de los datos estén basadas en la colaboración de diversos agentes, de acuerdo con un marco legal y normativo que evite la proliferación de monopolios y habilite la justa y proporcionada depuración de responsabilidades (Field *et al.*, 2003).

La quiebra de la confianza en el sector financiero: el surgimiento de Bitcoin

La transición desde una confianza ciega en lo tecnológico hacia un sensato recelo sobre los proveedores de servicios de Internet, no hace sino sobrepasar el dominio de discusión sobre la

2. Recordemos que Cambridge Analytica ha sido acusada de utilizar la información de 50 millones de usuarios de Facebook para influir en los votantes de EE UU en las elecciones presidenciales de 2016 en las que Donald Trump fue elegido como presidente norteamericano.

confianza para ir más allá de lo meramente tecnológico. De hecho, uno de los ejemplos más notorios sobre el cuestionamiento de entidades o autoridades centrales en las TIC viene de la mano del ámbito financiero: la aparición de la criptomoneda bitcoin[3].

Si antes destacamos el carácter central de los proveedores de servicio en la construcción de nuestro espacio social, no podemos ignorar que en ese espacio el intercambio de bienes pasa necesariamente por los bancos. Adquirimos nuevos productos pagando por ellos y todos esos pagos tienen por soporte lo que hoy en día se denomina *dinero fiat* o *dinero por decreto*[4].

En el periodo comprendido entre 1944 y 1971, el dinero estaba basado en el patrón oro, con lo que todo pago quedaba definido en términos de unidades de oro. Este tipo de dinero *fiduciario* establecía, pues, una promesa de pago en virtud del respaldo en oro que posee un cierto país. Dada la escasez de este metal, el Gobierno de Estados Unidos en la época de Richard Nixon decidió respaldar el dinero mediante decreto y no basándose en un material físico de valor como el oro o la plata. Las monedas, los billetes, los cheques, los pagos con tarjeta de crédito y toda variante de transacción o pago electrónico están respaldados únicamente por el Banco Central de cada país. No existe ningún elemento tangible que confiera a un billete valor alguno: ese valor viene dado por decreto del Gobierno de un país; el aval del mismo está dado por la confianza que los ciudadanos de ese país tienen en su Gobierno, así como por los acuerdos y convenios existentes a nivel internacional entre los bancos centrales de

3. De acuerdo con el criterio general utilizado en otros textos sobre la materia, a lo largo del libro emplearemos *Bitcoin* para designar la plataforma que habilita el registro de transacciones sin intermediarios, mientras que notaremos como *bitcoin* a la criptomoneda acuñada por dicha plataforma. Se seguirá el mismo criterio a la hora de mencionar la plataforma Ethereum y la criptomoneda asociada, *ether*.

4. De acuerdo con James S. Coleman (1994), en este libro hemos adoptado como referencia la taxonomía que distingue tres categorías de dinero: dinero primario, dinero fiduciaro y dinero *fiat*. Sin entrar en los entresijos y complejidades de cada uno de estos tipos de dinero, lo relevante en esta obra concierne al periodo de transición comprendido entre 1944 y 1971, y que supuso la progresiva preponderancia del dinero fiat por encima de cualquier otra forma de dinero (Cesarano, 2006). Una de las consecuencias más importantes de esta dinámica fue la consolidación de la figura del Banco Central en la gestión del monopolio del dinero.

todos los países. Consecuentemente, los bancos nacionales de cada país se crean bajo el amparo del Banco Central del mismo y se convierten en el único surtidor de dinero al alcance de los ciudadanos.

El ciudadano de a pie utiliza moneda acuñada por el Banco Central de su país, pero distribuida a través de la red de bancos nacionales. El Banco Central de un país imprime dinero, lo distribuye entre los diversos bancos nacionales que, a su vez, se lo hacen llegar a las distintas empresas, instituciones y organismos gubernamentales que pagan a los trabajadores y pensionistas en moneda fiat. Es más, los bancos tienen la capacidad de dar créditos a los ciudadanos, esto es, de prestar dinero para adquirir nuevos bienes. Estos préstamos deben ser devueltos con un interés estipulado mediante un contrato entre el ciudadano, empresa u organismo y el banco. Este contrato contará en todo momento con la aprobación del Banco Central del país en cuestión.

El Banco Central de cada país es el último garante del dinero que existe en dicho país y, por tanto, es el encargado de vigilar que los bancos de la red nacional aprueben créditos de modo correcto. En toda sociedad de mercado se confía en que el Banco Central realizará de modo riguroso su labor de supervisión, de forma que impedirá políticas de concesión de créditos que puedan poner en peligro el sistema financiero nacional. Esto, no obstante, es algo que actúa con disfunciones y fricciones que en periodos de bonanza económica son manejables, pero que pueden devenir en crisis sistémicas en momentos de incertidumbre e inestabilidad financiera.

Precisamente, en 2007, se originó uno de esos momentos de inestabilidad como consecuencia de la dinámica especulativa llevada a cabo en el sector inmobiliario a lo largo de la primera década del siglo XXI. La concesión de préstamos para la compra de inmuebles sin las suficientes garantías, la generación de toda una cadena de productos financieros que prometían ingentes beneficios sin dejar claro sobre qué se invertía y cuál era el riesgo, y algunos desajustes regulativos y legislativos generaron todo un clímax de entusiasmo que acabó eclosionando y dando origen a una de las crisis económicas más importantes de la historia. Todos aquellos clientes que se habían endeudado asumiendo

que la buena salud del sistema financiero les iba a permitir saldar sus deudas y aumentar sus beneficios en clave de patrimonio inmobiliario comprobaron que la entidad confiable sobre la que asentaban tal convicción no podía frenar las órdenes de embargo ni lograba depurar de modo adecuado las responsabilidades asociadas a muchos de los fraudes cometidos en los años de bonanza. Todo ello propició que una parte considerable de la población comenzara a cuestionar el sistema financiero. Así, en ciertos sectores sociales se fraguó un clima de desconfianza respecto a los bancos centrales de los distintos países o regiones, de forma que ciertos colectivos reaccionaron planteando modelos financieros alternativos orientados a eliminar la dependencia respecto al dinero fiat.

En ese caldo de cultivo surgió Bitcoin. Bitcoin es una solución tecnológica que permite la construcción de un sistema financiero alternativo al modelo del dinero fiat, y lo hace sustituyendo el rol de la autoridad central del banco por un protocolo de consenso entre múltiples entidades. Bitcoin fue propuesto de modo anónimo mediante un artículo publicado en 2008 a través de la lista de correo metzdowd.com (Nakamoto, 2008). Su autor, que responde al seudónimo de Satoshi Nakamoto, escogió un canal de difusión distinto del habitual a la hora de hacer públicas nuevas propuestas. Nakamoto eludió el proceso tradicional de revisión por pares de las proposiciones científicas, de forma que optó por realizar una propuesta teórica y presentar la correspondiente implementación práctica. Si bien es cierto que la propuesta teórica carece de algunos de los elementos básicos que se espera en trabajos de la envergadura que ha supuesto Bitcoin, la implementación práctica es totalmente funcional y pudo ser utilizada por el público general desde el primer momento. En definitiva, la iniciativa de Nakamoto tiene un doble cariz subversivo, en el sentido de que trata de eludir el control central del sector financiero e industrial y, al mismo tiempo, persigue evitar cualquier obstáculo que el mundo académico pudiera poner a su progreso. En lugar de someter su trabajo a la evaluación profunda y pausada de la comunidad criptográfica y de la ingeniería de sistemas de información, Nakamoto decidió someterla a una evaluación abierta, libre y sin ningún tipo de cortapisas.

¿Qué problemas viene a resolver la blockchain? ¿Cómo los resuelve?

Bitcoin se adscribe al ámbito de las monedas electrónicas, y en este sentido es continuadora del trabajo efectuado en la década de los ochenta del siglo XX por David Chaum (Chaum, 1983). A diferencia de las diversas soluciones propuestas por Chaum, la apuesta de Nakamoto evita la existencia de autoridad central alguna, lo que suponía eliminar el papel preponderante de los bancos en el sector financiero. El conjunto de protocolos desarrollados por Chaum como soporte de *Digicash*, no obstante, pusieron las bases de Bitcoin. En concreto, el uso de primitivas criptográficas para la validación anónima de transacciones es una de las grandes aportaciones de Digicash. Tales primitivas establecen la base de la creación de seudónimos, que serán usados en Bitcoin, para emular la propiedad de no trazabilidad del dinero físico. En efecto, de la misma forma que dado un billete o moneda no somos capaces de inferir el conjunto de propietarios que ha tenido, tampoco deberíamos poder detectar el origen de una moneda electrónica. El conjunto de primitivas criptográficas antes aludidas posibilita que la transferencia de dinero se realice de modo ciego, esto es, sin saber quién es el donante del dinero una vez se ha recibido el mismo (al igual que ocurre en las operaciones financieras realizadas con dinero en metálico). La protección de la identidad del propietario de la moneda electrónica se consigue mediante criptografía, y por ello a este tipo de moneda electrónica se le llama *criptomoneda*.

La no trazabilidad de una criptomoneda supone un problema en lo relativo a la satisfacción de las leyes para la persecución del lavado de dinero[5], pero es algo indispensable a la hora de

5. Esta es una diferencia fundamental entre el dinero electrónico y las transferencias electrónicas. En el dinero electrónico se produce una cesión de valor desde una identidad (seudo)anónima a otra identidad (seudo)anónima. En el caso de las transferencias financieras, el intercambio se lleva a cabo entre los propietarios de las diversas cuentas bancarias involucradas. Las leyes de la mayoría de los países, con la excepción de los paraísos fiscales, exigen que los propietarios de tales cuentas estén debidamente identificados y registrados. Este registro es, en último término, referido al banco central de cada país que, además, exigirá un control exhaustivo de las transferencias autorizadas y realizadas por los bancos de la red financiera nacional.

considerar que se cuenta con una alternativa real a la moneda física. El gran avance de Bitcoin con respecto a Digicash viene dado por la forma en la que se soluciona el "problema del doble gasto", que no es sino el desafío de impedir que una misma moneda pueda ser empleada por su propietario para pagar varias veces. Esto no es factible con la moneda física (si pago una vez pierdo el control físico de la moneda o el billete, con lo que no puedo volver a utilizarlo), y supone uno de los grandes retos a satisfacer cuando se trata de dinero digital. En el caso de Digicash, el doble gasto fue solucionado mediante la inclusión del banco como tercera parte confiable, paso que dificultaba sobremanera la interacción entre usuarios y entre comercios y usuarios. Este fue uno de los principales motivos que perjudicó la adopción de la tecnología. En el caso de Bitcoin se impulsó un protocolo que habilitaba de "forma natural" la interacción entre usuarios, para lo cual la criptomoneda bitcoin emulaba el tipo de dinero fiduciario que se generaba de acuerdo con un recurso limitado. Si a lo largo de la historia se han propuesto diversos modelos de dinero fiduciario utilizando como referencia un metal precioso, en el caso de bitcoin el recurso limitado es la resolución de un problema computacional complejo, esto es, un problema cuya resolución involucra un alto coste computacional. Dicho de otra forma, con bitcoin las criptomonedas empiezan a acuñarse mediante la ejecución de código *software* y el gasto energético asociado.

Tal y como veremos en el capítulo 2, el procedimiento de creación de nuevas monedas en Bitcoin se efectúa mediante la resolución de un desafío o puzle criptográfico de alto coste computacional (Nakamoto, 2008). Los usuarios involucrados en esta tarea son los denominados *mineros*, y el desafío sobre el que trabajan tiene por objeto el registro de transacciones realizadas por otros usuarios de la red. Tal registro se lleva a cabo en bloques de datos, donde cada bloque tiene un tamaño máximo preestablecido. Un minero está a la escucha de las diversas transacciones que son enviadas a través de la red Bitcoin. A medida que va recibiendo transacciones las incorpora en un bloque de datos. Una vez se completa el tamaño de dicho bloque, el minero comienza a resolver el desafío criptográfico al que hacíamos mención

anteriormente. El minero que resuelve el desafío enlaza el bloque creado con los bloques previos, difunde el resultado a través de la red y comienza a incorporar transacciones en un nuevo bloque. Dado que las transacciones en Bitcoin se escriben en bloques que son enlazados, se dice que la estructura de datos que almacena los apuntes de los intercambios monetarios es una cadena de bloques o *blockchain*.

La red Bitcoin carece de nodo central y se configura como una red entre pares (usuarios) o red P2P (del inglés *peer-to-peer*). Todos esos usuarios pueden acceder a una copia de la blockchain de Bitcoin, pero solo aquellos que actúan como mineros escribirán en dicha blockchain. Por último, cabe subrayar que el minado (es decir, la escritura) de bloques debe ser aceptado por la mayoría de los nodos de la red. En efecto, cuando un minero informa de la creación de un nuevo bloque difunde a través de su entorno la cadena resultante. Aquí hay que tener en cuenta que, al tratarse de una red no centralizada, normalmente un nodo va a recibir mensajes con cadenas de bloques provenientes de distintos mineros. En este supuesto, el criterio por defecto consiste en aceptar aquella cadena que tenga un mayor número de bloques y, tras ello, difundir la cadena aceptada. Con este procedimiento se alcanzará un consenso sobre la cadena de bloques aceptada por la red y, por ende, del conjunto de transacciones que han sido aceptadas. Dado que la estructura de datos en la cadena de bloques se constituye en una prueba del trabajo realizado por un minero, se dice que en Bitcoin se alcanza el consenso mediante "prueba de trabajo" o PoW (del inglés, *Proof-of-Work*)[6]. El minero responsable de la cadena de bloques consensuada recibirá una recompensa por el trabajo realizado. Esta recompensa es una fracción de la criptomoneda bitcoin, y constituye el procedimiento de impresión de nueva moneda.

6. La PoW es otra de las herencias que Bitcoin ha sabido incorporar de modo práctico a la hora de construir su sistema distribuido de registro de transacciones. En este caso, el trabajo seminal corresponde a Cynthia Dwork y Moni Naor, que en 1992 propusieron la PoW como parte de un sistema para evitar correo basura o *spam*. La primera aplicación de la PoW en el ámbito del dinero electrónico la encontramos en HashCash, una propuesta de Adam Back que data de 1997.

Contribución de la blockchain a la transformación digital: posibilidades y limitaciones

Al margen de los detalles del proceso de minado y la propagación del consenso, lo relevante es que la cadena de bloques aceptada de forma mayoritaria por la red P2P es, además, "inmutable". Es decir, ningún nodo de forma aislada puede modificar el contenido previo de los bloques que anteriormente han sido consensuados. Solo es factible la reescritura del registro distribuido mediante la colusión de un conjunto de nodos que acumulen más de la mitad de capacidad de cómputo de la red Bitcoin. Este factor, unido al hecho de que cualquiera pueda descargarse todas las transacciones efectuadas desde la primera que llevó a cabo Nakamoto el 3 de enero de 2009, hace que la blockchain de Bitcoin sea presentada como herramienta clave para sostener un modelo de gestión transparente de las actividades financieras. Sin embargo, cuando se recalca el carácter transformador de blockchain, se está pensando en ir más allá de su aplicación como mera tecnología para el registro contable distribuido o DLT (de las siglas en inglés de Distributed Ledger Technology)[7].

Pese a haber sido concebida inicialmente como un mero mecanismo de recolección de trazas financieras, los protocolos de la blockchain de Bitcoin permiten gestionar un conjunto de campos libres que la hacen atractiva para otro tipo de servicios. Así, existe un gran número de iniciativas que emplean la blockchain de Bitcoin como canal de control de sistemas de gestión o certificación documental (MaidSafe, Stampery), de trazabilidad de activos físicos (EverLedger), de compartición de recursos de transporte (La'Zooz), o de protección de derechos de autor (MediaChain).

El conjunto de soluciones construidas sobre la blockchain de Bitcoin, no obstante, requiere el despliegue de una arquitectura externa que, en cierta medida, involucra una recentralización en la gestión de la información. Dicha consecuencia solo puede ser evitada mediante la incorporación de mecanismos en

7. Aunque se suele utilizar DLT como sinónimo de blockchain, hay que tener bien presente que una blockchain es solo un tipo concreto de DLT.

la blockchain que habiliten la toma de decisión de modo autónomo y reduzcan, tanto como sea posible, cualquier dependencia respecto a agentes o elementos de computación externos. Pues bien, los denominados "contratos inteligentes" constituyen ese paso adicional en el desarrollo de la blockchain y, *de facto*, representan el surgimiento de la denominada *blockchain 2.0*. El ejemplo más representativo de esta generación de blockchain es la tecnología *Ethereum*, que será tratada en el capítulo 3. Esta plataforma incorpora una máquina virtual que habilita la codificación y ejecución de programas software que determinan los términos y condiciones para el intercambio de activos entre agentes. Dicho de otra forma, Ethereum incluye una funcionalidad para la concreción de acuerdos en clave de programas software. Esta funcionalidad es la piedra angular de las iniciativas de financiación colectiva o *crowdfunding* y de la transición (no exenta de problemas) desde las ofertas públicas iniciales de acciones (IPO, Initial Public Offering) a las ofertas iniciales de moneda (ICO, Initial Coin Offering). También es la base de las denominadas "propiedades inteligentes" que permiten definir, mediante códigos en la blockchain, la forma en la que los dispositivos de la Internet de las cosas (o IoT, Internet of Things) reaccionan ante eventos que tengan lugar en su entorno.

Tanto Bitcoin como Ethereum están basados en una blockchain pública, esto es, cualquiera puede insertar y leer información en y desde la blockchain que las soporta. Esto no es compatible con todos los modelos de negocio (Peters y Panayi, 2015). Ciertamente, sectores como el financiero tienen necesidades y restricciones normativas y legales que imposibilitan que toda la información de gestión sea pública. Por lo demás, las soluciones adoptadas por las blockchains públicas que existen hoy en día requieren que los usuarios almacenen un gran volumen de información si quieren supervisar las transacciones, lo que erosiona la escalabilidad de la red blockchain. Las blockchains "permisionadas" o "autorizadas" (como Hyperledger o Ripple) surgen para dar solución a este par de cuestiones, y serán analizadas en el capítulo 4.

Ejemplos como el incidente de seguridad de TheDAO (versión descentralizada de plataformas de *crowdfunding* como

Kickstarter)[8] o las fluctuaciones en la cotización de las criptomonedas no hacen sino indicar que el diseño de soluciones basadas en blockchain tiene un cariz problemático. Con vistas a dotar al lector de este libro con recursos para la identificación de riesgos y la adopción de soluciones blockchain de acuerdo con sus expectativas y modelo de negocio, en el capítulo 5 examinaremos los diversos contextos críticos a los que se enfrenta la ingeniería de la blockchain. El enfoque que se seguirá será de índole primordialmente práctica, de forma que en todo momento se identificarán las funciones y propiedades de los distintos ejemplos de blockchain. Posteriormente, se procederá a discutir dichas propiedades en los contextos más relevantes en los que la blockchain ha sido propuesta como elemento esencial de la transformación digital. En este último aspecto, se subrayarán las principales limitaciones que tienen en el momento actual tanto la aproximación teórica de la familia blockchain como las diferentes implementaciones prácticas. Finalmente, el libro concluirá con un resumen de los principales esfuerzos de investigación y desarrollo que tratan de adecuar la blockchain a las necesidades de negocio, los requisitos legales y las posibilidades tecnológicas en la actualidad.

8. TheDAO fue un intento fallido de organización descentralizada y autónoma basada en contratos inteligentes de Ethereum. En junio de 2016, la ICO vinculada a esta iniciativa consiguió un total de 250 millones de dólares. A los pocos días de comenzar su funcionamiento, TheDAO fue atacada y se tuvo que optar por dejar de explotar la plataforma. Aunque las consecuencias del incidente de TheDAO se analizarán en detalle en el capítulo 5, se puede encontrar un estudio pormenorizado de las implicaciones en términos de gobernanza en DuPont (2017).

CAPÍTULO 1

Fundamentos criptográficos de la gestión de la confianza

En este capítulo vamos a definir y considerar los conceptos criptográficos que son necesarios para comprender los fundamentos de las blockchains. En los capítulos siguientes estudiaremos cómo las blockchains los incorporan de forma que se posibilita un esquema de comunicación no centralizado. Todos estos mecanismos van encaminados a gestionar la confianza en la información que se utiliza. Se trata, en definitiva, de controlar la integridad de la información, es decir, de garantizar mediante protocolos criptográficos que la información que se almacena o se envía por cualquier medio no ha sido modificada y que procede de quien asegura ser su remitente.

La principal herramienta que se emplea con el fin de asegurar la integridad de una información son las denominadas "funciones resumen" o funciones *hash*. La verificación de que la información procede de su origen real, esto es, que el remitente no ha sido suplantado, se lleva a cabo mediante los protocolos de firma electrónica y, finalmente, los diferentes mecanismos para la optimización del almacenamiento e indexación de contenido calculado con resúmenes (*hashes*) criptográficos se realizan mediante los "árboles de Merkle".

No obstante, antes de considerar estos conceptos criptográficos, haremos un breve resumen de algunos aspectos de los sistemas de numeración que serán utilizados en otras partes de este libro.

Sistemas de numeración

Todo número está definido de acuerdo con una base numérica. De modo habitual, trabajamos en base decimal o base 10. Así, tenemos diez dígitos distintos de una cifra, del 0 al 9, mientras que todos los números con más de una cifra se obtienen sumando el producto de un dígito por sucesivas potencias de 10, considerando su posición (unidades, decenas, centenas, etc.). Por ejemplo, el número 123 equivale a computar

$$123 = 100 + 20 + 3 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0.$$

En informática, además de la base decimal se trabaja con base binaria y base hexadecimal. En el caso de la base binaria existen solo dos dígitos, 0 y 1, mientras que en base hexadecimal hay 16 dígitos distintos: 0,1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Los números expresados en base hexadecimal se suelen representar haciendo uso del prefijo 0x, y su valor en formato decimal se obtiene calculando la suma del producto de cada dígito por una potencia de 16. Además, se debe tener en cuenta que cada uno de los diez primeros dígitos hexadecimales equivale a cada uno de los dígitos decimales correspondientes, mientras que la A hexadecimal equivale al 10 decimal, la B corresponde al 11 decimal, etc. Así, 0xEF se transforma a decimal mediante la siguiente operación:

$$0xEF = 14 \cdot 16^1 + 15 \cdot 16^0 = 224 + 15 = 239.$$

Por otra parte, cuando se considera Base58, la base numérica es 58, que corresponde con el número total de caracteres (a, b, c, etc., en minúscula y en mayúscula) más el número de dígitos decimales, excluyendo los caracteres y dígitos que pueden generar ambigüedad (0 y O —o mayúscula—, l e I —i mayúscula—). Como se explicará en el capítulo 2, en Bitcoin se emplea una variante del formato Base58 denominada Base58Check.

Funciones resumen

Una de las primitivas criptográficas que en los últimos años ha adquirido mayor relevancia son las funciones resumen (*hash*, en inglés). Es importante destacar que tales funciones no cifran ni descifran mensajes, pero son las herramientas indispensables para comprobar la integridad de determinada información, al margen de otras aplicaciones de gran interés.

De modo muy resumido, se puede definir una función resumen como una función que es capaz de transformar una información o mensaje de cualquier longitud o tamaño en bits, en otro que tiene un tamaño fijado de antemano (Fúster Sabater *et al.*, 2012). Al resultado de tal transformación se le denomina resumen o *hash*.

Inicialmente, las funciones resumen fueron propuestas para ser empleadas en los protocolos de firma electrónica, de modo que estos fueran más eficientes, lo que se conseguía mediante el procedimiento de firmar, en lugar del mensaje que se quería firmar, el resultado de aplicar una función resumen al mismo, esto es, se firmaba un resumen del mensaje. Al ser dicho resumen una información de longitud mucho menor, se ganaba tiempo tanto en el proceso de firma —dado que los cálculos eran más sencillos— como en el de envío de la propia firma, puesto que se requería menor ancho de banda para la transmisión.

Con posterioridad a este primer uso de las funciones resumen, sus aplicaciones se han extendido de forma considerable a otros ámbitos relacionados, en general, con la protección de la información y, en particular, para garantizar su integridad.

Una función se dice que es una "función unidireccional con trampilla" (TOWF, *Trapdoor One-Way Function*) si la función definida entre los conjuntos X e Y

$$f: X \rightarrow Y, \text{ con } f(x) = y$$

cumple las siguientes condiciones:

1. f es una función unidireccional, esto es, es fácil, computacionalmente hablando, calcular $f(x) = y$, para todos los elementos x de X. Además, para la mayor parte de los

elementos del conjunto Y que proceden de algún elemento de X es difícil, desde el punto de vista computacional, encontrar algún elemento x de X, para el que $f(x) = y$.
2. Si se conoce alguna información adicional, denominada "trampilla", entonces es factible calcular, en un breve periodo de tiempo, un elemento x de X de modo que vaya a parar por f a un elemento y de Y dado.

Estas funciones son fácilmente construibles haciendo uso de herramientas matemáticas relacionadas con la aritmética modular (para una introducción elemental a este tipo de aritmética, puede consultarse Hernández Encinas, 2016), es decir, con una aritmética en la que el resultado final de una operación determinada es el resto que se obtiene una vez hecha la división, entre un número dado, del resultado inicial. El número dado se conoce como "módulo". En esta aritmética se incluye la expresión (mod n) para indicar el número por el que se hace módulo. Por ejemplo, si se considera como módulo el número entero 13, se tienen los siguientes resultados:

$$8 + 7 = 15 = 13 \cdot 1 + 2 \equiv 2 \pmod{13},$$
$$8 \cdot 7 = 56 = 13 \cdot 4 + 4 \equiv 4 \pmod{13}.$$

A partir de las funciones unidireccionales se pueden definir directamente las funciones resumen de la siguiente manera: una función resumen o una función hash es una función unidireccional que se aplica a un mensaje dado de tamaño variable, m, perteneciente a un conjunto de mensajes, M, de modo que proporciona un resumen del mismo que tiene un tamaño fijo y predeterminado de bits, sea este tamaño n. Así pues, el resumen o hash de un mensaje de longitud variable es una colección de bits (ceros y unos) de longitud prefijada. Por tanto, se puede considerar que una función resumen, h, está definida de la siguiente manera:

$$h: M \rightarrow \{0,1\}^n, \qquad \text{con } h(m) = \underline{m},$$

donde $\underline{m}$ es una secuencia de n ceros y unos.

Las funciones resumen son públicas, es decir, su definición concreta no se considera secreta y dado que transforman un mensaje de cualquier longitud en una colección de n bits, se deduce que el número de posibles resúmenes a obtener es finito y mucho menor que el de mensajes. Dicho de otro modo, siempre habrá mensajes diferentes cuyos resúmenes coincidan. Se dice que se produce una "colisión" cuando dos mensajes distintos proporcionan el mismo resumen.

En el caso particular de que se considere, por ejemplo, $n = 256$, tal función resumen proporciona resúmenes de 256 bits, esto supone que el número de posibles resúmenes será de 2^{256}; mientras que el número de mensajes de cualquier longitud es infinito.

Dado que las funciones resumen son unidireccionales, deben ser "fácilmente computables", esto es, el tiempo necesario para determinar el resumen de un mensaje dado debe ser pequeño (tiempo polinómico); mientras que el tiempo requerido para calcular un mensaje del que únicamente se conoce su resumen deber ser muy largo, del orden de cientos de miles de años (tiempo exponencial o subexponencial), incluso con los mejores algoritmos y la mejor tecnología disponible. Cuando el tiempo es tan grande, suele decirse que la tarea a realizar es "computacionalmente difícil" o inabordable.

Otras propiedades que deben cumplir estas funciones son las siguientes:

- Dependencia de bits: el resumen de un mensaje o documento, $h(m) = \underline{m}$, debe ser una función compleja dependiente de todos los bits del mensaje, de tal manera que si se cambia un bit del mensaje, su resumen debe cambiar, aproximadamente, en la mitad de sus bits.
- Resistencia a la preimagen: dado un resumen $\underline{m}$, es computacionalmente difícil obtener un mensaje m de modo que $h(m) = \underline{m}$. Dicho de otro modo, toda función resumen debe ser difícil de invertir.
- Resistencia a la segunda preimagen: dado un mensaje m_1, es computacionalmente difícil encontrar otro mensaje diferente, m_2, $m_1 \neq m_2$, cuyos resúmenes coincidan, esto es, tales que $h(m_1) = h(m_2)$.

- Resistencia a colisiones: es computacionalmente difícil encontrar dos mensajes distintos cualesquiera m_1 y m_2, con $m_1 \neq m_2$, de modo que $h(m_1) = h(m_2)$.

Conviene mencionar que la primera de las propiedades es la que relaciona este tipo de funciones con la integridad de los datos dado que, si tales datos son modificados, aunque solo sea en un bit, el resultado de la función muestra una gran diferencia entre la información original y la modificada, señalando su falta de integridad.

Además, las dos últimas propiedades son muy diferentes a pesar de su similar enunciado. En la resistencia a la segunda preimagen se supone conocido uno de los mensajes y se trata de encontrar un mensaje diferente que tenga el mismo resumen que el primero. Por su parte, en la resistencia a colisiones no se imponen condiciones sobre los mensajes; se trata de encontrar dos mensajes cualesquiera que tengan el mismo resumen. Por tanto, la resistencia a colisiones es una condición más débil que la resistencia a la segunda preimagen, por lo que es más fácil, computacionalmente hablando, encontrar colisiones.

Se exige que las funciones resumen verifiquen estas condiciones para garantizar, hasta donde sea posible, que no son vulnerables, es decir, que son seguras y cumplen su objetivo principal: impedir que a partir de un resumen dado pueda conocerse el mensaje del que procede y que nadie sea capaz de sustituir el mensaje original por otro diferente cuyo resumen sea el dado.

El ataque más sencillo y efectivo que puede montarse contra una función resumen se conoce como el "problema del cumpleaños".

Este problema puede enunciarse en los siguientes términos: "Si un año tiene $n = 365$ días, ¿cuántas personas debe haber en una sala para que la probabilidad de que dos de ellas celebren su cumpleaños el mismo día sea mayor del 50%?". Una forma alternativa de enunciar el mismo problema podría ser: "Si en una sala hay k personas, determinar cuándo se tendrá la confianza de que dos de ellas celebren su cumpleaños el mismo día, es decir, que la probabilidad de que esto suceda sea $P_k > 0{,}50$".

El enunciado se corresponde con la situación de las funciones resumen, dado que las personas juegan el papel de los mensajes y las fechas de los cumpleaños la de los resúmenes. Se trata de encontrar dos mensajes (personas) cuyos resúmenes (dos cumpleaños) coincidan. La solución a este problema es tan sorprendente que el problema se conoce también como la "paradoja del cumpleaños". De hecho, si $k = 23$, resulta que

$$P_{23} = 0{,}5072972343 > 0{,}50,$$

de modo que basta con que en la sala haya 23 personas para que la probabilidad de que dos de ellas celebren su cumpleaños el mismo día sea mayor del 50%.

El nombre de paradoja (aunque en sentido estricto el problema no sea una paradoja) se debe al hecho sorprendente de que la presencia de solo 57 personas, es suficiente como para que la probabilidad de que dos de ellas cumplan años el mismo día sea mayor del 99%.

A continuación, mencionamos las funciones resumen más utilizadas. Hasta hace unos años, la función más empleada fue la MD5 (Message Digest 5), propuesta por Rivest en (Rivest, 1992), que proporcionaba resúmenes de 128 bits. Una vez que se publicaron algunas debilidades de la función MD5, esta ha dejado de utilizarse. De hecho, en ambientes criptográficos donde la seguridad es una de las facetas primordiales, la función MD5 está prohibida.

La función SHA-1 o algoritmo resumen seguro (Secure Hash Algorithm), que proporciona un resumen de 160 bits, fue adoptada por el Instituto Nacional de Estándares y Tecnología norteamericano (NIST, National Institute of Standards and Technology) como función resumen estándar (NIST, 2002). Dado que sus resúmenes son más largos que los de sus antecesoras y que el diseño es similar, se considera más segura. SHA-1 sigue empleándose en la actualidad con bastante frecuencia, a pesar de que se han encontrado colisiones. También en este caso su uso está absolutamente descartado por la mayoría de los organismos e instituciones internacionales.

La familia de funciones resumen SHA-2 es la sucesora de la función SHA-1 y se ha propuesto una vez que se ha conocido

que es posible encontrar colisiones para la SHA-1 con menos operaciones de las esperadas. El algoritmo de la función resumen SHA-2 contiene varios subalgoritmos: SHA-224, SHA-256, SHA-384 y SHA-512, que proporcionan resúmenes de mensajes de longitudes 224, 256, 384 y 512 bits, respectivamente. Estos tamaños hacen que aumente considerablemente su seguridad (NIST, 2002).

