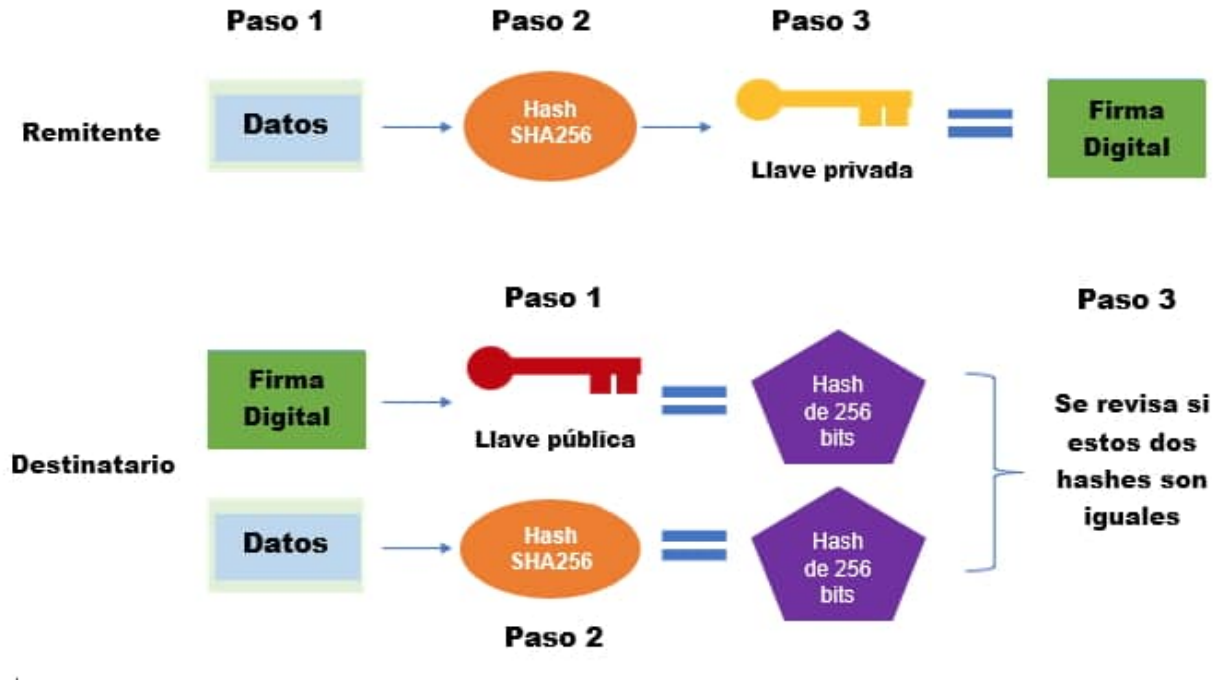
Todo sobre: Blockchain: bloques, transacciones, firmas digitales y hashes

19/11/2019 (https://www.centralbolivariana.org.ve/blockchain-bloques-transacciones-firmas-digitales-hashes/) Francisco Garcia Aaron (https://www.centralbolivariana.org.ve/author/francisco/) BANNER (https://www.centralbolivariana.org.ve/category/banner/), CRIPTO INVERSIONES (https://www.centralbolivariana.org.ve/category/cripto-inversiones/), Economia (https://www.centralbolivariana.org.ve/category/economia-2/)



CBSTinfo.- Las criptomonedas funcionan utilizando cadenas de bloques (blockchain), que son, básicamente, listas crecientes de registros de información cifrada, cada uno encadenado al anterior con criptografía. Esos ‘registros’ son los bloques en los que se asientan y validan las transacciones, pero eso no es lo único que hay dentro de ellos.

A continuación, daremos un paseo por el contenido de un bloque en una blockchain (https://www.criptonoticias.com/informacion/que-es-tecnologia-contabilidad-distribuida-blockchain/), tomando como referencia sobre todo la blockchain de Bitcoin. De esta forma, podremos aproximarnos a su funcionamiento.

Antes de empezar, debes tener en cuenta que este es un artículo para usuarios que ya dominan, al menos, las principales nociones sobre los conceptos de blockchain y criptomonedas. Si aún no lo haces, lo mejor es iniciar con información más básica

(https://www.criptonoticias.com/informacion/que-es-bitcoin/).

Hash criptográfico

Una función hash (https://komodoplatform.com/cryptographic-hash-function/) criptográfica es un algoritmo que cuenta con ciertas propiedades útiles para el cifrado de datos, esto es, **proteger contenidos mediante el uso de claves**. Al aplicarla, se toma un mensaje de cualquier tamaño, se cifra, y se consigue a cambio una cadena alfanumérica única de longitud fija (llamada *digest* o simplemente hash), sin importar el tamaño del mensaje original. Funciona para verificar que, en efecto, se trata de ese mensaje (o transacción) en particular y que no fue modificado antes de su entrega. Si una sola parte, aunque sea un solo punto del mensaje original cambia, el hash (digest) también lo hace de forma radical.

Por ejemplo, usando una herramienta (https://passwordsgenerator.net/sha256-hash-generator/) en línea para cifrar con el algoritmo SHA256 (del que hablaremos más adelante) podemos introducir el siguiente mensaje:

Bitcoin es la primera criptomoneda

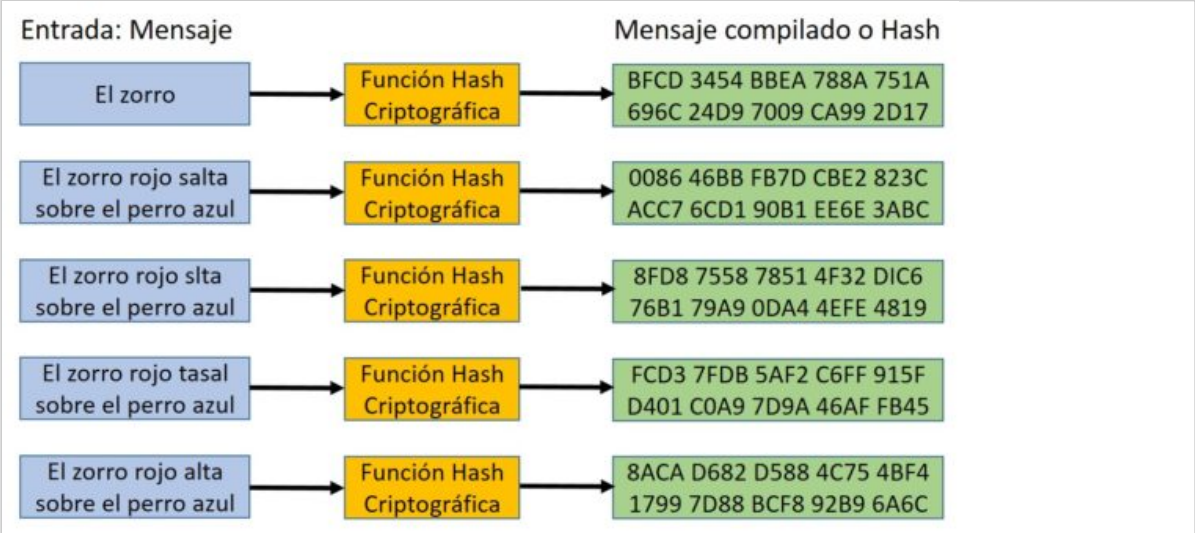
Y conseguir a cambio el siguiente resultado:

91D3081626672039C99F27323895D06B88376312706BF7AB035A662B6C5C0B1B

Si añadiéramos palabras o cambiáramos aunque fuera un punto en el mensaje original, el hash obtenido también cambiaría, aunque continuaría siendo de la misma extensión (64 caracteres). Veamos:

Bitcoin es la primera criptomoneda.

AE0B40E7FC912BCE189A06F3A8069776FB24DCCC493332F2D349F0A470DE1254



Esquema sobre el funcionamiento de un hash criptográfico. Fuente: Wikipedia. Traducción libre.

Nótese que sólo añadimos un punto al final de la frase. Aun así, el resultado es completamente distinto al primero. En otro caso, si las frases volvieran a cifrarse con este mismo algoritmo, pero con otra herramienta, sus digest continuarían siendo los mismos: **una entrada particular produce siempre el mismo resultado único.**

De esa forma, los mensajes se transmiten de manera segura e íntegra, pues es casi imposible averiguar el mensaje original a partir del digest, y, por tanto, tampoco sería posible modificarlo. A esto se le conoce como funciones de un solo sentido o unidireccionales. Podemos profundizar más al respecto.

Hash como función unidireccional

Una función unidireccional, en matemática, se define (https://en.wikipedia.org/wiki/One-way_function) como una función (relación entre los elementos de dos conjuntos) que tiene la característica de ser fácil de calcular, pero difícil de invertir. Nótese que se dice “difícil”, pero no imposible. En realidad, las funciones totalmente unidireccionales en ciencias de la computación aún son sólo una conjetura.

Las funciones hash, sin embargo, se hacen para ser lo suficientemente difíciles de invertir. Sólo así es posible que sean útiles para la criptografía, pues revertirlas tomaría una **cantidad contraproducente (para el atacante) de tiempo y recursos.**

Construir un hash es un proceso matemático complejo, pero una de las formas de hacerlo (http://www.cs.cornell.edu/courses/cs312/2008sp/lectures/lec21.html) es mediante funciones modulares, que asegurarían su ‘unidireccionalidad’. Puesto de forma sencilla, las funciones modulares producen el remanente de una división. Así, por ejemplo, 10 mod 3 = 1, porque 10 dividido entre 3 es 3 más un remanente de 1. En otra forma, 3 cabe 3 veces en 10, y queda un añadido de 1.

Ahora, digamos que para construir un hash tenemos una llave privada (X) mod 5 = 2. Sólo tú sabrías el valor de X, el cual, digamos que es 27, porque dividido entre 5 es igual a 5 más un remanente de 2. Supongamos, además, que 5 son los datos de una transacción que realizaste y 2 es el hash resultante de esa transacción. Aunque estos últimos datos sean públicos, es casi imposible averiguar que tu llave privada es 27, pues para llegar al resultado de 2 utilizando también 5 existen infinitas posibilidades. Tu “X” pudo ser 7, 52, 23390787 u otro: averiguarlo es casi imposible.

En la práctica, este mismo principio se aplica a algoritmos más sofisticados y cantidades de datos muy superiores, por lo que la dificultad de averiguar el dato de origen aumenta muchísimo más. Los datos resguardados por una función hash están seguros.

Propiedades de una función hash segura

Existen varios tipos de funciones hash, pero todos ellos, para ser seguros, deben poseer cuatro características principales:

1. **Computacionalmente eficiente**

Las funciones hash se utilizan en computadores, así que, aunque suene un poco obvio, estos computadores deben ser capaces de llevar a cabo la labor matemática necesaria para crear un hash en un período de tiempo muy corto. Si no fuera así, cada proceso que involucre la emisión de un hash tardaría demasiado y sería poco práctico utilizarlos. En la actualidad, esto no es un problema, pues una computadora promedio puede realizar la tarea en menos de un segundo.

2. **Determinista**

Esto implica que el mismo mensaje (entrada) debe producir siempre el mismo digest (salida) cada vez que sea utilizado o consultado. Si la función produjera un resultado al azar cada vez sería inútil, pues no serviría para verificar que se trata del mensaje original. El punto de un hash, para el caso que nos ocupa, es corroborar que una firma digital sea auténtica sin tener acceso a la llave privada.

3. **Resistente a preimagen**

Significa que la salida no debe revelar ningún dato en absoluto sobre la entrada. Es por eso que un hash debería tener siempre la misma longitud en el digest, independientemente del tamaño del mensaje. Tampoco debe darse ninguna pista sobre el contenido de tal mensaje, por lo que, al más mínimo cambio, el hash resultante debe ser por completo distinto.

4. **Resistente a colisión**

Dos (o más) entradas diferentes no deberían producir la misma salida (digest). Es necesario mencionar que ninguna función hash está por completo libre de colisión: es una simple probabilidad matemática. Las salidas tienen una longitud determinada, a diferencia de las entradas que pueden ser de cualquier tamaño, así que el número de resultados es finito y, por tanto, propenso a colisión. Sin embargo, esta probabilidad es bastante pequeña, y la meta de cualquier función hash es hacerla lo más pequeña posible.