Las funciones SHA-256 y SHA-512 se calculan con palabras de 32 y 64 bits respectivamente y las dos emplean diferentes constantes y número de rondas, pero su estructura es, en esencia, la misma. SHA-224 y SHA-384 son versiones truncadas de las funciones SHA-256 y SHA-512, respectivamente y se determinan usando valores iniciales diferentes. Hoy en día, SHA-2 es la función más utilizada debido a las debilidades encontradas en SHA-1.

El NIST lanzó, en noviembre de 2007, una competición internacional (SHA-3 Cryptographic Hash Algorithm Competition) con el fin de elegir una función resumen estándar que cumpliera determinados requisitos de eficiencia y seguridad. El anuncio de la función ganadora de esta competición se hizo público el 2 de octubre de 2012 y la elección recayó en la propuesta Keccak[9], que ha dado lugar a la familia de funciones SHA-3 (NIST, 2014), formada por las versiones de 256 y 512 bits.

Esta familia incluye las denominadas funciones esponja (*sponge functions*) que toman como entrada una cadena de bits de cualquier longitud y devuelven otra cadena binaria de longitud fija, de tal manera que su modo de operar se basa en una permutación de longitud fija y una regla de relleno (*padding*). De forma más concreta, la SHA-3 puede considerarse como una generalización tanto de las funciones resumen (salida de longitud fija) como de los cifradores en flujo (entrada de longitud fija).

A modo de ejemplo, en la siguiente tabla se presentan, en la segunda columna, los diferentes resúmenes que corresponden a las funciones que se indican en la primera columna, para este párrafo en particular. Conviene señalar que las correspondientes

9. La función Keccak (en sus versiones de 256 y 512 bits) fue adoptada como función resumen en Ethereum (véase capítulo 3) una vez propuesta por sus autores y antes de que fuera elegida como ganadora en este concurso por el NIST. Véase http://keccak.noekeon.org/(último acceso 11/03/2019).

salidas se han escrito en base 16 (hexadecimal) con el fin de ahorrar espacio (cada carácter hexadecimal representa 4 bits y el prefijo "0x" hace referencia a que lo que sigue está en hexadecimal).

FUNCIÓN	RESUMEN
MD5	0x6d4dc1ec30d537fdeeb71676667659d2
SHA-1	0x8239e330f6ad6134fa2b1a16c4fb5da8453267a3
SHA2-256	0x1d0eac3929792dff54ff1d94fe2d45f828d82d8b8ad57cc4287a788e0326ea9e
SHA2-512	0xb6f8fa068768be9b4dfa33af797a285194ae45b209e8ed904cb4045b6f237029df20dd2dad7d216f6b6c24207fab41cf8d7bb22afd374c30d5c3cb5e5076a0eb
SHA3-256	0xe11bf228806cac947406b0d6b50b987692430e675384ee9c1f6c3f0911a8232d
SHA3-512	0xd3c65fe0e496c4cf933006448504112b361deeb4441e50ab1f3209ee303b3c6fb867c6ad932433e1ccfe3bca6600d9cb5f42f8892fceea36a3f5e47c2cd099b1

Existen otras funciones hash también de uso extendido como las siguientes: Adler32, Haval, RipeMD128, RipeMD160, Tiger, Whirlpool, etc. Posiblemente, una de las más empleadas sea la verificación por redundancia cíclica o suma de verificación (también llamada *checksum* o CRC Cyclic Redundancy Check), pero su fin no tiene objetivos de seguridad ni de integridad, dado que los resúmenes solo tienen 32 bits (4 bytes). De hecho, el resumen del párrafo anterior para esta función es 0x15722ae8. Su principal objetivo es el de codificar información para detectar errores, esto es, verificar si se ha modificado algún bit de información y se emplea, sobre todo, en redes y en dispositivos de almacenamiento para detectar cambios en los datos.

Cálculo de la solución al problema del cumpleaños*[10]

La solución al problema del cumpleaños, esto es, determinar el número de personas, *k*, que debe haber en la sala para que la probabilidad de que al menos haya dos de ellas que tengan el mismo cumpleaños se calcula a través de la probabilidad complementaria,

10. A lo largo del texto existen secciones de alta complejidad técnica y que no son estrictamente necesarias para comprender los aspectos básicos de la tecnología blockchain. Estas secciones están identificadas mediante el símbolo (*).

Q_k, es decir, determinando la probabilidad de que no existan dos personas de entre las k que hay en la sala que no celebren su cumpleaños en la misma fecha y luego se determina la probabilidad complementaria, $P_k = 1 - Q_k$.

Una sencilla explicación de cómo resolver este problema podría ser la siguiente. Supongamos que el número de días de un año es de 365 (para evitar el caso de los bisiestos). Comencemos por el caso en el que en la sala hay una única persona. Si en la sala entra otra persona, la probabilidad de que las dos no cumplan años el mismo día es que la que acaba de entrar celebre su cumpleaños cualquiera de los 364 días del año que son diferentes del cumpleaños de la primera. Así pues, la probabilidad de que en una sala haya dos personas que no cumplan años el mismo día es el de casos favorables (364) dividido por el de casos posibles (365), es decir,

$$Q_2 = 364 / 365.$$

Si ahora entra otra persona en la sala, para que las tres no cumplan años el mismo día, la que acaba de entrar deberá cumplir años cualquiera de los 363 días restantes, por lo que la probabilidad de que las tres no cumplan años el mismo día es:

$$Q_3 = (364 \cdot 363) / 365^2.$$

Si entra una cuarta persona, deberá cumplir años cualquier día de los 362 días restantes del año, es decir, la probabilidad que buscamos en este caso será:

$$Q_4 = (364 \cdot 363 \cdot 362) / 365^3.$$

Repitiendo este argumento a medida que van entrando personas en la sala, en un momento dado en el que haya k personas, la probabilidad de que no haya dos de ellas que cumplan años el mismo día será:

$$Q_k = (n-1)\ (n-2) \cdots (n-k+1) / n^{k-1} = (n-1)! / (n^{k-1}\ (n-k)!).$$

Si ahora calculamos la probabilidad complementaria a la anterior, tendremos que la probabilidad de que habiendo k personas en una sala haya dos de ellas que cumplan años el mismo día es:

$$P_k = 1 - Q_k = 1 - ((n-1)! / (n^{k-1} (n-k)!)).$$

Para el caso particular de $k = 23$ personas, se obtienen los siguientes valores

$$Q_{23} = 0{,}4927027657, P_{23} = 1 - Q_{23} = 0{,}5072972343 > 0{,}50,$$

de modo que basta con que en la sala haya 23 personas para que la probabilidad de que dos de ellas celebren su cumpleaños el mismo día sea mayor del 50%.

Firmas electrónicas

La firma electrónica (o digital) de un documento es un procedimiento que permite asegurar su autoría, de forma similar a como lo puede garantizar la firma manuscrita, pero en formato electrónico. La firma electrónica nació por la necesidad de garantizar que los mensajes o documentos enviados por medios electrónicos procedían efectivamente de quienes eran sus remitentes, dado que podría darse el caso de que una persona suplantara al remitente verdadero, con lo que el destinatario podría ser engañado sobre la procedencia del mensaje o documento.

Esta necesidad se ha convertido hoy en día en una enorme ventaja, dado que toda persona que posea un certificado electrónico puede firmar digitalmente cualquier documento, de modo que quede constancia de su autoría. Este hecho ha supuesto que, en la actualidad, la firma electrónica sea, probablemente, el protocolo criptográfico más empleado (Fúster Sabater *et al.*, 2012).

La firma electrónica tiene en común con la manuscrita que debe ser fácil de calcular y difícil de falsificar; por el contrario, la firma electrónica se diferencia de la manuscrita en que esta es casi siempre la misma (solo hay pequeñas diferencias gráficas)

sea cual sea el documento que se firme, mientras que aquella se calcula para cada documento (fichero electrónico), de modo que el resultado de la firma electrónica no solo depende del remitente, sino también del documento que se firma. Dicho de otro modo, la firma de un mismo remitente es diferente para cada documento y, así, es única para cada uno.

Al margen de esta característica, la firma electrónica se lleva cabo mediante un protocolo criptográfico que permite no solo demostrar la autoría de un documento sino que también garantiza su integridad, esto es, que el documento firmado que se recibe en una transmisión no ha sido modificado a lo largo de la misma.

Desde el punto de vista puramente legal, la Ley 59/2003 de firma electrónica (Cortes Generales, 2003) señala la existencia de tres tipos de firma diferentes: la electrónica, la avanzada y la reconocida:

- La firma electrónica es el conjunto de datos en forma electrónica, consignados junto a otros o asociados con ellos, que pueden ser utilizados como medio de identificación del firmante (artículo 3.1).
- La firma electrónica avanzada es la firma electrónica que permite identificar al firmante y detectar cualquier cambio ulterior de los datos firmados, que está vinculada al firmante de manera única y a los datos a que se refiere y que ha sido creada por medios que el firmante puede mantener bajo su exclusivo control (artículo 3.2).
- La firma electrónica reconocida es la firma electrónica avanzada basada en un certificado electrónico reconocido y generada mediante un dispositivo seguro de creación de firma (artículo 3.3).

Además, la firma electrónica reconocida tendrá respecto de los datos consignados en forma electrónica el mismo valor que la firma manuscrita en relación con los consignados en papel (artículo 3.4).

Al margen de los aspectos legales, el protocolo criptográfico de firma electrónica consta de dos procedimientos: la elaboración de la firma, donde el firmante o remitente genera su firma para

un documento, y la verificación de la firma, en la que cualquier usuario que lo desee puede verificar la validez de dicha firma para el documento firmado.

Los protocolos de firma requieren de un criptosistema asimétrico (o de clave pública) para ser elaboradas o verificadas, de modo que el firmante debe poseer una clave pública y su correspondiente clave privada asociada. Vamos a resumir en qué consiste un criptosistema de este tipo. Para más información, remitimos al lector a cualquier libro de criptografía (por ejemplo, Fúster Sabater *et al.*, 2012; Hernández Encinas, 2016).

Un criptosistema asimétrico es un procedimiento criptográfico formado por una función de cifrado, C, y una de descifrado, D, de modo que los usuarios pueden cifrar y descifrar determinada información (en general un fichero electrónico correspondiente a un mensaje o documento que se desea hacer llegar de forma confidencial a un destinatario). Para llevar a cabo estos procedimientos, cada usuario posee un par de claves pública-privada. La clave pública (conocida públicamente) posibilita que cualquier otro usuario le cifre la información confidencial que le desea hacer llegar; mientras que su clave privada (secreta y que solo él conoce) es la que le permite descifrar la información cifrada recibida.

Como ya se ha dicho, los esquemas de firma digital constan de dos partes: elaboración de la firma y verificación de la misma. Para presentar ambos protocolos, veamos cómo un usuario elabora su firma digital, de modo que supongamos que Eduardo es el usuario que lleva a cabo el proceso de cálculo de su firma para un mensaje no secreto m y que las claves pública y privada de Eduardo se denotan por E y e, respectivamente.

El protocolo de elaboración de firma que debe seguir Eduardo para firmar electrónicamente m es el siguiente:

1. Da a conocer o pone a disposición de quien pueda verificar su firma, su clave pública E y selecciona una función resumen considerada segura, sea tal función h.
2. Calcula el resumen del documento a firmar m mediante la función h, obteniendo:
$$h(m) = \underline{m}$$

3. Aplica la función de cifrado, C, con su clave privada, e, al resumen del mensaje, m, obteniendo su firma para dicho mensaje, s:
$$C_e(\underline{m}) = s$$
4. Hace público el mensaje cuya firma acaba de calcular, así como su correspondiente firma para el mismo: (m, s).

Este protocolo garantiza que Eduardo ha sido el firmante del mensaje dado que es el único que conoce su clave privada.

Si otro usuario, Verónica, quiere verificar la validez de la firma de Eduardo, s, para el mensaje m, sigue el protocolo de verificación de firma siguiente:

1. Calcula el resumen del documento, m, mediante la misma función resumen que utilizó Eduardo y obtiene
$$h(m) = \underline{m}$$
2. Descifra con la clave pública de Eduardo, E (que conoce por ser pública), la firma del mensaje, s, obteniendo
$$D_E(s) = D_E(C_e(\underline{m})) = m'$$
3. Finalmente, comprueba si los valores de $\underline{m}$ y de m' coinciden. En caso afirmativo, la firma queda verificada. En caso contrario, la firma para el mensaje se rechaza.

Este protocolo permite asegurar que el mensaje es íntegro, es decir, que no ha sido manipulado durante la transmisión porque en otro caso, el valor m' no coincidiría con el de $\underline{m}$.

Actualmente, las firmas electrónicas más extendidas se llevan a cabo bien mediante el criptosistema asimétrico RSA (Fúster Sabater *et al.*, 2012; Hernández Encinas, 2016), bien mediante los basados en curvas elípticas de criptosistema asimétrico RSA.

Dado que en el caso de las blockchains más utilizadas el sistema de firma empleado se basa en los criptosistemas asimétricos con curvas elípticas, en particular, en el algoritmo estándar de firma digital con curvas elípticas o ECDSA (Elliptic Curve Digital Signature Algorithm) haremos un comentario muy rápido sobre el mismo. Para más información, remitimos al lector a publicaciones especializadas sobre el tema (ANSI, 2005; Fúster Sabater *et al.*, 2012; Hernández Encinas, 2016; Gayoso Martínez *et al.*, 2018).

En el caso de las blockchains de Bitcoin y de Ethereum, la curva elíptica que se utiliza se conoce como la curva de Koblitz secp256k1, que está definida en el estándar conocido por Standards for Efficient Cryptography (SEC)[11] y cuya ecuación es la siguiente:

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

Esta curva tiene

1157920892373161954235709850086879078528375642790749043826051631415181614943 37

puntos, considerándose como cuerpo base el que tiene el siguiente número primo de elementos

1157920892373161954235709850086879078532699846656405640394575840079088346716 63

que es un número de 256 bits (tamaño de la clave).

Una cuestión que conviene resaltar es la respuesta a la pregunta: ¿Cómo genera un usuario su clave pública? Como hemos señalado anteriormente, en realidad lo que el usuario genera en primer lugar es la clave privada, *e*, y a partir de ella calcula la clave pública correspondiente, *E*. De forma más concreta y en el caso particular de las claves utilizadas con la curva secp256k1, el usuario genera de modo aleatorio una colección de 256 bits y, siguiendo el protocolo definido para las curvas elípticas, deriva la clave pública correspondiente, que, en esta situación tiene una longitud de 65 bytes, esto es, 33 bytes en formato comprimido. Además, se debe tener en cuenta que como la clave pública en este caso es un punto de la curva secp256k1, que tiene dos coordenadas, esta viene dada por un par de enteros (x, y). Cada uno de estos números se codifica mediante el formato *big endian* de enteros de 256 bits, y la clave se escribe mediante la concatenación de esos números y el prefijo 0x04. Como resultado, la clave se codificará con 65 bytes. En el formato comprimido solo se codifica la coordenada *x*, ya que la

11. Certicom Research. Véase http://www.secg.org/sec2-v2.pdf (último acceso 11/03/2019).

coordenada *y* solo puede tomar dos valores (uno par y otro impar, dado que la curva es simétrica respecto al eje de abscisas). La clave en este caso se escribe con los 32 bytes de la coordenada *x* más un prefijo de un byte (0x02 para el valor par de *y*; 0x03 para el valor impar). Así, la clave queda codificada con 33 bytes.

Conviene señalar que existen diferentes tipos de firmas, cada uno de los cuales tiene su correspondiente protocolo, que ha sido diseñado específicamente para resolver determinadas situaciones particulares no incluidas en el planteamiento general y estándar de una firma electrónica clásica (Arroyo *et al.*, 2015). A continuación, se presentarán brevemente algunas de las firmas más importantes que serán utilizadas en capítulos posteriores.

Los esquemas de firmas múltiples (o multifirmas) son protocolos de firma en los que un grupo de usuarios firma un único mensaje, de modo que la firma solo es válida si todos ellos participan, de forma mancomunada, en el protocolo (Itakura y Nakamura, 1983). El planteamiento y las aplicaciones de las firmas múltiples dependen del criptosistema que se utilice en el protocolo. En el caso de las blockchains, existen diversas versiones de multifirmas con distintos grados de complejidad, y cuya motivación principal es aumentar la seguridad, impidiendo que una de las partes involucradas en una transacción pueda engañar al resto.

Las "firmas grupales" (o en nombre de un grupo) son firmas en las que un único miembro de un determinado grupo firma, anónimamente, un mensaje en nombre de todos los miembros del grupo (Chaum y Van Heyst, 1991). Cuando se va a verificar la firma elaborada, el verificador comprueba que la firma es válida si queda probado que la firma fue hecha por uno de los miembros del grupo, aunque no se pueda determinar quién en concreto la elaboró. Por ejemplo, esta situación puede darse cuando un empleado de una gran compañía firma un documento y para un verificador es suficiente saber que el mensaje fue firmado por un empleado, sin necesidad de conocer de qué empleado se trata.

En las firmas grupales hay un miembro destacado del grupo de firmantes que es su administrador. Su función, además de la de incluir o excluir miembros del grupo, es la de revelar el miembro del grupo que firmó determinado documento en el caso de que

exista una disputa o controversia posterior al hecho de la firma (ante un juez, por ejemplo).

Las firmas "en anillo" o *ring signatures* (Rivest *et al.*, 2001) son firmas digitales que pueden ser elaboradas por cualquier miembro de un determinado grupo. En este tipo de firma cada miembro tiene sus propias claves y no es posible calcular, de modo computacionalmente eficiente, cuál de las claves de los miembros del grupo fue la que se empleó para generar una firma dada. Las firmas en anillo son similares a las firmas grupales, pero tienen dos diferencias. La primera de ellas es que no hay forma de revocar el anonimato de una firma individual, y la segunda es que no es preciso que el grupo de usuarios deba cumplir ninguna condición previa. El nombre de firma en anillo se debe a la estructura en forma de anillo del algoritmo que se emplea.

Rivest (2001) describe dos aplicaciones de las firmas en anillo. La primera de ellas permite filtrar un secreto. Así, una de estas firmas podría ser utilizada por un alto funcionario de la Casa Blanca para dar a conocer un secreto, sin revelar qué funcionario fue el que firmó la filtración. Las firmas en anillo se pueden usar en este caso porque no hay forma de revocar el anonimato de tal firma y, además, porque el grupo se puede improvisar en cualquier momento dado que no hay condiciones previas sobre el mismo. Una segunda aplicación se da como firma negable. En este caso, el remitente y el destinatario de un mensaje son los únicos miembros del grupo. De este modo, cuando el remitente firme un mensaje, la firma será válida para el destinatario, pero para cualquier otra persona no será posible saber quién es el remitente y quién el destinatario. Por lo tanto, tal firma es convincente, pero cualquiera de los dos miembros del grupo puede negar ser el autor de la misma y no habría forma de comprobarlo.

Protocolo de firma ECDSA y parámetros de la curva secp256k1*

En esta sección se presenta el protocolo estándar de firma con curvas elípticas, denominado ECDSA. No obstante, antes de tratar dicho protocolo, mencionaremos algunos de los protagonistas que intervienen en el mismo. Para ello consideraremos el caso

más sencillo, aunque las implementaciones puedan contener aspectos y herramientas algo más complejas.

En este protocolo se considera que los resúmenes de los mensajes son puntos de una curva elíptica definida sobre un cuerpo finito (Hernández Encinas, 2016). Podemos suponer, en aras de su simplicidad, que un cuerpo finito es un conjunto de p números, denotado por $\mathbf{F}_p$, siendo p un número primo, en el que se define una operación producto entre sus elementos, sean a y b, de modo que, una vez realizada la multiplicación ordinaria de los mismos, $a \cdot b$, se considera como resultado al resto de la división de esa multiplicación entre el número p, representándose este resultado como

$$a \cdot b \pmod{p}.$$

En los cuerpos finitos se pueden realizar las operaciones de suma, resta, producto y división (salvo por cero) de todos sus elementos.

Por otra parte, las curvas elípticas, en su versión más sencilla, son curvas planas definidas por una ecuación del tipo

$$y^2 = x^3 + a\,x + b,$$

de modo que, conocidos los valores de a y de b, las coordenadas de sus puntos, $P = (x, y)$, además de verificar la ecuación anterior, pertenecen a $\mathbf{F}_p$, esto es, $a, b, x, y \in \mathbf{F}_p$.

En este conjunto finito de puntos que constituye la curva elíptica C se define una operación de suma de puntos mediante determinadas operaciones con sus coordenadas. La suma de puntos da como resultado otro punto de la misma curva, y tiene las propiedades asociativa, conmutativa y de elemento neutro, O, de modo que $P + O = P$, para cualquier punto P de la curva. La suma de varias veces el mismo punto se representa por

$$P + \overset{z}{\cdots} + P = z \cdot P.$$

El número de puntos de una curva elíptica se conoce como su *orden*, denotado por n, y que por simplicidad supondremos que es

un número primo. En este caso, el conjunto de puntos de la curva se dice que es un grupo cíclico y tiene la propiedad de contener generadores. Un "generador" no es sino un punto destacado de la curva que verifica la propiedad de que los demás puntos de la curva se pueden escribir como un múltiplo suyo, es decir, dado un generador G de C y un punto cualquiera de la curva, P, existe un número entero t de modo que

$$P = t \cdot G.$$

Al conjunto de parámetros que define una curva elíptica se le denota por $T = (p, a, b, G, n)$.

Como ya hemos señalado, el ECDSA es un protocolo de firma estándar, por lo que el mismo consta de dos partes: elaboración y verificación de la firma. En el ECDSA se considera un conjunto de parámetros prefijados y conocidos. Este conjunto de parámetros es el siguiente: una función resumen, h, una curva elíptica, C, un cuerpo finito base, $\mathbf{F}_p$, un generador del grupo de puntos de la curva, G, y las claves pública y privada del firmante, Eduardo, sean E y e, respectivamente. La clave e es un número entero y la clave pública es el punto de la curva dado por $E = e \cdot G$.

El protocolo que sigue Eduardo para elaborar su firma para el mensaje m, es el siguiente:

1. Calcula el resumen del mensaje:
$$H(m) = m.$$
2. Genera una clave de sesión (o efímera), esto es, un número aleatorio k con $1 \leq k \leq p - 2$, primo con $p - 1$, es decir, verificando $\text{mcd}(k, p - 1) = 1$.
3. Calcula los siguientes dos valores:
$$k \cdot G = (x_1, y_1), \qquad r = x_1 \pmod{p}.$$
Si resultara que $r = 0$, se elegiría otro k.
4. Calcula el inverso de k, es decir, un número que multiplicado por k da como resultado el 1, y que se denota por $k^{-1} \pmod{p}$. Luego determina el valor de
$$S = k^{-1}(m + b \cdot r) \pmod{p}.$$
5. La firma de Eduardo para el mensaje m es el par (r, s).

El protocolo de verificación de la firma que sigue un verificador es como sigue:

1. Calcula el resumen del mensaje:
$$h(m) = m.$$
2. Comprueba que los enteros r y s que constituyen la firma están en el intervalo dado por $[1, p - 1]$.
3. Determina el siguiente valor:
$$w = s^{-1} \pmod{p}.$$
4. Calcula
$$u_1 = m \cdot w \pmod{p}, \quad u_2 = r \cdot w \pmod{p}.$$
5. Determina
$$u_1 \cdot G + u_2 \cdot E = (x_0, y_0), \quad v = x_0 \pmod{p}.$$
6. La firma se considera verificada y válida si y solo si se cumple que
$$v = r.$$

La prueba de que el proceso de verificación es correcto se sencilla, puesto que si se denota por $(P)_x$ la primera coordenada del punto $P \in C$, se tiene lo siguiente:

$$\begin{aligned} v = x_0 \pmod{p} &= (u_1 \cdot G + u_2 \cdot E)_x = (m \cdot w \cdot G + r \cdot w \cdot E)_x \\ &= (m \cdot w \cdot G + r \cdot w \cdot e \cdot G)_x = ((m + r \cdot e) w \cdot G)_x = (k \cdot G)_x \\ &= x_1 \pmod{q} = r. \end{aligned}$$

Por otra parte, los parámetros de la curva que suele considerarse en las principales blockchains es secp256k1, cuyos parámetros $T = (p, a, b, G, n)$ son los siguientes:

p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
$= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
= 1157920892 3731619542 3570985008 6879078532
6998466564 0564039457 5840079088 34671663,
a = 0x00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000,
b = 0x00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000007,

G = 0x0479BE667EF9DCBBAC55A06295CE870B07029B-
FCDB 2DCE28D9 59F2815B 16F81798 483ADA77
26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419
9C47D08F FB10D4B8,
n = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE
BAAEDCE6 AF48A03B BFD25E8C D0364141
= 1157920892 3731619542 3570985008 6879078528
3756427907 4904382605 1631415181 61494337,

en definitiva, se trata de la curva de ecuación

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

Finalmente, una cuestión que conviene resaltar es la respuesta a la pregunta: ¿cómo genera un usuario su clave pública? Como hemos señalado anteriormente, en realidad lo que el usuario genera en primer lugar es la clave privada, e, y a partir de ella calcula la clave pública correspondiente, E. De forma más concreta y en el caso particular de las claves utilizadas con la curva secp256k1, el usuario genera de modo aleatorio una colección de 256 bits y, siguiendo el protocolo definido para las curvas elípticas, deriva la clave pública correspondiente, que, en esta situación tiene una longitud de 65 bytes, esto es, 33 bytes en formato comprimido. Además, se debe tener en cuenta que como la clave pública en este caso es un punto de la curva secp256k1, que tiene dos coordenadas, esta viene dada por un par de enteros (x, y). Cada uno de estos números se codifica mediante el formato *big endian* de enteros de 256 bits, y la clave se escribe mediante la concatenación de esos números y el prefijo 0x04. Como resultado, la clave se codificará con 65 bytes. En el formato comprimido solo se codifica la coordenada x, ya que la coordenada y solo puede tomar dos valores (uno par y otro impar, dado que la curva es simétrica respecto al eje de abscisas). La clave en este caso se escribe con los 32 bytes de la coordenada x más un prefijo de un byte (0x02 para el valor par de y; 0x03 para el valor impar). Así, la clave queda codificada con 33 bytes.

Punteros hash y árboles de Merkle

Tal y como su nombre indica, las blockchains son cadenas de bloques, esto es, sistemas de almacenamiento de información que se divide en bloques de datos debidamente enlazados. Pues bien, ese enlazamiento se realiza mediante funciones hash. En efecto, tal y como hemos visto al introducir el concepto de función resumen, un bloque de datos está asociado a un cierto valor de la función resumen, de forma que se puede crear un índice de datos sin más que crear una lista que incluya los valores de la función resumen para cada bloque de datos. Si, por otro lado, hacemos que cada bloque de datos contenga el resumen del bloque anteriormente añadido al sistema, entonces tendremos una lista enlazada (mediante punteros resumen) de bloques de datos. Al margen de las especificidades de cada implementación, este mecanismo de indexación de información es el fundamento general de agregación de bloques en las blockchains.

Por otro lado, la búsqueda de información en listas enlazadas mediante hashes no es una solución óptima, especialmente cuando se cuenta con un número muy elevado de bloques de información. Además, se debe tener en cuenta que los bloques de las blockchains contienen, a su vez, una elevada información sobre transacciones y/o registros, de forma que añadir un resumen por cada uno de esos datos supondría una reducción de la capacidad efectiva de almacenamiento de la blockchain. En consecuencia, es necesario buscar algún tipo de criterio para la gestión de hashes (en concreto, de punteros hash) que haga eficiente tanto el almacenamiento como la búsqueda de datos. Es en este punto donde entran en juego los árboles de Merkle.

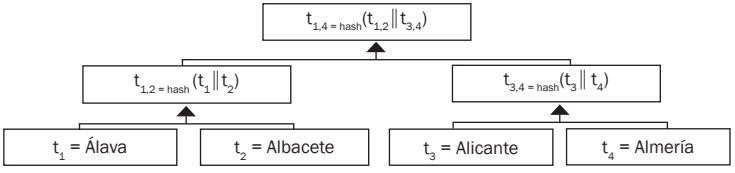
Un "árbol de Merkle" es una estructura de datos, propuesta por Ralph Merkle en 1979 (Merkle, 1982), que combina árboles binarios y funciones resumen. Un árbol binario es un grafo en forma de árbol (es decir, que no contiene ciclos) donde cada nodo tiene como máximo dos nodos hijos. En concreto, estos grafos se pueden dibujar de tal manera que haya un único nodo del que cuelgan todos los demás. Este nodo se conoce como "nodo raíz". Igualmente, los nodos que no tienen ningún hijo se conocen como "nodos hoja".

Usualmente, a cada nodo del árbol se le asigna un valor, y es aquí donde entra la peculiaridad de los árboles de Merkle: en lugar de asignar valores arbitrarios a todos los nodos, esto solo se hace para los nodos hoja. Una vez los nodos hoja tienen un valor asignado, el valor de cada nodo intermedio (desde los padres de los nodos hoja hasta el nodo raíz) se calcula combinando mediante una función hash los valores de sus nodos hijos.

Por ejemplo, en la figura 1 se almacenan los nombres de cuatro provincias españolas en un árbol de Merkle. Los nombres, etiquetados con t_1 hasta t_4 para facilitar la explicación, se asignan como valor a los nodos hoja. A continuación, el valor del nodo padre de las hojas t_1 y t_2 se calcula como $\text{hash}(t_1 \| t_2)$, donde el operador "$\|$" denota la concatenación. Lo mismo se hace para las hojas t_3 y t_4. Llamamos al resultado de ambas operaciones $t_{1,2}$ y $t_{3,4}$, respectivamente. Este mismo proceso se repite, ahora con $t_{1,2}$ y $t_{3,4}$, para dar lugar al hash etiquetado como $t_{1,4}$, que concentra todos los hashes anteriores.

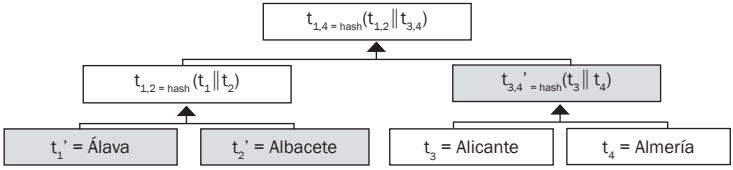
Después de llevar a cabo este proceso, quien almacene el valor $t_{1,4}$ obtenido para el nodo raíz, podrá comprobar qué valores se incluyeron en el árbol representado por dicha raíz. Esto es así debido a una de las propiedades de las funciones resumen (dependencia de bits): basta con que se modifique un único bit de los que se han utilizado para llegar al valor $t_{1,4}$ (esto es, cualquiera de los bits en los nodos t_1, t_2, t_3 o t_4 o en cualquiera de los nodos intermedios, $t_{1,2}$ o $t_{3,4}$) para que el valor obtenido para el nodo raíz del nuevo árbol será distinto al valor $t_{1,4}$ (lo que demuestra que se usaron valores distintos para calcularlo).

Figura 1
Ejemplo de árbol de Merkle



Más aún, si alguien quiere demostrar que un valor fue incluido en el árbol, no es necesario desvelar todos los valores del árbol, sino una cantidad bastante inferior que depende del logaritmo del número de nodos hoja que tiene dicho árbol. Siguiendo el ejemplo anterior, si alguien quisiera demostrar que el valor "Albacete" fue incluido en el árbol de Merkle con raíz $t_{1,4}$, simplemente tendría que transmitir los valores "Albacete", "Álava" y $t_{3,4}$ (figura 2). Con ellos, cualquiera podría verificar que el cálculo de los hashes concatenados da lugar al valor de hash almacenado en el nodo raíz, es decir, $t_{1,4}$.

Figura 2
Verificación de un valor incluido en un árbol de Merkle.



Volviendo al contexto de las blockchains, veremos en el capítulo 2 qué estructuras de este tipo permiten gestionar la integridad de la información sin más que incluir en los bloques el valor del nodo raíz del árbol de Merkle asociado a dicha información, en lugar de añadir el valor de todos los datos que son protegidos mediante los bloques. Esto proporciona importantes ventajas, como reducir la cantidad de información que se escribe de forma explícita en la cabecera de un bloque, o la implementación de lo que se conoce como clientes ligeros, que serán tratados más adelante.

CAPÍTULO 2

Desde la criptomoneda a la blockchain: introducción de la tecnología Bitcoin

Una vez se ha introducido la base criptográfica de blockchain, y tras una contextualización histórica, en este capítulo nos centraremos en la primera propuesta de blockchain: la blockchain de Bitcoin. A lo largo del capítulo subrayaremos los principales elementos criptográficos que permiten emular el comportamiento de la moneda física mediante tecnología digital. En concreto, explicaremos los principios criptográficos, los protocolos de red y el modelo de colaboración que hacen posible que una criptomoneda como bitcoin haga viable el intercambio de valor financiero con los mismos criterios con los que operamos al usar el dinero en metálico, esto es, los billetes y monedas de curso legal. Así, haremos hincapié en que el éxito de Bitcoin no se explica solo con su base criptográfica, sino que es necesario que en la red Bitcoin exista todo un conjunto de usuarios que pongan sus ordenadores y dispositivos de cómputo al servicio del ecosistema, actuando como nodos activos en la generación de criptomonedas. Por último, la sostenibilidad del sistema también precisa de la presencia de usuarios finales que instalen el software cliente de Bitcoin y realicen transacciones mediante la criptomoneda de la plataforma, esto es, el bitcoin, cuyo símbolo, por analogía con otras monedas es ₿ o **Ƀ**.

Orígenes de Bitcoin

El surgimiento de una tecnología rara vez corresponde a un único motivo o a una sola causa. La relación existente entre los intereses puramente técnicos y los objetivos económicos, sociales y políticos es algo que hemos venido experimentando a partir de la Segunda Guerra Mundial. Como señaló en su momento Bertrand Russell, y como recuerda Phillip Rogaway (Rogaway, 2015), el hecho científico está siempre anclado en un momento histórico. Cada contexto socio-político responde a una urdimbre económica, a un modelo que determina todo tipo de preferencias y de opciones, también las de proyectos científicos y técnicos. Aunque tendemos a pensar que las denominadas ciencias puras están exentas de toda intervención y determinación proveniente desde fuera de su dominio específico, lo cierto es que tanto las matemáticas como la física también están subordinadas a las líneas generales de la política científica de un país.

No obstante, el colectivo científico no es monolítico y, de hecho, el carácter indagador y hasta escéptico (en el sentido clásico de *skeptikós*, o 'el que examina') de la labor científica inevitablemente lleva a la disensión y al cuestionamiento. No es difícil encontrar ejemplos de "disidentes" en las diversas disciplinas de la tecnociencia, sin que ello tenga que comportar una necesaria adscripción o intención política.

En el caso de la comunidad criptográfica, esta dinámica se ha hecho presente a través de diversos movimientos y organizaciones como los *cypherpunk* (Rogaway, 2015). Este colectivo surgió a finales de 1980 con el fin de aplicar los principios de la criptografía al ámbito político. Herramientas como el gestor de cifrado PGP (Pretty Good Privacy), la red Tor (The onion router), la información de Wikileaks o la criptomoneda bitcoin son consecuencia de la acción política de este grupo de tecnólogos que extienden las contribuciones académicas en criptografía de modo que proporcionan implementaciones prácticas y casos de éxito reales. Son muchas las críticas que han recibido por parte del mundo académico las diversas iniciativas de este colectivo y casi todas ellas subrayan la falta de precisión y de formalismo de muchas de sus propuestas. Sin negar que en muchos

casos sería necesario un mayor desarrollo teórico, es de destacar la capacidad de los *cypherpunk* a la hora de proponer soluciones de fácil uso para cifrar información, para navegar por Internet protegiendo la identidad del usuario en la red, para acceder a información de interés público que no es públicamente accesible o para contar con sistemas de intercambio financiero prescindiendo de intermediarios.

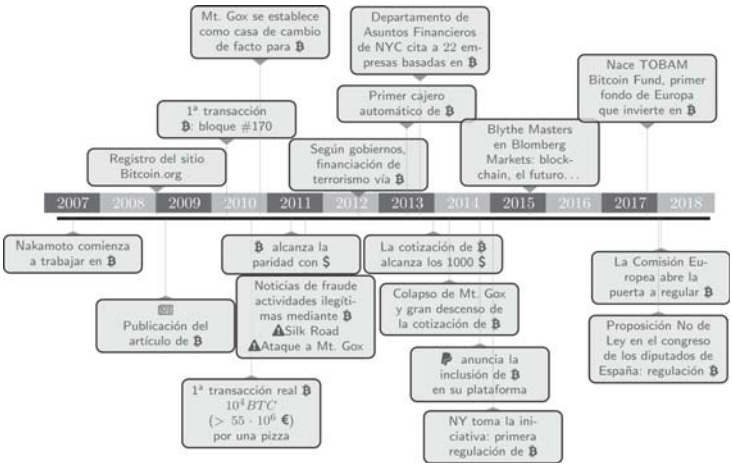
De hecho, los *cypherpunks* son muy responsables de que en estos momentos la criptografía sea una de las grandes protagonistas de la actualidad tecnológica (siempre lo ha sido aunque no hayamos sido conscientes). Y lo es fundamentalmente como resultado de toda la actividad e interés que ha despertado el fenómeno Bitcoin y el mundo de la criptoeconomía. Tal y como se explicó en la introducción, la criptomoneda bitcoin fue creada de forma anónima por una o varias personas bajo el seudónimo de Satoshi Nakamoto. Bitcoin es mucho más que un mecanismo de intercambio de valor financiero, ya que genera todo un ecosistema de actividad económica que se conoce bajo el nombre de criptoeconomía.

Desde la aparición del artículo de Satoshi Nakamoto (2008), el número de noticias sobre Bitcoin y la criptoeconomía ha ido creciendo. Este concepto de nuevo cuño, la criptoeconomía, se emplea para mencionar no solo la actividad financiera basada en criptomonedas, sino también para aludir a toda nueva propuesta que haga uso de la blockchain de alguna criptomoneda. Es más, en los foros de discusión sobre criptoeconomía se habla del mundo cripto o de expertos cripto para referir a toda la actividad adscrita a la criptoeconomía y a los expertos en esta nueva forma de economía digital. Con la criptoeconomía ocurre algo parecido a lo que tuvo lugar en el caso de los *cypherpunks*. En ambos casos, y al menos desde la perspectiva del gran público, la esfera académica pierde protagonismo. Dicho de otra forma (y en la línea de lo que subraya Phillip Rogaway), el ciudadano medio percibe que los *cypherpunks* y la criptoeconomía han alcanzado grandes resultados al emplear la criptografía como elemento transformador de las relaciones de poder. Ese éxito ha eclipsado el hecho de que los fundamentos teóricos de aquellos logros vienen del ámbito académico (Narayanan y Clark, 2017),

de forma que existe una tendencia a identificar la criptografía con sus aplicaciones.

Al margen de la cuestión de la autoría y de la más o menos fortuna de la sinécdoque, hay que admitir que la criptoeconomía ha venido a configurar la agenda de la investigación y desarrollo de los sistemas de información y de comunicación. Así, desde la publicación del artículo seminal de Nakamoto (2008), las reacciones al fenómeno Bitcoin han sido muy variadas (figura 3). Sobre todo, es significativo cómo ha ido variando la percepción de la tecnología por parte de los organismos e instituciones reguladoras de los distintos países.

Figura 3
Evolución temporal de Bitcoin.



En concreto se pueden diferenciar tres etapas en la línea temporal de la figura 3. Desde 2007 hasta mediados de 2011 se puede considerar que Bitcoin formaba parte de un grupo muy reducido de usuarios, de forma que los Estados y organizaciones no mostraban ni curiosidad ni recelo sobre la actividad que este grupo de programadores y usuarios avanzados de la tecnología llevaba a cabo con Bitcoin. A partir de finales de 2011 se empezaron a registrar evidencias de que Bitcoin no solo era cosa de friquis. En efecto, el anonimato que proporciona Bitcoin a la hora de

realizar transferencias era visto por terroristas y otros delincuentes como una gran oportunidad para eludir las medidas de control de los países y de sus fuerzas de seguridad. Además, los diversos bancos centrales de los distintos países advierten de la dificultad para imponer las medidas de lucha contra el blanqueo de dinero en el ecosistema Bitcoin. Este rechazo institucional y la mala fama de Bitcoin se extienden hasta 2015. En esa fecha, representantes de alto rango del mundo financiero ven que la tecnología base de Bitcoin —blockchain— tiene un gran potencial como vehículo de nuevos modelos de negocio y de actividad financiera. A partir de este momento se dispara el entusiasmo con blockchain y con la denominada criptoeconomía, entusiasmo que se inicia en Estados Unidos pero que recientemente también ha acabado contagiando al continente europeo.

Si en su inicio Bitcoin tuvo éxito, fundamentalmente como consecuencia de la quiebra de la confianza en los bancos tradicionales a comienzos de la crisis económica del 2008, posteriormente el crecimiento del número de usuarios dio lugar a un conjunto de iniciativas empresariales que empezaron a utilizar Bitcoin como soporte de su modelo negocio. En este estadio es especialmente relevante el caso de BitPesa en Kenia.

BitPesa es una empresa fundada en 2013 que trata de extender el éxito de la plataforma M-Pesa (acrónimo de móvil en inglés —Mobile— y dinero en swahili —Pesa—) como vehículo de inclusión financiera en países con una alta tasa de población sin posibilidad de tener acceso a una cuenta bancaria (Campbell-Verduyn, 2017). En el caso de M-Pesa, la innovación vino determinada por el uso de los dispositivos móviles como herramienta para realizar transacciones de pequeñas cantidades de dinero. Aunque M-Pesa es una plataforma creada por Vodafone para Safaricom (el principal proveedor de servicios de telecomunicaciones de Kenia), su éxito, desde su aparición en 2007, se basó en que todos los intercambios de remesas se realizaban de modo local. Así, todos los usuarios de M-Pesa utilizaban su móvil para enviar pequeñas cantidades de dinero a su círculo de confianza. Dicho de otro modo, la plataforma móvil de Safaricom hizo posible trasladar al dominio digital el modelo de reciprocidad sobre el que descansaba la sociedad

keniata. El alto grado de aceptabilidad de M-Pesa estuvo en gran medida deparado por la popularización de la telefonía móvil, específicamente de dispositivos móviles de bajo coste. Sin necesidad de contar con una red costosa de oficinas bancarias, la propuesta de Vodafone hizo viable un modelo de inclusión financiera que tampoco precisaba el uso de dispositivos móviles inteligentes. El envío de SMS (*short message service*) se convirtió en el canal financiero para transferir dinero a través de los operadores de telefonía. Es más, la recarga de tarjetas de prepago extendió el modelo original potenciando aún más, si cabe, el tráfico de micropagos a través de la telefonía móvil.

El carácter ubicuo del pago electrónico mediante móviles en M-Pesa estaba acotado geográficamente, ya que las características del modelo social sobre el que operaba estaban restringidas al ámbito de las redes de contactos familiares y de amistad. Consciente de las virtudes de M-Pesa y del alto coste del envío de remesas transnacionales, Elizabeth Rossiello, en 2013, fundó BitPesa con objeto de llevar el modelo M-Pesa más allá del ámbito nacional, haciendo para ello uso de Bitcoin. La propuesta fue bien recibida por la comunidad y rápidamente alcanzó un volumen significativo del total de transferencias transnacionales de África. Ahora bien, en 2015 el Banco Central de Kenia (CBK) declaró que la actividad de negocio de BitPesa no se amoldaba a las características establecidas por la legislación del sector de las transferencias y del envío de remesas. La consecuencia directa de esta declaración fue la interrupción por parte de Safaricom de cualquier colaboración entre M-Pesa y organizaciones basadas en tecnología Bitcoin[12].

Registro de transacciones en Bitcoin: ventajas e inconvenientes del seudoanonimato en el ámbito financiero

La consideración de Bitcoin como un ecosistema de riesgo no es exclusiva de Kenia. En términos generales, existe un recelo

12. Véase http://disrupt-africa.com/2017/09/how-kenyas-central-bank-is-holding-back-digital-currencies/ (último acceso 11/03/2019).

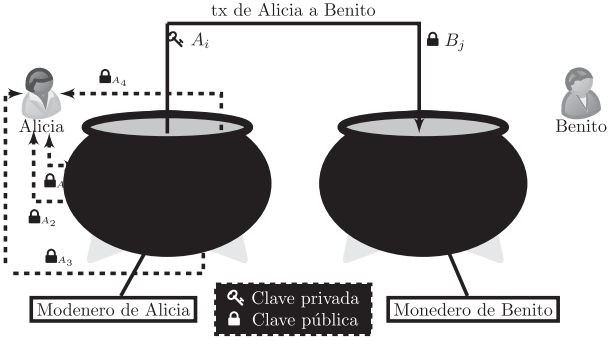
respecto a la forma en la que Bitcoin registra la cesión de capital (esto es, unidades de la criptomoneda bitcoin) desde un emisor a un cierto receptor. Tal apunte no se ajusta al paradigma clásico, en la medida que los usuarios involucrados (emisor y receptor) son designados a través de un seudónimo.

Tal y como referíamos en el capítulo 1, la firma digital es la primitiva criptográfica con la cual se gestiona la autoría de cualquier información. En el contexto de Bitcoin esta información viene dada por una transacción desde un emisor hacia un destinatario. La firma digital permitirá probar que el emisor es el propietario de una cierta cantidad de dinero que es transferida a un destinatario. De forma más precisa, este destinatario viene dado por una clave pública a la cual estará asociada la cantidad de dinero consignada en la transacción. Una vez completada la transacción, el destinatario podrá reutilizar el dinero recibido en una nueva transacción, para lo cual simplemente tendrá que probar que es el propietario haciendo uso de su clave privada.

En la figura 4 aparecen dos usuarios de Bitcoin, Alicia y Benito, cada uno de los cuales posee un monedero Bitcoin que incluye varios pares de claves públicas (representadas mediante un candado) y privadas (identificadas a través de una llave). Si Alicia desea enviar dinero a Benito mediante Bitcoin, solo necesita conocer una clave pública de Benito y hacer uso de una de sus claves privadas. De modo más detallado, en la figura 4 aparecen representados los monederos de Alicia y de Benito. Alicia tiene en su monedero cuatro pares A_i de claves, $1 \leq i \leq 4$, cada una de ellas asociada a una cantidad concreta de bitcoins. A la hora de realizar una transacción (denotada por trx) Alicia empleará los bitcoins asociados a uno o varios de los pares A_i. En el ejemplo de la figura 4 Alicia solo emplea un par de claves y, en concreto, hace uso de la clave privada incluida en A_i para probar que es la propietaria de los bitcoins correspondientes. Una vez que Alicia ha probado que es la dueña de los bitcoins a utilizar en trx, podrá enviar dinero a Benito. Para ello establece como destino de trx una dirección asociada a Benito, esto es, una clave pública incluida en uno de los pares B_j que están dentro del monedero de Benito.

Figura 4

Transacción entre dos usuarios de Bitcoin. Los monederos de Alicia y Benito contienen pares de claves, y cada uno de estos pares tiene asociada una cantidad de bitcoins.



La trazabilidad de cada uno de los usuarios de la red Bitcoin, por tanto, vendrá dada como la capacidad de establecer las operaciones realizadas por una clave pública y de vincular esta clave con una identidad física. Ahora bien, cada usuario de la red Bitcoin puede generar tantas claves públicas (con sus correspondientes claves privadas) como desee. Todas y cada una de esas claves están almacenadas en el monedero electrónico de un usuario, y no hay forma de saber qué usuario está detrás de una clave pública, a menos que ese usuario informe al respecto de su identidad.

En Bitcoin, por tanto, las direcciones desde la que se recibe dinero serán las claves públicas de los distintos usuarios. Cada una de esas claves públicas tiene asociada su correspondiente clave privada, de forma que aunque se conozca la clave pública no será posible determinar la clave privada asociada (la relación existente entre ambos tipos de claves está protegida por la imposibilidad computacional de resolver un problema matemático considerado muy difícil). Así, solo el poseedor de dicha clave privada podrá utilizarla para firmar información, la cual se constituye en prueba de autoría o de propiedad (capítulo 1). En el caso concreto de Bitcoin, esta propiedad determina el origen y destino de las transacciones, con lo que el dinero vinculado a una dirección se calculará como el balance contable entre las

transacciones que tienen por origen o destino dicha dirección. El dinero total que posee un usuario se calculará, pues, como la suma de todos los balances contables de las claves públicas que posee dicho usuario.

Como hemos venido remarcando, las credenciales de usuario (su clave pública y privada) en Bitcoin tienen por principal cometido la atribución de transacciones de modo que la identidad de los intervinientes en las mismas quede salvaguardada, a la vez que se garantiza la corrección y validez en el intercambio de valores entre las partes. En Bitcoin existen tres tipos de transacciones:

- P2PKH (*Pay-to-Public-Key-Hash*). Es el tipo estándar de transacción en Bitcoin y define la cesión de valores de criptomoneda de una dirección a otra.
- Multifirma (*multisignature*). En este tipo de transacción se impone que la operación deba ser validada por más de una clave privada. Esta restricción permite definir marcos colaborativos en los que una transacción precisa el acuerdo entre *n* partes. A la hora de definir una transacción hay que establecer la dirección de destino. Todas las direcciones vinculadas a transacciones normales se escriben con el prefijo 1, mientras que las direcciones de las transacciones de firma múltiple tienen por prefijo un 3.
- P2SH (*Pay-to-Script-Hash*). Se trata de un tipo de transacción en el que es necesario que se cumpla una serie de condiciones antes de que se lleve a término. Las transacciones de firma múltiple (capítulo 1) son un tipo especial de transacción P2SH. Las direcciones vinculadas a estas transacciones se escriben también con el prefijo 3, y las condiciones se definen a través de un *script*[13].

En el caso de Bitcoin, todas las operaciones vinculadas con la realización de transacciones van a estar escritas en los bloques de datos en los que se registra toda la actividad de la red.

13. En programación, un *script* es un fichero que contiene instrucciones que son interpretadas y ejecutadas por una máquina.

Las instrucciones que aparecen en los *scripts* están escritas de acuerdo con un lenguaje de programación muy simple: Script, que es una versión reducida del lenguaje Forth. Este lenguaje contiene un total de 256 posibles operaciones o instrucciones, y está orientado a ejecutarse a través de una pila[14].

Hay que tener en cuenta que una transacción en Bitcoin no es simplemente el apunte de un intercambio único entre dos usuarios. De hecho, la estructura de datos empleada para registrar las transacciones entre usuarios agrupa un conjunto de operaciones de entrada y de salida, de modo que cada operación de entrada hace referencia a una operación de salida contenida en un bloque previo. Para que esta referencia sea correcta, es necesario que la firma aportada en la operación de entrada del bloque actual se pueda verificar con la clave pública que figura en la correspondiente operación de salida.

Criptografía y protocolo de firma ECDSA en Bitcoin*

De forma más concreta, Bitcoin utiliza criptografía de clave pública mediante curvas elípticas (véase capítulo 1). Recordemos que la curva elíptica que se emplea con el protocolo ECDSA es la curva secp256k1. Los seudónimos de los usuarios se determinan a partir de sus claves públicas, codificándolas mediante el uso de dos funciones resumen: SHA-256 y RIPEMD-160, y una codificación en Base58Check[15], de acuerdo con el siguiente esquema:

1. Se calcula el resumen SHA-256 de la clave pública del destinatario de la transacción.
2. Al resultado de la operación anterior se le aplica la función resumen RIPEMD-160.
3. Se añade un prefijo a la salida del paso anterior. En Bitcoin se puede trabajar bien con la red principal, bien con una red de pruebas. Las direcciones de la red principal tienen

14. En programación, una pila es una estructura de datos en la que se introduce información (PUSH) y se extrae en orden inverso a su introducción. Es decir, una pila es un almacén LIFO (*Last In First Out*) de datos.
15. Véase https://es.bitcoin.it/wiki/Codificaci%C3%B3n_Base58Check (último acceso 11/03/2019).

el prefijo hexadecimal 0x00, mientras que las direcciones de la red de pruebas se notan mediante el prefijo 0xEF.
4. Una vez se ha añadido el adecuado prefijo, el bloque de bytes es resumido de nuevo mediante SHA-256.
5. Se seleccionan los primeros cuatro bytes de la salida obtenida en el punto 4. Esos cuatro bytes se utilizan como suma de comprobación o *checksum* (función CRC, véase capítulo 1).
6. Se añaden los cuatro bytes anteriores como sufijo de la salida obtenida en el paso 3.
7. Se codifica el resultado de la operación anterior utilizando formato Base58Check, que incluye como sufijo los cuatro bytes de verificación del punto 5.

Un ejemplo de seudónimo en Bitcoin podría ser

1MFjFmRhCYHt6GjPzXTEFuwSyuZTVAjYWX.

Es muy importante destacar que cada cliente, cada usuario de Bitcoin, genera sus claves de forma unilateral, sin que exista ninguna entidad que supervise esta generación de claves. Es cierto que existe la posibilidad de que se produzca una colisión en la generación de claves, es decir, es plausible que al generar aleatoriamente y de modo independiente varios pares de claves, estos sean idénticos, pero la probabilidad de que ello ocurra es tan baja que puede considerarse despreciable.

Transacciones mediante *scripts* en Bitcoin*

Las operaciones basadas en la ejecución de instrucciones a través de una pila se suelen escribir en "notación polaca inversa", que simplemente escribe los operadores después de los operandos. A modo de ejemplo, se tiene que:

- 3+4 mediante notación inversa se escribe 34+
- 2*3+4 se representa como 234+*

Las instrucciones del lenguaje Script empleado en Bitcoin siempre empezarán por el prefijo `OP_`. En el caso de la

operación de suma anterior, 34+, la instrucción quedaría codificada como

```
OP_3 OP_4 OP_ADD
```

Si quiere hacer una verificación del resultado se hace uso de la operación OP_EQUAL. Así, si hacemos

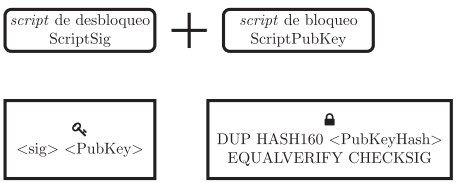
```
OP_3 OP_4 OP_ADD OP_5 OP_EQUAL
```

estaremos comparando el resultado de la operación suma de 3+4 con el valor 5; en este caso, el sistema devuelve FALSE.

Las instrucciones que hemos utilizado en los ejemplos anteriores son lo que denominan *opcodes*, es decir, códigos de operación[16]. A continuación, vamos a describir qué *opcodes* se aplican en el registro de una transacción en Bitcoin en el caso más simple: trasvase de dinero (dado como una porción de bitcoins) desde Alicia a Benito.

Figura 5

Composición de los scripts scriptSig y scriptPubKey en la creación de P2PKH.



Cuando Alicia crea una transacción, su cliente Bitcoin bloquea la transacción en cuestión mediante un *script*. En Bitcoin, cada transacción contiene un *script* que estipula las condiciones que han de satisfacerse para poder reclamar la salida de la

16. El conjunto total de *opcodes* que pueden emplearse en los *scripts* de bitcoin está definido en https://en.bitcoin.it/wiki/Script (último acceso 11/03/2019).

transacción, esto es, el dinero que pasa de Alicia a Benito. La concreción de estas restricciones puede ser más compleja, pero en el caso que nos ocupa simplemente se exigirá que Benito pueda probar que tiene la clave privada correspondiente a la clave pública del destinatario de la transacción creada por Alicia. *De facto*, hasta que Benito no muestra su clave privada, la transacción está bloqueada y su desbloqueo efectúa el flujo de operaciones definido por un *script* de tipo P2PKH (figura 5).

Este *script* en realidad es la combinación de dos *scripts*, uno de tipo *scriptPubKey* y otro de clase *scriptSig* (figura 5), y lleva asociada la ejecución de siete pasos a través de la pila:

1. En primer lugar, se introduce en la pila la firma del receptor de la transacción (es decir, de Benito) y su clave pública. El estado de la pila tras ello será
   <pubKey>
   <sig>
2. A continuación, se efectúa OP_DUP
   <pubKey>
   <pubKey>
   <sig>
3. La red Bitcoin no tiene acceso directo a la clave pública de ningún usuario. Aquí hay que recordar que solo son públicas las direcciones de los usuarios, y estas se obtienen mediante un doble resumen a partir de la clave pública. Esta operación de conversión de la clave pública se lleva término con OP_HASH_160:
   <pubKeyHash>
   <pubKey>
   <sig>
4. Se introduce en la pila la dirección de Benito
   <pubKeyHash>
   <pubKeyHash>
   <pubKey>
   <sig>
5. Se ejecuta la operación OP_EQUALVERIFY que se encarga de verificar si los dos últimos valores que hay en la pila

son iguales (se hacen dos extracciones de datos desde la pila y una comparación). Si no lo son, se interrumpe la ejecución

<pubKey>
<sig>

6. En caso de que se verifique de modo satisfactorio la igualdad, se comprueba la firma con OP_CHECKSIG

TRUE

7. Si la comprobación es correcta, se introduce en la pila el valor TRUE (FALSE en caso contrario).

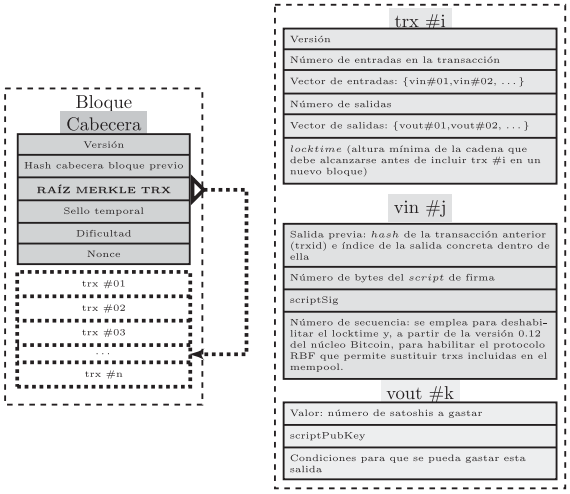
Como se puede observar en la explicación del proceso anterior, la transmisión de criptomoneda desde Alicia a Benito no precisa en ningún momento la inclusión de dato alguno sobre Alicia en los *scripts*. Esto está motivado por el modo en el que se atribuye la propiedad de moneda en Bitcoin según la concatenación de bloques de datos.

En la parte izquierda de la figura 6 podemos encontrar la estructura de los bloques de datos en Bitcoin. Cada bloque de datos contiene una cabecera y un conjunto de transacciones. Además de la versión del sistema Bitcoin, en la cabecera aparece el hash del bloque anterior, la raíz del árbol de Merkle que contiene todas las transacciones del bloque, el sello temporal con la fecha de creación del bloque y dos parámetros (la dificultad y el *nonce*) que jugarán un papel muy importante en el proceso de minado que explicaremos a continuación.

Como ya se mencionó en el capítulo 1, la función de los punteros hash en Bitcoin es garantizar la integridad referencial entre bloques mediante la construcción de una lista enlazada, mientras que los árboles de Merkle en Bitcoin se utilizan para optimizar el acceso a las transacciones incluidas en un bloque de modo que la raíz de un árbol de Merkle sirve de resumen de todas sus hojas, así como de verificador de su integridad. Esas hojas, por otro lado, son transacciones. En Bitcoin una transacción trx viene dada por un conjunto de entradas y un conjunto de salidas.

FIGURA 6

Estructura de un bloque de datos en Bitcoin.



En el esquema de la derecha de la figura 6 podemos ver que la *i*-ésima transacción (trx #i) consta de dos vectores: el vector vin de entradas y el vector vout de salidas. Además, contiene un campo para la versión del protocolo y otro para el valor *locktime,* que es un parámetro opcional que permite controlar cuándo debe incorporarse la transacción a un nuevo bloque y cuya utilidad se analizará en el capítulo 5. Cada uno de los elementos de vin hace referencia a un elemento de una salida contenida en una transacción de un bloque previo. Para un elemento dado de vin, la transacción referida se recupera buscando el valor trxid de su primer campo en el árbol de Merkle de las transacciones de cada uno de los bloques previos. Como podemos ver en la figura 6, cada elemento de vin y de vout contiene un *script* de desbloqueo (*scriptSig*) y de bloqueo (*scriptPubKey*), correspondientemente. Cuando recuperamos la salida de la transacción trxid en un elemento de vin, pues, estamos realizando la composición entre los dos *scripts* de acuerdo con el esquema de la figura 5.

En segundo lugar, la suma de todas las cantidades incluidas en las operaciones de entrada de un bloque (vin) tiene que ser menor o igual que la suma de todas las operaciones de salida (vout). Como resultado de todo ello:

- Un usuario, más bien la dirección que un usuario posee, tiene la capacidad de usar el conjunto de transacciones recibidas y que no han sido utilizadas. A estas transacciones se las nota con el acrónimo UTXO (*Unspent Transaction Output*), y constituyen el soporte de la Bitcoin.
- Un usuario no tiene forma de saber de dónde le llega una transacción. Eso sí, el usuario es el encargado de hacer públicas las direcciones de su monedero. Por tanto, si un usuario tiene especial interés en saber de dónde le llega el dinero en Bitcoin, puede adoptar como criterio el de generar una nueva dirección cada vez que inicia una relación de intercambio monetario en Bitcoin. Si sigue este criterio, el usuario tendrá una dirección de recepción de transacciones para cada uno de los posibles remitentes con los que interacciona en Bitcoin.

Protocolo de consenso distribuido en Bitcoin: la prueba de trabajo como mecanismo para eludir el intermediario financiero

Hasta ahora contamos con un procedimiento que permite transferir criptomonedas desde un usuario a otro. Es más, dicha transferencia se efectúa protegiendo la identidad de los usuarios; emulando, pues, esta característica esencial del dinero en efectivo. Por otro lado, el dinero en metálico tiene la cualidad de que una vez ha cambiado de propietario no puede volver a ser utilizado por el propietario original. Dicho de otra forma, la traslación de propiedad en el caso del metal o del billete imposibilita posteriores disposiciones por parte de quien paga. En el caso del dinero electrónico no existe bien tangible que evite el problema del doble pago (Karame *et al.*, 2012).

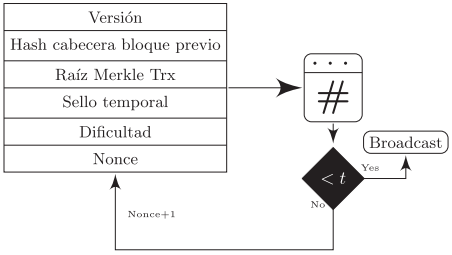
Es decir, cuando un usuario paga una cantidad de dinero en metálico por un bien adquirido, pierde el control de ese dinero, lo que no sucede en el caso del dinero virtual, dado que no hay nada material que se entregue a cambio del bien que se adquiere. Por ello, es importante garantizar que una vez hecho un pago con Bitcoin, la cantidad de criptomoneda utilizada deja de

estar disponible para el usuario que ha pagado con ella, de modo que no pueda volver a utilizarla.

En el caso de Bitcoin, la solución a la amenaza del doble gasto viene dada por la prueba de trabajo o PoW. La PoW es un puzle matemático que está basado en la dificultad para encontrar colisiones en funciones resumen (capítulo 1) y exige que la incorporación de un nuevo bloque a la blockchain se produzca única y exclusivamente si se obtiene un valor aleatorio que, concatenado a la cabecera del bloque, dé lugar a un resumen que tenga un cierto número de ceros a la izquierda. El valor aleatorio se conoce como número de un solo uso o *nonce* (*number used once*), y no tiene significado más allá de servir de apoyo al puzle matemático de la PoW.

En la figura 7 aparece resumido el procedimiento que se aplica en la resolución del desafío previo a la incorporación de un bloque por parte de un nodo. Como se puede observar en dicha figura, el resultado de la función resumen es comparado con un valor umbral *t*, que equivale a forzar un cierto número de ceros a la izquierda de la función resumen. En consecuencia, el desafío base de la PoW busca colisiones parciales de la función resumen: cuanto más pequeño es el valor de *t*, más ceros han de existir a la izquierda de la salida del resumen y, por tanto, más difícil es resolver el desafío.

Figura 7
Prueba de trabajo para la inclusión de nuevos bloques en Bitcoin.



Aunque pueda parecer lo contrario, lo cierto es que el anterior procedimiento de búsqueda de colisiones parciales es un problema de alto coste computacional. Hasta tal punto es así,

que solo una porción mínima de los usuarios del ecosistema Bitcoin participan de modo activo en la búsqueda de tales colisiones. Todos esos nodos son los denominados mineros, esto es, nodos que contribuyen al mantenimiento de la red mediante la incorporación de nuevos bloques tras resolver el puzle asociado a la PoW. Esta contribución implica tiempo de cómputo y, por ende, gasto energético y económico para los mineros. Por ejemplo, en España el minado de un bloque de Bitcoin a fecha de marzo de 2018 tiene un coste medio de más de 20.000 euros[17].

Ahora bien, esta contribución de los mineros no es desinteresada, ya que el minero que consigue agregar un bloque a la blockchain recibe una recompensa; proceso que equivale a la creación de moneda (nuevas bitcoins). Por este motivo, a la labor de los nodos que buscan colisiones se la denomina "minado", pues al fin y al cabo supone un esfuerzo por extraer el nuevo metal que conforma la criptomoneda bitcoin.