Sobre los tipos de hash

Existen numerosos tipos de algoritmos (https://en.wikipedia.org/wiki/Cryptographic_hash_function#Cryptographic_hash_algorithms) para crear hash en distintas plataformas, con diversas funciones, desde autenticación de documentos y verificación de contraseñas, hasta verificación de firmas digitales y, por supuesto, minería de criptomonedas (https://www.criptonoticias.com/informacion/que-es-la-mineria-de-bitcoins-criptomonedas/). Entre los que continúan siendo efectivos, podemos mencionar el BLAKE2, MD6, Streebog y, especialmente, la serie SHA (Secure Hash Algorithm).

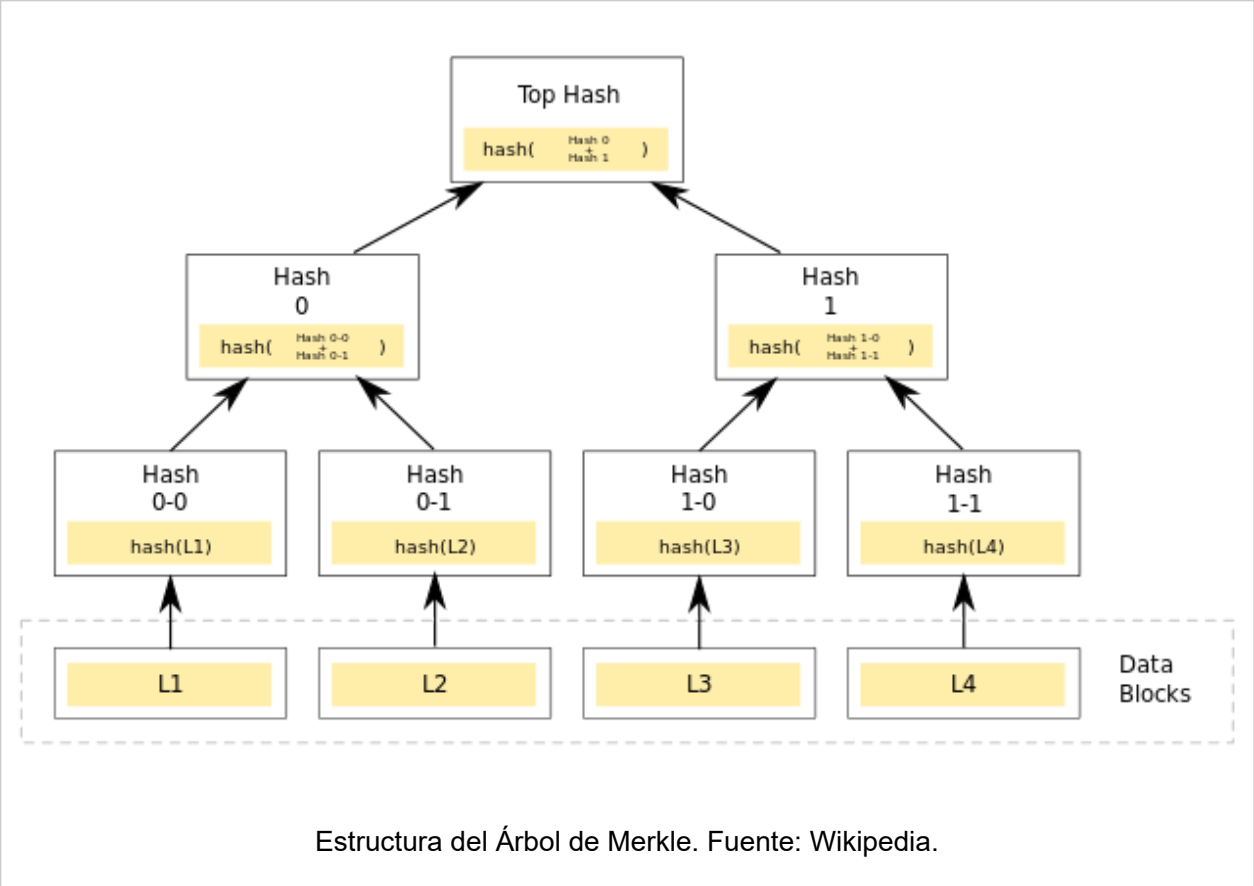
La serie SHA fue diseñada por la Agencia Nacional de Seguridad (NSA) estadounidense e incluye el SHA-256 (https://es.wikipedia.org/wiki/SHA-2), ampliamente utilizado en el criptomundo debido a que fue el algoritmo escogido por Satoshi Nakamoto para hacer funcionar la blockchain de Bitcoin, de la cual, a su vez, se han derivado otras criptomonedas que han conservado el mismo algoritmo (como Peercoin (https://cryptorival.com/algorithms/sha256/)).

El SHA-256 produce un digest de 256 bits y 64 caracteres. Por otro lado, tenemos el SHA-3 (https://en.wikipedia.org/wiki/SHA-3), incluido en el mismo estándar, pero con estructura diferente, pues produce digest de tamaño arbitrario. Este es el algoritmo utilizado para el sistema Ethash (https://en.wikipedia.org/wiki/Ethash) de Ethereum.

Árbol de Merkle

Sabiendo ya qué es un hash, hay que apuntar que las transacciones en una blockchain usan hashes para cifrar los datos, pero estas líneas alfanuméricas no sólo “flotan” libremente en una nube digital. **Se ordenan de forma estricta y se resumen** a medida que aumenta la cadena, proporcionando un método seguro, rápido y ligero para verificar los datos. Con ese propósito está implementado el Árbol de Merkle.

Este concepto, también conocido como Árbol de Hash, se trata de una estructura de datos en árbol (https://en.wikipedia.org/wiki/Tree_structure), es decir, aquella que simula un orden jerárquico de arriba hacia abajo, empezando por un único valor “raíz” que se va dividiendo en valores “hoja”, como un árbol al revés. En el caso del Árbol de Merkle (https://en.wikipedia.org/wiki/Merkle_tree), estos valores son todos hashes, y en una blockchain, esos hashes provendrían de los datos de las transacciones.