La cuestión de la recompensa, pues, es un elemento crucial en la estabilidad de Bitcoin. Al igual que ocurría con el dinero fiduciario, el soporte final de la moneda debe ser un bien escaso y, por tanto, caro (como el patrón oro, por ejemplo). En el caso de Bitcoin, el patrón de medida de moneda queda determinado por el consumo energético que los mineros deben realizar para minar nuevos bloques y generar nueva moneda. Si la recompensa por minar disminuye, eso equivale a encarecer el precio de la energía, lo que se puede considerar como una reducción de la capacidad de cómputo disponible en la red. Esta minoración de la recompensa se lleva a cabo cada $21 \cdot 10^5$ bloques, que equivale aproximadamente a cuatro años[18]. Desde junio de 2016 la recompensa que recibe un minero tras resolver la PoW es de 12,5 bitcoins y no volverá a cambiar hasta 2020.

Es más, el modelo de moneda en Bitcoin es deflacionario. Así, el número de criptomonedas que se puede generar está limitado a un número total de $21 \cdot 10^6$ bitcoins, lo que supone que no se podrán generar más criptomonedas después del año 2140.

17. En https://powercompare.co.uk/bitcoin-electricity-cost/ se puede encontrar un análisis del coste pormenorizado del minado en Bitcoin por país (último acceso 11/03/2019).
18. Véase https://www.bitcoinblockhalf.com/ (último acceso 11/03/2019).

El objetivo de esta limitación no es otro que incorporar un mecanismo que permita evitar la especulación con la criptomoneda. La paulatina merma del rédito que la actividad de minado pueda reportar es una suerte de dique de contención frente a la tendencia especuladora que hemos presenciado desde el origen de Bitcoin.

Dado que Bitcoin es una tecnología *cypherpunk*, quizá su adscripción como moneda deflacionista sea una inferencia precipitada por la dinámica de tensión que plantea tal colectivo entre los polos de lo instituido y de lo instituyente. En esta lid, dado que el modelo económico liberal de nuestra sociedad es inflacionario, lo lógico es que Bitcoin sea de espíritu deflacionario. Ahora bien, la tendencia observada en la comunidad Bitcoin parece indicar que no sigue patrones económicos muy distintos de otras actividades de negocio. Además, hay que señalar que un bitcoin es infinitamente divisible, lo que facilita la gestión de la moneda incluso en un contexto en el que no se pueda generar nueva criptomoneda. En efecto, la mínima unidad de valor que puede transferirse en Bitcoin es 1 *satoshi*, que equivale a 0,00000001($=10^{-8}$) bitcoins. Esta equivalencia viene derivada de que en Bitcoin se trabaja con precisión de ocho decimales. Ahora bien, este fue un criterio establecido de forma arbitraria por Satoshi Nakamoto al crear Bitcoin y que puede ser modificado aumentando el número de decimales cuando no se pueda generar moneda y se quiera controlar su valor en pos del interés común de la comunidad Bitcoin. Consecuentemente, a todas luces parece que (como en tantos otros casos) la naturaleza final de la criptoeconomía auspiciada por Bitcoin vendrá dada en función de sus condicionamientos técnicos, de acuerdo con las capacidades e intereses de sus usuarios.

Ese cruce de dominios no es una circunstancia exclusiva de Bitcoin, pero la forma en la que se articulan y entreveran sí que es de significada singularidad. Hasta aquí, hemos venido hablando de Bitcoin casi exclusivamente en términos técnicos. Aunque se han subrayado ciertos elementos relevantes sobre la intención política de su autor, así como algunas de sus consecuencias socio-económicas, no se ha incidido en que el funcionamiento de Bitcoin exige un correcto ensamblaje entre elementos criptográficos, el despliegue de una red de comunicación entre pares o P2P y la existencia de una comunidad que sea receptiva al uso de

esta tecnología. Si alguno de estos tres elementos no goza de una buena salud, Bitcoin dejará de funcionar.

Un elemento crucial para ese adecuado engranaje es la promoción de incentivos que estimulen la colaboración de los mineros. Como se ha reseñado previamente, los usuarios de Bitcoin engloban tanto a meros consumidores de la infraestructura como a aquellos que de forma activa van capturando nuevas transacciones para generar nuevos bloques que, una vez resuelto el puzle criptográfico de la PoW, son transmitidos a la red P2P de Bitcoin. De modo resumido, el proceso de minado y de propagación de nuevos bloques tiene el siguiente conjunto de características:

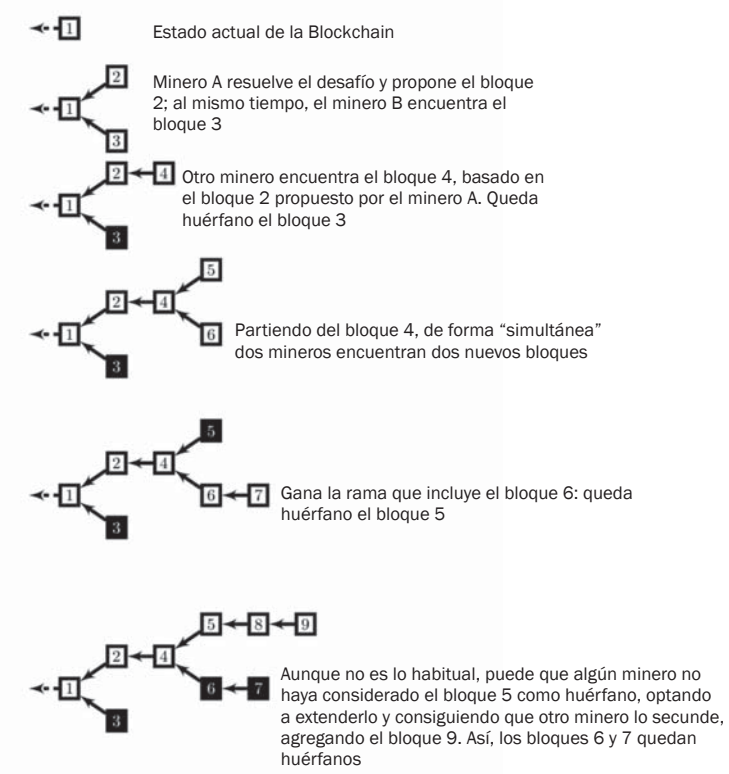
- La selección del minero que va a dar entrada a un bloque se realiza de modo no determinístico, dado que es el resultado de la competencia que existe por ser el primero en generar un nuevo bloque con nuevas transacciones. Cada vez que un usuario crea una nueva transacción envía por difusión (*broadcasting*) una petición de registro. Esa petición de registro es capturada por los nodos de la red P2P de Bitcoin a los que ese usuario tiene acceso, los cuales, a su vez, enviarán la petición a su entorno de vecindad. A la vez que se propaga la solicitud realizada por el emisor, los distintos nodos de la red guardan la transacción en una memoria temporal o *mempool*. Los nodos que están involucrados en la labor de minado de modo periódico seleccionan transacciones de su *mempool* para incluirlos en nuevos bloques.
- La integridad referencial de los bloques está garantizada por el hecho de que la cabecera de cada bloque referencia al bloque anterior a través de su resumen. De esta forma, cada vez que se extiende la cadena, un nodo de la red puede comprobar que el resumen incluido en la cabecera de cada bloque en verdad se corresponde con el resumen del bloque previo. Esta comprobación se puede efectuar desde el último bloque al bloque génesis, esto es, el primer bloque que Satoshi Nakamoto creó el 3 de enero de 2009, y a partir del cual se fue desplegando la blockchain de Bitcoin.
- Los mineros obtienen las transacciones de la memoria temporal, de forma que aquellas transacciones que incorporan un pago más elevado por minado son las primeras en ser seleccionadas.

En este punto hemos de tener presente que al crear una transacción se debe cumplir que la suma de los valores de entrada tiene que ser mayor o igual que la suma de los valores de salida. Esto significa que el balance de una transacción puede derivar en un balance positivo. Pues bien, este remanente es un pago extra que se les ofrece a los mineros para que seleccionen la transacción a la hora de construir un nuevo bloque.

- Hay que tener en cuenta que, en un momento dado, en la red Bitcoin pueden coexistir dos estructuras: los árboles de bloque y la cadena activa. Esta diferencia surge de la naturaleza P2P de la red de comunicación de Bitcoin. En determinado momento, todos los mineros empiezan a buscar un nuevo bloque que incorpore en una cabecera el resumen del último bloque existente en la cadena. Puede ocurrir que dos nodos mineros resuelvan la PoW y trasmitan por difusión el nuevo bloque. Esa difusión no llega de modo simultáneo a todos los nodos de la red Bitcoin. Por tanto, es posible que ulteriores mineros traten de incorporar nuevos bloques a partir del bloque incorporado por uno de los mineros, mientras que otros harán lo propio considerando el bloque de otro minero. En consecuencia, se produce una bifurcación o *fork* en la cadena de bloques, tal y como aparece representado en la figura 8. En este estadio se produce la aparición de un árbol de bloques con distintas ramas, una de las cuales es más exitosa que el resto al incorporar una mayor cantidad de bloques. La cadena activa vendrá determinada por la ruta que involucra mayor trabajo por parte de los mineros y que genera más consenso, que no es sino aquella determinada por la lista enlazada de resúmenes desde el bloque génesis a la hoja de esta rama exitosa. Como vemos en la figura 8, este criterio de poda involucra que existirán bloques correctamente minados que no formarán parte del estado de Bitcoin. Esos bloques se denominan "huérfanos", y aunque son bloques que no forman parte de la cadena activa, eso no significa que se pierdan las transacciones que llevan incorporadas. De hecho, las transacciones incluidas en bloques huérfanos son consideradas como transacciones no confirmadas que eventualmente volverán a la memoria temporal a la espera de que algún minero las incluya en un nuevo bloque.

FIGURA 8

Ramificación (*fork*) de la blockchain y aparición de bloques huérfanos.



- La búsqueda de colisiones de funciones resumen es una operación que en el contexto bitcoin hace tiempo que no se ejecuta con ordenadores personales convencionales. Cuando se realizó el primer minado de un bloque bitcoin en 2009, la recompensa era de 50 bitcoins. La alta rentabilidad del minado, pues, hacía interesante dedicar un ordenador con unidad de procesamiento (CPU) de múltiples núcleos a la labor de cómputo de resúmenes. Era interesante y rentable, ya que si se disponía de un par de ordenadores con buenas prestaciones, se podían ganar unos euros al día simplemente por minar. El retorno de inversión de la operación de minado se fue reduciendo, fruto de la confluencia de dos factores. Por un lado, la

entrada en juego de un mayor número de competidores; y, por otro, la caída de la recompensa que se produce en la red cada cuatro años. Basta considerar que los 50 bitcoins que se ganaban en 2009 tras minar un bloque, en el momento de escribir este libro eran 12,5 bitcoins. Frente a este encarecimiento del proceso de minado, en septiembre de 2010 Laszlo Hanyecz desarrolló e hizo público un nuevo cliente de minado basado en procesadores GPU (Graphics Processing Units) de tipo CUDA (Compute Unified Device Architecture). Este tipo de procesadores fue originalmente concebido para el procesamiento en paralelo de imágenes, pero Hanyecz explotó su capacidad de trabajo en paralelo para probar de modo simultáneo múltiples *nonces* y verificar si se conseguía superar la PoW. Sin embargo, este tipo de electrónica no es la más adecuada para buscar colisiones en el caso de que la función resumen sea de la familia SHA-2. En este supuesto es mejor trabajar con diseños basados en FPGA (Field Programmable Gate Arrays), diseños que son la base de los actuales ASIC (Application-Specific Integrated Circuits) de minado. Lo que es más, a medida que fue evolucionando la red Bitcoin, el rol del minero se fue optimizando y profesionalizando. Fruto de esta tendencia, empezaron a surgir iniciativas para la agrupación de mineros y la creación de las denominadas "granjas de minado". En la actualidad son estas granjas de minado las que sostienen la red Bitcoin, siendo imposible que un minero de forma individual y aislada pueda competir con magnates de la industria de minería de criptomonedas como Bitmain/Antpool, Bitfury, DiscusFish/F2Pool, BTCC/BTC China o Eligius.

- Por estricta decisión de diseño, el tiempo transcurrido entre la inclusión de dos bloques consecutivos en la blockchain de Bitcoin ha de ser de 10 minutos. Este es un criterio conservador que persigue garantizar la propagación del nuevo bloque a toda la red P2P de Bitcoin. Para garantizar esta tasa de generación de nuevos bloques, cada 2016 bloques se reajusta el parámetro de dificultad que figura en la cabecera de los bloques (figura 7) de acuerdo con la siguiente fórmula:

$$Dn = Dv \cdot 2016 \cdot 10 / Ti$$

donde *Dn* es la dificultad nueva, *Dv* es la dificultad vieja y *Ti* es el tiempo invertido en minar los últimos 2016 bloques. Aquí cabe preguntarse cómo la red Bitcoin puede imponer un valor determinado para el parámetro de dificultad sin que exista ninguna autoridad central. De nuevo hay que recordar que Bitcoin es un esquema criptográfico de protección de información, un protocolo de comunicación no centralizado y un modelo de consenso propiciado por el interés de los usuarios por la sostenibilidad del sistema. Este último punto es el que hace factible la imposición del criterio de actualización del parámetro de dificultad. En efecto, una vez que ha sido minado un bloque, cualquier nodo de la red puede verificar si el valor del campo de dificultad coincide con el esperado de acuerdo con la fórmula de más arriba. En caso de que se detecte alguna incoherencia bien en ese valor, bien en el resumen de la cabecera del bloque, el nodo descartaría el bloque. De este modo, el minero responsable de este bloque perdería la oportunidad de generar la cadena más larga y, por ende, no recibiría recompensa alguna.

Más allá de Bitcoin: aplicación de blockchain como soporte de protocolos generales de comunicación

Tal y como reza el llamado "principio de Skolnikoff" (Skolnikoff, 1994), los usos que se hacen de una tecnología en muchas ocasiones no coinciden con la utilidad que se le presuponía cuando fue creada. El acoplamiento entre la tecnología y sus usuarios, en efecto, contribuye a la configuración y reacondicionamiento de la tecnología. Al igual que ocurría con la aplicación del servicio SMS para realizar pagos en M-Pesa, la estructura de datos que soporta la infraestructura Bitcoin es un ejemplo paradigmático de este proceso de conformación de la novedad tecnológica.

Además de los *scripts* a los que hemos hecho mención, en Bitcoin existen otras funcionalidades debidamente codificadas mediante Script. Sin entrar en el detalle (para obtener más información véase Narayanan *et al.*, 2016), lo que sí conviene resaltar aquí es la forma en la que se ha venido configurando toda una serie de servicios explotando la definición abierta del *script*

Proof-Of-Burn. Este tipo de *script* tiene por objetivo usar criptomonedas sin que exista destinatario alguno. Para ello existen dos opciones: 1) se crea una transacción que tiene por destino una dirección que no tiene clave privada alguna[19], y 2) se hace uso de una instrucción específica del lenguaje Script o código de operación (en concreto del *opcode* OP_RETURN) que genera un error e interrumpe la ejecución del código contenido en una transacción, de forma que no importa la información que estuviera contenida después de un OP_RETURN. Esta circunstancia abre la posibilidad de utilizar la instrucción OP_RETURN como un canal de almacenamiento de información dentro de un protocolo de comunicación general. En la actualidad, este canal tiene una longitud máxima de 83 bytes, lo que constituye una clara limitación en cuanto al tipo de mensajes que se pueden enviar a través de él. Ahora bien, ese tamaño puede ser suficiente de cara a implementar un canal de control que habilite la verificación de información incluida en protocolos más complejos. De hecho, existe un muy amplio abanico de modelos de negocio que utilizan la información opcional del OP_RETURN para monitorizar activos de diverso tipo, para efectuar servicios de notaría documental o para desplegar mecanismos de protección de la propiedad intelectual (Bartoletti y Pompianu, 2017).

La instrucción OP_RETURN no es ni mucho menos la única instrucción que se puede utilizar para construir un protocolo de comunicación. Efectivamente, la estructura de datos de Bitcoin incluye campos libres en las siguientes estructuras de datos (Franco, 2014):

- Transacciones *coinbase*. Este tipo de transacciones es el que está relacionado con la creación de nuevas monedas. Cuando un minero gana "la lotería" de la PoW (Narayanan *et al.*, 2016) se crea una transacción distinta de las normales y se almacena en la primera hoja del árbol de Merkle

19. En Bitcoin, y en la mayoría de las blockchains, es posible incluir cadenas de texto arbitrarias como direcciones de destino. De hecho, es posible que no exista una clave pública cuya codificación corresponda con la cadena de texto indicada y, por lo tanto, tampoco habrá una clave privada relacionada. Esto hará que los bitcoins correspondientes queden bloqueados.

de las transacciones (capítulo 1 y figura 7). Las principales diferencias de una transacción *coinbase* respecto a las normales son:

  - Solo tiene una entrada y una salida.
  - La entrada no está respaldada por la salida de una transacción previa, con lo que el resumen que aparece en la transacción no apunta a bloque alguno y, por ello, se dice que es un puntero resumen nulo.
  - La recompensa que se ha recibido por minar el bloque. Su valor vendrá dado por la recompensa por defecto de la red más el excedente determinado por las entradas de la transacción.
  - Es un campo arbitrario que puede ser utilizado por el minero para escribir en la blockchain aquello que desee y, en consecuencia, hace factible su utilización como parte de un protocolo externo a Bitcoin.

- Direcciones falsas. Al igual que se hace al emplear `OP_RETURN`, se puede enviar dinero a una dirección falsa (es decir, que no tiene clave privada asociada) que codifica la información que queremos almacenar en la blockchain. Como toda transacción, su emisión tiene un coste, que como mínimo debe ser igual a 546 satoshis, que es el coste de una transacción polvo (*dust transaction*).
- Transacciones con múltiples firmas (*multisignature*). Tal y como se reseñó anteriormente, en Bitcoin existe un tipo especial de transacción en la que la recepción de la cantidad transferida puede requerir el concurso de más de un usuario. De modo general, deben firmar m usuarios de un total de n posibles (con $m \leq n$). En el caso que nos ocupa, podemos considerar transacciones con múltiples firmas de tipo 1-de-n, esto es, solo necesitaremos tener una dirección válida para poder reclamar la cantidad asociada a la transacción. Como resultado, el resto de direcciones pueden ser falsas y emplearse para codificar información extra. Eso sí, cabe subrayar que en la implementación actual el valor máximo de n es 3, lo que reduce considerablemente la capacidad de almacenamiento mediante este procedimiento.

La blockchain de Bitcoin, pues, ofrece un medio de persistencia de información de gran interés para el despliegue de nuevos protocolos de comunicación y de gestión de la información. El reto aquí consiste en saber conjugar la capa de confianza de blockchain con la pila de protocolos de los actuales sistemas de información y comunicación. Este ejercicio, por otro lado, debe efectuarse siguiendo el espíritu de los procesos de estandarización llevados a cabos en el dominio de la Internet desde sus inicios (Hardjono *et al.*, 2018), para de este modo configurar soluciones eficaces y eficientes que aprovechen las propiedades de la blockchain.

Limitaciones de Bitcoin: problemas de seguridad, de escalabilidad y de gobernanza

Sin ningún género de duda, Bitcoin supone un auténtico hito en la configuración de un entorno de compartición de información sin necesidad de que exista confianza expresa en un agente central (Narayanan y Clark, 2017). Por un lado, Bitcoin define una plataforma, un protocolo y una red de colaboración que van más allá del mero y simple intercambio financiero. Esa triple alianza configura una estructura de datos que es el embrión del fenómeno blockchain actual. Por otro lado, en el caso de Bitcoin, la potencia que se le presupone a este fenómeno adolece de una serie de características que en muchos casos se tornan en disfunciones estructurales que obstaculizan su papel como motor de la transformación digital.

El primer reparo que podemos observar respecto a la adopción de la infraestructura Bitcoin reside en su autoría anónima. Aunque esto no ha sido óbice para su expansión, el hecho de que no se sepa quién o quiénes están detrás de la figura de Satoshi Nakamoto no abona la confianza. Si no conocemos quién o quiénes están detrás del artículo seminal, si no tenemos información sobre la forma en la que se llevó a cabo el primer prototipo de la red Bitcoin, si no contamos con elementos suficientes para determinar y cuestionar los intereses de Satoshi Nakamoto, ¿por qué hemos de confiar en

esta tecnología?[20]. Según Nakamoto, hemos de confiar precisamente porque no está basada en la "confianza", sino en la criptografía.

Lo cierto es que es fácil encontrar ejemplos donde la criptografía no ha conseguido dar respuesta a las expectativas de sus usuarios. Así, en los últimos años podemos hallar ejemplos de errores en la implementación de protocolos criptográficos, robos de credenciales en plataformas digitales, incluso fallas graves de seguridad en dispositivos *hardware* resistentes a la manipulación. En el caso particular de Bitcoin existen tres sucesos muy significativos que ponen bajo cuarentena la sentencia de Nakamoto (Campbell-Verduyn, 2017):

1. La ramificación (*fork*) involuntaria de marzo de 2013. La actualización del software al realizar la transición desde la versión 0.7 a la 0.8 del núcleo del software de Bitcoin implicó la sustitución del motor de base de datos Berkeley DB por Level DB, que permitía sincronizar de forma más rápida. En un momento dado, tras la actualización del *software*, se generó un bloque de gran tamaño que no era compatible con la política de acceso bloqueado de Berkeley DB. Así, este bloque sí que era aceptado como válido por los nodos con la versión 0.8 de Bitcoin, pero era rechazado por los nodos que mantenían la versión anterior del *software*. Como resultado, se produjo una bifurcación en la cadena, con la singularidad de que los nodos mineros que tenía la versión 0.7 instalada tenían mayor capacidad de cómputo, lo que hizo que su rama de la cadena creciera más rápido. La gestión de este problema pone de relieve los sistemas de gobernanza que operan en Bitcoin. Después de muchas conversaciones y discusiones a través de foros abiertos y de correo, prevaleció el criterio de recomendar instalar la versión 0.7 en todos los nodos de la red Bitcoin. La forma en que se encauzó la toma de decisión puso de relieve la posición de privilegio del grupo de programadores encargados del mantenimiento del núcleo del

20. Este recelo sobre la fundación anónima de la criptomoneda bitcoin lo manifiesta Jeremy Allaire (Vigna y Casey, 2016), el fundador de la plataforma financiera Circle, al decir que *"Mysterious in the case of money is not so good"* (lo misterioso en el caso del dinero no es nada bueno).

software, así como la capacidad de influencia de operadores de granjas de minado como BTCGuild y Slush. Esto es, el suceso deja entrever una recentralización en la red Bitcoin.

2. La caída de la casa de cambio Mt. Gox en 2014. Mt. Gox fue una empresa intermediaria que operaba convirtiendo bitcoins en moneda fiat (es decir, en dólares, euros o cualquier otra divisa que cuenta con el respaldo del Banco Central correspondiente). Para ello, Mt. Gox tenía desplegada toda una infraestructura fuera de la cadena (*off-chain*), que fue atacada exitosamente en 2011 en primera instancia y de modo mucho más efectivo en 2014. El ataque en cuestión permitió la sustracción de un volumen considerable de criptomonedas de clientes de Mt. Gox. Tras el incidente de 2014, los responsables de la empresa intentaron culpabilizar a la infraestructura Bitcoin del robo de criptomoneda. La información que se pudo recabar directamente desde la blockchain y desde los diversos foros de discusión en redes sociales y listas de correo, unida a las pesquisas realizadas por el FBI norteamericano, permitió señalar a la plataforma de Mt. Gox y a la gestión de sus dirigentes como la verdadera fuente del problema de seguridad. De nuevo los mecanismos de gobernanza se establecen desde fuera de la blockchain, como consecuencia de la colaboración activa entre los usuarios y supervisores de la tecnología.
3. El caso de Silk Road. Bitcoin y Tor[21] suelen asociarse al mercado negro en muy buena medida como consecuencia de la actividad de Silk Road. Esta plataforma se presentó en 2011 como una opción para el libre intercambio de bienes sin tener que rendir cuentas a Estado o autoridad alguna. Con el tiempo, se demostró que el tipo de bien y de actividad a la que estaba dando cobijo Silk Road era de índole delictiva. Los pagos en Silk Road se producían en bitcoins, pero la interacción con sus usuarios se realizaba a través de servidores y servicios

21. Tor es un protocolo de red diseñado para ocultar la localización de los usuarios cuando navegan por Internet. En la medida que Tor permite navegar de modo anónimo, es utilizado en muchos casos para realizar actividades ilegítimas en la red (Christin, 2013). Ahora bien, Tor también es una herramienta para proteger la privacidad de los usuarios y una defensa frente a políticas de censura de servicios en la red (Brunton y Nissenbaum, 2015).

que estaban ubicados fuera de la blockchain. La protección de esa infraestructura era de alta complejidad, pues involucraba el uso de técnicas avanzadas de cortafuegos, enrutadores de tráfico y ocultación del origen de las comunicaciones mediante Tor. Esta complejidad tenía por meta evitar la trazabilidad de las operaciones y de los usuarios por parte de las autoridades y fuerzas de seguridad. Lo cierto es que los propios usuarios de Silk Road tenían dificultades para realizar transacciones y la compleja red de comunicaciones presentaba ciertas debilidades en puntos concretos que habilitaron las tareas de investigación del FBI. Como resultado de todo este trabajo, el FBI logró dar con los responsables de Silk Road.

La principal conclusión que hemos de extraer de las tres disfunciones señaladas en Bitcoin es que la blockchain por sí sola no soluciona todos los problemas que existen en los sistemas de información y de comunicación. En su haber, blockchain permite impulsar modificaciones orientadas a crear soluciones más robustas, pero tal proceso exige pensar muy en profundidad qué tipo de imbricación ha de tener la blockchain en la actual infraestructura TIC. En esa reflexión hemos de considerar de modo muy sosegado dos cuestiones capitales: la escalabilidad y la sostenibilidad de la tecnología.

La "escalabilidad" de Bitcoin afecta al tamaño en memoria secundaria (el disco duro de nuestro ordenador) que ocupa la blockchain. En la medida que la blockchain es un registro de actividad en el que se añade información y no se borra, y teniendo en cuenta que el número de usuarios y la actividad irá creciendo a lo largo de los próximos años, el número de bloques y el espacio que ocupan en disco duro puede resultar prohibitivo[22]. De hecho, en el momento de redacción de este libro, el tamaño de la blockchain de Bitcoin es de 254 GB, tamaño que no es ni mucho menos manejable si estamos trabajando con dispositivos con poca capacidad de almacenamiento y con restricciones en términos de consumo energético. Es cierto que en estos

22. Véase https://www.blockchain.com/es/charts/blocks-size (último acceso 11/03/2019).

escenarios se pueden buscar opciones apoyadas en clientes ligeros de Bitcoin en tales dispositivos y un nodo central con un cliente que cuente con una réplica completa de las transacciones de Bitcoin, pero lo cierto es que este tipo de restricciones han de ser bien tenidas en cuenta a la hora de diseñar arquitecturas de sistemas basadas en algún tipo de blockchain.

En lo relativo a la "sostenibilidad", hemos dicho que el recurso limitado que Bitcoin utiliza como patrón-dinero es la energía. La búsqueda de colisiones en estos momentos se realiza fundamentalmente en las denominadas granjas de minado e involucra un gasto energético muy elevado. De hecho, a finales de 2017 existían 159 países cuyo consumo energético estaba por debajo del consumo de energía involucrado en la generación de bitcoins. Eso por lo que respecta al consumo global de Bitcoin, pero además es de tener en cuenta que el reparto de la capacidad de cómputo no es homogéneo, pues existen granjas de minado que sobrepasan con creces la capacidad de cómputo del resto de mineros del ecosistema Bitcoin. Es particularmente relevante el caso de GHash.IO, empresa que surgió a mediados de 2013 como agrupación de grupos de mineros y que a mediados de 2014 alcanzó una capacidad de cómputo del 51% de la capacidad total de la red. En ese caso los responsables de GHash.IO emitieron un comunicado señalando que no pensaban utilizar esa situación de privilegio para subvertir la blockchain.

Efectivamente, recordemos que el minado de un bloque en la blockchain de Bitcoin requiere encontrar un *nonce* tal que se consiga un resumen de la cabecera del bloque con un número determinado de ceros a la izquierda. Si existiera un nodo con una capacidad de cómputo mayor que la mitad de la capacidad, eso significaría que ese nodo podría utilizar la misma cantidad de criptomoneda para realizar más de un pago (algo que atenta contra los principios del dinero electrónico). Para ello, a la hora de realizar un pago, el nodo en cuestión crea una transacción para el destinatario correspondiente. Tras ello, envía por difusión la transacción y empieza a minar bloques en privado haciendo crecer la blockchain en una rama distinta de la cadena activa. Si el nodo malicioso consigue minar n bloques antes de que se confirme la transacción real, entonces habrá conseguido utilizar varias veces la misma criptomoneda. En caso de que esto no ocurra

de forma general, el resto de nodos eventualmente acabarán detectando todos los intentos de doble gasto, algo que pondría en peligro la confianza en la infraestructura Bitcoin.

Este ataque, llamado "por mayoría" o "ataque del 51%", afecta, pues, sobre todo a la credibilidad de Bitcoin. Las características de la red P2P y la naturaleza criptográfica de la estructura de datos de Bitcoin no hace posible el borrado de información de forma trivial, pero el descrédito que este ataque depararía es más que suficiente para dejar inservible a Bitcoin.

No obstante, esta no es la única vulnerabilidad del modelo de consenso de Bitcoin. A la hora de explicar la creación de los bloques de datos, vimos que las diversas transacciones eran seleccionadas por los mineros desde la memoria temporal (Narayanan *et al.*, 2016). Por defecto, en el proceso de selección se priorizan las transacciones en función del remanente que los emisores han dejado como pago para los mineros. Asimismo, cuando se decide sobre qué bloque se debe continuar minando, el criterio de decisión por defecto apunta a escoger la cadena de bloques más larga (la que tiene más altura). Si existen dos cadenas con la misma altura, se escoge aquella que se vio primero. Por último, otro comportamiento por defecto atañe al criterio que se sigue una vez se ha minado un bloque. Aquí la pauta base es informar de la creación inmediatamente después de la creación del bloque, esto es, se enviará el mensaje de difusión a los nodos vecinos sin dilación. Modificaciones sobre estos comportamientos, así como otras pautas de decisión básicas, hacen factible cribar ciertas transacciones y establecer colusiones entre nodos en la red con objeto de perjudicar a usuarios y/o nodos concretos, o de buscar beneficio particular cuando sea posible.

CAPÍTULO 3

Contratos inteligentes y Ethereum

En este capítulo vamos a presentar los denominados contratos inteligentes, sus principales características y su relación con la tecnología blockchain. De forma más pormenorizada introduciremos el sistema Ethereum, su criptomoneda asociada, ether, y cómo realizar contratos inteligentes con esta tecnología. Se comentarán otros aspectos como el ciclo de vida de un contrato inteligente y la visión de futuro de Ethereum.

Los contratos inteligentes (*smart contracts*), aunque de moda en los últimos años, son en realidad un concepto ideado hace más de dos décadas por Nick Szabo. En un ensayo, en 1994, Szabo previó la confluencia de la informática, la criptografía, las leyes y la economía, para dar lugar a "entes autónomos" que codificarían comportamientos complejos con el fin de resolver, de forma eficiente y segura, diferentes situaciones cotidianas y de negocios. Por ejemplo, pagos automatizados y "propiedad inteligente" capaz de gestionarse a sí misma sin ayuda humana.

Mucho de lo que se ha propuesto en los últimos años bajo el paraguas de las cadenas de bloques tiene su fundamento de negocio en las visionarias palabras de Szabo. Entre otros, su ejemplo de un coche financiado por un banco, que evita ser utilizado en el caso de que su propietario no realice un pago (siempre y cuando el coche no esté en marcha).

Szabo señala que, por un lado, las tecnologías subyacentes y su difusión estaban aún muy inmaduras y, por otro, los casos de uso eran poco conocidos por los tecnólogos. No obstante, con la llegada de la criptomoneda bitcoin en 2008 (Nakamoto, 2008) y la tecnología blockchain que la sustenta, se empezó a vislumbrar la aplicación práctica de estos conceptos. Como veremos a continuación, no es casualidad: los sistemas blockchain combinan las tecnologías que Szabo menciona como potenciales pilares de sus contratos inteligentes.

Contratos inteligentes y funcionamiento en sistemas blockchain

Un "contrato inteligente", en su representación fundamental, no es más que un programa informático que codifica las operaciones que deben llevarse a cabo dependiendo de sucesos externos al programa. El término "contrato" es debido a que dichas operaciones suelen enmarcarse dentro de un acuerdo entre partes, como el acuerdo de pagar la cuota del coche en el ejemplo anterior. Otro ejemplo frecuente es el de una cerradura de una habitación de hotel que solo se abre para el usuario que ha pagado dicha habitación cuando este lo requiere.

Esta simplificación de los contratos inteligentes ha llevado a muchos a afirmar que estos no son inteligentes, sino que simplemente siguen una serie de instrucciones predefinidas; también se dice que no son contratos, ya que no suelen llevar respaldo legal. Es cierto que en el ensayo de Szabo no se formalizan sus propiedades, más allá de enmarcarlos dentro de un mundo ciberfísico donde lo digital o virtual interactúa con lo físico. No obstante, a continuación se describen de soslayo las tecnologías que los harán posibles y que dotan a los contratos inteligentes de propiedades fundamentales no presentes en un programa informático tradicional. Estas propiedades son:

- El resultado producido por un contrato inteligente no es simplemente el obtenido por un ordenador tras ejecutar un código determinado; es, además, el resultado que ha

sido consensuado por un conjunto de ordenadores comunicados entre sí.
- Dicho resultado se escribe en un registro público de tal manera que es posible detectar posteriores alteraciones del mismo.
- El registro público es mantenido para siempre por todos los ordenadores del sistema, o por una parte importante de ellos.

Estas propiedades se heredan, como hemos dicho, de las tecnologías que finalmente se han utilizado para implementar la visión de los contratos inteligentes de Szabo. La primera propiedad anterior se resume en que el resultado de un contrato inteligente es "consensuado" por la red de ordenadores que componen el sistema. Es decir, existe un "protocolo de consenso" que gobierna esta red mediante el cual los ordenadores que la forman deciden cuándo un resultado es válido y cuándo no. La segunda propiedad se refiere a que se preserva la *integridad* de los resultados que produce un contrato inteligente mediante mecanismos criptográficos. Por último, la tercera propiedad indica que los datos estarán disponibles siempre y cuando sigan quedando ordenadores en la red (y se cumpla el requisito de que al menos la mitad de los ordenadores que componen el sistema son honestos).

Combinando las tres propiedades ya citadas, se consigue una propiedad adicional y esencial, conocida como "inmutabilidad". Es decir, dado que el resultado de un *smart contract* es consensuado por toda la red, se escribe de forma que se garantiza su integridad y el registro en el que se escribe se mantiene siempre. Así, en cualquier momento futuro se podrá consultar cuál fue el resultado, con la certeza de que lo que esté escrito en el registro es el resultado que efectivamente produjo el contrato inteligente.

De esta manera, llegamos al punto común entre los contratos inteligentes y los sistemas blockchain: un sistema blockchain está, *grosso modo*, compuesto por los elementos anteriores (protocolo de consenso, criptografía y registro público), con el fin de proporcionar inmutabilidad sin necesidad de

autoridades centrales, de tal manera que es posible ejecutar contratos inteligentes "a la Szabo" usando un sistema blockchain como base tecnológica.

Para concretar lo expuesto, vamos a resumir el comportamiento de un contrato inteligente en el contexto de un sistema blockchain. Omitiendo detalles técnicos, como interfaces y aspectos relacionados con la programación, y simplificando, supongamos que el sistema blockchain del ejemplo contiene un contrato inteligente para bloquear un coche en el caso de que su propietario no satisfaga una mensualidad del préstamo concedido por un banco. Si fuera así, entonces:

- El propietario ejecuta la operación de apertura del coche. Esta operación llega a todos los ordenadores de la red (a través del coche, que es uno de estos ordenadores).
- Cada ordenador comprueba en el registro inmutable que el propietario no ha satisfecho el último pago. El resultado de la operación de apertura es denegado.
- Dicha denegación se escribe en el registro por parte de alguno de los ordenadores que componen la red.
- Al detectar la escritura en el registro, el coche tiene la certeza de que el resultado es válido. Rechaza abrirse, informando al propietario del motivo.

Por último, para afianzar el papel de las tecnologías subyacentes, destacamos que la primera operación incluye típicamente una firma digital, de tal manera que se puede verificar quién la origina. La segunda operación implica una iteración del protocolo de consenso de la red, de tal manera que la mayoría de los ordenadores de la red llegan a la misma conclusión. La tercera operación incluye mecanismos de control de integridad, como resúmenes criptográficos, que garantizan que se detectará cualquier cambio no deseado. Todas estas propiedades garantizan la correcta ejecución de las condiciones del contrato inteligente, dotándolo, por lo tanto, de cierta similitud con un contrato tradicional, ya que se cumplen sus condiciones. A la vez dotan al contrato de cierta "inteligencia", ya que este cumplimiento se hace de forma (prácticamente) automática.

En las siguientes secciones, presentaremos ejemplos concretos de contratos inteligentes, usando como base el sistema blockchain Ethereum.

Fundamentos y diseño de Ethereum

Como hemos visto en el capítulo 2, Bitcoin fue el primer sistema en utilizar una cadena de bloques para representar la información y su evolución. Aunque se suele decir que Bitcoin no soporta contratos inteligentes, esto no es del todo cierto: toda transacción de Bitcoin es en sí misma un contrato inteligente muy sencillo que, de hecho, está compuesto por instrucciones en un lenguaje de programación propio de Bitcoin. Dado que este lenguaje es bastante limitado, las capacidades de un contrato inteligente en Bitcoin son restringidas, aunque los contratos inteligentes que pueden definirse con este lenguaje permiten llevar a cabo todas las tareas para las que fue creado.

Estas limitaciones de Bitcoin dieron lugar al nacimiento de Ethereum, un sistema blockchain diseñado para sustentar contratos inteligentes capaces de ejecutar prácticamente cualquier cómputo.

De forma más concreta, Ethereum es un sistema descentralizado en el que los nodos que lo componen (computadores ejecutando el *software* específico de Ethereum) se comunican de igual a igual, es decir, forman una red entre pares (P2P). Estos nodos mantienen una copia de la cadena de bloques de Ethereum, que incluye toda la información del estado del sistema. Para entender este estado, es necesario conocer varios conceptos fundamentales, que veremos a continuación.

Cuentas de usuario. Para poder operar en Ethereum, es necesario que cada usuario se cree una cuenta. Cada cuenta es, en realidad, un par de claves pública-privada de un criptosistema asimétrico basado en curvas elípticas (de hecho, se emplea la curva secp256k1, al igual que en Bitcoin). De este par de claves, la parte pública se utiliza para obtener de forma unívoca lo que se conoce como "dirección de la cuenta". No existe limitación en el número de cuentas que puede tener cada usuario, basta con que

se generen varias parejas de claves pública-privada. Cada vez que un usuario quiera ejecutar una transacción en Ethereum, tendrá que elaborar una firma digital con su clave privada. Como se vio en el capítulo 1, cualquiera podrá verificar dicha firma con la clave pública correspondiente. Un aspecto esencial es que, dada la naturaleza descentralizada de estos sistemas, la gestión de las claves privadas es responsabilidad única del usuario. En concreto, si un usuario pierde la clave privada de una de sus cuentas, efectivamente pierde el acceso a dicha cuenta (y todo lo que tuviera en ella). Del mismo modo, si una tercera persona consiguiera el acceso a dicha clave privada, podría controlar la cuenta como si fuera su propietario legítimo. Por ello, es conveniente tomar todas las medidas necesarias para mantener las claves privadas en secreto y sus correspondientes copias de respaldo.

Ether. Cada cuenta tiene un balance asociado en la criptomoneda interna de Ethereum: el ether. Este ether se puede utilizar de dos maneras: bien en transferencias "tradicionales", en el sentido de enviar una cantidad determinada desde una dirección *A* a una dirección *B*; o bien como pago por la creación o ejecución de un contrato inteligente. El balance de cada cuenta está escrito en la cadena de bloques de Ethereum. De hecho, por defecto todas las cuentas existen; simplemente, la mayoría de ellas tienen un balance 0 y no tienen un "propietario", es decir, nadie conoce o ha generado el par de claves pública-privada correspondiente. La unidad indivisible del ether (el equivalente a los céntimos en el euro) se conoce como *wei*. Un ether equivale a 10^{18} wei. Hay otras denominaciones intermedias como *szabo* (equivalente a 10^{12} weis) o *finney* (correspondiente a 10^{15} weis). Como curiosidad, todos estos nombres se refieren a personajes importantes en la historia de las criptomonedas.

Gas. En Ethereum, cualquier operación que implique cambios sobre el estado de la red (como las transacciones, que veremos a continuación) se traduce en instrucciones en código máquina de Ethereum. Este código será ejecutado por los nodos de la red, lo cual supone que tales nodos dedicarán recursos a satisfacer las peticiones de otros usuarios. Del mismo modo, cualquier valor

que sea necesario almacenar en la cadena de bloques consumirá espacio en los nodos que componen el sistema. Por ello, es necesario incentivar a los nodos a que contribuyan con sus recursos de procesamiento y almacenamiento; por lo que es necesario establecer formas de medir cuántos recursos se utilizan, para recompensar justamente a todos los nodos que contribuyan. Para ello, Ethereum utiliza el concepto de *gas*. Cada operación básica en el código máquina de Ethereum tiene un coste predefinido en unidades de gas, al igual que cada byte de almacenamiento requerido, al que el emisor debe hacer frente.

Transacciones. Son las operaciones que permiten hacer cambios en la cadena de bloques. Es decir, son operaciones de escritura que tienen como objetivo modificar el estado actual de la cadena, por lo que deben estar sujetas a validación. Podemos distinguir tres tipos principales de transacciones: de transmisión de valor, de creación de un contrato inteligente y de ejecución de un contrato inteligente. Los campos que componen una transacción son los siguientes:

- Origen (obligatorio): es la dirección de la cuenta que ejecuta la transacción. En el caso de una cuenta de usuario es la dirección que corresponde a la clave pública; en el caso de un contrato inteligente es la dirección asignada al contrato inteligente en el momento de su creación. Técnicamente, puede haber varios "niveles" de origen. Es decir, un usuario *U* puede crear una transacción para ejecutar un contrato *A*. Como resultado, este contrato *A* puede crear otra transacción invocando a otro contrato *B*, y así sucesivamente. Para evitar ambigüedades, Ethereum distingue entre el origen de la transacción (txOrigin) y el emisor del mensaje (msgSender). El primero siempre será la cuenta del usuario que inició la cadena de llamadas, ya que en el extremo inicial siempre debe haber una cuenta de usuario. El segundo se refiere al salto inmediatamente anterior, que será un contrato inteligente. En el ejemplo anterior, desde el punto de vista de la transacción de *A* a *B*, *U* es el origen de la transacción y *A* es el emisor del mensaje.

- Destino (obligatorio y condicional): es la dirección de la cuenta destinataria de esta transacción. Puede ser una cuenta de usuario, en el caso de transferencias de criptomoneda, o la dirección de un contrato inteligente, en el caso de querer ejecutar alguna de sus funciones. Si es una transacción de creación de contrato, el campo debe dejarse vacío.
- Número de transacción (obligatorio). Cada transacción debe llevar un número entero como identificador que debe ser único dentro de las transacciones enviadas por una misma dirección de origen. En Ethereum, estos números son secuenciales para cada dirección de origen.
- Valor (opcional): es la cantidad de ether a transferir del origen al destino. Es un valor opcional dado que no todas las transacciones son para realizar pagos.
- Gas máximo (obligatorio). Como no siempre es posible determinar de antemano cuál será el coste de una transacción, lo que se hace es especificar una cantidad máxima de gas por la que se está dispuesto a pagar: el gas máximo. Una vez se llegue al límite establecido, la transacción se cancela aunque no haya llegado a completarse. El nodo que ejecute la transacción y la incluya de forma satisfactoria en un bloque será quien se quede con el importe asociado a las unidades de gas consumidas por la transacción.
- Precio (obligatorio). Aunque el gas podría medir cuánto cuesta ejecutar una transacción y el gas máximo cuánto se debe pagar por ella, Ethereum incluye un nivel de abstracción más: el "precio", esto es, el valor que el emisor de una transacción está dispuesto a pagar por cada unidad de gas. Dado que estos conceptos (gas, ether y precio) pueden llevar a confusión, para fijar ideas, consideremos el siguiente escenario ficticio. Supongamos que una persona tiene un coche con un depósito de capacidad máxima, M, con la peculiaridad de que, cuando va a usar el coche, debe llenar el depósito, no necesariamente al máximo de su capacidad. Además, esta persona conoce una gasolinera a la que puede hacer ofertas: puede decirle al propietario cuánto dinero está dispuesto a pagarle por cada litro de combustible. Por supuesto, el propietario puede negarse a servirle si cree que el precio es muy bajo. Por otro lado, si el

dueño del coche ofrece un precio muy alto por cada litro de combustible, estará derrochando dinero. Como es lógico, esta persona se preocupará por hacer un buen análisis en dos aspectos: cuál es el precio óptimo a ofrecer por litro y cuánto debe repostar, para no quedarse tirado a mitad de camino. Volviendo a Ethereum, la transacción sería el equivalente a moverse entre dos puntos en el ejemplo del coche. Las unidades de combustible (en litros) que se repostan antes de cada trayecto, y que permitirán hacer trayectos más o menos largos, son en Ethereum las unidades de gas máximo que se especifican en la transacción, de modo que solo se podrán ejecutar, como máximo, tantas operaciones como permitan las unidades de gas especificadas. El precio en euros de cada litro de combustible que el dueño del coche ofrece al dueño de la gasolinera corresponde con el precio en ether que ofrece el emisor de la transacción al minero que la incluya en un bloque. Por último, Ethereum tiene un parámetro global, que es el límite máximo de gas que se puede usar por transacción y que es común a todos los usuarios. En el ejemplo del coche, este parámetro prefijado sería la capacidad máxima del depósito. La introducción de este parámetro adicional tiene dos ventajas importantes. La primera es que crea una especie de mecanismo de priorización: recordemos que el nodo que consigue ejecutar la transacción e incluir el resultado en un bloque, se queda con el importe asociado a la misma. Este importe total se calcula multiplicando el gas total utilizado por el precio que se le ha dado a cada unidad de gas. Por ello, la recompensa será mayor cuanto más alto sea el importe total, y hay más incentivo por ejecutar primero las transacciones con mayor importe. La segunda ventaja es un mecanismo de protección ante fluctuaciones del precio del ether en monedas fiat (en general, euros o dólares): si el precio del ether sube, se puede bajar el precio del gas para amortiguar el coste final de una transacción; mientras que si el precio del ether baja, se puede subir el precio del gas para que ejecutar la transacción siga siendo atractivo para nodos que luego intercambian el ether por dinero fiat.

- Datos (opcional). En el caso de que la transacción represente la intención de desplegar un nuevo contrato inteligente en

Ethereum, es necesario especificar el código fuente de dicho contrato para que el resto de nodos puedan comprobar si es un contrato válido. Igualmente, si la transacción pretende ejecutar una función de un contrato inteligente ya existente (más adelante detallaremos con precisión el concepto de función de un contrato), puede ser necesario que dicha función requiera parámetros de entrada. En ambos casos, los datos asociados se incluyen en este parámetro de la transacción.

Por último, conviene enfatizar el "modelo transaccional" adoptado por Ethereum. En el capítulo 2 vimos que Bitcoin sigue el modelo conocido como UTXO. En el caso de Ethereum, se optó por un modelo basado en cuentas, que es al que estamos acostumbrados en nuestro día a día cuando hacemos operaciones bancarias, es decir, cada cuenta tiene directamente un balance en ether, escrito en la cadena de bloques, de modo que cuando una cuenta es el origen de una transacción, su balance se reduce, y aumenta cuando es el destino.

Contratos inteligentes. Como ya hemos mencionado brevemente, un contrato inteligente es un programa creado por algún usuario de Ethereum y desplegado dentro del sistema, de modo que cada contrato tiene asociada una dirección única. Desplegar un contrato no es más que "empaquetar" el código del programa dentro de una transacción de Ethereum, formateada para indicar que se quiere crear un nuevo contrato en el sistema. Los contratos inteligentes de Ethereum están compuestos por:

- Variables globales: son similares a los atributos de un objeto en programación orientada a objetos y se pueden utilizar a modo de "memoria persistente" del contrato.
- Funciones públicas: son funciones ejecutables desde el exterior, es decir, por parte de usuarios o de otros contratos inteligentes. Por similitud con lenguajes de programación orientada a objetos, es normal referirse a las funciones como métodos.
- Funciones internas o restringidas: son funciones ejecutables únicamente por parte de otras funciones del contrato.

Análogamente a las funciones públicas y por similitud con lenguajes de programación orientada a objetos, es habitual referirse a ellas como métodos.

- Balance propio. Dado que un contrato puede ejecutar transacciones (incluyendo la creación de nuevos contratos y llamadas a otros contratos), también debe ser capaz de realizar pagos. Por ello, los contratos inteligentes en Ethereum tienen un balance asociado.

En cierto modo, un contrato inteligente en Ethereum puede verse como una cuenta o incluso como un usuario del sistema. No obstante, la dirección de un contrato inteligente, al contrario que la dirección de la cuenta de un usuario, no se deriva de la clave pública de un par asimétrico de claves. En su caso, la dirección del contrato se deriva de varios valores públicos obtenidos en el momento de desplegar el contrato. Ya hemos mencionado que un usuario puede generar tantas cuentas como quiera; un contrato, no obstante, no puede generar cuentas (sí otros contratos) ni puede tener varias direcciones. De hecho, la forma de calcular la dirección de los contratos procede de una propiedad inherente de los mismos: no tienen almacenamiento privado. Esto es así porque tanto su código como sus variables son almacenadas "en claro" en la cadena de bloques. Por lo tanto, todos los nodos de la red tienen acceso a su información. De esta manera, si la dirección de un contrato se calculara basándose en una clave pública, todas las transacciones que emitiese ese contrato deberían firmarse con la clave privada correspondiente. Esto hace que las cuentas de usuario se suelan llamar también Externally Owned Account o EOA (algo así como "cuenta de propietario exterior"). Más adelante veremos por qué el resultado de una transacción que se escribe en la cadena de bloques es correcto.

EVM. Un componente fundamental, y una de las principales innovaciones de Ethereum, es lo que se conoce como máquina virtual de Ethereum o EVM (Ethereum Virtual Machine). La EVM define un conjunto de instrucciones fundamentales en un lenguaje de programación, tipo ensamblador, denominado Solidity, al que se compilan los contratos inteligentes de alto nivel.

Además, proporciona los mecanismos para ejecutar de forma unívoca estas instrucciones en ensamblador. Esto garantiza que, partiendo del mismo estado inicial (que se obtiene del último bloque de la cadena de bloques de Ethereum), todos los nodos del sistema producirán el mismo resultado. Dicho de otro modo, la EVM se encarga de evitar que el resultado obtenido por los nodos del sistema sea diferente, dado que estos utilizan entornos de procesamiento distintos, aunque todos ellos empleen el mismo estado inicial. Es decir, garantiza el "determinismo" del sistema. Pero además, tiene otro papel fundamental: ya hemos comentado que, en Ethereum, cada operación tiene un coste asociado, donde "operación" en realidad significa instrucción de código máquina (por ejemplo, una instrucción de salto tiene un coste de 1 unidad de gas). Esto permite calcular, en tiempo de ejecución, el coste real, en unidades de gas, que supone ejecutar un método de un contrato. Más aún, gracias al determinismo de la EVM, este coste será igual para todos.

Consenso. El mecanismo de consenso en Ethereum es de tipo PoW, al igual que en Bitcoin. Es decir, hay que dedicar esfuerzo computacional a una tarea predefinida. Recordemos que en el caso de consenso por prueba de trabajo, a los nodos que se dedican a hacer estos cómputos se los conoce como mineros. En el caso de Ethereum, los mineros utilizan un algoritmo llamado Ethash, diseñado para ser más resistente a optimizaciones *hardware*. Es decir, hace que sea más complicado y costoso crear *hardware* específico para esa tarea, lo cual ha provocado la creación de granjas de minado que suponen un riesgo de recentralización. Brevemente, Ethash sigue los siguientes pasos:

- Crea una semilla aleatoria utilizando la función resumen Keccak-256 (véase capítulo 1) sobre valores derivados de los bloques previos de la cadena.
- Crea una memoria de 16 MB utilizando la función resumen Keccak-512 sobre valores derivados de la semilla aleatoria previa.
- Crea un conjunto de datos aleatorios de 1 GB utilizando la función resumen Keccak-512 sobre valores derivados de la caché de 16 MB.

Este proceso se repite aproximadamente cada 30.000 bloques. El cálculo de una memoria de 16 MB, además del conjunto de datos final de 1 GB, permite que clientes con *software* de pocos recursos puedan verificar los nuevos bloques sin necesidad de dedicar 1 GB de almacenamiento. La resistencia a optimizaciones *hardware* se debe a que el algoritmo es muy intenso en cuanto a operaciones de lectura y escritura en memoria RAM, tarea que es difícil de optimizar en *hardware*, al contrario que el cálculo de una función resumen, que consume pocos recursos.

No obstante, Ethereum está trabajando para migrar a un consenso del tipo prueba de interés o PoS (*Proof of Stake*). Este tipo de protocolos de consenso, más recientes que los de prueba de trabajo, se basan en la asunción de que quienes hayan depositado más "interés" en el sistema (típicamente en forma de inversión económica directa o indirecta), probablemente tengan más motivos para querer garantizar su correcto funcionamiento y, por lo tanto, se puede confiar más en sus decisiones. Este "interés" suele traducirse de una forma u otra en la criptomoneda interna del sistema blockchain. El protocolo al que Ethereum migrará se conoce como Casper. Esta migración tendrá lugar en la versión conocida como Serenity, aún sin fecha pública de lanzamiento.

Otras propiedades y conceptos importantes

Ethereum tiene propiedades importantes que, aunque pueden deducirse de las explicaciones anteriores, conviene comentar explícitamente, de forma breve.

Por un lado, público y no permisionado. Todas las transacciones que tienen lugar en Ethereum, al igual que las de Bitcoin, son accesibles públicamente por cualquiera que se instale un nodo, es decir, son cadenas de bloques públicas. De la misma manera, cualquiera puede instalarse un nodo en Ethereum y llevar a cabo cualquier tarea; es decir, es un sistema que no requiere permisos (no permisionado).

En segundo lugar, seudoanonimato. Ethereum proporciona lo que se conoce como seudoanonimato. Cuando un usuario quiere operar en Ethereum, debe crearse un par asimétrico de

claves pública-privada. La clave pública, o un valor derivado de la misma, funciona a modo de identificador para realizar operaciones. Por supuesto, se pueden crear tantos pares de claves como se quiera, pero toda operación llevada a cabo por una misma clave será trazable por dicha clave. Esto es lo que hace que sea un sistema seudoanónimo en lugar de anónimo.

En tercer lugar, encontramos el lenguaje de programación de contratos inteligentes. Ya hemos comentado que el lenguaje de programación de Bitcoin, utilizado para definir qué condiciones deben cumplirse para que una transacción sea válida, es de bajo nivel y relativamente limitado. Ethereum ha tenido varios lenguajes de programación de contratos inteligentes (LLL, Serpent y Solidity). El utilizado actualmente, Solidity, es un lenguaje de alto nivel bastante similar a los lenguajes tradicionales de programación orientada a objetos. De hecho, se suele decir que Solidity es un lenguaje Turing completo, lo que quiere decir que puede calcular cualquier función computable. Aunque, siendo estrictos, esto no es completamente cierto, ya que un contrato inteligente en Ethereum, implementado en Solidity, no puede ejecutarse de manera indefinida debido al límite de gas que se define de antemano.

Por último, el almacenamiento. También hemos comentado que los contratos inteligentes pueden tener variables globales, que son persistentes. Es decir, su valor no se restablece tras acabar la ejecución del contrato inteligente. Esto implica que debe escribirse en la cadena de bloques, lo cual significa que la propia cadena puede usarse para almacenar información. No obstante, también hemos comentado que hay que pagar por cada byte utilizado y, precisamente, el almacenamiento es uno de los recursos más caros en Ethereum, así que suele ser conveniente mantenerlo en el mínimo imprescindible.

Ejecución de un contrato inteligente en Ethereum

Conocidos los principales conceptos relacionados con Ethereum, presentaremos el ciclo de vida de un contrato inteligente desde su creación hasta su (posible) destrucción, pasando por las ejecuciones de sus funciones.

Supongamos que ya existen n cuentas de usuario, cada una con su par de claves asimétricas, y que todos los usuarios conocen las direcciones de los demás. Sin entrar en detalles de código fuente para facilitar la comprensión, trataremos simplemente con un contrato inteligente que, al crearse en la red, permite inicializar una variable interna con un valor entero arbitrario. Al inicializarse, además, el contrato memoriza la dirección de su creador, a quien nos referiremos como usuario *U*. Posteriormente, solo este creador podrá cambiar el valor de la variable interna, aunque cualquiera podrá consultar cuál es el último valor almacenado. En general, el proceso de creación de un contrato es como sigue:

1. El usuario *U* crea una transacción especificando los valores siguientes: origen, datos, gas máximo, precio (en ether) y número de transacciones del emisor.
2. El usuario *U* firma la transacción utilizando su clave privada. El resultado de esta firma se incluye como parte de la transacción que se envía a la red.
3. El protocolo de comunicaciones P2P subyacente hace que la transacción se difunda a (gran parte de) los nodos de la red. Entre estos nodos, habrá mineros buscando crear nuevos bloques.
4. Los mineros compiten por crear un nuevo bloque, en el que incluyen las transacciones que más incentivos les ofrecen (en forma de importes más altos, como ya vimos). Suponiendo un importe razonable, antes o después algún minero incluirá la transacción enviada por *U* anteriormente. Cada minero comprueba que cada una de las transacciones que quiera incluir en el siguiente bloque es correcta. Cuando algún minero incluya en el bloque que cree la transacción construida en el punto 1 y enviada en el punto 2, será el momento a partir del cual la transacción existirá en la cadena de bloques y será cuando el código se convierta en un contrato inteligente. Se le asignará una dirección única que depende de la dirección del usuario que lo envía y del número de transacciones que este haya enviado hasta el momento.

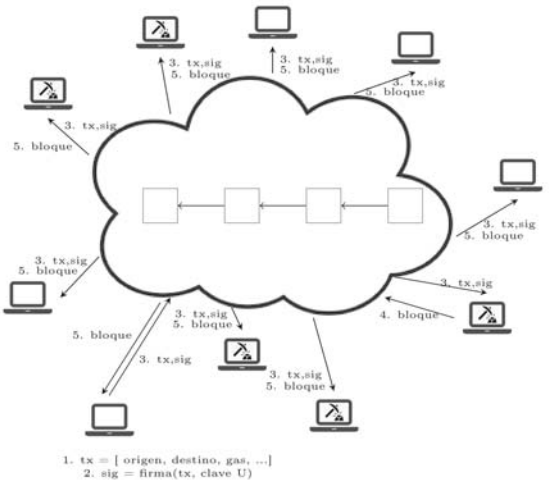
Nótese que cualquier nodo del sistema puede comprobar que la dirección asignada es correcta. Por otra parte, aunque la transacción ya exista en la cadena de bloques, no significa que sea irreversible. De hecho, al igual que en Bitcoin —y que en cualquier sistema blockchain basado en prueba de trabajo— es necesario que se añadan nuevos bloques después del recién creado para ir aumentando la certeza de que no se revertirá la transacción.

5. Cada nodo del sistema que reciba el nuevo bloque verificará que los cálculos que hizo el minero que ha propuesto el bloque son válidos. En caso de que lo sean, aceptará el nuevo bloque.

Llegados a este punto, y suponiendo que se ha producido un número razonable de confirmaciones después del bloque que contiene el contrato inteligente recién creado, cualquier nodo de la red tendrá acceso al mismo. La creación de la transacción, firma digital, bloque y los intercambios de mensajes asociados se indican esquemáticamente en la figura 9.

FIGURA 9

Esquema de la creación de una transacción, firma digital, bloque y los intercambios de mensajes.



Supongamos ahora que el mismo usuario *U* quiere establecer el valor de la variable interna a un valor concreto. El proceso que seguiría sería el siguiente:

1. *U* crea una transacción con la dirección del contrato como dirección de destino y en la que, en el campo *data*, especifica qué método quiere ejecutar dentro del contrato, además del valor al que quiere establecer la variable interna. Por supuesto, también especifica el resto de los valores obligatorios: dirección origen, gas y precio.
2. *U* firma la transacción con su clave privada, envía esta transacción al resto de nodos de la red, incluyendo el resultado de la firma dentro de la propia transacción.
3. El protocolo de comunicaciones P2P hace que la transacción se difunda a los nodos de la red, entre los que habrá mineros deseando crear nuevos bloques.
4. Los mineros compiten por crear un nuevo bloque. Cada minero comprueba que cada una de las transacciones que quiera incluir en el siguiente bloque esté bien formada; en este caso, al tratarse de la ejecución de un método de un contrato inteligente, simula esta ejecución dentro de su EVM. Si, como resultado de esta ejecución no se produce un error o una excepción, el minero actualiza la memoria persistente del contrato para reflejar la ejecución de la transacción. En este ejemplo, una posible excepción sería que quien invoca el método del contrato para establecer el valor de la variable interna no es el usuario *U*. Otra excepción, más genérica, podría ser que el gas máximo especificado no sea suficiente para ejecutar el contrato inteligente. Suponiendo que todo es correcto, en este caso, la ejecución de la transacción implica que la variable interna del contrato creado anteriormente se establece al valor indicado en la invocación anterior. El nuevo bloque calculado por el minero incluye esta transacción (entre otras, potencialmente) y los cambios en el estado de la cadena de bloques.
5. Cada nodo del sistema que reciba el nuevo bloque verificará que los cálculos que hizo el minero que ha propuesto

el bloque son válidos. En caso de que lo sean, aceptará el bloque. Por cada nuevo bloque que extienda la cadena, más difícil será revertir la transacción.

Por último, supongamos que un usuario distinto de *U*, por ejemplo *V*, intenta modificar el valor de la variable interna. *V* seguirá un proceso similar al que se ha explicado para *U*, con la diferencia de que firmará la transacción con una clave privada distinta de la de *U*. Los mineros que reciban dicha transacción, al ejecutar la lógica del contrato inteligente existente en la red, se darán cuenta de que la dirección de *V* no coincide con la de *U*, que había sido apuntado como el propietario del contrato. Habiéndose especificado como parte del código del contrato que solo su propietario puede modificar la variable interna, esto provocará una excepción, y la ejecución del código se detendrá, rechazando la transacción.

Componentes de Ethereum

A grandes rasgos, los apartados anteriores nos permiten hacernos una idea de las propiedades que garantiza Ethereum y de su modo de funcionamiento. Para un mayor detalle, lo más recomendable es consultar el conocido como *yellow paper* (Wood, 2015), en el que se expone el modelo formal de Ethereum.

En lo que sigue, presentaremos cómo se puede interactuar con Ethereum y cuáles son las mejores aplicaciones y herramientas para ello.

Si lo que queremos es crear una cuenta para Ethereum y gestionar ether, probablemente la mejor opción sea lo que se conoce como *wallet* o billetera. Un wallet no es más que una herramienta que permite gestionar las claves criptográficas asociadas unívocamente a cada cuenta (dirección) que se crea. Ya hemos visto que en Ethereum, para demostrar ser el propietario de una cuenta concreta (cuya dirección es una codificación de una clave pública), hay que emitir una firma digital con la clave privada asociada. Por lo tanto, es necesario guardar de forma segura dicha clave privada, para lo que se hace uso de la herramienta

wallet. El nombre viene de que, en última instancia, un wallet permite gestionar dinero en forma de ether y, si se pierde dicho wallet, se pierde el dinero asociado; si lo roban, el ladrón puede hacer uso del dinero. En concreto, un wallet en Ethereum (y en cualquier criptomoneda) no almacena el ether en sí mismo, sino que almacena las claves privadas que permiten hacer uso de dicho dinero. Que haya criptomonedas asociadas a una dirección concreta viene definido por los "apuntes contables" escritos en la cadena de bloques y que están asociados a dicha dirección.

Hay diferentes wallets; entre los más populares están Ethereum-Wallet, de la fundación Ethereum; MetaMask, disponible como *plugin* para los navegadores principales; y MyEtherWallet, disponible como aplicación web y como aplicación de escritorio.

No obstante, han surgido herramientas más avanzadas que un wallet. Sin ir más lejos, el tipo de aplicaciones que se construyen basados en contratos inteligentes en un sistema blockchain se conoce como "aplicaciones descentralizadas" (*dapps*). Son descentralizadas porque no hay un único servidor que decida si se acepta o no una ejecución, sino que todos los nodos de la red deben ponerse de acuerdo.