Estructura del Árbol de Merkle. Fuente: Wikipedia.

El nombre del árbol criptográfico proviene de su inventor, Ralph Merkle (https://en.wikipedia.org/wiki/Ralph_Merkle), un científico computacional estadounidense que patentó esta estructura en 1979. Cabe resaltar que Merkle, a 2019 con 67 años de edad, también es el inventor del hash criptográfico y uno de los inventores de la criptografía de llave pública.

Raíz de Merkle

El principal propósito del árbol de hash es crear una raíz de Merkle (https://komodoplatform.com/whats-merkle-tree/), es decir, el valor raíz que mencionamos antes. En este caso, se podría creer que de este valor provienen los valores “hoja”, pero, en realidad, **el valor raíz es un resumen de todos los valores hoja.**

Este resumen se crea agrupando todos los hashes de las transacciones en pares a los que, a su vez, les será aplicada de nuevo la función hash criptográfica pertinente para crear un nuevo digest que equivale a ambos. Si acaso el número de entradas fuera impar, la última se copiaría a sí misma y se emparejaría con esa copia para permitir el proceso. Los digest resultantes volverán a organizarse por pares y a repetir la misma técnica, hasta que sólo quede una única línea hash como resumen de todas las que pasaron por este proceso de fusión. Esa es la raíz de Merkle, y hay una sola por bloque en una blockchain.

> Así, por ejemplo, si en un bloque están contenidas 512 transacciones, el árbol de Merkle se encargaría de agruparlas en 256 pares, que se reducirían luego a 128, luego a 64, de ahí a 32, después 16, 8, 4, 2 y la última. Una sola línea alfanumérica se queda allí para representar esas 512 transacciones, en lugar de recargar el bloque con 512 hashes.

Recordemos que cada transacción en una blockchain tiene su propio hash, y, si hablamos de hashes creados con SHA-256, cada uno pesa 32 bytes. Volviendo al ejemplo de los 512 hashes, ese bloque soportaría entonces (además de otros datos) 16.384 bytes cuando, usando la raíz de Merkle, puede contener sólo 32 bytes. Ella contiene al resto matemáticamente, así que no es necesario incluirlos todos.

Como vemos, a largo plazo, con miles y miles de transacciones realizadas por usuarios de todo el mundo, el espacio para almacenar completa una blockchain **podría llegar a convertirse en un problema** de no ser por la raíz de Merkle, que resume una gran cantidad de datos en un solo hash.

El propio Satoshi Nakamoto (https://www.criptonoticias.com/informacion/quien-es-satoshi-nakamoto/) en el Libro Blanco (https://criptonoticias.com/documentos/libro-blanco-bitcoin-satoshi.pdf) de Bitcoin explica que una vez que una transacción está enterrada bajo suficientes bloques, las transacciones anteriores a esa pueden ser descartadas para salvar espacio. Para lograrlo sin romper el hash que las asegura y conecta al resto de la blockchain, los viejos bloques se compactan con el árbol de Merkle, mientras que los hashes interiores que formaron la raíz no necesitan ser conservados.

> Una cabecera de bloque sin transacciones sería de unos 80 bytes. Si suponemos que los bloques se generan cada 10 minutos, 80 bytes * 6 * 24 * 365 = 4,2MB por año. Con sistemas informáticos siendo típicamente vendidos con 2 GB de RAM a partir de 2008, y la Ley de Moore prediciendo un crecimiento actual de 1,2 GB por año, el almacenamiento no debería ser un problema incluso si las cabeceras de bloque deben mantenerse en la memoria.
>
> Satoshi Nakamoto, libro blanco de Bitcoin.

Firmas digitales

Antes de definir directamente qué son las firmas digitales por sí mismas, es necesario explorar el concepto de la criptografía asimétrica (https://en.wikipedia.org/wiki/Public-key_cryptography) o de llave pública, pues esta es parte esencial de su funcionamiento. Se trata de un sistema criptográfico que genera para sus usuarios, mediante la aplicación de algoritmos específicos, **dos "claves" o "llaves":** una pública, que puede ser distribuida a cualquiera sin riesgo, y otra privada, que sólo debe ser conocida por su dueño. Estas "llaves" son líneas alfanuméricas de determinada extensión. No hay gran diferencia entre los formatos de una y de otra: para un usuario promedio, cuál es la pública y cuál la privada es decidido por el sistema que use para crearlas.

Utilizando este sistema, la persona remitente puede cifrar cualquier mensaje usando la llave pública del destinatario. Una vez ese mensaje esté cifrado con esa llave pública, **sólo la llave privada de ese receptor puede descifrarlo**, pues ambas llaves están relacionadas matemáticamente. En este sentido, la clave pública puede ser comparada a una dirección de correo electrónico, mientras que la privada sería la contraseña de ese correo.

> En el caso de las criptomonedas (https://www.criptonoticias.com/etiquetas/criptomonedas), la criptografía de llave pública se utiliza en cualquier monedero para intercambiar fondos. Las direcciones públicas de cartera que otorgamos para recibir fondos son una versión en hash (https://www.dummies.com/software/other-software/bitcoin-public-private-keys/) de la llave pública, mientras que las doce (o más) palabras que proporcionan muchos software como "semilla" para recuperar los fondos sirven para derivar la llave privada.

En medio de este proceso es que entran las firmas digitales. Estas son (https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/digital-signatures), básicamente, **la combinación entre una llave privada y un hash de los datos a firmar** (como los de una transacción), lo que otorga una identificación digital única para establecer la autenticidad e integridad del mensaje, sin revelar la llave privada del firmante.

Para lograr ese propósito, utiliza típicamente tres algoritmos: uno para la generación de una clave privada al azar, de donde se deriva una clave pública correspondiente; otro para producir la firma como tal basándose en la llave privada y los datos, y un último que determina si el mensaje es o no auténtico basándose en los datos, la clave pública y la firma. Cada transacción realizada en una blockchain necesita, adicional a otros requerimientos, de la firma de su remitente y de la verificación de esa firma por parte del destinatario y la red para volverse válida.