Las dapps utilizan información de la cadena de bloques y procedimientos de los contratos inteligentes asociados. Por lo tanto, saber leer esa información y operar según la misma es algo que queda fuera del alcance de un wallet, que solo permite emitir firmas digitales. Para ello, Ethereum proporciona navegadores de dapps, que son herramientas que permiten buscar e interactuar con aplicaciones descentralizadas, ejecutando los métodos que definen.

Este ecosistema de dapps se conoce en Ethereum como la *web 3.0*. Para entender el concepto más fácilmente, un navegador de dapps se puede considerar como el equivalente a cualquiera de los navegadores web que solemos emplear al acceder a Internet. Si un navegador web permite encontrar e interactuar con el contenido almacenado en Internet, los navegadores de dapps también ofrecen la funcionalidad de wallet. Los principales navegadores de dapps son Mist, de la fundación Ethereum, y Parity, de la compañía homónima cofundada por uno de los cofundadores de Ethereum.

Tanto los wallets como los navegadores de dapps incluyen nodos para conectarse a la red de Ethereum. Pueden ser "nodos completos", que se descargan una copia completa de la cadena de bloques de Ethereum, o "nodos ligeros", que solo se descargan un subconjunto de la misma (normalmente, las cabeceras de los bloques y de las transacciones que afectan directamente a las cuentas de usuario que almacenan). No obstante, es posible descargarse directamente un nodo que no incorpore funcionalidad de alto nivel para navegar e interactuar con dapps o sin la interfaz gráfica de un wallet. De hecho, las distintas implementaciones de los nodos de Ethereum permiten hacer prácticamente cualquier operación y, por ello, se puede decir que son la "navaja suiza" del ecosistema. El nodo más extendido, *geth*, lleva integrado un wallet y permite desplegar e invocar contratos inteligentes en cualquier red de Ethereum, además de explorar la cadena de bloques desde una línea de comandos.

Por último, hay que destacar que, utilizando los nodos de Ethereum que acabamos de comentar, es posible crear redes "privadas" de Ethereum, en el sentido de que pueden hacerse accesibles únicamente a los ordenadores que se configuren para conectarse a ellas (cabe tener en cuenta que "securizar" una red de este estilo puede ser muy complejo). Sin embargo, es importante tener en cuenta que, mientras Ethereum siga teniendo un algoritmo de consenso de tipo PoW, cualquier red privada de Ethereum con pocos nodos puede ser bastante vulnerable a ataques.

Más allá de contratos inteligentes

Para finalizar este capítulo, es importante hacer una breve mención a la visión de futuro de Ethereum. Hasta ahora, hemos hablado principalmente de los principales actores del sistema, de los principales aspectos criptográficos y hemos comentado los principios que están detrás de los contratos inteligentes.

No obstante, la visión de Ethereum va más allá. En realidad, Ethereum se compara con un ordenador global al que todo el mundo tiene acceso. Los contratos inteligentes, ejecutados de forma descentralizada a través de la mencionada EVM, representan

la capacidad de cómputo de dicho ordenador global. Pero para poder considerarse un ordenador completo, hay otros aspectos que necesitan ser cubiertos.

En concreto, es necesario proveer de un sistema de almacenamiento persistente escalable. Aunque ya hemos visto que los contratos inteligentes pueden almacenar información, esta información se replica en todos los nodos completos de la red, es decir, los que guardan una copia total del sistema. Este aspecto no es escalable dado que no basta con añadir un nuevo nodo para aumentar la capacidad total de almacenamiento. Por ello, cada byte de almacenamiento persistente en la cadena de bloques es muy costoso. La visión de futuro de Ethereum es incorporar un sistema de almacenamiento distribuido, criptográficamente protegido, que haga más eficiente este almacenamiento persistente. El subsistema del ordenador global de Ethereum que intenta resolver este aspecto se conoce como Swarm. Cabe destacar que, aparte de Ethereum, hay sistemas que hacen de la tarea de crear un sistema seguro de almacenamiento distribuido su objetivo principal, como el sistema de ficheros interplanetario o IPFS (InterPlanetary File System).

Por otro lado, también es necesario incorporar sistemas eficientes de comunicaciones de información que no necesariamente deba ser escrita en la cadena de bloques. Por ejemplo, dos dapps, como parte de su funcionamiento normal, podrían necesitar intercambiar información antes de hacer una transacción para saber qué direcciones de usuario utilizar al ejecutar una transacción que vendrá a continuación. Utilizar la cadena de bloques para este intercambio sería contraproducente y, de hecho, como hemos mencionado en el párrafo anterior, bastante caro. Por eso, en Ethereum se está trabajando para incorporar mecanismos que permitan realizar este tipo de comunicaciones. El subsistema que se encargaría de esta funcionalidad se conoce como Whisper.

CAPÍTULO 4

Modelos alternativos de blockchain: otros esquemas de consenso y autorización en blockchain

En los capítulos 2 y 3 hemos analizado las bases de las plataformas Bitcoin y Ethereum. La primera plataforma surgió con el objetivo de crear un nuevo tipo de moneda sin necesidad de contar con bancos centrales. Como efecto paralelo de la creación de la criptomoneda bitcoin, la propuesta de Satoshi Nakamoto propició todo un conjunto de proyectos orientados a aprovechar las características de la estructura de datos en la que se apoya la plataforma Bitcoin. Así, la inmutabilidad de la blockchain de Bitcoin ha venido siendo interpretada como un recurso de gran potencial para entornos en los que se necesita registrar el estado de mercancías, de artículos y de todo tipo de productos. Con Ethereum, esas virtudes de la blockchain se ensanchan mediante la introducción de mecanismos para automatizar la ejecución de acciones en función del estado registrado en la blockchain. Ahora bien, tanto en el caso de Bitcoin como en el de Ethereum el registro de información en sus blockchains es un proceso con un coste computacional muy elevado. En efecto, en esas plataformas se asume que los agentes encargados de escribir la información no son confiables y, por ello, la inclusión de nuevos datos se realiza tras alcanzar un acuerdo explícito entre todos esos agentes. La PoW o prueba de trabajo es el procedimiento escogido en Bitcoin y Ethereum para hacer explícito dicho acuerdo y, tal y como hemos indicado en el capítulo 2, requiere un consumo energético muy elevado por parte de los agentes de escritura.

En este capítulo vamos a analizar modelos de blockchain en los que existe cierta confianza respecto a la actividad de los agentes de registro, lo que habilita el despliegue de esquemas de consenso menos exigentes que la PoW en términos computacionales. El desarrollo de estos esquemas ha dado origen a propuestas como Hyperledger, Corda y Ripple que son de alto interés en el ámbito empresarial a la hora de optimizar la consolidación y transferencia de información en consorcios y organismos. A diferencia de lo que ocurre en las blockchains públicas de Bitcoin y Ethereum, en estos modelos alternativos de blockchain se privilegia la eficiencia a costa de admitir el papel de los agentes de escritura como intermediarios de confianza. En las siguientes secciones se introducirán alternativas a las blockchains públicas. Las diversas variantes conservan la estructura de datos de las blockchains de Bitcoin y Ethereum, pero plantean modificaciones en el algoritmo de consenso y/o en la gestión de permisos de lectura y escritura en la blockchain.

El problema de los generales bizantinos y las dificultades del consenso en sistemas asíncronos

En los capítulos previos hemos visto que los ecosistemas blockchain están construidos sobre un sistema distribuido de almacenamiento y una red de comunicación descentralizada. Tanto en Bitcoin como en Ethereum, la escritura de información puede realizarla cualquier usuario que instale el *software* correspondiente y ponga su ordenador o nodo a trabajar en el minado de nuevos bloques. Este proceso de minado lleva asociada una competencia entre los nodos mineros que se resuelve de modo no determinístico, lo que impide que se sepa de antemano qué nodo será el encargado de escribir el siguiente bloque de la cadena. Por este motivo se dice que el minado en blockchain se asemeja a una lotería entre los mineros (Narayanan *et al.*, 2016), característica que constituye la principal defensa frente a nodos que actúen de modo malicioso.

Desde un punto de vista general, la inclusión de transacciones en los bloques por parte de los mineros responde a la

replicación de estado en sistemas distribuidos. Dado que no existe un nodo central que ordene los bloques de datos, ha de existir una coordinación entre los nodos de forma que la transición de estados se realice de forma consistente. Si estamos hablando de criptomoneda, la transición de estado se refiere a la actualización del balance financiero tras nuevas transacciones. Paxos es uno de los primeros sistemas en los que se considera la replicación de estado (Lamport, 1998) en una red con canales de comunicación no confiables y con una minoría de nodos que envían mensajes desactualizados tras perder la conexión con la red temporal o definitivamente. Como extensión de Paxos, aparecieron modelos de consenso en sistemas distribuidos cuyos nodos pueden comportarse de modo errático, debido a un fallo natural o intencionado. Este tipo de comportamiento se conoce como bizantino, de acuerdo con el modelado que Lamport propuso del consenso en sistemas distribuidos usando como analogía el problema de los generales bizantinos (Lamport *et al.*, 1982).

En el "problema de los generales bizantinos" tres o más generales están sitiando una ciudad enemiga y deben consensuar si atacan o se retiran, teniendo en cuenta que si no llegan a un acuerdo serán derrotados. Los generales se comunican a través de mensajeros, siendo conscientes de que los mensajes pueden ser interceptados por el enemigo, manipulados por los mensajeros o incluir información falsa introducida intencionadamente por generales traidores. Lamport y sus colaboradores demuestran que se puede alcanzar consenso siempre que no haya más de un tercio de generales traidores.

Los sistemas de consenso basados en el modelo introducido por Lamport y colaboradores se conocen como "sistemas bizantinos tolerantes a fallos" o sistemas BFT (Byzantine Fault Tolerance). En 1999, Miguel Castro y Barbara Liskov (Castro y Liskov, 1999) extendieron el esquema BFT adecuándolo a contextos con redes de comunicación no confiables y creando los modelos PBFT (Practical Byzantine Fault Tolerance). Paxos, BFT y PBFT han dado lugar a un gran número de estudios teóricos en los últimos años y, de hecho, constituyen la base formal más sólida del problema del consenso en

sistemas blockchain. Ahora bien, la implementación de estos esquemas requiere un mayor control sobre la identidad de los nodos de la red habilitados para realizar inserciones y validaciones de bloques.

En el caso del algoritmo de consenso utilizado en Bitcoin, no existe un enfoque tan sólido como el anterior. Así, el consenso de Nakamoto se ha mostrado efectivo en la práctica, si bien no han podido demostrarse de forma teórica las condiciones específicas en las que el sistema es resiliente frente a un volumen considerable de mineros maliciosos. Ataques como los destacados al final del capítulo son indicios sobre el marco de restricciones en las que es aplicable la PoW, pero todavía queda mucho trabajo por realizar. En cualquier caso, frente a BFT y PBFT, la PoW sí que consigue dar cabida a validadores anónimos de bloque, es decir, sí proporciona una respuesta efectiva al despliegue de un sistema totalmente descentralizado para la gestión del consenso.

En el capítulo 2 hemos subrayado que la PoW erosiona la sostenibilidad energética del ecosistema Bitcoin, algo que también es aplicable al modelo actual de consenso de Ethereum[23], además de implicar problemas de gran calado en lo concerniente a la gobernanza de la blockchain. Como reacción a estos dos inconvenientes, ha surgido una pléyade de modelos de consenso distribuido. Dada la limitación de espacio de la presente obra, a continuación resumimos algunas de las principales propuestas e implementaciones en este campo:

- PoS (*Proof of Stake*). En este caso, en el proceso de aceptación de bloque, se da preferencia a aquellos nodos que tienen en su posesión un mayor número de criptomonedas. Así, en lugar de solicitar a los validadores que son capaces de realizar una operación de alto coste computacional, lo que se les exige es que muestren que tienen en su poder un volumen suficiente de criptomonedas. Este esquema asume que los nodos que tienen una mayor

23. Ethereum, en el momento actual, también está siendo construido utilizando minado de bloques, si bien su modelo de consenso es una variante del algoritmo de Nakamoto (Lewenberg *et al*., 2015).

cantidad de criptomonedas tienen especial interés en que la plataforma blockchain siga operativa y, por ello, acceden a resolver el desafío matemático asociado a la aceptación de nuevos bloques. PeerCoin es un ejemplo de implementación de PoS. Dentro de las propuestas más prometedoras de PoS cabe destacar el caso de Ouroboros (Kiayias *et al.*, 2017).

- DPoS (*Delegated Proof of Stake*). La PoS plantea ciertas dudas desde el punto de vista ético, así como desde el punto de vista de la gobernanza, dado el papel privilegiado de aquellos nodos que hacen acopio de criptomoneda. Para superar este tipo de limitaciones y como una iniciativa de descentralización de PoS, surge DPoS. En este nuevo modelo de consenso los usuarios de la plataforma eligen delegados y testigos encargados de escribir y verificar nuevos bloques. El sistema de elección de estos intermediarios se realiza por voto electrónico o a través de reuniones en las que los diversos candidatos a delegados y testigos presentan su historial como aval de su actividad. Algunas de las iniciativas blockchain que emplean DPoS son Bitshares, Steem y EOS.
- PoA (*Proof of Activity*). Este modelo combina la PoW y la PoS. En la PoA el proceso de minado comienza con una PoW con varios mineros compitiendo por conseguir que su bloque sea el que finalmente quede consolidado en la blockchain. Cuando se encuentra un nuevo bloque, el sistema comienza a utilizar PoS y el nuevo bloque solo incluirá una cabecera y la dirección del minero que obtendrá la recompensa. Esa cabecera será posteriormente utilizada para seleccionar de forma aleatoria un nuevo grupo de validadores de entre los diversos nodos mineros de la red blockchain. Esta selección no es puramente aleatoria, ya que existe un sesgo en la selección, de forma que la probabilidad de ser seleccionado como validador aumenta con la cantidad de criptomoneda que se posea. Un ejemplo de plataforma que hace uso de PoA es Decred.

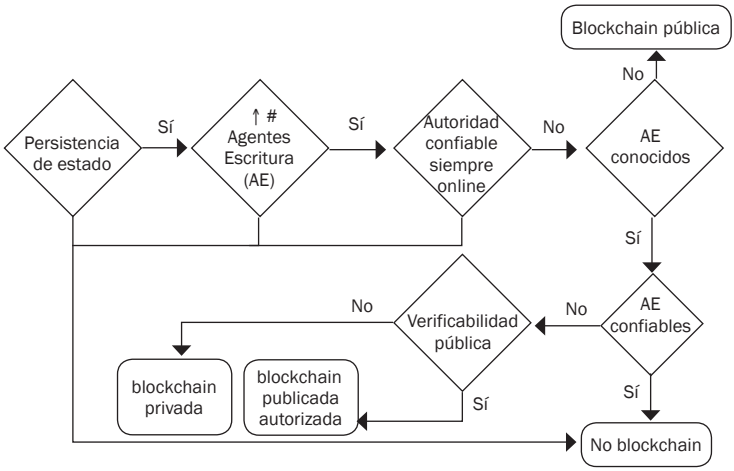
Modelos de control de acceso para la blockchain: blockchain públicas, permisionadas y privadas

Wüst y Gervais (2017) señalan que a la hora de emprender un proyecto blockchain la primera pregunta que hemos de plantearnos es si la estructura y características que ofrece blockchain son mejores que las que nos proporcionan las bases de datos tradicionales. Tal y como se puede observar en la figura 10, la respuesta a esta pregunta está fundamentalmente condicionada por el modelo de control de acceso que queremos para nuestros datos. El modelo requiere la concreción del tipo de agentes de escritura que van a tener capacidad de insertar información en el sistema de persistencia de información, así como la disponibilidad que debe tener el mecanismo de consolidación de información.

El análisis anterior lleva a identificar contextos en los que las blockchains no proporcionan una ventaja competitiva respecto a las bases de datos. Tal es el caso de aquellas situaciones en las que están claramente identificados los agentes que realizan la escritura de información que, además, son a todos los efectos interpretados por los usuarios como terceras partes confiables. En otros casos, el volumen de agentes de escritura existente, así como la necesidad de tener siempre disponible el medio de consolidación de información, lleva a considerar como conveniente la adopción de alguna suerte de blockchain. En este supuesto, la selección del modelo de blockchain va a depender de la naturaleza de los agentes de escritura y de la capacidad para acceder a la información que han de tener los usuarios del sistema blockchain. Si los agentes de escritura, que en este contexto no son confiables, no son conocidos, entonces habremos de optar por una blockchain pública como Bitcoin o Ethereum. Por el contrario, si los agentes de escritura son conocidos, entonces utilizaremos una blockchain permisionada o autorizada (como es el caso de Ripple o Hyperledger Fabric) si se desea que la verificación de la información se pueda realizar públicamente, o una blockchain privada (por ejemplo, Corda) si no se precisa verificabilidad pública.

Figura 10

Metodología para la elección del modelo de blockchain.



Fuente: Adaptada de la figura 1 de Wüst y Gervais (2017).

Blockchain privadas: Corda

Como ejemplo de blockchain privada vamos a considerar Corda. En 2014 se creaba R3[24], compañía que posteriormente lideraría un consorcio internacional de empresas —muchas de ellas financieras— para fomentar el uso de tecnología blockchain en este ecosistema. Como resultado se diseñó Corda (Brown *et al.*, 2016), un sistema que, pese a las reticencias de un grueso importante de la comunidad blockchain, es muy similar en su naturaleza a todos los sistemas que hemos visto. De hecho, de este esfuerzo de R3 por distanciar Corda de los sistemas blockchain tradicionales surgió el término DLT que, como mencionamos en la introducción, puede ser interpretada como una generalización de los sistemas blockchain.

Para justificar que Corda no es un sistema blockchain estrictamente hablando, bastaría con considerar que no existe el concepto de bloque. De hecho, en Corda, las actualizaciones

24. Véase https://www.r3.com/ (último acceso 11/03/2019).

sobre el estado global se hacen directamente al nivel de transacciones.

Corda se define como un sistema DLT semiprivado (Hearn, 2016), en el sentido de que solo pueden unirse a la red y operar en ella quienes cuenten con una identidad validada. Además, es permisionado ya que, como veremos, hay nodos que tienen roles especiales, no pudiendo ser asumidos por todos. El foco principal de Corda son las aplicaciones financieras. Por ello, aunque permiten contratos inteligentes, en principio se limitan a funcionalidades financieras.

Corda ha sido también un sistema bastante pionero en muchos aspectos. Por ejemplo, en Corda no se comparten todas las transacciones con todos los nodos. En concreto, al crear un contrato inteligente, se establece quiénes van a ser los validadores de dicho contrato, es decir, quiénes se van a encargar de votar a favor o en contra de la aceptación de transacciones realizadas dentro de este contrato. Esto aumenta la agilidad en la validación de transacciones, ya que el resto de nodos son totalmente indiferentes al contrato en cuestión. Esto se asemeja bastante a los productos financieros: solamente las partes implicadas toman parte en la validez o no de un producto. De hecho, esto facilita la gestión de conflictos: en caso de que haya un contrato con un histórico conflictivo, las partes afectadas pueden ponerse de acuerdo en eliminar este histórico, lo cual es mucho más sencillo que tener que poner de acuerdo a toda la red.

Por otro lado, sí existe un conjunto de nodos que deben encargarse de vigilar que no se utiliza una misma transacción más de una vez. Este es el conocido como "servicio de unicidad" (*uniqueness service*). Los nodos que forman parte de este servicio se conocen como "notarios", que además componen un servicio común a toda la red. El mínimo requisito que se impone a estos notarios es comprobar que una transacción no se ha utilizado anteriormente, sin necesidad de comprobar sus valores internos. De hecho, en este caso, es suficiente con enviar a estos notarios algo que les permita comprobar que la transacción es única. En concreto, se les envía la raíz del árbol de Merkle que se obtiene al agrupar los campos que forman la transacción de acuerdo con este tipo de árbol. De esta manera, los notarios pueden verificar que no se ha generado y utilizado

previamente una transacción igual. Además, también aportan propiedades de privacidad, ya que no hay necesidad de que exista un único subconjunto de nodos que tenga acceso a toda la información de la red. Este tipo de notarios se conoce como "notarios no validadores". No obstante, el hecho de que no validen los datos internos de cada transacción podría llegar a utilizarse en algún ataque. Por ello, existe lo que se conoce como "notarios validadores" que, además, deben tener acceso a los campos de la transacción y validarlos. La ventaja de los notarios validadores es que evitan determinados ataques, aunque la desventaja es que tales notarios sí tienen acceso a todos los datos de todas las transacciones de la red, degradando la privacidad total del sistema.

En cualquier caso, los notarios se ponen de acuerdo entre sí sobre la unicidad de una transacción utilizando algoritmos de consenso. Corda no obliga a utilizar uno concreto, aunque los más utilizados son los sistemas de consenso bizantino. En cuanto al modelo transaccional, Corda sigue el modelo UTXO de Bitcoin, que ya estudiamos en el capítulo 2. Es decir, cada transacción debe recibir una o más entradas y producir una o más salidas. Posteriormente, tales salidas podrán ser utilizadas como entradas para otras transacciones. En cuanto a la implementación de los contratos inteligentes, Corda se basa en la máquina virtual de Java, que restringe la posibilidad de conseguir un entorno determinista.

Otro aspecto diferenciador de Corda, derivado también de su foco en el ecosistema financiero, es el hecho de que cada contrato lleva asociado el código fuente que especifica las operaciones y condiciones que se deben cumplir y ejecutar. No obstante, también, incluye un texto en forma de prosa legal que especifica lo que se espera de dicho contrato. Aunque no es un campo obligatorio, sí se hace énfasis en que se espera que tal campo exista siempre, ya que en caso de disputa entre los participantes del contrato, toma precedencia el texto legal sobre el código del contrato.

Corda también introduce de forma nativa un componente que en otros sistemas tardó más en tenerse en cuenta. Nos referimos a los "oráculos". Este componente, que veremos con más calma en el siguiente capítulo, permite introducir información

del mundo exterior dentro del sistema blockchain (en este caso, DLT). Por ejemplo, el valor de una acción en un instante temporal concreto. Para ello, en Corda es posible adjuntar esta información, proveniente de un oráculo confiable, junto con una firma digital de la información, firmada por el oráculo. De esta forma, cualquier transacción que dependa de un valor que requiera información del exterior puede hacer referencia a esta información adjunta.

En resumidas cuentas, Corda, sin ser un sistema blockchain en sentido estricto, está fuertemente inspirado por estos sistemas. De hecho, combina aspectos de varios de ellos, reduciendo su caso de uso al contexto financiero y, de este modo, enfocando sus principios de diseño para ser más eficiente en este entorno.

Principales propuestas de blockchain permisionadas

Las blockchains permisionadas han recibido especial atención por parte de empresas y asociaciones de empresas. Estas posibilitan introducir ciertos controles adicionales que hacen que sea más fácil compatibilizar las propiedades de resiliencia y trazabilidad que proporcionan las blockchains públicas, con los casos de uso más restrictivos del mundo corporativo.

Estos sistemas se sitúan, pues, en un punto intermedio entre el extremo de los sistemas descentralizados sin un único punto de fallo, y el opuesto de los sistemas centralizados tradicionales donde una única autoridad lo decide todo. Consecuentemente, no están exentos de crítica por parte de ambos mundos, con frecuencia con matices éticos y con eco en cuestiones de gobernanza corporativa. Al margen de posibles prejuicios y reservas, parece oportuno reconocer el mérito de estos sistemas en su intento de incorporar nuevas arquitecturas tecnológicas en entornos adscritos a regulaciones estrictas, y con las que los sistemas de difícil regulación (como son las blockchains públicas) son directamente incompatibles.

A continuación se comentarán con más detalle las dos blockchains permisionadas más extendidas: Hyperledger y Ripple.

Hyperledger

Más que un sistema blockchain concreto, Hyperledger es un consorcio de organizaciones que se han unido para fomentar el desarrollo de la tecnología blockchain para entornos corporativos a través de código abierto[25]. A marzo de 2019, cuentan con diez proyectos, enfocados en distintos aspectos o aportando aspectos diferenciales con respecto al resto. Por ejemplo, cuentan con un conjunto de herramientas para facilitar el despliegue y gestión de sistemas blockchain (el proyecto Cello), o un explorador de cadenas de bloques (el proyecto Explorer).

No obstante, el proyecto bandera de Hyperledger, iniciado por IBM con el nombre de Openblockchain, y posteriormente cedido al consorcio, es Fabric: un sistema blockchain privado y permisionado que pretende ser altamente configurable, de manera que se puedan utilizar diferentes algoritmos de consenso, diferentes modos de gestión de identidad, etc., con el fin de poder adaptarse a la mayor cantidad de escenarios posibles.

En efecto, Hyperledger Fabric o, simplemente, Fabric, está basada en un diseño modular que permite diferenciar las fases de replicación de estado. Para ello, Fabric abandona la arquitectura "ordenación-ejecución" de otras blockchains permisionadas como Tendermint[26], Chain[27] o Quorum[28], de modo que estará basada en un esquema "ejecución-ordenación-validación" (Androulakis *et al.*, 2018a). De acuerdo con sus diseñadores, este cambio de arquitectura permite mejorar el rendimiento de la blockchain al mismo tiempo que habilita una mayor protección frente a ataques por denegación de servicio y una gestión más eficiente de la confidencialidad.

La incorporación de una nueva capa de abstracción en Fabric está asociada a la identificación de distintos nodos de acuerdo con funcionalidades específicas en las etapas de consenso y de replicación de estado de una blockchain. En concreto, en Fabric se distinguen cuatro tipos de nodos:

25. Véase https://www.hyperledger.org/ (último acceso 11/03/2019).
26. Véase https://tendermint.com/ (último acceso 11/03/2019).
27. Véase https://chain.com/ (último acceso 11/03/2019).
28. Véase https://www.jpmorgan.com/global/Quorum (último acceso 11/03/2019).

- Clientes que se encargan de enviar propuestas de transacciones, y que también colaboran en las tareas de ejecución de transacciones y de envío por difusión (*broadcast)* de transacciones para su ordenamiento.
- Nodos ejecutivos (*peers*) que se encargan de ejecutar y validar transacciones, además de almacenar el registro completo de transacciones. Esto es, existirá una réplica del registro de transacciones en cada uno de los nodos ejecutivos.
- Nodos verificadores (*endorsers)* que son un subconjunto de los nodos ejecutivos que tienen por misión ejecutar las propuestas de transacciones. El conjunto de nodos verificadores de una transacción está dictaminado por el contrato inteligente correspondiente, que en el caso de Fabric se denomina *chaincode*.
- Nodos de ordenación (*orderers*) que tienen por objetivo establecer el orden correcto de transacciones en Fabric, lo cual involucra la identificación de las transacciones que implican un cambio de estado, de las dependencias que han sido procesadas durante la fase de ejecución y de las firmas digitales de los nodos verificadores.

La asignación de roles en Fabric depende de una entidad central que actúa como autoridad de certificación de acuerdo con lo establecido en el estándar X.509 (Cooper *et al.*, 2008). Esta entidad se denomina MSP (Membership Service Provider) y es la piedra central de la infraestructura de clave pública asociada a una blockchain de tipo Fabric. Asimismo, las versiones actuales de Fabric incorporan la funcionalidad necesaria para que la MSP pueda generar y administrar identidades seudoanónimas[29], lo que supone una contribución relevante en términos de protección de la privacidad de los usuarios y entidades gestionadas por la correspondiente blockchain.

Tal y como se ha reseñado anteriormente, la asignación de roles en Fabric está asociada a la realización de las diversas tareas

29. Véase https://hyperledger-fabric.readthedocs.io/en/release-1.4/idemix.html (último acceso 11/03/2019).

de las tres capas lógicas del protocolo de consenso y de replicación de estado de la blockchain. Así, durante la fase de ejecución, los clientes firman sus propuestas de transacción y, en virtud del *chaincode* correspondiente, las envían a uno o más nodos verificadores. Cada nodo verificador ejecuta la transacción en un entorno virtual que está aislado del entorno normal de ejecución del resto de la red. Es importante tener en cuenta que la ejecución de la transacción depende del flujo de operación definido en el *chaincode* y del estado del registro de transacciones, y que el resultado de la ejecución no modificará el estado del registro de transacciones. En la medida que el resultado de la ejecución tiene que modificar el estado de la blockchain en algún momento, dicho resultado ha de almacenarse de modo temporal. Esto lo lleva a cabo una nueva entidad, la PTM (Peer Transaction Manager), que crea una estructura llamada *writeset* con la actualización de estados, y una segunda estructura denominada *readset* que contiene las dependencias de la transacción que ha sido ejecutada en el entorno de simulación. Tras la simulación, el nodo verificador devuelve al cliente este par de estructuras y un conjunto de metadatos, todo ello como parte de un paquete de respuesta que está firmado digitalmente.

Para que se inicie la siguiente fase es necesario que el cliente tenga un número mínimo (que está definido en el *chaincode*) de respuestas firmadas y congruentes entre sí (esto es, con el mismo conjunto de valores en *writeset* y *readset*). Una vez se ha satisfecho esta condición, el cliente crea una nueva transacción y la envía por difusión a los nodos de ordenación. Estos nodos se encargan de recibir transacciones y agruparlas en bloques, de forma que la consolidación de un bloque requerirá alcanzar un consenso de forma distribuida. En Fabric, el protocolo de consenso del servicio de ordenación no está definido de modo cerrado, sino que se deja abierta la posibilidad de incorporar nuevas implementaciones de acuerdo con la interfaz de programación definida en la especificación de Fabric. No obstante, por defecto, Fabric trabaja con la implementación de ZooKeeper de Kafka[30].

30. Kafka es un producto *software* de la Apache Software Foundation para la gestión de mensajes de protocolos de red. El protocolo de sincronización que emplea Kafka es ZooKeeper.

Aquí conviene tener en cuenta que los nodos de Fabric son nodos autenticados, lo que abre el camino para utilizar variantes de BFT como base del servicio de ordenación de transacciones[31].

Una vez se ha construido un bloque en la fase de ordenación, los nodos ejecutivos inician la verificación de cada una de las transacciones. Este proceso de verificación se realiza de modo paralelo para cada una de las transacciones del bloque, y comprende la comparación de los datos de tipo *readset* con el estado en el registro de la blockchain, así como la actualización del estado de dicho registro en función de los valores incluidos en los datos de tipo *writeset*. En este punto es importante destacar que Fabric almacena todas las transacciones, incluidas aquellas que fueran rechazadas durante el proceso de verificación. Esto es de vital importancia para el proceso de auditoría, ya que Fabric permite identificar de modo biunívoco el nodo que ha originado una cierta transacción.

Sin duda, una de las aportaciones más relevantes de Fabric a la tecnología blockchain viene dada por su capacidad para efectuar trazabilidad de operaciones y atribución de responsabilidad. Ahora bien, también hemos de recalcar las ventajas que Fabric ofrece en lo concerniente a la escalabilidad y la protección de la confidencialidad gracias a la inclusión de una nueva capa de abstracción de su modelo de datos, los denominados canales (*channels*). En una red de tipo Fabric, es posible tener múltiples blockchains conectadas a un único servicio de ordenamiento, pero permitiendo que el conjunto de nodos ejecutivos de cada blockhain sea distinto. La ordenación de transacciones en un canal es independiente del resto y, en consecuencia, el consenso y replicación de estado también lo son. Por ello, podemos interpretar los canales de Fabric como un procedimiento de autorización adicional a la autenticación gestionada por la MSP. Es más, la funcionalidad de estos canales se puede extender para construir protocolos avanzados de comunicación entre canales y para la implementación de oráculos mediante canales constituidos por nodos ejecutivos seleccionados a tal efecto (Androulakis *et al.*, 2018b).

31. Véase https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html#pluggable-consensus (último acceso 11/03/2019).

Ripple

Hacia 2007, antes de la publicación de Bitcoin, Ghosh y sus colaboradores publicaban un estudio de redes de confianza en el contexto de pagos (Ghosh *et al.*, 2007) que daría lugar a RipplePay, antecesor directo de Ripple. Tanto el sistema original como su sucesor se centran en el problema de permitir pagos en una red en la que no hay una confianza directa entre el pagador y el receptor. En concreto, se describe el proceso de pago como la transferencia de "obligaciones" en el sentido de "pagaré" (IOU, abreviación de *I Owe You*), de modo que los pagos en sistemas monetarios equivalen a la confianza que tiene el receptor del pago en que el "mecanismo" que utiliza el pagador está respaldado por algo fiable, garantizado de algún modo por un emisor de dicho "mecanismo" de pago.

No obstante, en RipplePay no se parte de la existencia de una autoridad al estilo de un Banco Central en el que confían todos los usuarios del sistema. Al contrario, el problema que se pretende resolver se basa en un contexto más descentralizado en el que puede que la autoridad que respalda las obligaciones del pagador no sea considerada, de forma directa, como una autoridad válida por el receptor del pago. Es decir, RipplePay se centra en encontrar un "camino de confianza". Por ejemplo, puede que el receptor *A* no confíe en los IOU del pagador *C* pero sí confíe en los de un tercero, *B*, que a su vez confía en los de *C*. De este modo, el camino de confianza sería *A*-*B*-*C*, y *C* cambiaría sus IOU por el equivalente de *B*, para darle a *A* estos últimos. Cada uno de estos saltos se puede ver como líneas de confianza entre los dos extremos.