Proceso de una firma digital

En este ejemplo, nos centraremos en la creación y recorrido de la firma digital como tal. Es importante mencionar que en una transacción con criptomonedas intervienen otros elementos y se validan otros datos, que mencionaremos más adelante.

Esquema sobre el funcionamiento de una firma digital. Fuente: Blair Marshall / Medium. Traducción libre.

Por ahora, digamos que Alice ya generó su par de llaves (o lo que es lo mismo, consiguió una cartera digital) y quiere hacer una transacción (https://medium.com/@blairlmarshall/how-does-a-bitcoin-transaction-actually-work-1c44818c3996) con bitcoins hacia Bob. Para que esta transacción sea válida, debe venir firmada por la llave privada de Alice, y Bob debe verificar que esa firma sea auténtica. Los pasos que sigue el sistema para lograr completar ese proceso son los siguientes:

1. Por parte de Alice, se toman los datos de la transacción y se utiliza el algoritmo SHA-256 (ya que hablamos de Bitcoin (https://www.criptonoticias.com/informacion/que-es-bitcoin/)) para cifrarlos en un hash de 64 caracteres.
2. El hash obtenido se “combina” o se “firma” con la llave privada de Alice, dando como resultado (https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm) **dos números conocidos como r y s**, con un peso variable entre 71 y 73 bytes. Esa, de forma más concreta, es la firma digital.
3. Se envían a Bob entonces los datos de la transacción, la firma digital y la clave pública de Alice.
4. Utilizando la llave pública de Alice, el sistema por parte de Bob podrá descifrar la firma digital (sin revelar la clave privada de Alice) para conseguir el hash de 64 caracteres correspondiente a los datos de la transacción, que antes Alice había cifrado con SHA-256 y combinado con su llave privada.
5. Como los datos de la transacción también fueron recibidos por Bob, el sistema repite el proceso de cifrarlos con SHA-256 para conseguir el hash correspondiente.

6. Se verifica que los hashes de los pasos 4 y 5 **sean exactamente iguales**. Si no lo son, esto indicaría que alguien alteró los datos o la clave pública de Alice no corresponde con su clave privada. Por tanto, la transacción sería inválida, ya que fue modificada durante su tránsito o no corresponde al dueño de los fondos.

Propiedades de una firma digital segura

Existen varios algoritmos (https://en.wikipedia.org/wiki/Digital_signature) para crear firmas digitales: en el caso de Bitcoin, se usa el algoritmo de firma digital de curva elíptica (ECDSA (https://medium.com/@blairlmarshall/how-does-ecdsa-work-in-bitcoin-7819d201a3ec)), que toma la matemática detrás de los campos finitos y las curvas elípticas para generar las llaves públicas a partir de las privadas. Todos los algoritmos, sin embargo, deberían proveer las siguientes características para otorgar la seguridad necesaria entre los participantes:

1. **Autenticación**

Utilizar una firma digital debería asegurar (https://en.wikipedia.org/wiki/Message_authentication) al destinatario que el mensaje (o transacción) proviene de un remitente en específico, cuya identidad puede ser verificada más allá de una firma escrita: con matemática. La firma generada se basa en datos precisos y es casi imposible de falsificar.

2. **Integridad**

Esta propiedad garantiza que los datos llegarán intactos (https://en.wikipedia.org/wiki/Data_integrity) al remitente, es decir, que no serán modificados de ninguna forma durante su transferencia. En teoría, los datos podrían ser modificados sin ser vistos por algún hacker habilidoso, pero si esto sucede la firma cambiaría también y, por tanto, dejaría de ser válida.

3. **No repudio**

El usuario que utilizó su firma digital personal no puede negar (https://en.wikipedia.org/wiki/Non-repudiation) que lo hizo. Usualmente, el no repudio se trata de un concepto legal: una vez firmado un documento, el autor no debería poder negar que él se comprometió mediante su firma. Por tanto, las firmas digitales también son vinculantes y auditables. Entre los países que las reconocen legalmente se encuentran Estados Unidos, Suiza, Brasil, México, India, Turquía y la Unión Europea.

Transacciones

Las transacciones **son agrupaciones de datos con firma digital** que almacenan (https://en.bitcoin.it/wiki/Transaction) las transferencias de fondos entre los remitentes (conocidos como *entradas* dentro de esas agrupaciones) y los destinatarios (salidas). Se transmiten a la red y, a medida que son validadas, se juntan y ordenan para formar los bloques en una blockchain.

En Bitcoin (y en muchas otras criptomonedas), las transacciones deben autorizarse con el uso de firmas digitales, pero no están cifradas; así que es posible ver los datos que incluyen a través de un explorador (https://www.criptonoticias.com/informacion/anatomia-exploradores-blockchain-cadena-

bloques/) de la cadena.

Transacciones pendientes de gasto (UTXO)

Antes de avanzar más dentro de la estructura de una transacción en la blockchain, es necesario que sepamos de qué trata una UTXO (https://komodoplatform.com/whats-utxo/) (Unspent Transaction Output). **Una UTXO, o transacción pendiente de gasto, es una salida (de fondos) que un usuario recibe para poder gastar en el futuro como una entrada para alguien más.** El balance total en la cartera de cualquier usuario está compuesto por UTXOs de distinto tamaño, o bien por una sola que puede recibir a cambio cuando es gastada a menos de su totalidad.

Para ponerlo en perspectiva, podemos pensar que las UTXO **son el equivalente a monedas o billetes individuales dentro de la blockchain**. Tal como el sistema de dinero en efectivo está diseñado sólo con billetes o monedas de determinado monto (en el dólar, por ejemplo, sólo existen seis billetes distintos), que se combinan para formar nuevas cantidades o se entregan en su totalidad esperando recibir el remanente de la compra a cambio, en la blockchain también se combinan distintas UTXO, o bien se otorga un cambio a una UTXO más grande cuando se realiza una transferencia.

De este modo, supongamos que tienes $100 en bitcoins dentro de tu cartera (https://www.criptonoticias.com/informacion/como-elegir-monedero-cartera-bitcoin-criptomonedas-criptoactivos/) y necesitas pagar 45$. Como usuario común no lo ves a simple vista, pero esos 100$ puede que estén conformados por varias UTXO: quizás dos de 50$, o cuatro de 25$. En este último caso, al realizar el pago de 45$, estarías enviando en realidad dos UTXO de $25 y recibiendo otra a cambio de 5$.

A diferencia del dinero en efectivo, **las UTXO pueden ser de cualquier cantidad**, dependiendo del monto intercambiado entre los participantes. Otra divergencia con el efectivo es que, además de los fondos que damos para un pago o una compra, debemos incluir también un pago de una comisión para quienes mantienen la red ayudando a validar las transacciones para incluirlas en nuevos bloques (los mineros).

Así, continuando con el ejemplo anterior, en realidad, la UTXO que recibirías de vuelta al final no sería de 5$, sino de $4,5, pues el total a pagar sería de $45,5 ($45 de pago + $0,5 de comisión, por ejemplo). Según sea la criptomoneda y también según las circunstancias del momento (pues, en ocasiones, las cadenas de bloques se saturan (https://www.criptonoticias.com/sucesos/mientras-core-corre-la-arruga-nace-un-mercado-de-comisiones-bitcoin-por-demora-en-transacciones/)), la cantidad pagada en comisión varía, aunque suele ser de mucho menos de un dólar.

Dado que una sola UTXO puede incluir cualquier cantidad, incluso algunas muy pequeñas que se repitan de forma indefinida con su peso respectivo; a largo plazo, el espacio en la cadena puede llegar a ser un problema. Por ello, los desarrolladores de las carteras digitales deben mantenerlas en tamaños eficientes que ocupen el menos espacio posible para permitir mejores velocidades de procesamiento. Así, si tu balance es de 100$, es mejor tenerlo dividido en dos UTXO de 50$ o en

cuatro de 25$ que en 100 UTXO de 1$. Si volvemos a imaginar las UTXO como billetes en una cartera física, el problema de tener demasiadas UTXO de poco valor en un solo balance queda más claro.

Es importante tener en cuenta que, más que de monedas, **toda la blockchain es una red de UTXO que esperan a ser desbloqueadas** y enviadas a alguien más como una nueva UTXO.

Estructura de una transacción

Para hacernos una idea de la estructura de una transacción, podemos revisar de cerca cómo se compone (https://es.wikipedia.org/wiki/Anexo:Estructura_de_transacciones_y_bloques_en_Bitcoin) una de ellas en la blockchain de Bitcoin. Puede dividirse en tres partes: el encabezado, las entradas y las salidas.

Tamaño	Campo	Descripción
4 bytes	Versión	Reglas a las cuales se apega la transacción
1-9 bytes	Total de entradas	Número de entradas que se incluyen
Variable	Entradas	Una o más entradas de la transacción
1-9 bytes	Total de salidas	Número de salidas que se incluyen
Variable	Salidas	Una o más salidas de la transacción
4 bytes	Bloqueo	Fecha en formato UNIX o un número de bloques

Estructura de una transacción en Bitcoin. Fuente: Wikipedia

El encabezado se compone de cuatro partes: el hash de la transacción, la versión del software que debería usarse para validar ese bloque, el número de entradas y salidas y una fecha o bien una altura de bloque (cualquiera de las dos) para indicar cuando esa transacción fue añadida a la cadena.

Las entradas incluyen el hash de la salida previa apuntando hacia la o las UTXO disponibles, un índice de la lista de salidas de la transacción previa para identificar la que puede gastarse en la nueva entrada y el ScriptSig (https://www.mycryptopedia.com/scriptpubkey-scriptsig/), un “programa simple” de desbloqueo que solicita cierta condición para acceder a los fondos. La condición principal es la llave privada personal del destinatario.

Las salidas, a su vez, incluyen el monto a pagar en satoshis y el ScriptPubKey, el par contrario del ScriptSig que es el encargado de bloquear los fondos con la clave pública del destinatario para que sólo él pueda desbloquearlos después con su clave privada.

Una vez realizada la transacción, esta se envía a los mineros, que son los encargados de validarla, entre otros pasos, comparando ambos Script.

Bloque completo

El bloque (https://dev.to/damcosset/blockchain-what-is-in-a-block-48jo) de una blockchain se trata de **un “contenedor” de datos de tamaño variable**. La mayor parte de esos datos lo conforman las transacciones (en Bitcoin, un promedio (https://www.blockchain.com/es/charts/n-transactions-per-block) de 2.188), que a su vez utilizan hashes, firmas digitales y UTXO para completarse, como ya

hemos visto. Además, dentro de un bloque encontramos su cabecera, en la cual registra la metadata del propio bloque; es decir, información técnica sobre su composición y validación dentro de la cadena.

En seguida veremos qué conforma esa cabecera, pero antes es necesario pasar por otro concepto: el nonce.

Nonce y minería

Debemos empezar afirmando que cada bloque (https://medium.com/coinmonks/blockchain-for-beginners-what-is-blockchain-519db8c6677a) posee una identificación única en forma de hash. Este se crea pasando la cabecera del bloque a través del algoritmo SHA-256 (en el caso de Bitcoin). Dentro de esa cabecera se encuentra el hash del bloque anterior, así que, de forma automática, ambos bloques quedan entrelazados.

Sin embargo, **no basta con sólo darle cualquier hash al bloque** para que este se vuelva válido: tiene que ser un hash muy específico, que inicie con un número consecutivo de ceros, pues debe ser igual o estar por debajo de cierto valor denominado Objetivo (https://en.bitcoin.it/wiki/Target) (target): un número de 256 bits (bastante largo) determinado por la dificultad establecida por el sistema. Encontrar ese hash aceptable para hacer válido un bloque es lo que se llama **minería de criptomonedas**, y se implementa con la intención de hacer casi imposible la modificación de la cadena de hashes.