En Ripple, esta filosofía se mantiene en cierto modo, aunque poniendo más énfasis en el cambio de divisa. En concreto, dentro de la red de Ripple existen unas entidades especiales conocidas como pasarelas o *gateways*. Por un lado, una pasarela en Ripple permite establecer líneas de confianza con el mundo exterior. Es decir, un usuario *A* puede hacer un depósito de 1.000 euros a una pasarela *P* y, a partir de ese momento, *A* tendrá un crédito de 1.000 euros con *P*, que podrá utilizar para hacer pagos dentro de Ripple. Es decir, *A* confía en que *P* vaya a

mantenerle su dinero. Por otro lado, las pasarelas pueden permitir también intercambios de divisas internamente. Si la pasarela *P* soporta tanto euros como dólares, el usuario *A* puede pedir a *P* cambiar parte de los 1.000 euros a dólares. Sin embargo, para que esta funcionalidad de intercambio de divisas no dependa únicamente de las pasarelas, cualquier usuario de Ripple puede hacer ofertas de compraventa de divisas de un tipo por divisas del otro. En este sentido, este proceso funciona como un mercado tradicional: la oferta de compra a un precio más bajo será más atractiva para el comprador, y viceversa. Las pasarelas también son responsables de mantener procesos para prevención de blanqueo de dinero o AML (Anti Money Laundering) o de recabar la información legal necesaria sobre sus usuarios del tipo "conoce a tu cliente" o KYC (Know Your Customer), que serán ampliadas en el capítulo 5. Hasta aquí todo se parece bastante al sistema bancario tradicional, incluyendo la confianza que el usuario deposita en la pasarela y los mecanismos de control y liquidez.

La primera diferencia aparece con la existencia de una moneda interna de Ripple, conocida como *XRP*. Una opción típica de una pasarela es aceptar pagos en dinero fiat tradicional (euros, dólares, etc.) a cambio de crédito interno en XRP. Una vez dentro de Ripple, estos XRP pueden enviarse a cualquier otra pasarela. Es decir, el XRP actúa como moneda universal dentro de Ripple.

Otra diferencia, también fundamental, es que cada transacción en Ripple, implique las divisas que implique, "destruye" una cantidad variable de XRP. Esta variación depende de forma directa de lo sobrecargada que esté la red. Es decir, si hay mucha actividad, las tasas serán más altas; si hay poca, serán más bajas. Este es un mecanismo nativo de protección ante ataques de denegación de servicio, ya que un atacante que quiera bloquear la red emitiendo muchas transacciones (por ejemplo, de un valor muy bajo), acabará provocando que las tasas en XRP aumenten y, por lo tanto, tendrá que gastar más dinero en enviar sus transacciones, quedándose cada vez más rápido sin los XRP necesarios para seguir emitiendo transferencias y mantener su ataque de denegación.

El mecanismo de consenso utilizado se conoce como Ripple Protocol Consensus Algorithm o RPCA (Schwartz *et al.*, 2014). Este algoritmo, ejecutado por los nodos validadores de la red, procede de forma iterativa y por fases. En cada iteración, durante la primera fase, cada validador almacena las transacciones nuevas que haya recibido y las distribuye como su conjunto candidato. En la segunda fase, cada validador vota cuáles son las transacciones que considera válidas de entre todas las que ha recibido de sus validadores de confianza. En una tercera fase, cada validador descarta las transacciones que no hayan recibido un respaldo de un determinado porcentaje de sus validadores de confianza. Este proceso se repite, incrementando el porcentaje de "corte", hasta que al menos el 80% de los validadores de confianza vote favorablemente. Normalmente, el porcentaje inicial es del 50%. El conjunto de "validadores de confianza" que hemos mencionado se conoce como la UNL (Unified Node List) de cada nodo. Es, básicamente, el conjunto de nodos validadores en los que cada nodo deposita su confianza para decidir la evolución del sistema. Cada nodo es libre de elegir sus validadores de confianza (e incluso el porcentaje de respaldo exigido). De esta manera, Ripple "esquiva" la limitación tradicional de escalabilidad en protocolos de consenso distribuido tradicionales, ya que en lugar de que cada nodo espere a que todos los demás nodos se pongan de acuerdo, simplemente tiene que esperar a que se pongan de acuerdo los nodos de su UNL. Esto no implica que cada nodo vaya a ver un estado distinto del sistema: siempre y cuando haya una mínima intersección entre las UNL de cada nodo y no haya más de un determinado porcentaje de validadores maliciosos, el consenso global está garantizado. En concreto, según Schwartz *et al.* (2014), RPCA soporta un 20% de nodos maliciosos y requiere al menos un 20% de intersección entre las UNL de los nodos de la red. En la práctica, existe un listado de nodos validadores fiables[32], entre los cuales todos los nodos de la red suelen elegir a sus validadores de confianza. Esto ayuda a garantizar la

32. Disponible en https://xrpcharts.ripple.com/#/validators (último acceso 11/03/2019).

intersección mínima entre todas las UNL, pero también supone un riesgo de recentralización.

Como podemos apreciar, Ripple tiene muchos aspectos en común con el sistema bancario al que estamos acostumbrados: el establecimiento de líneas de confianza por medio de IOU, la orientación al cambio de divisa, el concepto de pasarelas, quienes además tienen como requisito soportar procedimientos de KYC y AML, etc. En la práctica, también se favorece que haya ciertos validadores confiables dado que cualquiera puede crear un validador. En todo caso, si alguien incluye en su UNL validadores que no son comúnmente aceptados por la mayoría de la red, corre el riesgo de ver una versión distinta del histórico de transacciones y, por lo tanto, ser vulnerable a ataques. Este tipo de consideraciones hacen que, *de facto*, Ripple funcione como un sistema permisionado ya que, aunque cualquiera podría actuar como pasarela o como validador, solo unos pocos acaban siéndolo.

CAPÍTULO 5

Aplicaciones prácticas de la tecnología blockchain: casos de uso y limitaciones

Tras unos primeros años en los que los primeros usuarios de Bitcoin se familiarizaron con su funcionamiento, empezaron a proponerse nuevos casos de uso que, más que basarse en Bitcoin, aprovechaban propiedades intrínsecas de la tecnología subyacente: blockchain. Pero al ritmo que se iban "ampliando" los límites de la tecnología y dando un mayor uso a la misma, se iban detectando problemas. En este capítulo, nos centraremos en estas aplicaciones y sus problemas.

Desde la notarización a la gestión de la Internet de las cosas, pasando por la gestión de la identidad

Durante unos años, únicamente existía Bitcoin. Pero pronto empezaron a surgir cadenas de bloques que variaban algún aspecto de su funcionamiento con respecto a Bitcoin. Estos sistemas se conocen como "cadenas de bloques alternativas", o simplemente, cadenas alternativas. Una de las primeras cadenas de bloques alternativas en crearse fue Namecoin. Su propósito fue el de crear una versión descentralizada del sistema de gestión de asignaciones de nombres de dominio en

Internet[33], el conocido DNS (Domain Name System). La motivación de este proyecto fue (y es) evitar el control centralizado de DNS de nivel superior que gestiona la Corporación de Internet para la asignación de nombres y números o ICANN (Internet Corporation for Assigned Names and Numbers). La forma de evitar este control centralizado a través de una cadena de bloques es bastante intuitiva. En lugar de traspasar un valor monetario, el bien transferido en las transacciones del sistema son nombres de dominio. Así, en la cadena de bloques queda registrado quién es el propietario de un dominio concreto. Implementándose encima de un sistema blockchain, también se dotó a esta forma de registro de la funcionalidad de pago, para poder pagar por esa adquisición de dominios, utilizando una moneda nativa de la cadena. Lo importante, en cualquier caso, es que no habría un único propietario de la tabla que asignara un propietario a cada dominio, sino que todos los nodos de Namecoin mantendrían una copia y, por lo tanto, cualquier actualización debería someterse a consenso.

El hecho de que la gestión descentralizada de nombres fuera el primer caso de uso de una cadena de bloques no puramente monetario es significativo, dado que evitar que haya una única entidad que controle quién es el titular de una identidad es algo bastante delicado. Además, el uso de sistemas blockchain también permite evitar la actuación de intermediarios en el proceso de autenticación, con lo que puede ser una pieza importante para cumplir ciertas propiedades de privacidad. Como por ejemplo, que haya una entidad que controle a qué sitios web accede una identidad concreta, algo que pasa con frecuencia en sistemas de Single Sign On[34]. Lo cual no evita que escribir información de identidad en un medio inmutable y

33. En Internet, un nombre de dominio es la cadena de texto que utilizamos para identificar un sitio de Internet. Por ejemplo, "es" es el dominio de nivel superior destinado a sitios de Internet españoles y "wikipedia.es" es el nombre de dominio para la versión española de la Wikipedia.

34. Single Sign On es un mecanismo de control de acceso a través del cual varios sistemas pueden "asociarse" de manera que sus usuarios puedan acceder a sus servicios utilizando un identificador común para todos. Por ejemplo, como cuando utilizamos nuestras credenciales de Facebook para acceder a sitios que no tienen nada que ver con Facebook.

que facilita la trazabilidad sea en sí mismo un riesgo para la privacidad.

En cualquier caso, la gestión descentralizada de la identidad a través de sistemas blockchain es un área bastante activa, en la que existen numerosas iniciativas. Por ejemplo, la W3C, una organización para la estandarización de sistemas y protocolos basados en la World Wide Web, tiene activas dos iniciativas: 1) el modelo de datos y especificación para la identidad descentralizada (Decentralized Identity)[35], y 2) el grupo de trabajo para afirmaciones verificables (Verifiable Claims Working Group)[36] sobre estas identidades digitales. El primero está directamente relacionado con sistemas descentralizados, principalmente cadenas de bloques, utilizándolos como plataforma de respaldo de los datos de identidad. Estos datos, a su vez, pueden ser atestados en forma de las afirmaciones verificables.

Quizá, el interés suscitado por esta posible aplicación de las cadenas de bloques viene derivado de los procedimientos KYC y AML (véase el capítulo 4). Ambos procedimientos son fundamentales en la industria financiera. El primero, para poder identificar de forma fehaciente a los clientes de las entidades financieras; el segundo, para prevenir usos fraudulentos de los activos que manejan. De hecho, cualquier entidad financiera tiene el requisito legal de implementar sistemas que cumplan estos procedimientos para poder llevar a cabo su actividad. En ambos casos, la trazabilidad que proporcionan las cadenas de bloques, si se dotan de sistemas de identificación robustos, puede utilizarse como palanca para implementar sistemas robustos de KYC y AML. No obstante, como ya hemos comentado, todo esto puede entrar en conflicto con la privacidad de los usuarios, aspecto en el que se enfocan otras normativas, como el reciente RGPD (el Reglamento General de Protección de Datos).

Otra propiedad que rápidamente se aprovechó para diseñar casos de uso es la de inmutabilidad. En concreto, para

35. Véase https://w3c-ccg.github.io/did-spec/ (último acceso 11/03/2019).
36. Véase https://www.w3.org/2017/vc/WG/ (último acceso 11/03/2019).

permitir sistemas de notarización centralizados. Hasta que se propusieron estos sistemas, la única forma de obtener una prueba digital y fehaciente de que una determinada información existía en un instante temporal era por medio de una autoridad de sellado temporal o TSA (Time Stamping Authority). Estas autoridades son entidades con un reloj que debe cumplir unas propiedades estrictas de precisión y que, al recibir un documento, añaden el instante temporal en el que lo han recibido. También incluyen el documento en una firma digital. Por lo tanto, debe haber confianza en que estas autoridades actúen de forma honesta.

Gracias a las cadenas de bloques, es posible crear un equivalente descentralizado. El proceso es simple: en una transacción se incluye un resumen del documento a sellar temporalmente, además de una transferencia "simbólica" (por ejemplo, de bajo importe y entre dos cuentas propiedad del emisor de la transacción), y se envía dicha transacción. Cuando esa transacción sea aceptada por la red, aparecerá en un bloque que incluirá en su cabecera el instante temporal "aproximado" en el que se creó. El matiz de aproximado es importante dado que, aun en el caso de que cada nodo de la red sea capaz de especificar el instante temporal de una manera precisa, al tratarse de una red, es tremendamente complicado que todos los nodos se pongan de acuerdo en el mismo instante temporal. Por eso, las cadenas de bloques suelen manejar rangos aproximados de tiempo y, en cualquier caso, la granularidad con la que se puede demostrar el instante temporal en el que existía dicha información depende de la frecuencia del bloque (por ejemplo, aproximadamente 10 minutos en Bitcoin). El mecanismo específico de escribir la información en la cadena depende del sistema blockchain en concreto. Por ejemplo, en Bitcoin se utiliza el comando `OP_RETURN`, que permite incluir en la salida de una transacción un comando con unas pocas decenas de bytes arbitrarios (por ejemplo, el resultado de una función resumen). Aunque dicha salida se evaluará como FALSE siempre y, por lo tanto, no podrá gastarse, dichos bytes quedarán registrados para siempre en la cadena.

Como hemos comentado, este es también uno de los primeros casos de uso alternativos que se les dio a los sistemas blockchain. Ejemplos concretos de aplicaciones y sistemas en este campo son Proof of Existence[37], OpenTimeStamps[38] y Stampery[39]. Por supuesto, de este caso de uso "básico" se pueden derivar casos de uso más avanzados, como servicios de notarización, certificación, etc.

Otro uso tradicional es el de gestión de activos, de nuevo basándose en la propiedad de inmutabilidad. De hecho, este caso de uso puede verse como una generalización de transferencia de dinero. El envío de dinero representa en realidad la transferencia de propiedad de unas "monedas" de una persona a otra. Pero si la propiedad que se transfiere es sobre cualquier otra cosa (de ahí que este caso de uso se haya asociado a la Internet de las cosas), el razonamiento sigue siendo muy parecido. Por ejemplo, si una empresa de coches compartidos (*carsharing*) escribe en una blockchain que el propietario actual del coche *C* es el usuario con dirección *D*, entonces solo el usuario que controle la clave privada asociada tiene derecho a utilizar el coche. La información de quién es el propietario se almacena por lo tanto de forma descentralizada en la cadena. Y más aún, la propia empresa podría ser descentralizada y sus normas representadas en forma de un contrato inteligente. Estos principios se han utilizado para diseñar sistemas que reflejan, en transacciones de la cadena de bloques, quién es el propietario en un determinado momento de un bien en concreto, ya sea físico o digital.

Quizá el primer concepto que empezó a surgir en este ámbito es el de monedas coloreadas (*colored coins*). Se dio este nombre a los sistemas que construían el concepto de activo basándose en criptomonedas y su tecnología subyacente (de Bitcoin, principalmente). El nombre se deriva simplemente de que a cada moneda (más precisamente, transacción de poco valor) se le asociaba una metainformación que permitía determinar qué tipo de bien representaba esa moneda. En sistemas derivados de Bitcoin, esta asociación se hacía muchas veces utilizando el comando `OP_RETURN` en una de las salidas de la transacción. Al

37. Véase https://poex.io/ (último acceso 11/03/2019).
38. Véase https://opentimestamps.org/ (último acceso 11/03/2019).
39. Véase https://stampery.com/ (último acceso 11/03/2019).

tener cada metainformación un significado distinto (por ejemplo, todas las monedas con la etiqueta 0x1234 son coches), se dice que las monedas están coloreadas. Para transferir la propiedad, basta con que el propietario de la *colored coin* gaste la salida que aparece en la misma transacción junto con la salida incluida en el OP_RETURN (que recordemos que es una salida no gastable porque siempre se evalúa como falso).

Pese a ser un caso de uso totalmente válido, hay, no obstante, matices importantes que conviene destacar. Por un lado, puede que la blockchain subyacente no entienda de la lógica de aplicación. Es decir, que no se preocupe por verificar que una determinada transacción sea válida según las reglas de la *colored coin* que represente. Esto pasa por ejemplo en Bitcoin, donde a su sistema de consenso solo le preocupa la semántica en términos de bitcoins, no de una u otra *colored coin*. Esto puede no ser así en otras cadenas de bloques. Por ejemplo, en un contrato inteligente de Ethereum se puede especificar, como parte de un método del contrato, cualquier mecanismo de transferencia de propiedad y el sistema se preocupará de que la ejecución de dicho método (y por lo tanto, del mecanismo de transferencia) sea correcto.

Finalmente, sobre todo en el caso de los sistemas blockchain que no soportan esta semántica, existe la opinión de que este uso de la cadena correspondiente es inapropiado. Se alegan dos motivos: (1) los mecanismos que se suelen utilizar "queman" dinero (por el hecho de que la salida que incluye un OP_RETURN sea no gastable) y (2) crea transacciones que se alejan del fin principal del sistema blockchain en cuestión y, por lo tanto, hacen que sea más difícil conseguir un "hueco" en un bloque. Este hecho crea demanda por dicho espacio y produce una subida de las tasas recomendadas para que una transacción sea aceptada.

Propuestas para la integración y trasvase de información entre plataformas blockchain: transición hacia la web 3.0

Inicialmente solo existía Bitcoin, pero pronto surgieron cadenas de bloques alternativas. Aunque las primeras eran poco más que mínimas variaciones de Bitcoin, con el tiempo estas cadenas

alternativas han ido aportando diferencias importantes. En el capítulo 3 hemos hablado de Ethereum, que permite realizar cómputos arbitrarios sobre la cadena, y de Ripple en el capítulo 4, especialmente pensado para intercambio de divisas.

Lo cierto es que, conforme el ecosistema va siendo más rico y aunque cada sistema blockchain se centre en un caso de uso, no es raro que surjan contextos en los que como parte de una operación en una blockchain, pudiera tener sentido hacer una operación en otro. Por ejemplo, si un usuario tiene cuentas en Bitcoin y en Ethereum, ¿sería posible que ejecutara un contrato inteligente en Ethereum, pero pagando con bitcoins?

Una respuesta afirmativa a esta pregunta consiste en utilizar lo que se conoce como verificación de pago simplificado o SPV (Simplified Payment Verification). Básicamente, un proceso SPV permite verificar que un pago ha tenido lugar sin necesidad de tener una copia completa de la cadena. En concreto, solo es necesario incluir una prueba de que la salida correspondiente al pago se ha incluido en un bloque y una lista de cabeceras de bloque que lo incluyan. No obstante, esta ganancia viene a costa de perder seguridad, ya que al no utilizar una copia completa de la cadena, no siempre es posible detectar determinados ataques. La ventaja es que son procesos más ligeros y que pueden implementarse dentro de una cadena de bloques, por ejemplo, como parte de un *smart contract* en Ethereum.

Supongamos que existe un contrato inteligente en Ethereum que permite verificar pruebas SPV relativas a transacciones de Bitcoin. Entonces, si un usuario *A* tiene bitcoins y quiere pagar a un usuario *B* (que también tiene al menos una dirección en Bitcoin) por ejecutar una operación en Ethereum, lo que tiene que hacer *A* es pagar en bitcoins a una de las direcciones de *B* en Bitcoin y, desde Ethereum, construir la prueba SPV y enviarla al contrato que verifica y ejecuta la operación indicada por *A*. Este proceso, explicado de forma muy resumida y que incluye periodos adicionales de espera como mecanismo de protección ante reorganizaciones de la cadena, es la base de lo que se conoce como *pegged sidechains*[40], algo así como

40. Véase https://blockstream.com/sidechains.pdf (último acceso 11/03/2019).

cadenas pinzadas. De hecho, hay sistemas que implementan este tipo de operaciones, como BTCRelay[41].

Protocolo de *Atomic Swaps**

Otra forma de intercambiar valor entre cadenas, en este caso sin SPV, son los intercambios atómicos o *atomic swaps*. Estos intercambios atómicos siguen un ingenioso protocolo introducido en un foro de Bitcoin por el usuario TierNolan[42] que utiliza transacciones temporizadas para evitar comportamientos deshonestos. Este protocolo, que permite que un usuario *A* con *mA* criptomonedas en una cadena *cA* las intercambie con *mB* monedas que un usuario *B* tiene en otra cadena *cB* es como sigue (figura 11).

1. El usuario *A*, en posesión de un secreto (por ejemplo, una contraseña) *s*, prepara una transacción en *cA*, moviendo *mA* monedas de una de sus cuentas a la cuenta especificada por *B* en *cA*. Llamaremos a esta transacción Transacción 1. Hay dos posibles opciones de gastar la Transacción 1:
   - Proporcionando un valor *s,* tal que el resumen de *s* valga *V*, y una firma digital emitida desde una dirección de *B*. Esta será la opción utilizada en caso de que todo vaya correctamente.
   - Proporcionando una firma digital doble, firmada tanto por *A* como por *B*. Esta será la opción utilizada en caso de que haya que dar marcha atrás.

   A no publica aún la Transacción 1.
2. *A* prepara una transacción temporizada, es decir, que no se pueda ejecutar hasta un momento concreto para devolver las *mA* monedas a una cuenta de *A*. Esto se puede hacer en muchas cadenas de bloques especificando a partir de qué número de bloque es aceptable la operación. Por ejemplo, 288 bloques después del actual.

41. Véase http://btcrelay.org/ (último acceso 11/03/2019).
42. Véase https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949 (último acceso 11/03/2019).

Llamaremos a esta transacción, Transacción 2.
A manda la Transacción 2 a *B*.

3. *B* devuelve la Transacción 2 firmada. De esta forma, en caso de que algo vaya mal, *A* solo tendrá que esperar el tiempo necesario (288 bloques) para recuperar sus monedas.
4. *B* realiza un proceso similar. Crea una transacción moviendo, en la cadena *cB*, *mB* monedas de una de sus cuentas a una cuenta especificada por *A* en *cB*. Llamaremos a esta transacción Transacción 3. Es posible gastar la Transacción 3 de dos formas:
   - Proporcionando un valor *s* tal que el resumen de *s* valga *V*, y una firma digital emitida desde una dirección de *A*.
   - Proporcionando una firma digital doble: firmada tanto por *A* como por *B*.

   B no publica aún la Transacción 3.
5. *B* prepara una transacción temporizada, devolviendo las *mB* monedas a una de sus direcciones. El plazo en este caso debe ser menor que el plazo de la Transacción 2. Por ejemplo, 144 bloques después del actual. Llamaremos a esta transacción, Transacción 4. *B* envía la transacción a *A*.
6. *A* envía la Transacción 4 firmada de vuelta. A partir de este momento, *B* tiene la certeza de que podrá recuperar sus monedas si algo va mal, pasados 144 bloques.
7. Una vez se han intercambiado las transacciones temporizadas (la Transacción 2 y la Transacción 4), que hacen de seguro por si algo funciona mal, *A* y *B* pueden publicar las transacciones que permiten hacer el intercambio atómico *per se* (la Transacción 1 y la Transacción 3).
   En este paso 7, *A* publica la Transacción 1 en la cadena de bloques *cA*. Esto significa que, siguiendo el *script* de bloqueo de la Transacción 1, correspondiente al caso 1.a), en el momento en el que el secreto *s* caiga en manos de *B*, este podrá gastar las *mA* monedas de *cA* especificadas en la Transacción 1, enviándolas a una dirección de *cA* bajo su control.
8. *B* publica la Transacción 3 en la cadena de bloques *cB*. Como *A* conoce el secreto *s*, a partir de este momento,

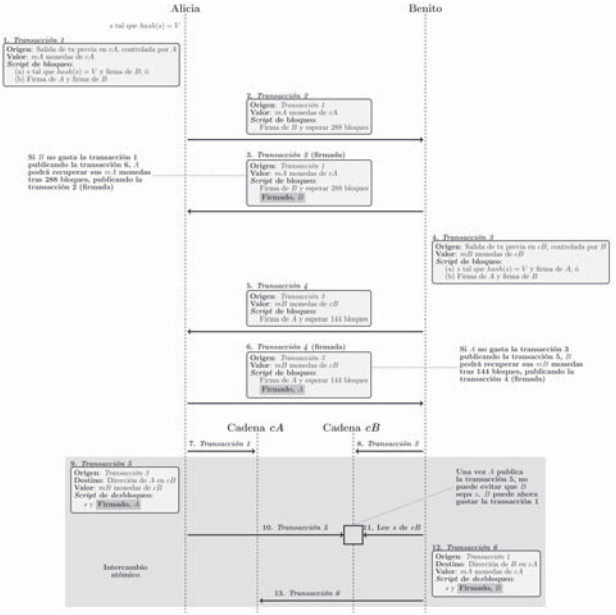
A puede gastar las *mB* monedas de *cB* especificadas en la Transacción 3, enviándolas a una dirección de *cB* bajo su control.

9. Una vez *A* detecta que *B* ha publicado la Transacción 3 en *cB*, crea una nueva transacción, que llamaremos Transacción 5, en la que envía las *mB* monedas de *cB* de la Transacción 3 a una dirección de *cB* bajo su control. Pero lo importante es que, para desbloquear estas monedas, debe incluir el secreto *s* dentro de la Transacción 5. Además, este secreto debe cumplir la condición de que su resumen valga *V* o, de lo contrario, la Transacción 5 sería inválida.
10. Como *A* conoce el valor *s* adecuado, la Transacción 5 se acepta.
11. *B* está continuamente monitorizando la cadena *cB*, por lo que en el momento en el que la Transacción 5 se difunde, el valor *s* llega a conocimiento de *B*. Es fundamental para B que actúe antes de que venza el plazo especificado en la Transacción 2 ya que, de lo contrario, A podrá recuperar también sus *mA* monedas de *cA*.
12. Dado que en el paso anterior, *A* ha desvelado el secreto *s*, ahora *B* también puede crear una transacción para mover las *mA* monedas de *cA* incluidas en la Transacción 1. Llamaremos a esta nueva transacción Transacción 6.
13. Para terminar, *B* envía la Transacción 6 a la cadena de bloques *cA*. Nótese que esta transacción debe ser válida, ya que la Transacción 5 lo era, y ambas dependían del mismo secreto *s*. Por lo tanto, la Transacción 6 es aceptada en la cadena *cA*, y las *mA* monedas de *cA* pasan a una dirección de *cA* controlada por *B*.

Estas operaciones y otras similares permiten intercambiar valores entre cadenas de bloques distintas, lo cual da mucha flexibilidad, evita tener que "casarse" con una cadena en concreto y permite crear aplicaciones mucho más ricas en funcionalidad. No obstante, sigue habiendo un problema: ¿cómo conectamos una blockchain con el mundo exterior? Es decir, ¿es posible tener en cuenta datos de la web tradicional para tomar decisiones en una transacción dentro de una blockchain?

FIGURA 11

Protocolo por el que un usuario *A* con mA criptomonedas en una cadena cA las intercambia con mB monedas que un usuario B tiene en otra cadena cB.



Este es un problema relevante, especialmente teniendo en cuenta que gran parte de la información del mundo real de la que dependemos se encuentra en la web tradicional. Y no es tan sencillo como hacer una consulta a una API[43] (Application Programming Interface) desde, por ejemplo, un contrato inteligente. Esto es debido a que, para evitar poner en peligro la capacidad de consenso de los sistemas blockchain, cada operación de lectura (dentro de la blockchain o fuera), siempre debe devolver lo mismo.

Supongamos que es posible hacer consultas al exterior. Entonces, en un instante t_1, un nodo ejecuta un contrato para leer un valor de una web dinámica y actúa en consecuencia; posteriormente, en un instante t_2, otro nodo intenta validar la ejecución del

43. Una API es una interfaz que expone un sistema informático para ofrecer al exterior la funcionalidad que implementa.

contrato efectuada por el primer nodo pero, al hacer la consulta a la web, recibe otro valor distinto. Lógicamente, el segundo nodo no puede fiarse del resultado obtenido por el primer nodo, por lo que rechaza la transacción pese a que fue generada honestamente.

Para evitar esta casuística, una aproximación común es utilizar el concepto de oráculos (ya mencionado en el capítulo 4), que son entidades en las que se confía para traer datos del exterior. Por ejemplo, una forma de implementar un oráculo es escribiendo en un contrato inteligente una clave pública bien conocida o certificada de algún modo. Para proporcionar datos fiables procedentes de dicha fuente, alguien tendrá que ejecutar dicho contrato inteligente, proporcionando una firma digital sobre dichos datos, y que es emitida con la clave privada asociada a la clave pública certificada previamente. Dada la confianza depositada en la clave pública inicialmente, podemos tomar dicho valor como verídico.

Aunque es cierto que esta confianza choca con la filosofía de los sistemas blockchain (principalmente en el caso de ser públicos), no parece posible hacerlo de otra forma. En cualquier caso, existen servicios que implementan esta funcionalidad de una forma bastante fiable (haciendo uso de sistemas criptográficos), como es el caso de Oraclize[44].

Riesgos y problemas en plataformas blockchain

En esta sección vamos a comentar algunos de los riesgos y problemas más importantes relacionados con la tecnología blockchain.

Privacidad

Como ya hemos comentado, la gestión de la identidad es un aspecto fundamental dentro de cualquier plataforma blockchain. Esto es así porque es imprescindible conocer quién ejecuta cada acción para saber si tiene derecho a hacerlo en términos que entienda la lógica del protocolo de consenso subyacente: si tiene fondos suficientes, si tiene permisos para ejecutar una función

44. Véase http://www.oraclize.it/ (último acceso 11/03/2019).

determinada, etc. La práctica común, heredada de Bitcoin, es utilizar las direcciones (recordemos que, en jerga blockchain, una dirección es una codificación de una clave pública) para determinar la validez de una operación. Ello tiene mucho sentido dado que el único capaz de demostrar ser el propietario de una dirección será quien controle la clave privada asociada a esa clave pública. Pero en un sistema en el que todos tienen acceso a todo, el hecho de apuntar junto a cada acción que se ejecuta la dirección de quien la ejecuta, tiene un efecto colateral importante: todos serán capaces de asociar la dirección con la acción.

A priori, esto podría no parecer muy grave. Al fin y al cabo, estas direcciones son valores binarios que poco sentido tienen para un humano. Además, lo normal (al menos en las blockchains públicas) es que cada usuario pueda crear tantas direcciones como quiera, con lo que, en teoría, podría utilizar una dirección distinta para cada operación. Sobre el primer aspecto, el problema es que no solo debemos preocuparnos por adversarios de carne y hueso, sino también por programas capaces de procesar y clasificar información binaria. Además, sobre el segundo aspecto, no es difícil dar con buenas heurísticas que minimicen el efecto de usar múltiples direcciones por usuario. Un ejemplo muy claro es Bitcoin: si una transacción en Bitcoin tiene varias entradas, es bastante probable que las direcciones de cada una de esas entradas pertenezcan todas a la misma persona. Parece bastante lógico, al fin y al cabo, dado que podemos equiparar una transacción con varias entradas en Bitcoin a una persona que utiliza varios billetes para hacer una compra en una tienda física. Esto es, aunque se utilicen varios billetes (entradas) en una misma compra (transacción), el propietario de todos los billetes (claves asociadas a las direcciones en las entradas) es el mismo. Por supuesto, hay excepciones, como cuando un amigo le presta un billete a otro para completar el pago. Pero normalmente son eso, excepciones. Heurísticas de este estilo han sido aplicadas sobre Bitcoin y otras criptomonedas (Karame *et al.*, 2013) para ser capaces de determinar qué direcciones pertenecen a una misma persona. Su efectividad es a veces asombrosa.

Dicho esto, en el aspecto del anonimato de las transacciones (concepto directamente relacionado con la privacidad del

sistema), es justo puntualizar que no es lo mismo analizar este asunto desde una perspectiva de privacidad de la información que desde una perspectiva legal. En el primer caso, ya hemos argumentado que, en muchas blockchains, es posible "desanonimizar" con una tasa de acierto bastante elevada las identidades que hay detrás de una transacción seudónima. Se entiende por "desanonimizar" el hecho de diferenciar si cualquier par de acciones distintas las lleva a cabo una misma persona o no, independientemente de si utiliza direcciones distintas. No obstante, desde el punto de vista legal, situándonos por ejemplo en el contexto de la prevención de blanqueo de dinero o los requisitos de sistemas de KYC, probablemente no sea suficiente con saber con un alto porcentaje de certeza que dos operaciones han sido ejecutadas por la misma entidad; más bien, sería necesario tener una certeza total y, además, saber cuál es la identidad real detrás de esa entidad; es decir, un identificador seudónimo no es suficiente.