Ahora bien, ¿cómo se puede encontrar ese hash aceptable para que el bloque se vuelva válido? Como mencionamos, se supone que ese hash debe salir de la cabecera del bloque. Pero allí se presenta un problema, porque todos los datos de esa cabecera son esenciales y no pueden ser modificados: implicaría cambiar, por ejemplo, los montos de una transacción. ¿Qué tal si entonces el hash no coincide con el Objetivo? Para que coincida, tendrían que cambiarse los datos en la cabecera de alguna manera, pero cada dato es vital.

Allí es donde entra el nonce (https://en.wikipedia.org/wiki/Cryptographic_nonce): se trata de un número por completo al azar que es añadido a la cabecera del bloque como un pequeño dato adicional, sin más propósito que **ser cambiado una y otra vez por los mineros para poder encontrar un hash válido**. Si el primer nonce no funciona, se quita y se añade uno nuevo, hasta dar con una hash válido que satisfaga la condición de dificultad de la red (esté dentro del objetivo). Una vez hecho este proceso, si el hash del bloque fuera cambiado, la red se daría cuenta fácilmente gracias a las propiedades del hash, el bloque se invalidaría y se desencadenaría de la blockchain.

Más que de número de intentos, la minería se trata de suerte, pues cada nonce es al azar, así que otorga también un número al azar. No obstante, aquellos mineros que cuenten con mejores equipos tendrán más oportunidades, pues, después de todo, el número y la velocidad de esos intentos aumenta sus probabilidades de hallar el hash correcto en el menor tiempo posible.

Cabecera de bloque

Sabiendo todo lo anterior, podemos comprender entonces cómo se conforma la cabecera de bloque en una blockchain. Incluye (http://learnmeabitcoin.com/guide/blocks) seis datos:

Versión: se trata del número que indica el nivel de desarrollo del software en el momento en que el bloque fue minado. Las computadoras lo utilizan para leer el contenido de cada bloque de manera correcta. El software de Bitcoin, por ejemplo, ha tenido (https://bitcoin.org/en/version-history) para 2019 unas 56 versiones sólo en el cliente Bitcoin Core.

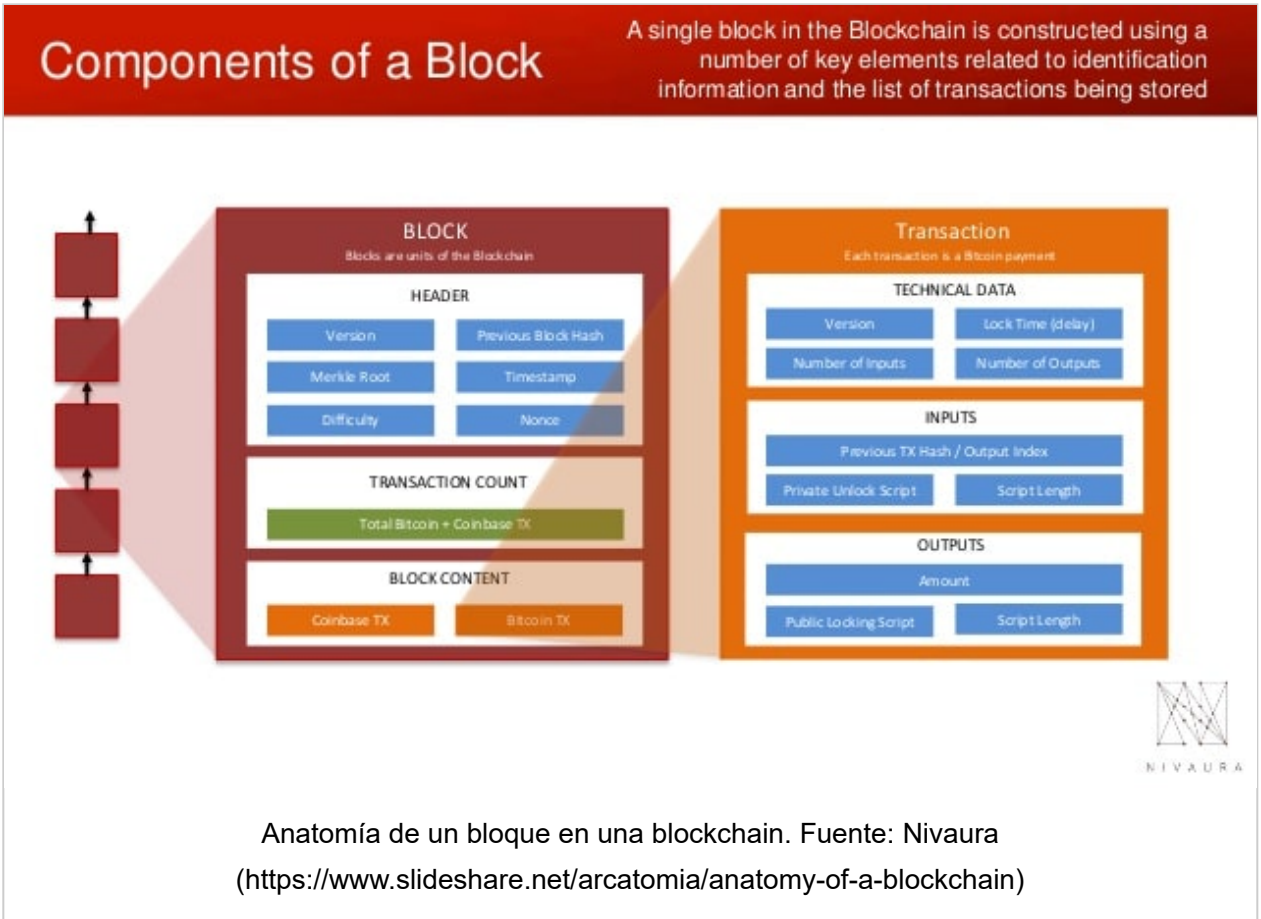
Hash del bloque anterior: es una larga línea alfanumérica que empieza con varios ceros. En Bitcoin, tiene 64 caracteres. Ya vimos cómo se forma.

Raíz de Merkle: como explicamos antes, todas las transacciones en el bloque se unen en un solo hash, que es esta raíz.

Marca de tiempo: indica el momento exacto en que fue minado el bloque. En Bitcoin, se pone el número (https://es.wikipedia.org/wiki/Anexo:Estructura_de_transacciones_y_bloques_en_Bitcoin) de segundos pasados desde enero de 1970.

Objetivo (target): es el número de 256 bits que indica a los mineros cuál puede ser el hash correcto.

Nonce: número adicional al azar que los mineros utilizan para encontrar un hash válido para el bloque.



Anatomía de un bloque en una blockchain. Fuente: Nivaura (https://www.slideshare.net/arcatomia/anatomy-of-a-blockchain)

Otros datos

Además de la cabecera y las transacciones, un bloque (https://en.bitcoin.it/wiki/Block#Block_structure) también contiene otros datos para su funcionamiento dentro de él. Veamos:

Número mágico: en programación, se trata (https://www.anintegratedworld.com/unravelling-the-mysterious-block-chain-magic-number/) de un número constante que se utiliza para identificar el formato de un archivo o protocolo. En el caso de una blockchain, este número sirve para identificar cuándo empieza y cuando termina un bloque. En Bitcoin es siempre el mismo: 0xD9B4BEF9, y pesa 4 bytes.

Tamaño de bloque: número en bytes para indicar el volumen del bloque. A su vez, ese número pesa 4 bytes en Bitcoin.

Contador de transacciones: se representa mediante un número íntegro positivo de extensión variable. En Bitcoin, pesa de 1 a 9 bytes.

En total y en orden, un bloque cuenta entonces con cinco secciones:

1. Número mágico
2. Tamaño de bloque
3. Cabecera (que a su vez contiene 6 secciones)
4. Contador de transacciones
5. Transacciones (pueden ser miles)

En este sentido, podríamos decir que un bloque es menos un “bloque” como tal y más **una agrupación determinada de datos encadenados a otros con criptografía**. A su vez, estos bloques de información (financiera, en el caso de las criptomonedas) se van encadenando secuencialmente hasta el infinito, creando una cadena criptográfica básicamente irrompible.

(https://www.centralbolivariana.org.ve/tag/blockchain-bloques/)

(https://www.centralbolivariana.org.ve/tag/firmas-digitales/)

(https://www.centralbolivariana.org.ve/tag/hashes/)

(https://www.centralbolivariana.org.ve/tag/transacciones/)

DEJA UNA RESPUESTA

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

Guarda mi nombre, correo electrónico y web en este navegador para la próxima vez que comente.

PUBLICAR EL COMENTARIO

Este sitio usa Akismet para reducir el spam. Aprende cómo se procesan los datos de tus comentarios (https://akismet.com/privacy/).

« Ejército de China planea usar blockchains para administrar datos del personal militar (https://www.centralbolivariana.org.ve/ejercito-china-planea-usar-blockchains-administrar-datos-del-personal-militar/)

Piyaron en un audio a Francisco Santos y Claudia Blum (aquí lo que dijeron +audio) » (https://www.centralbolivariana.org.ve/piyaron-en-un-audio-a-francisco-santos-y-claudia-blum-aqui-lo-que-dijeron-audio/)

Search...

Tweets por @CBSTinfo

CBST INFO
@CBSTinfo

#SemanaDeLucha con @SoyRevolucion4 Desde el complejo Cementero Pertigalete encuentro con los trabajadores cemnteros y petroleros
@Mtulio2 @hernandbm en el marco del @motorconstr
Tomas Rincón *Nicolás Maduro*

11h

CBST INFO
@CBSTinfo

#SemanaDeLucha de @SoyRevolucion4 Reunion de Trabajado con el Ministro

Insertar Ver en Twitter

OPINION

La Voz Obrera (Semanario de los Trabajadores de Venezuela #1) (https://www.centralbolivariana.org.ve/la-voz-obrera-semanario-los-trabajadores-venezuela-1/)Francisco Garcia Aaron
Aquí te dejamos el numero 1 del semanario que quiere convertirse en La voz de la Clase Obrera Venezolana Bajalo Aqui: VOZ OBRERA edición 1

Programa para You Tobe: Concejos Productivos #1 Para Los CPTT (https://www.centralbolivariana.org.ve/programa-you-tobe-concejos-productivos-1-los-cptt/)Francisco Garcia Aaron

(https://www.centralbolivariana.org.ve/programa-you-tobe-concejos-productivos-1-los-cptt/) https://www.youtube.com/watch?v=-edfgVW098Q&feature=youtu.be Consejos Productivos es un programa para redes sociales conducido por el Constituyente y dirigente sindical Francisco García, en esta oportunidad participaran como personajes invitados, el diputado Francisco Torrealba, Vicepresidente para la CLASE OBRERA DEL PSUV y el ministro para el proceso social del trabajo Eduardo Piñate* Suscríbete a Nuestro Canal de YOU TOBE.

---

31 años “El Caracazo” es oportuno recordarlo para los que piden guerra e invasión en Venezuela (https://www.centralbolivariana.org.ve/31-anos-caracazo-oportuno-recordarlo-los-piden-guerra-e-invasion-venezuela/)Francisco Garcia Aaron
Es oportuno que aquel genocidio grotesco del cual hoy se cumplen 31 años, sirva para recordarles a los que hoy piden intervención militar, a los que hoy piden guerra; que para eso hay que estar preparado para; Meterse 5 días en una trinchera, comer, defecar y dormir en ella, aprender a tener los pies mojados …

(https://www.centralbolivariana.org.ve/category/opinion-2/)

POLÍTICA (HTTPS://WWW.CENTRALBOLIVARIANA.ORG.VE/CATEGORY/POLITICA-POLITICA-2/)

REGIONALES (HTTPS://WWW.CENTRALBOLIVARIANA.ORG.VE/CATEGORY/REGIONALES/)

VÍDEO (HTTPS://WWW.CENTRALBOLIVARIANA.ORG.VE/TYPE/VIDEO/)

PRESIDENTE MADURO: A MÁS SANCIONES, MÁS ELECCIONES PARA VENEZUELA (HTTPS://WWW.CENTRALBOLIVARIANA.ORG.VE/PRESIDENTE-MADURO-MAS-SANCIONES-MAS-ELECCIONES-VENEZUELA-2/)