Por último, conviene señalar que este no es un problema recientemente descubierto, sino que se sabe desde hace años. De hecho, ya hay muchas alternativas que intentan resolver este problema. Por un lado, pronto surgieron servicios de mezcla (*mixing*), a través de los cuales se reenviaban bitcoins múltiples veces entre direcciones no relacionadas entre sí, terminando en una dirección nueva del usuario de origen, lo cual permitía desacoplar las direcciones de origen y destino (ambas del mismo usuario). Por otro lado, existen propuestas para mejorar la privacidad de forma nativa. Así, en Bitcoin tenemos lo que se conoce como transacciones confidenciales (*confidential transactions*), que permiten ocultar las cantidades implicadas en cada entrada y salida de una transacción; pero también hay criptomonedas creadas desde cero con la privacidad en mente, como ZCash o Monero. ZCash es una criptomoneda basada parcialmente en Bitcoin que permite hacer transacciones totalmente anónimas utilizando primitivas criptográficas conocidas como "pruebas de conocimiento nulo" (*zero knowledge proofs*). Por su parte, Monero combina las *confidential transactions* con otra primitiva criptográfica conocida como firmas en anillo (capítulo 1), lo que permite ocultar el firmante de una transacción entre un conjunto de posibles firmantes.

Escalabilidad

Otra problemática importante es la de la escalabilidad. Podemos hablar de escalabilidad principalmente en dos sentidos: en cuanto a coste computacional y a coste de almacenamiento. El problema de la escalabilidad computacional se hace evidente cuando se leen noticias que comparan el consumo energético de Bitcoin con el de algunos países o el de un porcentaje del consumo mundial. Desde el punto de vista computacional, un sistema que consuma tal cantidad de potencia de cálculo, y por tanto de energía, no parece ser un sistema eficiente. El problema es que este coste es intrínseco al funcionamiento de Bitcoin: para que sea seguro, el mecanismo de prueba de trabajo requiere que se gaste potencia computacional. De hecho, el sistema será más seguro cuanta más potencia de cálculo se dedique a dicha tarea, siempre y cuando dicha potencia no se concentre en unos pocos participantes de la red. Precisamente esta cuestión es la que ha fomentado la búsqueda de nuevos sistemas de consenso, como el *Proof of Stake* (PoS) que ya mencionamos en el capítulo 4; o *Proof of Elapsed Time*, basado en procesadores seguros de Intel. Además, también están los que se han venido a conocer como algoritmos de tipo *Proof of Authority*, que vienen siendo derivados de los tradicionales sistemas de consenso bizantino, conocidos en la literatura académica desde los años ochenta (Lamport *et al.*, 1982).

La otra vertiente de la problemática sobre la escalabilidad la encontramos en los requisitos de almacenamiento. Realmente, una cadena de bloques es una estructura de datos en la que cada bloque va enlazado inequívocamente al bloque precedente de forma que, además, es necesario tener el bloque precedente para poder comprobar la validez del siguiente. Y así desde el bloque más reciente hasta el bloque inicial o génesis. Como hemos mencionado en el capítulo 2, la cadena de bloques de Bitcoin completa, a marzo de 2019, ocupa aproximadamente 254 GB. Es cierto que se pueden hacer compromisos, por ejemplo, borrando transacciones poco interesantes para el usuario de un cliente Bitcoin; no obstante, esto viene siempre a costa de pérdida de seguridad. También se está investigando en esta línea. Por ejemplo, el protocolo CODA busca utilizar de nuevo un tipo especial

de pruebas de conocimiento nulo de forma recursiva para conseguir una blockchain de tamaño constante.

En cualquier caso, es importante tener claras las implicaciones de esta falta de escalabilidad en términos de potencia computacional y almacenamiento requeridos, ya que no es suficiente con añadir más recursos. Sobre el aspecto de potencia computacional, el hecho de que esté siendo tan costoso energéticamente impide *de facto* a muchos usuarios participar en el proceso de consenso, a no ser que adquieran *hardware* especializado (algo que no está al alcance de cualquiera). Hacerlo implicará que pierdan dinero para pagar la energía necesaria para llevar a cabo la tarea, ya que, con una probabilidad altísima, nunca conseguirán minar un bloque y conseguir su recompensa. Sobre los requisitos de almacenamiento, de nuevo, no todo el mundo puede dedicar 254 GB (cifra que, además, solo puede crecer) de su espacio de almacenamiento para albergar una copia de la cadena de bloques completa. En ambos casos, esto conduce a que haya pocos que sean capaces de llevar a cabo estas tareas, lo que significa que no se consigue el nivel de descentralización que se perseguía inicialmente. Todo ello restringe, claramente, el tipo de dispositivos en los que se puede ejecutar un cliente completo de un sistema blockchain.

Rendimiento

Otra problemática es el rendimiento o *throughput*. En la mayoría de las blockchains no se puede alcanzar más de unas decenas o centenas de operaciones por segundo. Este es un límite difícil de eliminar, ya que está relacionado con la frecuencia de bloque. Por ejemplo, en Bitcoin, se producen de promedio, como ya hemos mencionado, un bloque cada 10 minutos. Además, cada bloque tiene un tamaño máximo; hasta finales de 2017 este límite era de 1 MB, desde entonces el límite es algo más complicado de calcular, pero pueden llegar a ocupar bastante más. Esto implica que también hay un límite máximo de transacciones por bloque: las que caben en él. Por hacer unas cuentas rápidas, pongamos que son 4.000 transacciones las que caben en un bloque (aproximadamente 250 bytes por transacción). Como se produce un bloque aproximadamente cada 10 minutos, se podrán publicar

como máximo 4.000 transacciones cada 10 minutos, o cerca de 7 transacciones por segundo. Esto no es competitivo con el número de transacciones que soportan los sistemas bancarios actuales, que son del orden de varios miles por segundo.

De nuevo, esta es un área en la que se está trabajando. Desde hace ya unos años se propuso la Lightning Network, que proporciona mecanismos para hacer transacciones *off-chain*, es decir, transacciones que no necesitan ser escritas en la cadena de bloques salvo en determinadas ocasiones. Estas transacciones son, por tanto, prácticamente instantáneas y también menos costosas en cuanto a las tasas que llevan asociadas. Algunas cadenas de bloques ya han empezado a introducir medidas para soportarlas, como Litecoin y Bitcoin.

Gestión de claves

La mayoría de las operaciones en una cadena de bloques requieren, como hemos mencionado en varias ocasiones, de una firma digital. A su vez, una firma digital requiere un par de claves: una pública y una privada (ver capítulo 1), siendo además imprescindible almacenar de forma segura las claves privadas. Esto trae consigo la problemática de la gestión de claves que siempre ha existido en criptografía. En el caso de las cadenas de bloques y, especialmente, en el de las criptomonedas, existe el agravante de que estas claves privadas dan acceso directo a un valor (en forma de dinero electrónico, de bienes digitales, etc.). Es decir, quien controla la clave privada, controla el valor asociado; y en el sentido contrario, quien pierde una clave privada, pierde el control del valor asociado. En el caso de blockchains públicas, perder el control en este caso es un hecho irreversible, ya que no hay una autoridad central que pueda decidir revertir una operación[45].

¿Cómo hacer para almacenar las claves privadas de forma segura y usable? La máxima seguridad requiere almacenar las claves en un dispositivo específicamente diseñado para tal fin, sin conectividad a Internet y guardado a buen recaudo. Las

45. Salvo casos excepcionales no exentos de polémica, como el *hack* a TheDAO en Ethereum y las medidas que se adoptaron. Véase https://www.criptonoticias.com/opinion/ethereum-duras-consecuencias-ataque-dao/ (último acceso 11/03/2019).

billeteras especialmente diseñadas para actuar como tal (no necesariamente sin conectividad) son las conocidas como *hardware wallets* (billeteras hardware), como Trezor[46]. A las wallets sin conectividad se las conoce con el término de *cold wallets* (billeteras frías). No obstante, en términos de usabilidad, probablemente estas no siempre sean prácticas para operaciones cotidianas, ya que sería demasiado incómodo. Por otro lado, si simplemente guardamos las claves criptográficas en un fichero, en nuestro móvil o portátil, corremos el riesgo de que nos lo roben o lo perdamos, lo que implicaría perder el control de las criptomonedas asociadas. También podemos cifrar el fichero con las claves, pero entonces estamos simplemente trasladando el mismo problema a la clave con la que ciframos el fichero. Otra opción puede ser utilizar un servicio en línea específicamente diseñado para ello, como Coinbase[47]. En este caso, es normal que no tengamos las claves en ningún momento, sino que sea el servicio quien las controla, asociando simplemente un balance a nuestro usuario. Quizá un punto intermedio son aplicaciones que han sido creadas específicamente para gestionar claves. En este caso, es el usuario quien debe asegurarse de conocer bien los mecanismos de copias de seguridad que proporciona cada aplicación; así como de las buenas prácticas para evitar accesos no deseados a estas aplicaciones. Un ejemplo de este tipo de billeteras es MyEtherWallet[48], una aplicación web (por lo tanto, accesible desde cualquier navegador) que permite generar claves en el propio navegador, sin enviarlas a ningún servidor, y mantenerlas cifradas en el ordenador del usuario. O las billeteras que normalmente ofrecen los equipos de desarrollo oficiales de cualquier sistema blockchain y se ejecutan como una aplicación de escritorio más. Ejemplos de esto son Bitcoin Core[49], de Bitcoin, o Mist[50], de Ethereum.

Por otro lado, en casos en los que es imprescindible asociar una identidad real a una clave, ¿qué procedimientos son

46. Véase https://trezor.io (último acceso 11/03/2019).
47. Véase https://www.coinbase.com (último acceso 11/03/2019).
48. Véase https://www.myetherwallet.com (último acceso 11/03/2019).
49. Véase https://bitcoin.org/es/wallets/desktop/windows/bitcoincore/ (último acceso 11/03/2019).
50. Véase https://github.com/ethereum/mist/releases (último acceso 11/03/2019).

necesarios y qué protocolos se pueden usar para ello? Una blockchain pública no es *a priori* compatible con las infraestructuras de clave pública tradicionales, como certificados electrónicos (tradicionalmente del tipo X.509), en las que disponemos de una o varias autoridades certificadoras que garantizan, en cierto grado, la identidad real asociada a un par de claves. El simple hecho de ser autoridades parece algo incompatible con un sistema descentralizado. En el caso de cadenas de bloques privadas, las infraestructuras de clave pública existentes son más apropiadas, pero ello no hace que sea un asunto sencillo: es necesario mantener listas actualizadas de los miembros del sistema, revocar las claves comprometidas o caducadas, etc. Todas estas tareas son delicadas y complejas, de forma que no se alcanza una solución idónea para la gestión de la identidad digital en el ecosistema blockchain.

Inmutabilidad

Una de las propiedades estrella de las cadenas de bloques es que, una vez que algo se escribe en ellas (en realidad, después de que se incluyan algunos bloques adicionales), es imposible borrarlo en la práctica. Esto es lo que se conoce como la inmutabilidad de las cadenas de bloques, y que se deriva de la estructura en cadena y de la robustez de los algoritmos de consenso.

No obstante, esta inmutabilidad también puede crear problemas, principalmente en blockchains públicas. Desde situaciones derivadas de fallos no intencionados, como equivocarse al poner la dirección de destino de un pago; hasta usos claramente indeseables, como incluir fotos pedófilas en la cadena de bloques. En el primer caso, una vez escrita la transacción con el destinatario incorrecto, dado que no hay un banco o similar que controle la cadena de bloques, la única opción sería pedirle a dicha persona que te devolviera el dinero. Para ello, primero debe existir tal persona ya que es perfectamente posible enviar dinero a direcciones inexistentes (es decir, cuya clave privada aún no se ha generado); e incluso en caso de existir, y de tener la suerte de conocerla, dicha persona tendría que devolver el dinero por decisión propia. En resumidas cuentas, parece complicado. En el ejemplo de contenido ilegal, como el pedófilo, es

perfectamente posible, ya que hay maneras de escribir información arbitraria en la cadena de bloques. De hecho, hay estudios que desgraciadamente afirman la existencia de este preciso ejemplo[51]. Y de nuevo, una vez escrito dicho contenido, la propiedad de inmutabilidad implica que no se puede borrar. Del mismo modo, también podrían utilizarse para guardar para siempre una copia de un determinado *malware*, etc.

Igual que con las otras problemáticas ya mencionadas anteriormente, también han surgido propuestas en este aspecto. Cabe destacar una propuesta de un modelo de cadena de bloques editable o *redactable blockchain*[52]. Explicado de manera muy breve, este modelo se basa en una primitiva criptográfica conocida como "resúmenes camaleón" en los que quien esté en posesión de una clave criptográfica privada es capaz de encontrar colisiones en estas funciones resumen camaleón (recordemos del capítulo 1 que encontrar una colisión en una función resumen criptográfica segura es algo computacionalmente imposible). Utilizando estas funciones resumen camaleón, quien posea la clave privada para calcular colisiones puede cambiar la historia de la cadena de bloques encontrando una colisión para el puntero resumen de un bloque al bloque anterior. Por supuesto, quien controle dicha clave privada tiene mucho poder y debe ser confiable. Esto hace que este modelo sea difícil de compatibilizar con cadenas de bloques públicas.

Gobernanza de la blockchain: principales desafíos tecnológicos y éticos en la gestión descentralizada de la información

Desde un punto de vista técnico, la descentralización es un medio para conseguir determinadas propiedades de seguridad, como protección ante censura o ataques de denegación de servicio. En definitiva, la disponibilidad y perdurabilidad de la información. Por otra parte, tiene cierto aire romántico, ya que todos los participantes del sistema son, *a priori*, iguales. La cuestión es que, pese

51. Véase https://fc18.ifca.ai/preproceedings/6.pdf (último acceso 11/03/2019).
52. Véase https://eprint.iacr.org/2016/757.pdf (último acceso 11/03/2019).

a que esta igualdad es muy deseable e incluso necesaria, conseguir una plataforma sin asunciones de confianza puede llegar a complicar bastante determinadas tareas.

Por ejemplo, en un entorno centralizado, desplegar una nueva versión de un sistema o un protocolo es algo delicado, pero que, con buena coordinación, se puede llevar a cabo de forma eficiente. Casos que vienen rápido a la mente son las actualizaciones del sistema operativo de un teléfono móvil, por ejemplo. Incluso casos algo más descentralizados, como los estándares de la IETF (Internet Engineering Task Force) o como los protocolos de seguridad de la capa de transporte o TLS (Transport Layer Security). En estos escenarios, basta con que la organización de referencia cree una nueva versión del sistema/protocolo en cuestión y siga los pasos establecidos para el despliegue de la misma.

Podría pensarse que el proceso sería el mismo en un entorno descentralizado, pero hay una diferencia fundamental: no existe una autoridad que pueda dar instrucciones al resto del mundo. Por supuesto, suele haber organizaciones o asociaciones que se convierten en referentes, pero sus decisiones o recomendaciones no son ni mucho menos incontestables. Además, como no podía ser de otro modo, en el caso de sistemas blockchain, el consenso es crucial. Por un lado, ya sabemos que cuanto mayor sea la proporción de nodos honestos frente a nodos deshonestos, más estable será el estado de la red (y esto vale tanto para sistemas con consenso de tipo prueba de trabajo, como de prueba de interés, o cualquier otro tipo).

Por otro lado, en este tipo de sistemas, las normas definidas en el proceso de consenso a partir de las cuales se decide qué transacciones son válidas parten de la definición del protocolo. Combinando estos dos aspectos podemos empezar a vislumbrar la problemática: si, ante una nueva versión del protocolo, un porcentaje elevado de los participantes de la red se opone a aplicarla (algo totalmente posible ya que no hay una autoridad que tome la decisión por todos), el sistema puede verse en la situación de que, en caso de aplicarse dicha actualización, haya una porción del sistema que la acepte y otra que la rechace. Esto significa que, potencialmente, un porcentaje de los nodos verán unas transacciones como válidas, y el porcentaje complementario las verá como

inválidas. En la práctica, esto significará que se han creado *de facto* dos redes, cada una con una versión parecida, pero distinta, del protocolo original. Y lo peor de todo es que la tolerancia ante actores deshonestos se reduce proporcionalmente en ambas redes.

En la práctica, hay distinción entre los tipos de actualizaciones. Una actualización se conoce como "bifurcación blanda" (*soft fork*) si los cambios que introduce son retrocompatibles. Es decir, un nodo que no se actualice podrá seguir generando transacciones válidas, aunque probablemente no entienda un subconjunto de las transacciones nuevas. Por otro lado, una actualización se conoce como "bifurcación dura" (*hard fork*) si no es así. Es decir, cualquier nodo que no se actualice dejará de ser capaz de emitir transacciones que sean aceptadas por el resto de la red. Por lo tanto, una *hard fork* es mucho más peligrosa ya que de no haber un acuerdo muy amplio, se corre el riesgo de crear dos redes totalmente incompatibles.

Históricamente, ha habido bifurcaciones de ambos tipos en las cadenas principales. Las *hard forks* más conocidas se han utilizado para crear nuevas criptomonedas, como *Bitcoin Cash*[53], que se bifurcó de Bitcoin para aumentar el tamaño máximo del bloque. De hecho, el historial del cambio de límite de tamaño de los bloques en Bitcoin es un buen ejemplo de los problemas de gobernanza en redes descentralizadas. Algo tan, en principio, sencillo como aumentar el tamaño máximo de los bloques del protocolo ha generado un debate que ha durado años y que ha tenido como consecuencia la creación de varias cadenas alternativas a Bitcoin. Incluso ha habido casos en los que la versión "oficial" de Bitcoin parecía que recibía menos soporte que una alternativa (el famoso *fork* de Segwit2x[54], al que llegaron a respaldar actores importantes del ecosistema, como granjas de minado, casas de intercambio e incluso quienes llegaron a ser desarrolladores principales del Bitcoin "oficial").

Otro ejemplo de *hard fork* —de distinta naturaleza pero igualmente relevante— fue el que se llevó a cabo en Ethereum para resolver el *hack* a TheDAO, un contrato inteligente que

53. Véase https://www.bitcoincash.org (último acceso 11/03/2019).
54. Véase https://en.wikipedia.org/wiki/SegWit2x (último acceso 11/03/2019).

pretendía usarse como vehículo de financiación descentralizada (DAO son las siglas de Decentralized Autonomous Organization, esto es, organización autónoma descentralizada) para proyectos basados en Ethereum. Después de una de las mayores rondas de financiación hasta la fecha, alguien descubrió una vulnerabilidad en el código de TheDAO a través de la cual podía desviar los fondos recaudados a una dirección de su elección. Aunque el ataque se paró, el atacante consiguió robar una cantidad sustancial de ether. Para recuperarlo, una parte importante de la comunidad (entre ellos, Vitalik Buterin, uno de los fundadores de Ethereum) decidió desplegar una actualización en la que se devolvía la cantidad robada a sus dueños originales. Como hemos dicho, pese a que los principales referentes en el ecosistema respaldaban esta opción, hubo muchos a quienes no les parecía correcto, porque pensaban que debía prevalecer la propiedad de inmutabilidad, o porque consideraban que "el código es ley" y, si algo estaba escrito de una forma en código y alguien lo había conseguido utilizar en su provecho, debería mantenerse así. En cualquier caso, la crítica principal era que abrir esta vía de actuación podía poner de manifiesto que Ethereum no era tan descentralizado como se suponía. El resultado final fue que tuvo lugar la bifurcación dura, y se creó una versión alternativa de Ethereum en la que, pese a tener la misma funcionalidad, el histórico de transacciones era diferente. Esta versión pasó a llamarse Ethereum Classic, y hoy en día sigue siendo una de las criptomonedas principales.

Para concluir, cabe mencionar que, como no puede ser de otra forma, hay sistemas blockchain que incluyen mecanismos para agilizar la gobernanza. Por ejemplo, en Tezos[55] el proceso de votación sobre posibles nuevas versiones del protocolo se hace sobre la propia blockchain (en jerga, se dice que se hace *on-chain*), mientras que en Bitcoin o Ethereum, este proceso debe hacerse por otros medios (es decir, *off-chain*).

55. Véase https://tezos.com (último acceso 11/03/2019).

Glosario

AML	Anti Money Laundering
API	Application Programming Interface
ASIC	Application-Specific Integrated Circuits
BFT	Byzantine Fault Tolerance
CBK	Banco central de Kenia o Central Bank of Kenya
CPU	Unidad de procesamiento central o Central Processing Unit
CRC	Verificación por redundancia cíclica o Cyclic Redundancy Check
CUDA	Compute Unified Device Architecture
DAO	Organización autónoma descentralizada o Decentralized Autonomous Organization
DLT	Registro contable distribuido o Distributed Ledger Technology
DNS	Sistema de nombres de dominio o Domain Name System
DPoS	Delegated Proof of Stake
ECDS	Algoritmo estándar de firma digital con curvas elípticas o Elliptic Curve Digital Signature Algorithm
EOA	Externally Owned Account
EVM	Máquina virtual de Ethereum o Ethereum Virtual Machine
FPGA	Field Programmable Gate Arrays
GPU	Graphics Processing Units

ICAN	Corporación de Internet para la asignación de nombres y números o Internet Corporation for Assigned Names and Numbers
ICO	Oferta inicial de moneda o Initial Coin Offering
IETF	Internet Engineering Task Force
IPFS	Sistema de ficheros interplanetario o InterPlanetary File System
IPO	Oferta pública inicial de acciones o Initial Public Offering
IoT	Internet de las cosas o Internet of Things
KYC	Know Your Customer
LIFO	Last In First Out
MD5	Message Digest 5
NIST	Instituto Nacional de Estándares y Tecnología norteamericano o National Institute of Standards and Technology
P2P	Red entre pares o Peer-to-Peer
P2PK	Pay-to-Public-Key-Hash
P2SH	Pay-to-Script-Hash
PBFT	Practical Byzantine Fault Tolerance
PGP	Pretty Good Privacy
PoA	Proof of Activity
PoS	Prueba de interés o Proof of Stake
PoW	Prueba de trabajo o Proof-of-Work
RGPD	Reglamento General de Protección de Datos
RPCA	Ripple Protocol Consensus Algorithm
SEC	Standards for Efficient Cryptography
SHA	Algoritmo resumen seguro o Secure Hash Algorithm
SMS	Short Message Service
SPV	Verificación de pago simplificado o Simplified Payment Verification
TIC	Tecnologías de la Información y de la Comunicación
TLS	Transport Layer Security
TOWF	Función unidireccional con trampilla o Trapdoor One-Way Function
TSA	Autoridad de sellado temporal o Time Stamping Authority
Tor	The onion router
UNL	Unified Node List
UTXO	Unspent Transaction Output

Bibliografía

Androulaki, E. *et al.* (2018a): "Channels: Horizontal Scaling and Confidentiality on Permissioned Blockchains", en J. López *et al.* (eds.), *Computer Security, 23rd European Symposium on Research in Computer Security (ESORICS, 2018), Proceedings, Part I*, Springer, Cham, pp. 111-131.

— (2018b): "Hyperledger Fabric: a Distributed Operating System for Permissioned Blockchains", *Proceedings of the Thirteenth EuroSys Conference*, ACM, Nueva York, p. 30.

ANSI, American National Standards Institute (2005): "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", National Bureau of Standards, ANSI X9.62.

Aoyagi, J. y Adachi, D. (2018): "Fundamental Values of Cryptocurrencies and Blockchain Technology", Social Science Research Network. Disponible en doi 10.2139/ssrn.3132235.

Arroyo D. *et al.* (2015): "Non-conventional Digital Signatures and Their Implementations-A Review", en Herrero, A. *et al.* (eds.), *International Joint Conference*, Springer, Cham, pp. 425-435.

Bartoletti, M. y Pompianu, L. (2017): "An Analysis of Bitcoin Op_ RETURN Metadata", en A. Kiayias (ed.), *Financial Cryptography and Data Security, 21st International Conference, Revised Selected Papers,* Springer, Cham, pp. 218-230.

Brown, R.G. *et al.* (2016): "Corda: An Introduction", R3 CEV.

Brunton, F. y Nissenbaum, H. (2015): *Obfuscation: A user's guide for privacy and protest*, MIT Press, Cambridge.

Campbell-Verduyn, M. (2017): *Bitcoin and Beyond: Cryptocurrencies, Blockchains, and Global Governance*, Routledge, Nueva York.

Castro, M. y Liskov, B. (1999): "Practical Byzantine Fault Tolerance", MIT Press, Cambridge. Disponible en http://pmg.csail.mit.edu/papers/osdi99.pdf.

Cesarano, F. (2006): *Monetary Theory and Bretton Woods: the Construction of an International Monetary Order*, Cambridge University Press, Cambridge.

Chaum, D. (1983): "Blind Signatures for Untraceable Payments", *Advances in Cryptology-Proceedings of Crypto 82*, 3, Springer, Boston, pp. 199-203.
Chaum, D. y van Heyst, E. (1991): "Group signatures", *Lecture Notes in Computer Science*, 547, pp. 257-265.
Christin, N. (2013): "Traveling the Silk Road: A Measurement Analysis of a Large Anonymous Online Marketplace", en D. Schwabe, *Proceedings of the 22nd international conference on World Wide Web*, ACM, Nueva York, pp. 213-224.
Coleman, J. (1994): *Foundations of social theory*, Harvard University Press, Cambridge.
Cooper, D. *et al.* (2008): "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (Crl) Profile", RFC 5280 (May): 1-151. Disponible en http://dblp.uni-trier.de/db/journals/rfc/rfc5200-5299.html.
Cortes Generales (2003): "Ley 59/2003, de 19 de diciembre, de firma electrónica", BOE-A-2003-23399. Disponible en https://www.boe.es/eli/es/l/2003/12/19/59/con
DuPont, Q. (2017): "Experiments in Algorithmic Governance: A History and Ethnography of The DAO, a Failed Decentralized Autonomous Organization", en M. Campell-Verdyuin, *Bitcoin and Beyond*, Routledge, Nueva York, pp. 157-177.
Enserink, M. y Chin, G. (2015): "The End of Privacy", *Science*, 347 (6221), pp. 453-580. Disponible en http://www.sciencemag.org/content/347/6221.toc.
Field, T. *et al.* (2003): *OECD E-Government Studies the E-Government Imperative*, OECD Publishing.
Franco, P. (2014): *Understanding Bitcoin: Cryptography, Engineering and Economics*, John Wiley & Sons, Chichester.
Fúster Sabater, A. *et al.* (2012): *Criptografía, protección de datos y aplicaciones. Una guía para estudiantes y profesionales*, RA-MA, Madrid.
Gayoso Martínez, V.; Hernández Encinas, L. y Martín Muñoz, A. (2018), *Criptografía con curvas elípticas*, CSIC, Madrid.
Ghosh, A. *et al.* (2007): "Mechanism Design on Trust Networks", *Internet and Network Economics, 3rd International Workshop on Web and Internet Economics, Proceedings,* Springer, Berlín, pp. 257-268.
Hardjono, T. *et al.* (2018): "Towards a Design Philosophy for Interoperable Blockchain Systems", MIT Press, Cambridge, pp. 1-27. Disponible en http://arxiv.org/abs/1805.05934.
Hearn, M. (2016): *Corda: A distributed ledger*, R3 CEV. Disponible en https://www.corda.net/content/corda-technical-whitepaper.pdf.
Hernández Encinas, L. (2016): *La criptografía*, CSIC-Los Libros de la Catarata, Madrid.
Itakura, K. y Nakamura, K. (1983): "A public-key cryptosystem suitable for digital multisignatures", NEC Res, Development, 71, pp. 1-8.
Karame, G. *et al.* (2012): "Double-Spending Fast Payments in Bitcoin", *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 906-17, ACM, Nueva York.
— (2013): "Evaluating User Privacy in Bitcoin", *Financial Cryptography and Data Security, 17st International Conference, Revised Selected Papers*, Springer, Cham, pp. 34-51.

Kiayias, A. *et al.* (2017): "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol", *Advances in Cryptology – CRYPTO 2017, 37th Annual International Cryptology Conference, Proceedings, Part I*, Springer, Cham, pp. 357-388.

Lamport, L. (1998): "The Part-Time Parliament". *ACM Transactions on Computer Systems* (TOCS) 16 (2), ACM, Nueva York, pp. 133-169.

Lamport, L.; Shostak, R. y Pease, M. (1982): "The Byzantine Generals Problem", *ACM Transactions on Programming Languages and Systems* (TOPLAS) 4 (3), ACM, Nueva York, pp. 382-401.

Lewenberg, Y.; Sompolinsky, Y. y Zohar, A. (2015): "Inclusive Block Chain Protocols", *Financial Cryptography and Data Security, 19st International Conference, Revised Selected Papers*, Springer, Heildelberg, pp. 528-547.

Merkle, R. C. (1982): "Method of providing digital signatures", US Patent n. 4309569, 5 enero.

Nakamoto, S. (2008): "Bitcoin: A Peer-to-Peer Electronic Cash System". Disponible en https://bitcoin.org/bitcoin.pdf.

Narayanan, A. *et al.* (2016): *Bitcoin and cryptocurrency technologies*. Princeton University Press, Princeton.

Narayanan, A. y Clark, J. (2017): "Bitcoin's academic pedigree", *Communications of the ACM* 60, 12, pp. 36-45.

NIST, National Institute of Standards and Technology (2002): "Secure Hash Standard (SHS)", NIST Federal Information Processing Standard Publication FIPS180-2.

— (2014): "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", NIST Federal Information Processing Standards, FIPS PUB 202.

Peters, G. W. y Panayi, E. (2015): "Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money", Social Science Research Network, Preprint arXiv:1511.05740, 1-33. Disponible en doi 10.2139/ssrn.2692487.

Rivest, R. (1992): "RFC 1321: The MD5 message-digest algorithm", Internet Request for Comments, 1321, Rump session of Crypto'91.

Rivest, R.; Shamir, A. y Tauman, Y. (2001): "How to leak a secret", *Lecture Notes Computer Science*, 2248, pp. 552-565.

Rogaway, P. (2015): "The Moral Character of Cryptographic Work", IACR Cryptology ePrint Archive, 1162.

Sánchez-Gómez, A. *et al.* (2018): "Review of the Main Security Threats and Challenges in Free-Access Public Cloud Storage Servers", en K. Daimi (ed.), *Computer and Network Security Essentials*, Springer International Publishing, pp. 263-281. Disponible en doi 10.1007/978-3-319-58424-9_15.

Schwartz, D. *et al.* (2014): "The Ripple Protocol Consensus Algorithm", Ripple Labs Inc White Paper 5.

Skolnikoff, E. B. (1994): *The elusive transformation: science, technology, and the evolution of international politics*, Princeton University Press, Princeton.

Strandburg, K. J. (2014): "Monitoring, Datafication and Consent: Legal Approaches to Privacy in a Big Data Context", en J. Lane *et al., Privacy, Big Data, and the Public Good: Frameworks for Engagement*, Cambridge University Press, Nueva York.

Szabo, N. (1994): "Smart Contracts". Disponible en http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html.
Vigna, P. y Casey, M. J. (2016): *The Age of Cryptocurrency: How Bitcoin and the Blockchain Are Challenging the Global Economic Order*, MacMillan, Nueva York.
Wood, G. (2015): "Ethereum: A Secure Decentralised Generalised Transaction Ledger". Disponible en http://yellowpaper.io/.
Wüst, K. y Gervais, A. (2017): "Do you need a Blockchain?", IACR Cryptol, ePrint. Disponible en https://eprint.iacr.org/2017/375.pdf.

¿QUÉ SABEMOS DE?

Blockchain

De la mano del dinero virtual, en los últimos años *blockchain* ha acaparado el foco de interés de la actualidad tecnológica. Blockchain ofrece un mecanismo descentralizado de recolección de trazas inmutables y permite, por ejemplo, la construcción de sistemas financieros alternativos, sin bancos centrales, o de organizaciones autónomas basadas en contratos inteligentes; aquí destacan Bitcoin y Ethereum, con sus monedas virtuales, el *bitcoin* y el *ether*. Esta obra explica los fundamentos de los componentes tecnológicos y criptográficos de blockchain, analiza sus limitaciones en la gestión de la identidad y privacidad, y discute críticamente su papel protagonista en la configuración de la web 3.0.



David Arroyo es ingeniero y doctor de Telecomunicación, y científico titular en el ITEFI (CSIC). Su actividad investigadora se centra en el análisis, diseño y evaluación de sistemas para la protección de la seguridad y de la privacidad de la información. **Luis Hernández** es licenciado y doctor en Matemáticas, e investigador del Departamento TIC en el ITEFI (CSIC). Su investigación incluye la criptografía de clave pública, esquemas de firma digital, autenticación, identificación y compartición de secretos, blockchain, criptobiometría, entre otros. **Jesús Díaz** es ingeniero y doctor en Informática, y *research scientist* en BBVA Next Technologies. Entre sus campos de investigación destacan la criptografía aplicada a la privacidad, el análisis de seguridad de protocolos de comunicaciones y los sistemas distribuidos.

ISBN: 978-84-00-10478-8

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
CSIC
CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS