Universidad de Buenos Aires

Facultades de Ciencias Económicas,

Cs. Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final de la Especialización

Criptomonedas
Estudio de las redes de Bitcoin y Ethereum

Autor:

Eladio José Cousiño Godoy

Tutor del Trabajo Final:

Pedro Hecht

2018

Cohorte 2017

Declaración jurada de origen de los contenidos

“Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual”.

_____

Eladio José Cousiño Godoy

DNI: 95736769

Resumen

El presente trabajo de investigación recopila información referente a las criptomonedas Bitcoin y Ethereum: elementos, funcionamiento, primitivas criptográficas, vulnerabilidades y ataques registrados a ambas redes. Se inicia con una reseña histórica del concepto de criptomonedas. Luego se estudian las características del protocolo de Bitcoin y del de Ethereum, enfocándose en la implementación del blockchain de cada uno de los protocolos y las primitivas criptográficas utilizadas. Posteriormente se describen las vulnerabilidades existentes en los protocolos así como los ataques más relevantes registrados. Finalmente, se realiza una valoración sobre el futuro de las criptomonedas y se concluye sobre el estado actual de las criptomonedas, las problemáticas con las que se enfrentan y se define cuál protocolo es mejor al momento de redactar este trabajo.

Palabras clave: Bitcoin, Ethereum, criptomonedas, criptografía, blockchain, vulnerabilidades.

Índice General

Agradecimientos

A mi familia y amigos por todo el apoyo incondicional brindado durante esta especialización.

1- Dinero Digital: Antecedentes y Objetivos

La historia del dinero digital comienza con el auge del comercio electrónico, el cual introdujo una serie de variantes sobre la forma tradicional de comercio: las transacciones se realizaban por una red, donde la ubicación física del vendedor es difícilmente identificable por el cliente o comprador y las actividades son susceptibles de ser observadas por entidades que no estén involucradas en la operación.

Para realizar transacciones en la red, se precisó establecer credibilidad: en el caso del comprador, debe conocer la identidad del vendedor, y en el caso de este último, debe contar con la garantía de que cobrará el importe correspondiente a su mercancía a pesar de no conocer la identidad del comprador. La solución a esta problemática fue el registro de los datos pero a su vez, estos registros pusieron en peligro el anonimato de los involucrados en el acto de comerciar[1].

La propiedad clave del dinero es la anonimidad: al extraer dinero de un banco, es imposible para el banco saber qué y cuándo se compra con ese dinero y el vendedor no sabe quién le está comprando. En contraste, cuando se compra con tarjetas de crédito en línea, se debe proveer información al vendedor sobre quién está comprando y se debe proveer información a la compañía de tarjetas de crédito sobre a quién se está comprando. Esta problemática de invasión de privacidad al realizar compras utilizando medios de pago electrónicos deriva en la aparición del dinero digital.

Se denomina dinero digital/criptomoneda/criptomoneda a aquel que utiliza una red *peer-to-peer* como medio de intercambio[1], descentralización[2], y utiliza fundamentos criptográficos para garantizar la seguridad en su uso [2].

Si bien es cierto que Bitcoin es la primera implementación extendida del dinero digital, el concepto del mismo, aparece por primera vez en una publicación del matemático norteamericano David Chaum[3]. En dicha publicación, Chaum establece que un sistema automático de pagos debe contar con las siguientes propiedades:

- Los terceros son incapaces de determinar el beneficiario, el tiempo o la cantidad de los pagos realizados por un individuo.

- Los individuos pueden demostrar los pagos mediante pruebas o determinar la identidad de los beneficiarios bajo circunstancias excepcionales.

- Se puede detener el uso de medios de pago que hayan sido reportados como robados.

[1] Al ser *peer-to-peer,* los costos de operación son bajos porque no se necesita de una infraestructura compleja para mantener la red.

[2] Los usuarios pueden transferirse dinero entre sí, sin la necesidad de que este pase por una entidad reguladora como son los bancos en el sistema económico tradicional.

Para lograr esto, idea un sistema criptográfico denominado *Blind Signatures*[3] y funda en 1989 la compañía DigiCash, con el objetivo de crear una divisa digital tan segura y privada como las divisas del mundo real. El problema era que los vendedores se rehusaban a aceptarlo, por lo tanto, no se podía conseguir consumidores para usarlo. Y como el comercio electrónico empezaba a florecer, resultó ser que las divisas preferidas fueron Visa y MasterCard, no el dinero digital en sí[5].

En 1998 DigiCash se declaró en bancarrota y no pasó mucho tiempo para que todas las demás compañías de criptomoneda tuvieran el mismo destino. Como otro ejemplo se puede nombrar a Flooz, criptomoneda creada por Robert Levitan en 1999, la cual era utilizada como certificados de regalos electrónicos, cesó sus actividades en 2001 al declararse en bancarrota luego de que la compañía reporte haber sido víctima de fraude[6].

En 1998, Wei Dai crea la propuesta de b-money[7], introduciendo así la idea de crear dinero a través de la resolución de problemas matemáticos y el consenso distribuido. Pero la propuesta escaseaba de detalles sobre cómo debería ser implementado el consenso.[8] En 2005, Hal Finney introdujo el concepto de pruebas de trabajo reutilizables, un sistema que utiliza las ideas de Wei Dai junto a los rompecabezas de Hashcash para crear un concepto de criptomoneda, pero falló al depender de la computación confiable[4] como *backend*.

En resumen, el dinero es una representación abstracta de un valor, respaldado por una autoridad y ampliamente admitido y aceptado para la realización de transacciones. Se pretende que el dinero electrónico sea lo mismo que el dinero tradicional, pero sustituyendo el soporte convencional (papel o metal) por el soporte electrónico (bits)[1]. De los emprendimientos fallidos presentados anteriormente, se puede deducir los factores clave que deben tener las criptomonedas para su adopción[10]:

a. **Aceptación universal:** un medio de pago requiere ser aceptado ampliamente para ser adoptado. Esta aceptación puede ser adquirida mediante la cooperación con bancos, gobiernos y compañías reconocidas.

b. **Confianza:** los medios de pago que sean confiables, reconocidos y fidedignos son más propensos a ser adoptados por vendedores y compradores.

c. **Seguridad:** refiriéndose a la capacidad de reducir al mínimo el fraude y proteger al consumidor del robo de sus fondos y de su información personal.

d. **Simplicidad:** la facilidad de uso es un factor determinante en la adopción de cualquier tecnología, la masificación requiere que los usuarios realicen el menor esfuerzo en entender sobre cómo utilizar el medio de pago.

---

[3] Es un protocolo entre dos partes: remitente A y firmante B. Básicamente, A envía una pieza de información a B, el cual la firma y luego retorna a A. A entonces puede computar la firma de B en un mensaje M que elija. Al finalizar el protocolo, B no conoce el mensaje M ni la firma asociada a A [4].

[4] En la computación confiable, la computadora se comportará de la manera esperada, y estos comportamientos serán impuestos por el hardware y el software[9].

Por último, la masificación de los dispositivos móviles hace que el acceso a tecnologías como las billeteras virtuales sea mucho más sencillo que en la década de los 90, esto significa que cada vez serán más los consumidores que opten por utilizar medios de pago alternativos[5] al dinero físico como son las criptomonedas, y a su vez, serán más los vendedores que las acepten como medios de pago válidos.

2- Comparativa Bitcoin y Ethereum

Tanto Bitcoin como Ethereum son criptomonedas peer-to-peer, esto significa que las transferencias de información se realizan directamente de un usuario a otro, sin la necesidad de una entidad central para generarla, controlarla y distribuirla (como ocurre con el sistema financiero tradicional), los encargados de llevar a cabo estas tareas son los nodos que conforman la red.

Los denominados nodos de la red, tanto la de Bitcoin como la de Ethereum, son los ordenadores de los usuarios, en dichos ordenadores se almacena el *blockchain* de las redes y se realiza el proceso de *mining*, del cual se ahondará más adelante en este trabajo. Los nodos pueden ser de dos tipos:

- *Full Nodes*: cuando los nodos almacenan localmente una copia completa del *blockchain*.
- *Lightweight*: cuando los nodos almacenan solamente una parte del *blockchain*.

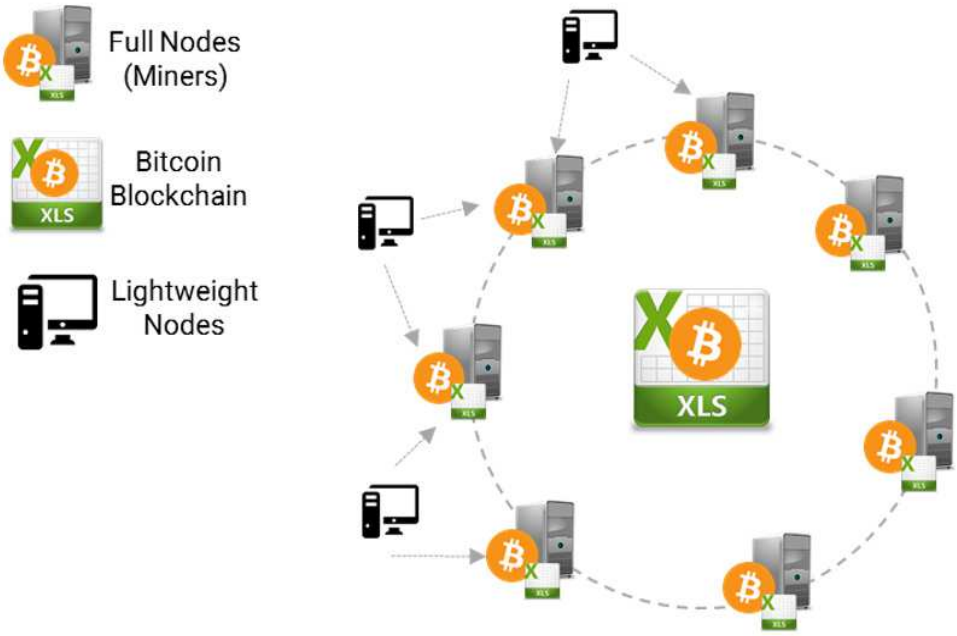

Ilustración 1: Red Bitcoin. Fuente: [11].

[5] La compañía Blockchain.info reporta que actualmente tiene más de 17 millones de usuarios de billeteras.

2.1- Blockchain

Antes de adentrarse en detalles acerca de cada arquitectura en particular, es necesario definir de forma genérica los conceptos de *Blockchain* y Prueba de Trabajo (PoW).

Se puede definir al *blockchain* como una base de datos distribuida compuesta por listas de registros, denominados bloques, los cuales están enlazados entre sí, ordenados cronológicamente, en constante crecimiento y asegurados mediante criptografía [12]. En la Ilustración 1 se puede observar el funcionamiento genérico de un *blockchain*.

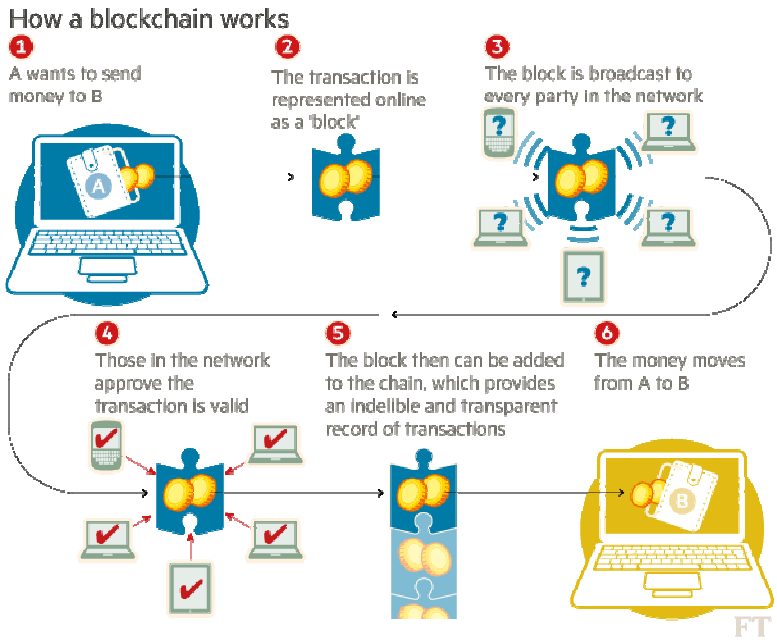

Ilustración 2: How Blockchain Works. Fuente: [13]

Las principales características del *blockchain* son las siguientes [14]:

- Base de datos distribuida. Cada nodo en una cadena de bloques tiene acceso a toda la base de datos y su historial completo. Además, ninguna de las partes controla los datos y puede verificar los registros de sus pares directamente, sin un intermediario.
- Transmisión punto a punto. La comunicación ocurre directamente entre pares sin pasar por un nodo central. Y cada nodo almacena y envía la información a todos los otros nodos.
- Transparencia con el seudo anonimato. Cada transacción y su valor asociado son visibles para cualquier persona con acceso al sistema. Cada nodo (en el caso de un *miner*[6]) o usuario en una cadena de bloques puede tener una o más direcciones alfanuméricas de 30 caracteres que la identifica. Los usuarios pueden elegir permanecer en el anonimato o proporcionar una prueba de su identidad a otros. Las transacciones ocurren entre las direcciones de cadenas de bloques.

[6] Nodo que dedica sus recursos computacionales al proceso de mining.

- Registros no modificables en el tiempo. Una vez que se ingresa una transacción en la base de datos y la cadena de bloques se actualiza, los registros no pueden ser alterados, ya que están vinculados a cada bloque hacia atrás. Se utilizan diversos algoritmos y enfoques computacionales para asegurar que el registro en la base de datos sea permanente, ordenada cronológicamente y disponible para todos los demás en la red.
- Lógica Computacional. La naturaleza digital del libro mayor significa que las transacciones de la cadena de bloques pueden estar ligadas a la lógica computacional y en esencia programadas. Así, los usuarios pueden configurar algoritmos y reglas que activen automáticamente las transacciones entre nodos usando las reglas de script.

En cuanto a la Prueba de Trabajo, es un reto a ejecutar por un nodo de la red para poder unir un bloque verificado a la cadena de bloques. Este reto debe ser computacionalmente difícil de resolver por fuerza bruta y, una vez resuelto, fácil de comprobar que es correcto. En la Ilustración 3, se observa el mecanismo genérico de una Prueba de Trabajo.

Cabe destacar que la Prueba de Trabajo no es el único método de consenso que existe, pero es el más extendido actualmente.

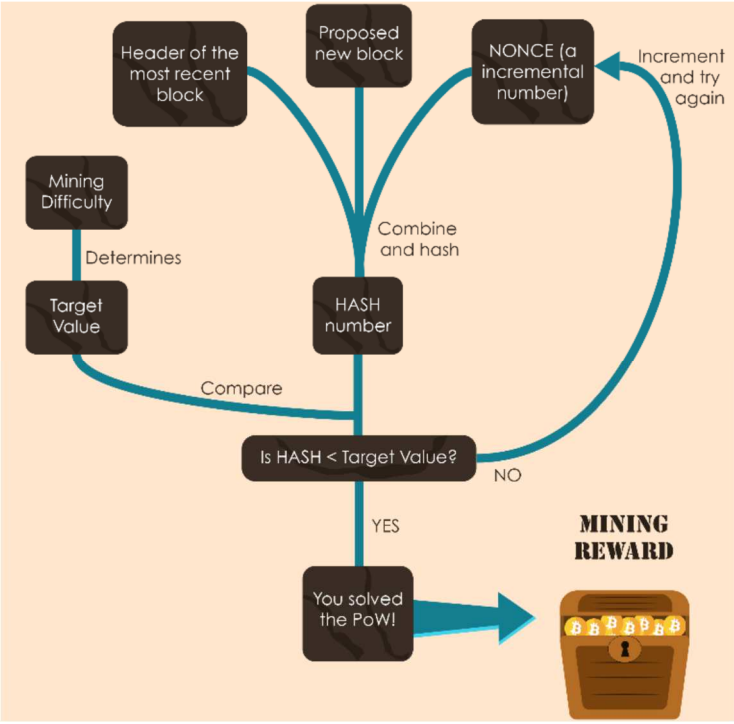

Ilustración 3: Prueba de Trabajo. Fuente: [15]

Si bien la arquitectura del *blockchain* de Ethereum está basada en la de Bitcoin, presenta ligeras diferencias, a continuación se detallará la arquitectura de ambos casos:

2.1.1- Blockchain de Bitcoin

Como se puede observar en la Ilustración 2, los bloques en la arquitectura Bitcoin tienen los siguientes componentes:

- Header:
  - Version: número de versión del bloque, basado en la correspondiente del software.
  - Previous Block Hash: es un hash SHA256^2 del bloque anterior.
  - Transactions Merkle Root: el hash raíz de una estructura de datos llamada árbol de Merkle[7]
  - Epoch Timestamp: es la marca de tiempo de la creación del bloque en segundos contados a partir del 01/01/1970.
  - Bits / Difficulty Target: nivel de dificultad del POW[8] definido para el bloque.
  - Nonce: Valor utilizado para generar el bloque. Permite variaciones del *header* y computar diferentes hashes para el POW.
- Body: contiene los datos sobre las transacciones.

[7] Un árbol de Merkle es un tipo de árbol binario, compuesto de un grupo de nodos con un gran número de nodos hoja en el fondo del árbol que contienen datos, un grupo de nodos intermedios donde cada nodo es el hash de sus dos hijos y finalmente un solo nodo raíz o cabecera, el cual es también el hash de sus dos hijos[16].

[8] *Proof Of Work* o prueba de trabajo, este tema será tratado más adelante en transacciones.

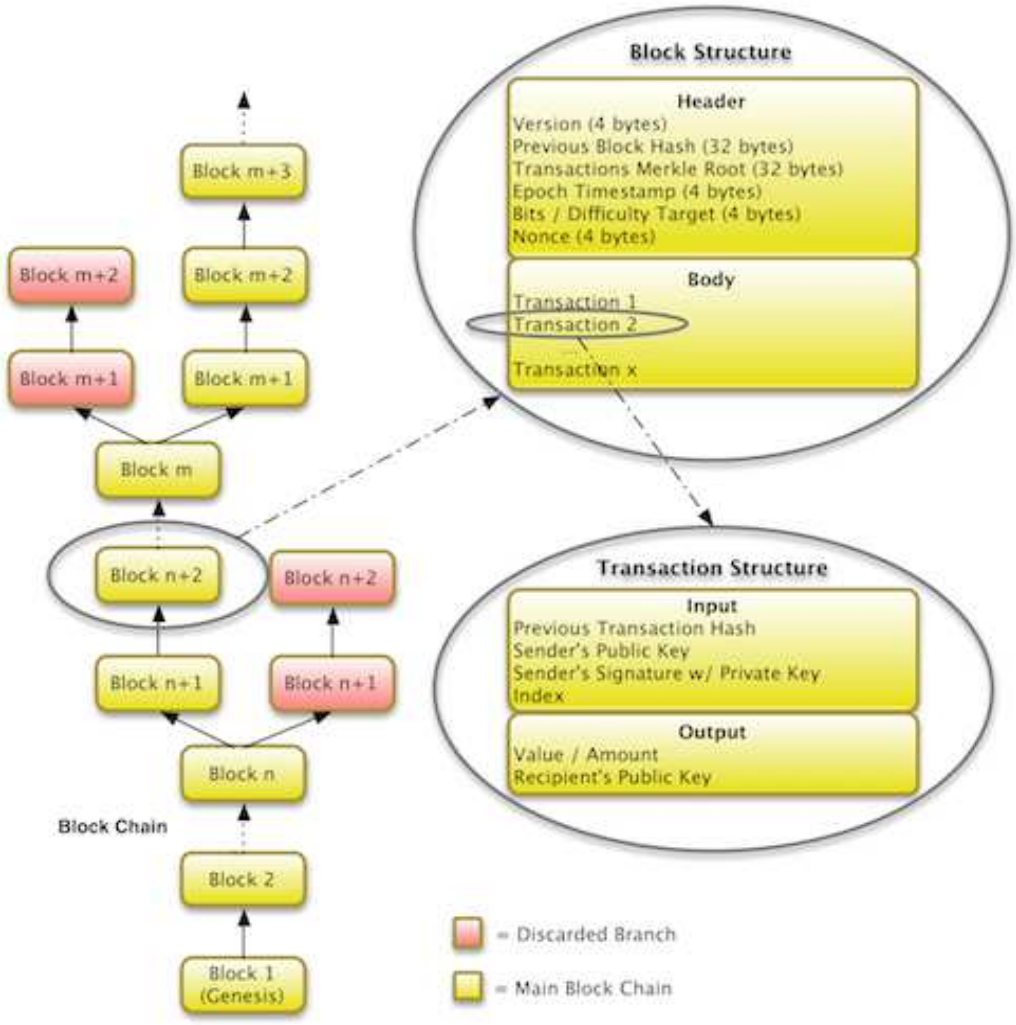

Ilustración 4: Bitcoin Blocks. Fuente: [17]

Cabe destacar en la figura anterior que se observan dos bifurcaciones de la cadena, estas son conocidas como *Forks* y suelen ocurrir cuando los nodos están desorientados debido al cambio de las reglas de consenso.

Las reglas de consenso son un conjunto específico de reglas, las cuales deben ser cumplidas infaliblemente por todos los *full nodes* de la red Bitcoin cuando se considera la validez de un bloque y sus transacciones. Por ejemplo, las reglas de consenso requieren que un bloque cree un determinado número de bitcoins, si un bloque crea más bitcoins de las permitidas, todos los *full nodes* rechazarán el bloque, inclusive si todos los demás nodos y *miner*s del mundo lo aceptan. Agregar nuevas reglas de consenso generalmente pueden ser realizadas como un *softfork* mientras que remover reglas requiere un *hardfork* (En la Ilustración 4 puede visualizarse un *softfork* y en la Ilustración 5 un hardfork). Las reglas acerca del comportamiento de la red no son consideradas reglas de consenso, inclusive si un cambio en el protocolo de la red rompe la compatibilidad para atrás. Las reglas de consenso solo se refieren a la validez de los bloques y transacciones.[18]

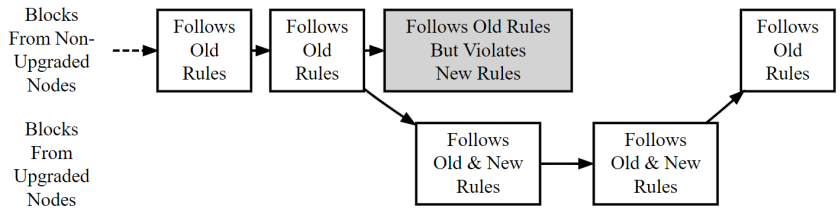

Ilustración 5: Softfork. Fuente: *[19]*

Un *softfork* es un cambio en el protocolo del software donde solamente los bloques/transacciones previamente válidos se convierten en inválidos. Debido a que los viejos nodos reconocerán los nuevos bloques como válidos, un *softfork* es compatible para atrás. Este tipo de *fork* requiere que la mayoría de los nodos se actualicen para obligar a cumplir las nuevas reglas.[20]

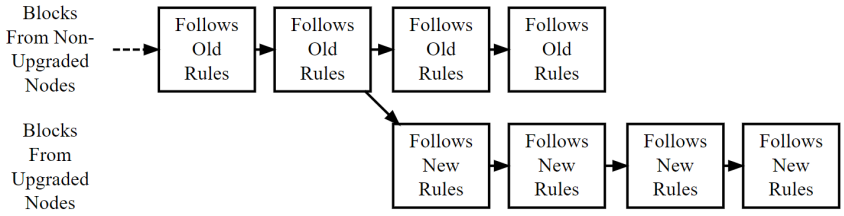

Ilustración 6: Hardfork. Fuente: [21]

Un *hardfork* es un cambio radical del protocolo que hace válidos bloques/transacciones previamente inválidos, y por lo tanto requiere que todos los nodos se actualicen a la nueva versión del protocolo del software. Es decir, un *hardfork* es una divergencia permanente de la versión previa del *blockchain* y los nodos que ejecutan la versión previa no podrán ser aceptados por la nueva versión.[22]

Los detalles sobre el POW se discutirán más adelante en este trabajo.

2.1.2- Blockchain de Ethereum

Por otra parte, se observa en la Ilustración 7 que los bloques que conforman el *blockchain* de Ethereum se diferencian de los bloques de bitcoin con respecto a la cantidad de campos que contienen y la información almacenada en los mismos.

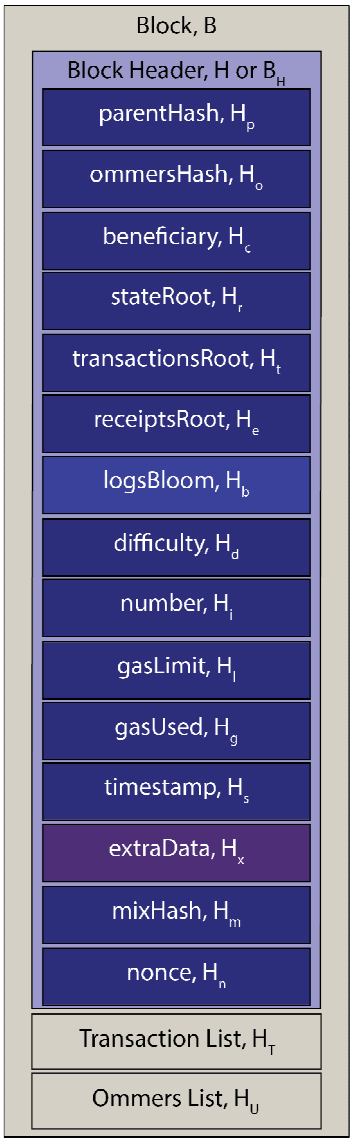

Ilustración 7: Ethereum Block. Fuente: [23]

Block, B: es la colección de piezas de información relevantes denominado *Block Header* (H o B_H), junto a información correspondiente a las transacciones comprimidas (T o B_T) y un conjunto de otros *Block Headers* (U o B_U) que tienen como bloque padre al abuelo del bloque actual (dichos bloques se conocen como ommers[9]). El bloque contiene varias piezas de información:

- parentHash: El hash[10] Keccak 256-bit del *header* del bloque padre, formalmente H_p.
- ommersHash: El hash Keccak 256-bit de la lista de *ommers* de este bloque; formalmente H_o.

[9] Ommers es la palabra en inglés que describe con género neutral al hermano/hermana de un padre, Gender_neutral_language#Family_Terms

[10] La función de hash KECCAK-256 de Ethereum es la versión creada por el equipo de Keccak, no aquella que fue modificada por el NIST luego de ser seleccionado el algoritmo ganador de la competencia SHA-3 [24]. De acuerdo a [25], el NIST modificó el padding original para el estándar SHA-3.

- beneficiary: La dirección de 160-bit a la cual se transfieren todos los honorarios por haber minado exitosamente un bloque; formalmente H_c. Los detalles sobre este campo se presentan en el capítulo X.
- stateRoot: El hash Keccak 256-bit del nodo raíz del *state trie*, luego de ejecutar las transacciones y aplicar las finalizaciones; formalmente H_r.
- transactionsRoot: El hash Keccak 256-bit del nodo raíz de la estructura trie poblada con cada transacción de la lista de transacciones del bloque, Formalmente H_t.
- receiptsRoot: El hash Keccak 256-bit del nodo raíz de la estructura *trie* poblada con los recibos de cada transacción de la lista de transacciones del bloque, formalmente H_e.
- logsBloom: El filtro Bloom[11] compuesto de información indizable (dirección registradora y tópicos referentes a los registros) contenidos en cada entrada de registro de los recibos de cada transacción en la lista de transacciones; formalmente H_b.
- difficulty: Un valor escalar correspondiente al nivel de dificultad del bloque. Este valor puede ser calculado a partir del nivel de dificultad del bloque previo y del *timestamp*; formalmente H_d.
- number: Un valor escalar igual al número de bloques ancestros. El bloque génesis tiene como valor cero en este campo; Formalmente H_i.
- gasLimit: Un valor escalar igual al límite actual de gasto de gas[12] por bloque; formalmente H_l.
- gasUsed: Un valor escalar igual al total de gas utilizado en las transacciones del bloque; formalmente H_g.
- timestamp: Un valor escalar razonable igual a la salida de la función time() de Unix al inicio de este bloque; formalmente H_s.
- extraData: Un array de bytes arbitrarios que contienen datos relevantes del bloque. Debe tener una longitud de 32 bytes o menos; formalmente H_x.
- mixHash: Un hash de 256-bit que, en conjunto al *nonce*, prueban que se llevó a cabo suficiente computación en el bloque, formalmente H_m.
- nonce: Un hash de 64-bit que, en conjunto con el mix-hash, prueban que se llevó a cabo suficiente computación en el bloque, formalmente H_n.

Los restantes dos componentes en el bloque son simplemente una lista de *headers* de los bloques *ommers* (en el mismo formato que arriba) y una serie de transacciones. Formalmente se puede referir a un bloque B:

$$B \equiv (B_H, B_T, B_U)$$

[11] Un filtro Bloom es una estructura de datos eficiente en cuanto a espacio, la cual es utilizada para probar si un elemento es miembro de un conjunto. Los falsos positivos son posibles pero los falsos negativos no, en otras palabras, un query puede retornar "posiblemente en el conjunto" o "definitivamente no en el conjunto"[26].

[12] Gas es el nombre de una unidad especial utilizada en Ethereum. Mide cuanto trabajo computacional requiere una acción o un conjunto de acciones para ejecutarse.

2.1.3- Transacciones en la red Bitcoin

En la Ilustración 8 se puede observar en resumen cómo funcionan las transacciones en la red Bitcoin. Posteriormente se describen los componentes, el proceso de *mining*, la prueba de trabajo y el enlazamiento de bloques.

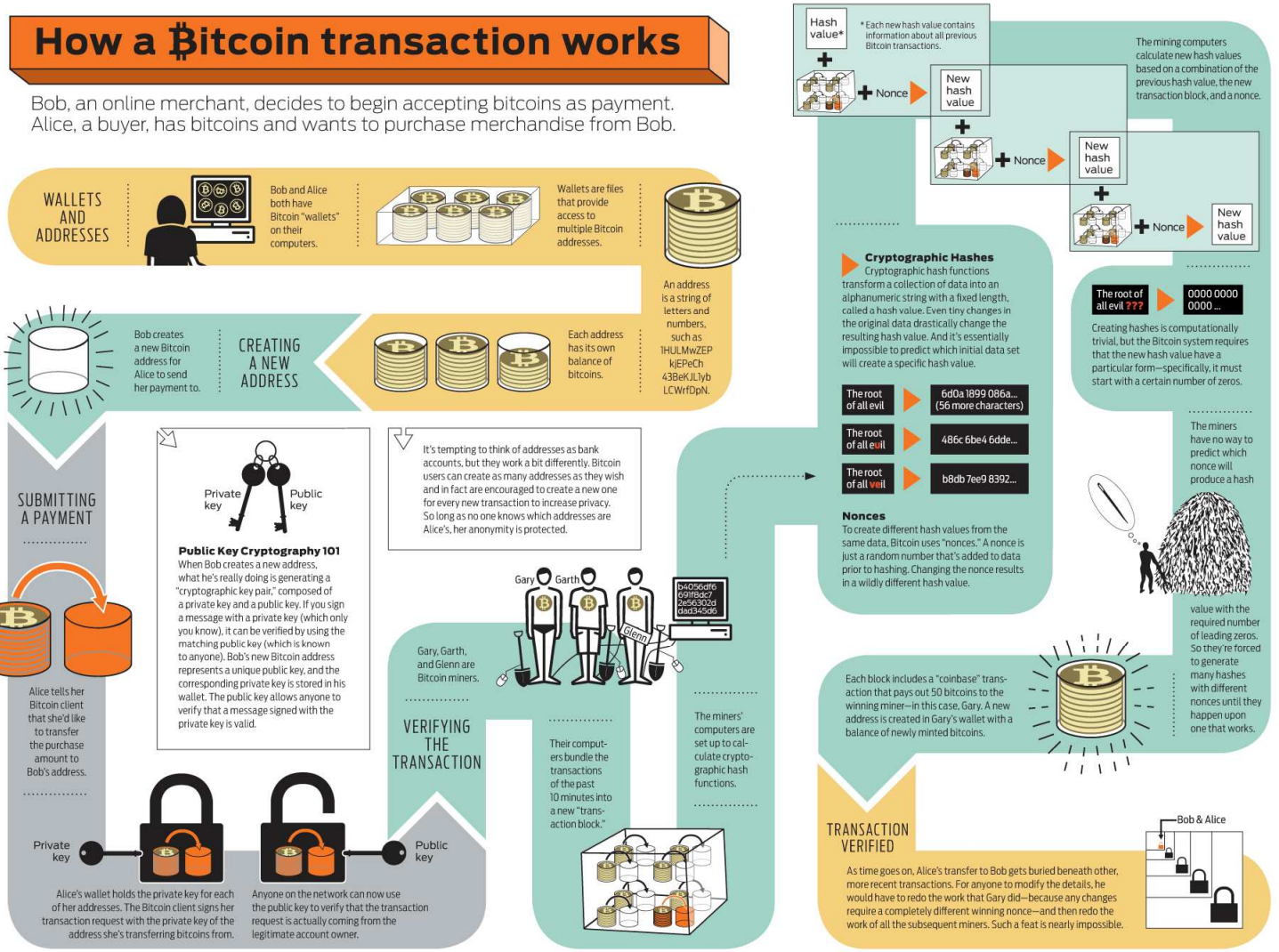

Ilustración 8: Bitcoin Transaction Workflow Fuente: [27]

2.1.3.1- Direcciones y Billeteras

Las direcciones son identificadores de entre 27 y 34 caracteres alfanuméricos, comenzando por el número 1 o el 3, que representan destinos de pagos en bitcoins. [28]

Un nuevo par de claves (pública y privada) es generado por cada dirección. Bitcoin permite la creación de tantas direcciones como desee el usuario, y se recomienda que se utilice una dirección nueva por cada transacción[29]. Más adelante se discutirá en detalle sobre la generación de direcciones a partir de una clave pública ECDSA[13].

Las billeteras son archivos que contienen las claves privadas asociadas a las direcciones pertenecientes a un usuario, existe una clave privada por cada dirección. Estas claves se utilizan para firmar las transacciones con el objetivo de garantizar el origen como su integridad. Asimismo las billeteras permiten mostrar la cantidad de bitcoins que contiene (balance de BTC), si bien los bitcoins no son almacenados en ningún lugar, la billetera provee la propiedad sobre el balance a un usuario, motivo por el cual suele estar cifrado mediante una contraseña maestra[30].

2.1.3.2- Transacciones

Una transacción es una transferencia de valor en Bitcoins que se transmite a la red y se recolecta en bloques (vistos anteriormente)[31]. Están compuestas por inputs y outputs para la combinación y separación de valores. Un input es una referencia a un output de una transacción previa mientras que un output contiene instrucciones para enviar Bitcoins. En la tabla 1 se muestra la estructura de las transacciones en formato raw.

Bytes	Nombre	Tipo de dato	Descripción
4	version	uint32_t	Número de versión de la transacción, actualmente versión 1. Los programas que crean transacciones usando nuevas reglas de consenso pueden usar números de versión mayores.
Varía	tx_in count	compactSize uint	Número de inputs en esta transacción.
Varía	tx_in	txIn	Los inputs de la transacción.
Varía	tx_out count	compactSize uint	Número de outputs de la transacción.
Varía	tx_out	txOut	Los outputs de la transacción.
4	lock_time[14]	uint32_t	Un tiempo (Unix epoch time) o un número de bloque.

Tabla 1 Estructura de una transacción en Bitcoin.

[13] Algoritmo de Firma Digital basado en Curvas Elípticas.

[14] También indica el primer momento en el que una transacción puede ser añadida al blockchain (el valor 0 indica que debe ser ejecutada inmediatamente) y permite crear transacciones bloqueadas que serán válidas en el futuro, dando así a los firmantes la posibilidad de invalidar la transacción (se invalida dicha transacción si se crea una nueva transacción no bloqueada)[32].

En la tabla 2 se observan los detalles de la estructura TxIn.

Bytes	Name	Data Type	Description
36	previous_output	outpoint	TxID de la transacción previa.
4	index	uint32_t	El índice de un output específico del cual se debitará en la transacción. El primer output es 0x00000000.
Varía	script bytes	compactSize uint	El número de bytes del *signature script*. El máximo es 10,000 bytes.
Varía	signature script	char[]	Un lenguaje de script que satisface las condiciones del Pubkey Script.
4	sequence[15]	uint32_t	Número de secuencia. Versión de la transacción como la define el emisor.

Tabla 2 Estructura de un TxIn

En la tabla 3 se observan los detalles de la estructura TxOut.

Bytes	Name	Data Type	Description
8	value	int64_t	Número de satoshis[16] a transferir.
1+	pk_script bytes	compactSize uint	Número de bytes del pubkey script. Maximo 10,000 bytes.
Varía	pk_script	char[]	Define las condiciones que deben ser satisfechas para gastar los outputs.

Tabla 3 Estructura de un TxOut

El *Pubkey Script* es el campo donde se definen las condiciones bajo las cuales los satoshis son gastados. Existen tres tipos de scripts actualmente:

- *Pay to Public Key Hash* (P2PKH) o Pago a hash de clave pública: es un script simple que indica que los valores son bloqueados para un hash de una dirección pública de Bitcoin,
- *Pay to Public Key* (P2PK) o Pago a clave pública: similar al anterior con la diferencia de que se utiliza la clave pública en lugar del hash de la misma,
- MULTISIG o Multifirma: similar al primero pero con la opción de incluir hasta 15 firmas en un script de bloqueo hasta la versión actual de Bitcoin. Una desventaja de este tipo de scripts es su longitud y por lo tanto difíciles de transmitir a la red. Y por último,
- *Pay to Script Hash* (P2SH) o Pago a hash de script: combina la simplicidad de P2PKH y la complejidad de MULTISIG, permite bloquear los valores con varias firmas y desbloquearlos con menos del total.[14]

En cuanto a las transacciones inválidas, se puede decir que se consideraran inválidas y por ende rechazadas, aquellas transacciones en las que[2], [32], [34]:

- Algún dato dentro de los campos sean incorrectos o inválidos.

[15] El campo lock_time junto a sequence demuestran si una transacción está finalizada.
[16] Satoshi es actualmente la unidad mínima de bitcoins: 1 Bitcoin equivale a 100.000.000 satoshis.[33]

- El BTC del output está categorizado como gastado o el input ya fue utilizado.
- La transacción tiene el mismo número de identificación que otra.
- La firma digital es inválida.
- Formato de la transacción inválido.

2.1.3.3- Prueba de trabajo (POW)

Bitcoin utiliza como prueba de trabajo una metodología del tipo desafío-respuesta: el desafío consiste en generar un hash con una cantidad de ceros al inicio. La cantidad de ceros es fijada por la red, aumenta cuando los bloques están siendo generados muy rápido y disminuye en caso contrario. Para crear diferentes hashes de la misma cadena de caracteres, los *miner*s calculan los hashes basados en combinaciones de: el hash de las transacciones anteriores, el nuevo bloque de transacciones y un *nonce*.

El *nonce* es un valor que varía en el tiempo y que tiene a lo sumo una posibilidad insignificante de ser repetido. [35]

Cuando un *miner* consigue un hash que cumpla con los requerimientos del desafío, lo transmite a la red y los demás nodos se encargan de verificar si se cumple el desafío, en caso positivo, se lo recompensa con una cantidad de BTC y en caso negativo el bloque es rechazado, siguiendo así la competencia por encontrar un bloque válido. Únicamente el primer hash en ser encontrado es válido (pueden existir varias respuestas), el resto es descartado.

En la Ilustración 9 se observa un ejemplo del funcionamiento de la prueba de trabajo.

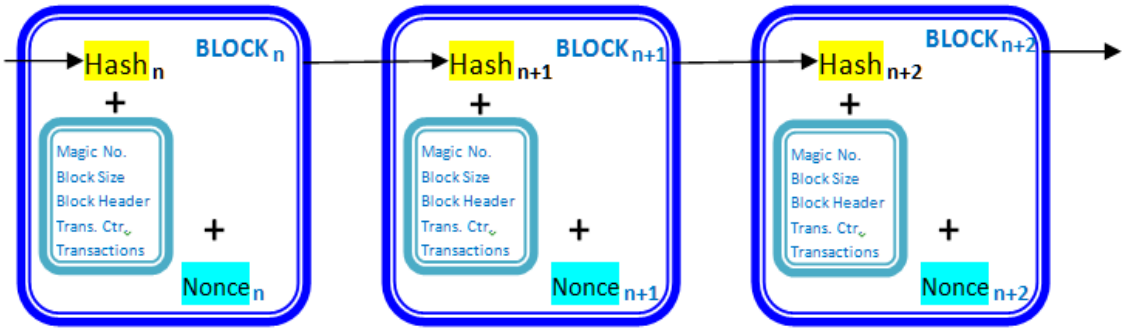

Ilustración 9: Fuente: [36]

2.1.4- Transacciones en la red de Ethereum

En la Ilustración 10 se puede observar el mecanismo de transacciones en la red de Ethereum:

El proceso se inicia cuando se detecta que una o varias transacciones nuevas han sido transmitidas a la red, en ese momento los *miner*s comienzan la competencia de ser el primero en hallar un hash que satisfaga el *challenge* del POW, cuando se encuentra dicho hash, el ordenador del *miner* compila las transacciones a ser incluidas en el bloque.

Las transacciones son compiladas en una lista pero a su vez son utilizadas para derivar el hash raíz del trie de ethereum. La estructura conocida como *trie* será estudiada en el capítulo 3.

Posteriormente determina los *ommers* del bloque y se incluye los *headers* de éstos en el campo *Ommers List*. Se aplican las recompensas y finalmente se computa un estado válido. Se concluye que un bloque tiene validez holística si y sólo si satisface las siguientes condiciones: debe tener consistencia interna con los hashes de los *ommers* y del bloque de transacciones, cuando se ejecuta en orden sobre el estado base σ (el cual se derivó del estado final del bloque padre) resulta en un nuevo estado de la identidad H_r.

Finalmente se comprueba que el hash generado satisfaga el *challenge* del POW y se transmite a la red.

Posteriormente se describen los detalles de los componentes, al igual que en la sección anterior.

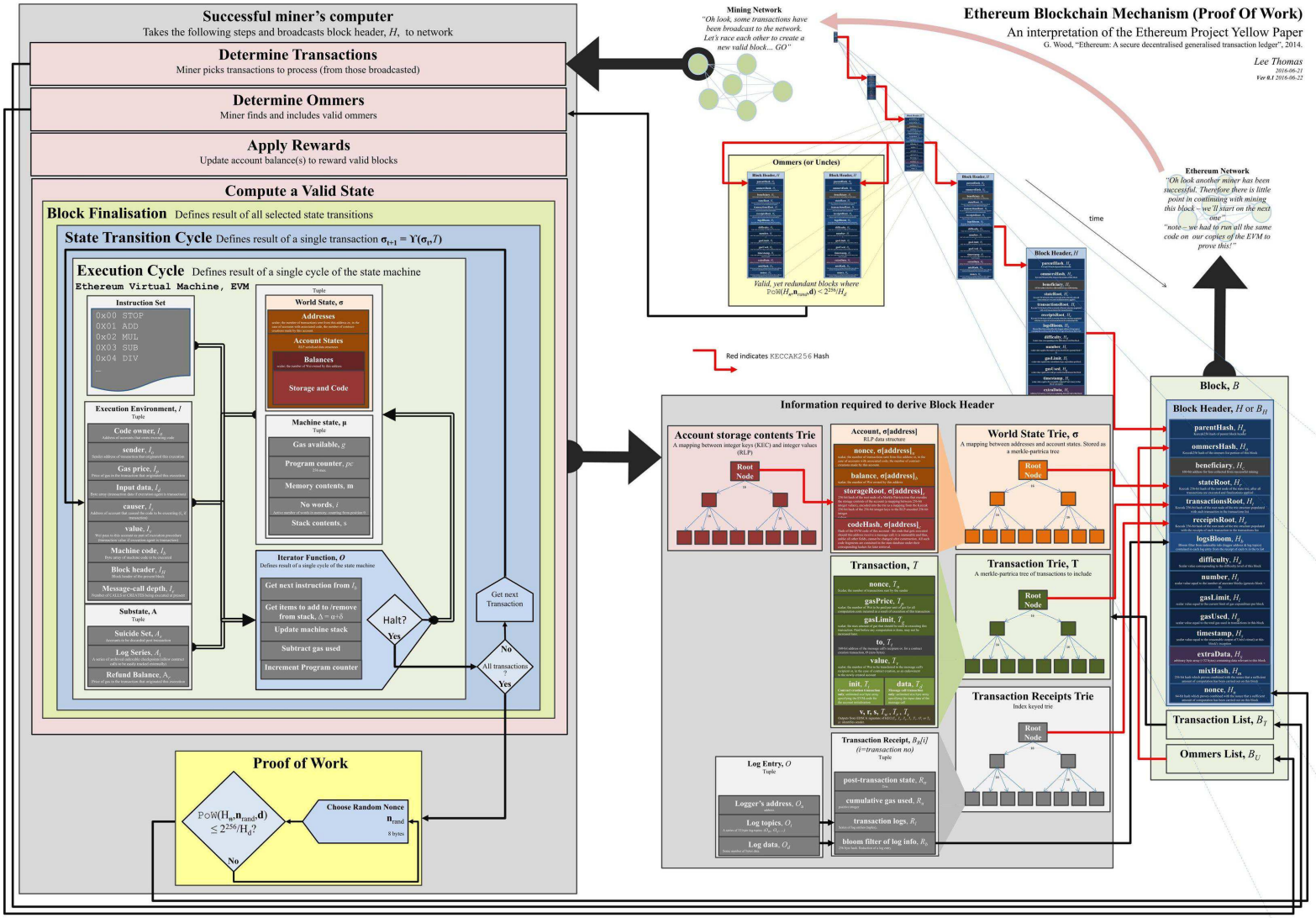

Ilustración 10: Ethereum blockchain mechanism. Fuente:[23]

2.1.4.1- Direcciones y Cuentas

Existen dos tipos de cuentas[17] en la red de Ethereum: cuentas de propiedad externa (EOAs) y cuentas de contratos. En el presente trabajo se referirá a las primeras como cuentas y a las segundas como contratos. El término genérico de cuenta que incluye tanto las cuentas de propiedad externa como los contratos se justifica porque estas entidades se denominan objetos de estado. Estas entidades tienen un estado: las cuentas tienen balance y los contratos tienen balance y almacenamiento por contrato. El estado de todas las cuentas es el estado de la red Ethereum, que se actualiza con cada bloque y, sobre el cual, la red realmente necesita llegar a un consenso. Las cuentas son esenciales porque los usuarios las utilizan para interactuar con el *blockchain* a través de las transacciones.

Más adelante se discutirá en detalle el proceso de generación de direcciones y claves para el *Keyfile*.

Al igual que en Bitcoin, todos los pares de claves/direcciones son almacenados en un archivo llamado *keyfile*. El keyfile es un archivo con formato JSON[18] que puede ser abierto con cualquier software de edición de textos. Los componentes críticos del keyfile, las claves privadas del usuario, están siempre encriptadas con la contraseña maestra ingresada al momento de crear la cuenta.[37] [39]

2.1.4.2- Prueba de trabajo (POW)

Ethereum basa su prueba de trabajo en un algoritmo propio de la red llamado Ethash, la particularidad del algoritmo radica en que depende de un *dataset pseudorandom*, inicializado por la longitud actual del *blockchain*. Dicho *dataset* se denomina DAG[19]. El flujo del algoritmo se puede resumir en los siguientes pasos:

1- El *Header* pre-procesado (derivado del bloque anterior) y el *nonce* actual (el intento actual), se combinan utilizando un algoritmo parecido al SHA3 para crear el Mix inicial, el Mix 0.
2- El Mix es utilizado para computar cuál de las páginas de 128 bytes del DAG se debe recuperar, representado por el bloque *Get* DAG *Page*.
3- El Mix se combina con la página del DAG recuperada para generar el siguiente Mix, Mix 1.
4- Los pasos 2 y 3 se repiten 64 veces, dando como resultado el Mix 64.
5- El Mix 64 se vuelve a procesar, dejando como resultado un *digest* de 32 bytes.
6- El *digest* se compara contra el Target *Threshold* de 32 bytes predefinido. Si el *digest* es menor o igual al Target *Threshold*, entonces el *nonce* actual se considera válido y será transmitido a la red de Ethereum. De otro modo, el *nonce* actual se considera inválido, y el algoritmo se vuelve a ejecutar

[17] Las cuentas representan identidades de agentes externos (por ejemplo: personas, nodos *miner*s o agentes automáticos). [37]
[18] Acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. [38]
[19] Grafo dirigido acíclico.

con un nonce diferente (ya sea incrementando el nonce actual, o eligiendo uno nuevo aleatoriamente).

En la Ilustración 10, se puede observar el mecanismo completo del algoritmo.

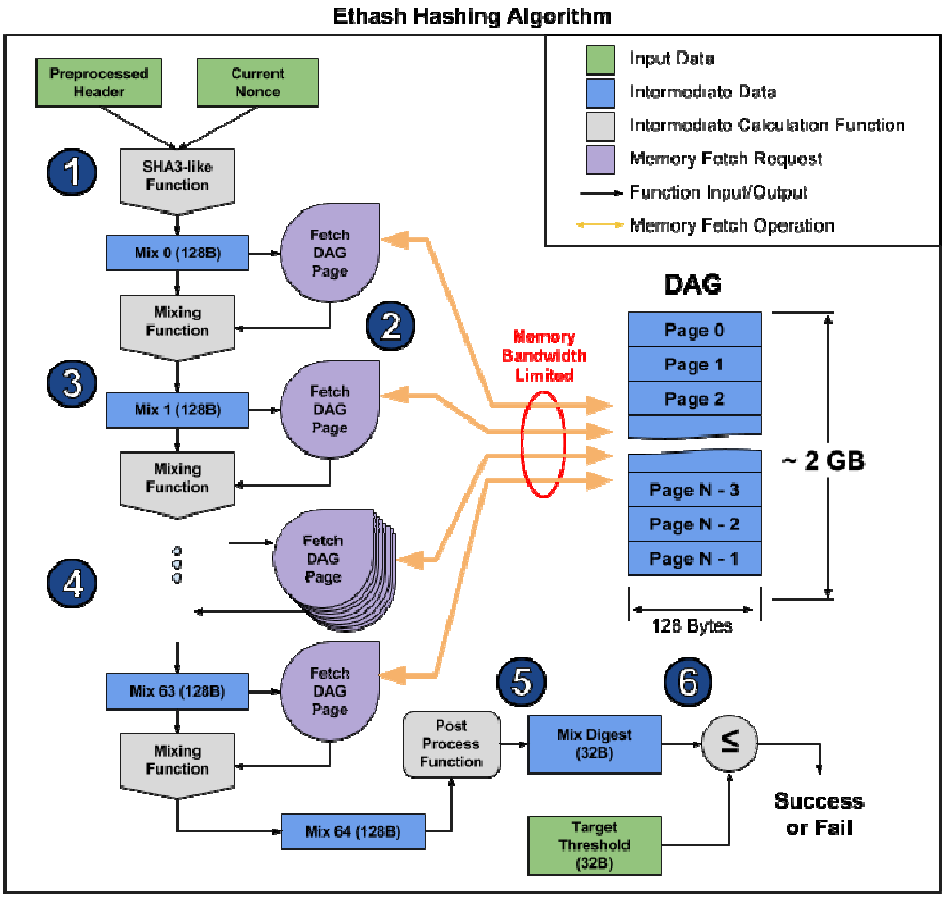

Ilustración 11: Ethash hashing Algorithm. Fuente: [40]

Cabe destacar que Ethereum prevé reemplazar la prueba de trabajo por la prueba de participación (*proof of stakes*), un tipo de algoritmo de consenso para *blockchains* públicas en el cual se selecciona al azar a un grupo de validadores, los cuales votan sobre el siguiente bloque, y el peso de cada voto depende del tamaño del balance del validador, es decir, aquel votante que posea más Ether, tendrá mayor poder de decisión. [41], [42]

3- Fundamentos Criptográficos

Firmas digitales

Las firmas digitales permiten la verificación del origen sin exponer la identidad del emisor o el receptor de las transacciones. Como se vio anteriormente, las transacciones son firmadas con la clave privada asociada a la dirección del emisor. Posteriormente, cuando se requiera verificar que una transacción efectivamente proviene del dueño de una dirección, cualquier nodo de la red puede verificar la identidad del emisor mediante su clave pública.

Como se mencionó con anterioridad, tanto en Bitcoin como en Ethereum, las claves y direcciones se generan mediante la criptografía de curvas elípticas. En específico se utiliza ECDSA, el cual es un análogo del algoritmo de firma digital (DSA) utilizando curvas elípticas. La seguridad de los criptosistemas de curvas elípticas se basa en la dificultad computacional de resolver problemas de logaritmos discretos de curvas elípticas (ECDLP). Este sistema provee una seguridad por bit de la clave muy superior en relación a los sistemas de logaritmo discreto[20] convencionales[21], esto se traduce en la posibilidad de utilizar parámetros de menor tamaño obteniendo niveles de seguridad equivalentes. Las ventajas de utilizar parámetros más cortos incluyen velocidad y claves y certificados más pequeños, esto es ideal para ambientes con escasos recursos computacionales.

En la tabla X se puede observar una comparación entre el tamaño de claves para ECDSA y para el módulo de la clave para RSA/DSA para obtener niveles de seguridad equivalentes. Es notable la diferencia de tamaño de claves entre ambos sistemas.

Parámetros[22]	Fortaleza (bits)	Tamaño ECDSA (bits)	Tamaño RSA/DSA (bits)
secp128r1	64	128	704
secp160k1	80	160	1024
secp192k1	96	192	1536
secp256k1	128	256	3072

Tabla 4 Fuente: Elaboración propia en base a información de [43]

Tanto Bitcoin como Ethereum implementan la curva secp256k1. Esta curva es la recomendada por el Grupo de Estándares para la Criptografía Eficiente, es una curva del tipo Koblitz[23] y se define por los siguientes parámetros: la ecuación es $y^2 = x^3 + 7$ (a=0, b=7), definida sobre el campo finito F_p donde $p= 2^{256} – 2^{32} – 2^9 – 2^8 – 2^7 – 2^6 – 2^4 – 1$, donde con el punto generador P de orden n es posible derivar la clave privada, siendo ésta un número aleatorio s<n y la clave pública es un punto en la curva tal que Q = s•P.

La implementación de curvas Koblitz en ECDSA presenta los siguientes beneficios cuando se realizan multiplicaciones complejas:

1) Son no supersingulares, esto previene el uso de la reducción Menezes-Okamoto-Vanstone de logaritmos discretos de curvas elípticas a campos finitos como un ataque a los criptosistemas de curvas elípticas.
2) El orden del grupo tiene un factor primo grande, esto previene la computación de logaritmos discretos por el algoritmo baby-step/giant-step, así como el algoritmo Pollard-Rho. Típicamente la curva definida sobre GF(2^k) tiene K>=160 y primo.

[20] El problema del logaritmo discreto: Dado $g, h \in E(\mathbb{F}_p^x, \cdot)$ encontrar k tal que $g^k = h \, mod \, (p)$

[21] Un ejemplo de esos tipos de criptosistemas es ElGamal.

[22] Parámetros recomendados por el Stardards for Efficient Cryptography Group.

[23] Las curvas Koblitz describen curvas binarias anómalas sobre GF(2^k) y definidas sobre el campo finito F_p son de la forma $y^2=x^3+ax+b$

3) Doblar los puntos en la curva es muy eficiente, puede llevarse a cabo de forma casi tan efectiva como lo hace Vanstone con las curvas supersingulares. Uno previene el problema de que el problema del logaritmo discreto puede ser resuelto utilizando algoritmos de cálculo de índice sobre curvas supersingulares desde K<=6.
4) Las curvas son fáciles de encontrar, esto facilita la implementación. El SECG tiene disponibles en línea varias curvas Koblitz ya definidas[44].

Adicionalmente, en cuanto a la curva secp256k1, al haber sido elegidos los parámetros de forma predecible por el SECG, se reduce la posibilidad de inserción de puertas traseras o *backdoors* por parte del creador de la curva.

El algoritmo ECDSA para la generación de claves públicas y privadas tanto en Ethereum como en Bitcoin funcionan de la siguiente manera: un usuario U quiere firmar un mensaje m para autorizar el pago a un vendedor M [45].

Para la firma:

(a) Un usuario U elige un número aleatorio (diferente por cada vez que se necesite firmar) tal que 1 <= k < r y computa $R = (x_R, y_R) = \mathrm{kP} \in \mathrm{E}(\mathbb{F}\mathrm{q})$ R=(xR,yR)=.
(b) Luego U computa $s = k^{-1}(H(m) + ax_R)$ en $\mathbb{Z}/r\mathbb{Z}$. La firma de U es entonces (x_R, s).

Para la verificación:

(a) M computa $u_1 = s^{-1}H(m)$ y $u_2 = s^{-1}x_R$ en $\mathbb{Z}/r\mathbb{Z}$, y luego $V = (x_V, y_V) = u_1P + u_2Q$ en E.
(b) M verifica la firma del usuario U al cumplirse que $x_R = x_V$ en $\mathbb{Z}/r\mathbb{Z}$.

Donde:

E es una curva elíptica $\mathrm{E}/\mathbb{F}\mathrm{q}$

$\mathbb{F}\mathrm{q}$ es un campo finito

$\mathrm{P} \in \mathrm{E}(\mathbb{F}_q)$ es un punto base (la base del algoritmo)

r es el orden

H es una función de hash (sus valores son considerados modulo r)

K es un entero

P, Q son puntos de la curva elíptica E.

a es un entero secreto $a \bmod r, a \in \mathbb{Z}/r\mathbb{Z}$.

Q es un punto que cumple $Q = aP$.

Generación de direcciones

En cuanto a la generación de direcciones, se presentan diferencias en los esquemas de ambas redes. En el caso de Bitcoin, el mecanismo es el siguiente:

1- Se genera la clave privada mediante el algoritmo ECDSA visto anteriormente
2- Se toma la clave pública correspondiente, se realiza el hash sobre la clave pública (la función de hash es SHA-256)
3- Al resultado se le aplica la función de hash RIPEMD-160 (se describirá esta función en la siguiente sección)
4- Se agrega el byte de versión a la izquierda del resultado anterior[24].
5- Se vuelve a aplicar la función SHA-256 sobre la cadena obtenida en el punto 3.
6- Se aplica una vez más la función SHA-256 sobre el resultado del punto 4.
7- Se toman los primeros 4 bytes de la cadena del punto 6 para utilizarlas como *checksum* de la dirección.
8- Se agregan los 4 bytes del punto 6 a la derecha del resultado del punto 3. Esto da como resultado una dirección binaria de 25 bytes.
9- Finalmente se convierte la dirección binaria a una cadena utilizando la codificación base58[25] modificada, denominada Base58Check.

Según Satoshi Nakamoto, la justificación de la utilización de base-58 en lugar del estándar base-64 es que no se desea que ciertos caracteres luzcan iguales en algunas fuentes para evitar la creación de direcciones que sean idénticas visualmente, las cadenas con caracteres no alfanuméricos no son aceptadas fácilmente como números de cuenta, los e-mails usualmente no romperán la línea si no existen signos de puntuación donde romperlas y al hacer doble click sobre una cadena se selecciona en su totalidad si es completamente alfanumérica [46]. En la Ilustración 12 se puede observar el mecanismo de generación de direcciones en Bitcoin.

[24] Para la red principal se utiliza 0x00.
[25] Existen otras formas de generar direcciones pero esta es la más común.

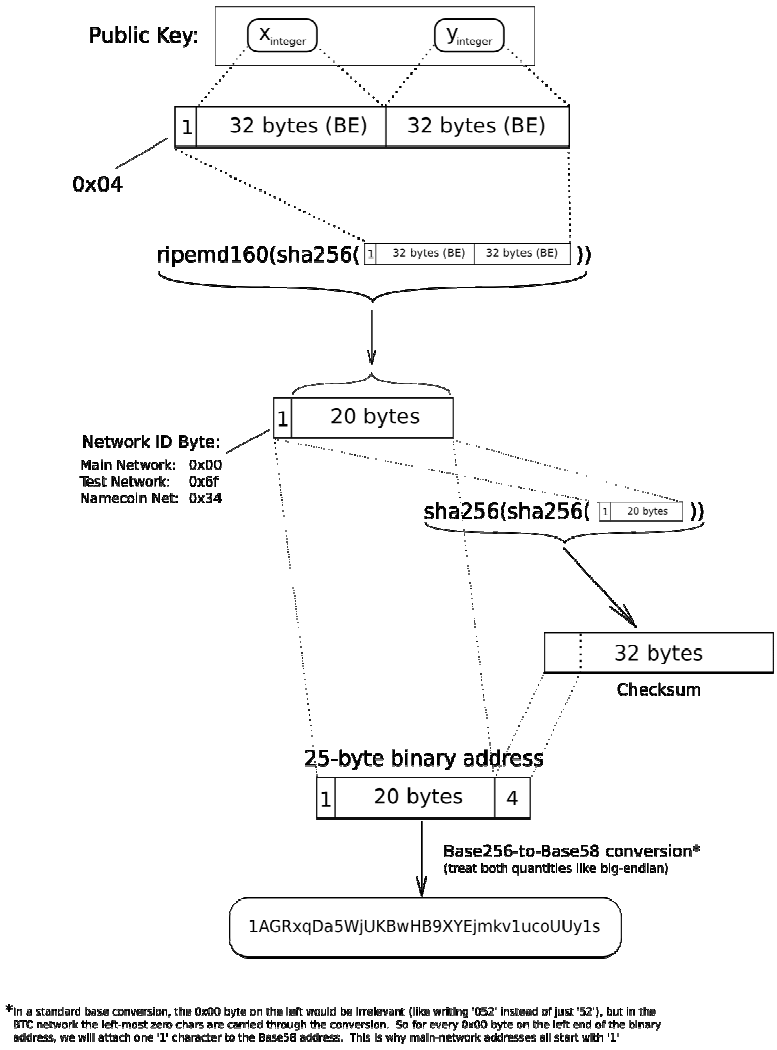

Ilustración 12: Generación de direcciones en Bitcoin. Fuente: [47]

En cuanto a Ethereum, la generación de direcciones sigue una metodología diferente:

1- Se generan la clave privada y pública mediante el algoritmo ECDSA visto anteriormente.
2- Se toma la clave pública y se le aplica la función de hash KECCAK-256
3- Al resultado del punto 2 se le eliminan los primeros 24 caracteres (12 bytes), en otras palabras, se toman los últimos 40 caracteres y se le agrega el prefijo 0x quedando así generada una dirección de 42 caracteres.

En la Ilustración 13 se puede observar el mecanismo de generación de direcciones en Ethereum.

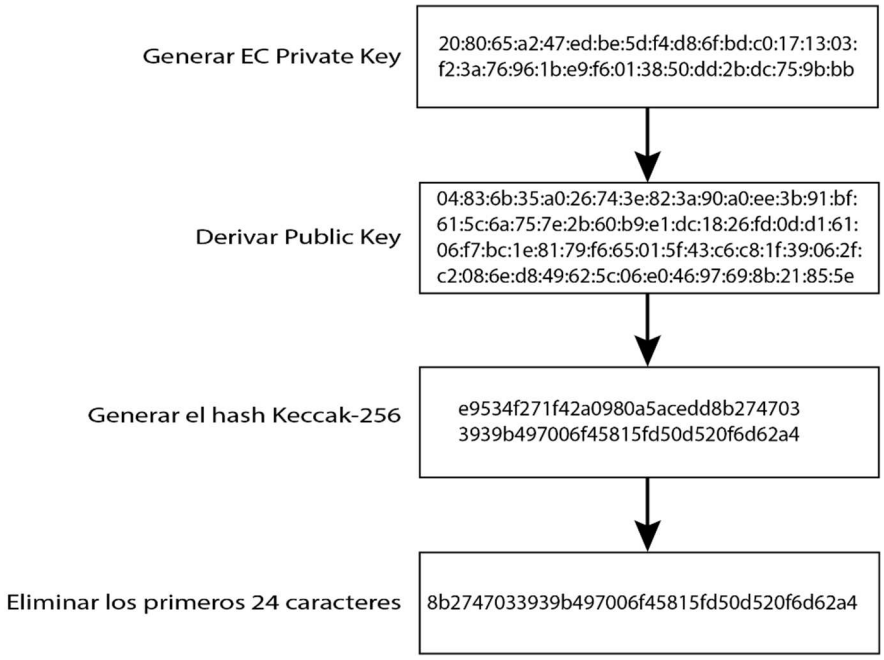

Ilustración 13: Generación de direcciones Ethereum. Elaboración propia en base a [48]

Hash y Árboles

Una función de hash es una función computacionalmente eficiente que mapea cadenas binarias de longitud arbitraria a cadenas binarias de una longitud determinada, denominada digesto.

Para que una función de hash h sea de uso criptográfico, típicamente cumple que: no es factible computacionalmente encontrar dos entradas distintas que den como resultado un valor común (por ejemplo, dos entradas x e y que colisionen de tal forma que $h(x) = h(y)$), y que dado un hash específico y, no es factible computacionalmente encontrar una entrada x tal que $h(x) = y$.[4]

SHA-256

El proceso de generar un hash con el algoritmo SHA-256 es el siguiente:

1- Al mensaje a ser convertido en hash se le agrega un *padding* de modo que el resultado es un múltiplo de 512 bits de longitud.
2- Se divide el mensaje en bloques de 512 bits $M^{(1)}, M^{(2)}, \dots, M^{(N)}$.
3- Los bloques son procesados de a uno por vez comenzando con un valor inicial pre establecido $M^{(0)}$, y se computa secuencialmente

$$H^{(i)} = H^{(i-1)} + C_{M^{(i)}}(H^{(i-1)})$$

4- Donde C es la función de compresión SHA-256 y $+$ representa a la adición en $mod\ 2^{32}$. $H^{(N)}$ es el hash de M.

El algoritmo da como salida una cadena de 256 bits de longitud.

RIPEMD-160

Es una variante del algoritmo MD4, por lo cual opera con palabras de 32 bits de longitud. El algoritmo divide una cadena de longitud arbitraria en bloques de 512 bits, para garantizar que la longitud de la entrada es un múltiplo de 512 se utiliza un *padding*[26]. Cada bloque se divide nuevamente en 16 cadenas de 4 bytes, y cada cadena de 4 bytes se convierte en una palabra de 32 bits utilizando la convención *little-endian*[27].

El resultado son cinco palabras de 32 bits de longitud, las cuales forman el estado interno del algoritmo. Finalmente el contenido de las cinco palabras se convierte en una cadena de 160 bits utilizando nuevamente la convención *little-endian*.

El estado interno del algoritmo se inicializa con un grupo determinado de cinco palabras de 32 bits, ese grupo de palabras se llama valor inicial. La parte principal del algoritmo se conoce como la función de compresión: esta computa el nuevo estado a partir del antiguo estado y el siguiente bloque de 16 palabras. Consiste en 5 rondas paralelas, las cuales contienen 16 pasos cada una. El número total de pasos es entonces $5 \times 16 \times 2 = 160$. Primeramente, se hacen dos copias del antiguo estado (cinco registros de 32 bits a la derecha y cinco a la izquierda). Ambas son procesadas de forma independiente. Cada paso computa un nuevo valor para uno de los registros basado en los otros cuatro registros y una palabra del mensaje. Al final de la función de compresión, se computa un nuevo estado al agregar a cada palabra del antiguo estado un registro de la mitad derecha y uno de la mitad izquierda[49].

En la Ilustración 14, puede observarse la configuración para el algoritmo RIPEMD-160.

[26] En los miembros de la familia de algoritmos MD4, el padding consiste en agregar un 1 seguido de tantos 0s como bits falten para completar los 512 bits.

[27] El término inglés *endianness* designa el formato en el que se almacenan los datos de más de un byte en un ordenador. El sistema *big-endian* consiste en representar los bytes de derecha a izquierda, en cambio, con *little-endian* se representan a la inversa.

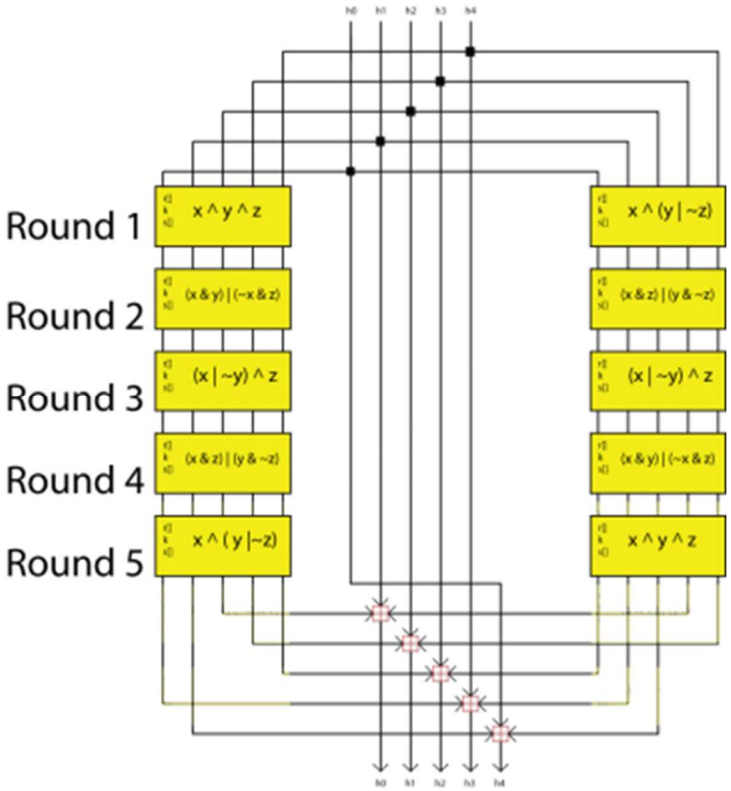

Ilustración 14: RIPEMD-160 Algorithm. Fuente: [49]

KECCAK-256

Es una variación de 256 bits del algoritmo KECCAK, el cual fue seleccionado por el NIST[28] para el estándar SHA-3.

Teniendo como entrada una cadena N, una función de padding pad, una función de permutación f que opera en bloques de bits de longitud b, un ratio r y una salida de longitud d, se tiene una capacidad $c = b - r$ y la función de esponja[29] $Z = sponge[f, pad, r](N, d)$, produciendo una cadena de bis Z de longitud d, el algoritmo funciona de la siguiente manera:

1- Se rellena el input N utilizando la función de padding, produciendo una cadena P con una longitud divisible entre r (tal que $n = len(P)/r$ es entero).
2- Se divide P en n bloques consecutivos de r bits $P_0, \dots, P_{n-1}$.
3- Se inicializa el estado S a una cadena de b 0 bits.
4- Se absorbe la entrada en el estado: Por cada bloque P_i,
    a. Se extiende P_i al final por una cadena de c 0 bits, produciendo una de longitud b,
    b. Se realiza un XOR del resultado del punto anterior con S

[28] National Institute of Standards and Technology
[29] En el contexto de la criptografía, una función de esponja es un tipo de algoritmo que toma una entrada de cualquier cantidad de bits y produce una salida de cualquier cantidad de bits [50].

c. Y se aplica la función de permutación f al resultado, produciendo un nuevo estado S.

5- Se inicializa Z como una cadena vacía.
6- Mientras que la longitud de Z sea menor a d:
   a. Se adjuntan los primeros r bits de S a Z
   b. Si Z aun es de longitud menor a d, se aplica f a S, produciendo un nuevo estado S.
7- Se trunca Z en d bits.

Cabe destacar que Ethereum utiliza la versión original del algoritmo KECCAK, propuesto por el equipo del mismo nombre, no la versión estandarizada del NIST. Esto se debe a que Ethereum se puso en producción antes de que el NIST declare ganador del concurso SHA-3 al KECCAK y por lo tanto antes de que el NIST lo modifique como parte del proceso de revisión. Es por esta razón que las librerías de generación de hash de Ethereum producen resultados diferentes a las librerías estandarizadas del NIST.[51], [52], [53],[54],[55].

Ethash

Es el algoritmo utilizado para la POW en Ethereum. Al inicio fue una implementación de la última versión del algoritmo Dagger-Hashimoto, luego pasó a llamarse Ethash debido a los drásticos cambios en ambos algoritmos durante la etapa de investigación y desarrollo. La ruta general que sigue el algoritmo:

1- Existe una semilla la cual puede ser computada por cada bloque, al escanear sus *headers* hasta ese punto.
2- A partir de la semilla, se puede computar un caché *pseudorandom*, de $J_{cacheinit}$ bytes[30] de valor inicial. Los clientes ligeros almacenan el caché.
3- A partir del caché, se puede generar un *dataset*, de $J_{datasetinit}$ bytes[31] de tamaño inicial, con la propiedad de que cada ítem del *dataset* depende solamente de un número pequeño de ítems del caché. Los clientes full almacenan el *dataset*. El *dataset* crece linealmente con el tiempo.

El proceso de *mining* consiste en tomar aleatoriamente piezas del *dataset* y aplicarles una función de hash[32]. La verificación se puede realizar con escasa memoria al utilizar el caché para regenerar específicamente las piezas del *dataset* que son necesarias, de este modo únicamente se necesita almacenar el caché. El *dataset* se actualiza cada J_{epoch} bloques[33], de modo que la mayor parte del esfuerzo de los *miner*s se enfocará en leer el *dataset*, no realizando cambios en el. [48], [56], [57]

[30] Actualmente son de 16Mb.
[31] Actualmente son de 1Gb.
[32] Es la función de hash Keccak, vista anteriormente.
[33] Actualmente son 30.000 bloques.

Árbol de Merkle

Es un árbol binario en el cual cada nodo hoja[34] contiene el TxID de su bloque correspondiente y cada nodo interno[35] así como el nodo raíz[36] contiene el hash de los TxID de sus nodos hijos. En el caso de que exista un TxID que no tenga par, se realiza el hash consigo mismo[37]. En la ilustración 15 se puede observar un árbol de Merkle como el utilizado en Bitcoin[16], [32],[58].

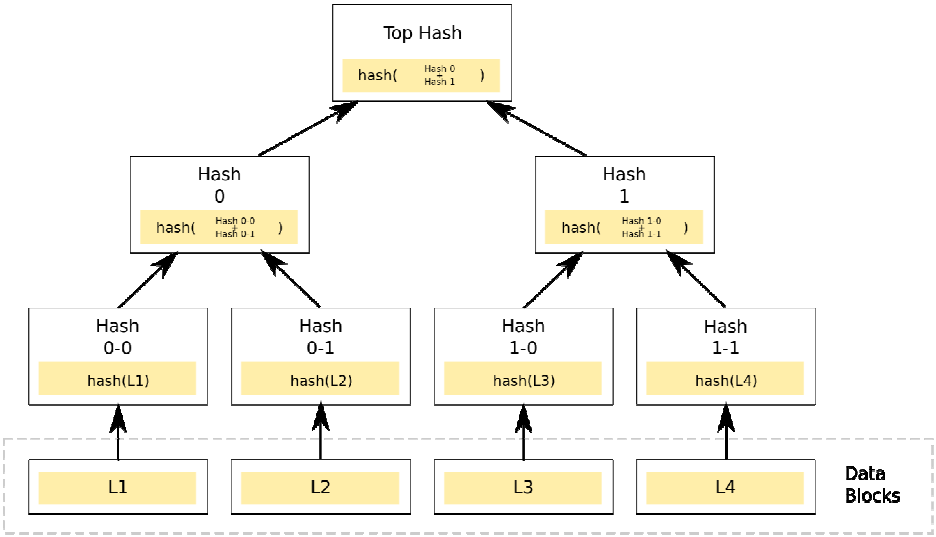

Ilustración 15: Árbol de Merkle. Fuente https://en.wikipedia.org/wiki/Merkle_tree

Esta estructura de datos permite a los usuarios de Bitcoin verificar por sí mismos si una transacción fue incluida en un bloque mediante la obtención de la raíz del árbol perteneciente al bloque que contiene la transacción buscada y una lista de los hashes intermediarios de un nodo full. Esta verificación se denomina prueba de Merkle, y al realizarla satisfactoriamente, uno puede verificar que, al menos para esa rama, es consistente recorriendo todo el árbol.

El encadenamiento en el tiempo junto a la criptografía fuerte, aseguran la integridad de los datos. Por ejemplo, para cambiar el valor de una transacción de 1 a 1000 en un bloque “v” deberá cambiarse la raíz del árbol de Merkle que viene incluido en la cabecera y el hash del bloque “v”, adicional deberá cambiarse todos los hashes de las cabeceras de los bloques posteriores v+1, v+2, y así hasta el último bloque de la cadena lo cual supone un esfuerzo computacional. A esto debe añadirse que cada bloque posee en la cabecera una prueba de trabajo que se genera con el contenido de la cabecera del bloque.[14]

[34] Son los nodos que no tienen hijos.
[35] Los nodos que no son hojas ni raíz.
[36] El nodo al final o principio del árbol.
[37] Esto se da en el caso de que exista un número impar de transacciones.

Árbol de Patricia

Los árboles de Merkle posibilitan la alta escalabilidad del *blockchain* al resumir grandes trozos de datos en hashes de una longitud minúscula (en comparación).

El *trie*[38] o *radix trie*, es un tipo de árbol de búsqueda, el cual es utilizado para almacenar un set dinámico de datos o un array asociativo donde las claves son generalmente cadenas. En la ilustración 16 se observa un ejemplo de un *radix trie*.

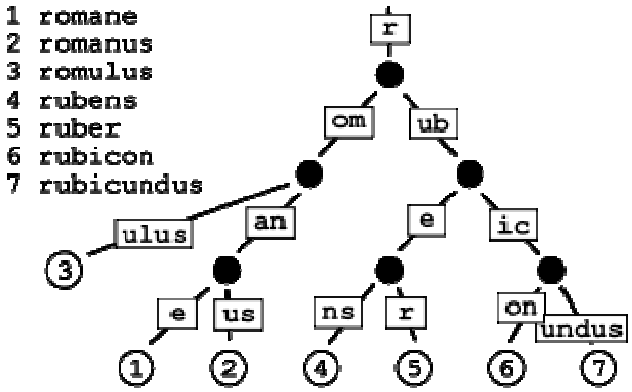

Ilustración 16: Árbol de Rádix. Fuente: [59]

El problema de este tipo de árboles es su ineficiencia. Si se desea almacenar una clave de 64 caracteres de longitud necesitará un espacio extra de 1 Kb y cada búsqueda y borrado tomará 64 pasos.

Para resolver este problema de ineficiencia, Ethereum introduce complejidad extra a la estructura de datos: primeramente, para que el árbol sea criptográficamente seguro, cada nodo se referencia por su hash (como en un árbol de Merkle)[39]. Este esquema, al igual que en el árbol de Merkle, convierte a la raíz en una huella criptográfica de toda la estructura de datos. Luego, se introducen los tipos de nodos:

- nodo *blank*, que simplemente está vacío,
- el nodo hoja que es una lista de $[clave, valor]$,
- los nodos de extensión que también son una lista $[clave, valor]$, pero en lugar de valor, se encuentra el hash de otro nodo,
- nodos ramas, los cuales son una lista de longitud 17. Los primeros 16 elementos corresponden a los 16 posibles caracteres hexadecimales en una clave, y el elemento final contiene un valor si existe un par $[clave, valor]$ donde la clave finaliza en el nodo rama.

Se puede observar un ejemplo del Trie de Ethereum en la ilustración 17.

[38] Del inglés reTRIEve.

[39] En la implementación actual, el hash se utiliza para una búsqueda en la base de datos leveldb.

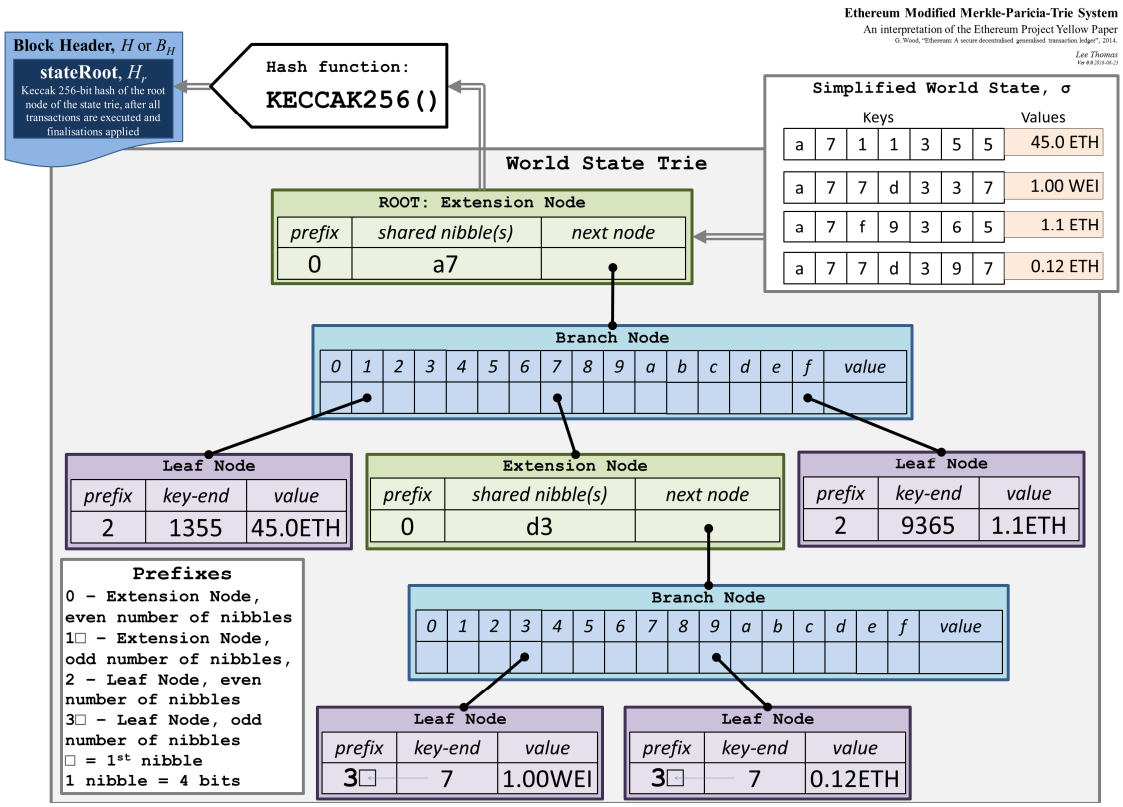

Ilustración 17: Ethereum Patricia Trie. Fuente: [23]

Una limitación particular de la implementación en Bitcoin de los árboles de Merkle es que a pesar de poder probar que una transacción fue incluida en un bloque, no se puede probar nada relacionado al estado actual (por ejemplo el balance actual). Para evitar esa limitación, Ethereum implementa tres árboles diferentes para tres tipos de objetos: transacciones, recibos y estado[60],[61],[62],[63].

Firmas Lamport

Las Firmas Lamport[40] o Esquema de Firmas de un solo uso, es un método de construcción de firmas digitales a partir de cualquier función de una vía criptográficamente segura, usualmente se utilizan funciones de hash[64], [65].

A continuación se describe, mediante un ejemplo, el mecanismo del esquema:

1- Alice utiliza un generador de números aleatorios para producir 256 pares de números aleatorios, siendo cada número de 256 bits de longitud, en total la longitud es de:

$$2 \times 256 \times 256 = 16\ KiB$$

Esta es la clave privada de Alice.

[40] Si bien las Firmas de Lamport aún no fueron implementadas en la red de Ethereum, resulta importante incluirla en este trabajo debido a que su implementación, se estima, es resistente al ataque de las computadoras cuánticas si se implementa en conjunto a funciones de hash de longitud criptográficamente aceptada.

2- Para generar la clave pública, se aplica una función de hash sobre cada uno de los 512 números aleatorios en la clave privada, obteniendo 512 hashes, cada uno de ellos de 256 bits de longitud (al igual que en la clave privada, la longitud es de 16 KiB). Estos 512 hashes representan la clave pública de Alice
3- Cuando Alice va a firmar un mensaje para Bob, aplica la función de hash a su mensaje, luego, por cada bit en el hash, basándose en el valor del bit (1 o 0), toma un número del par correspondiente de números que componen su clave privada (en caso de que sea 0 toma el primer número, de lo contrario, toma el segundo número).
   Estos números aleatorios conforman la firma de Alice y como cada número tiene longitud 256 bits, su firma en total será de

$$256 \times 256\ bits = 8\ KiB$$

4- Cuando Bob requiere verificar la firma de Alice en su mensaje, primeramente aplica la función de hash al mensaje.
5- Luego, utiliza los bits del digesto obtenido para tomar 256 de los digestos de la clave pública de Alice, siguiendo el mismo procedimiento del paso 3, si el bit es 0, toma el digesto del primer conjunto, de lo contrario toma el hash del segundo conjunto. Hasta obtener los 256 digestos.
6- Posteriormente, Bob aplica la función de hash a cada uno de los 256 números aleatorios de la firma de Alice. Obteniendo, 256 digestos más.
7- Bob compara los 256 digestos obtenidos en el paso 5 y 6, si concuerdan significa que el mensaje contiene la firma correcta.

En la ilustración 18 se puede observar el el funcionamiento de las Firmas Lamport.

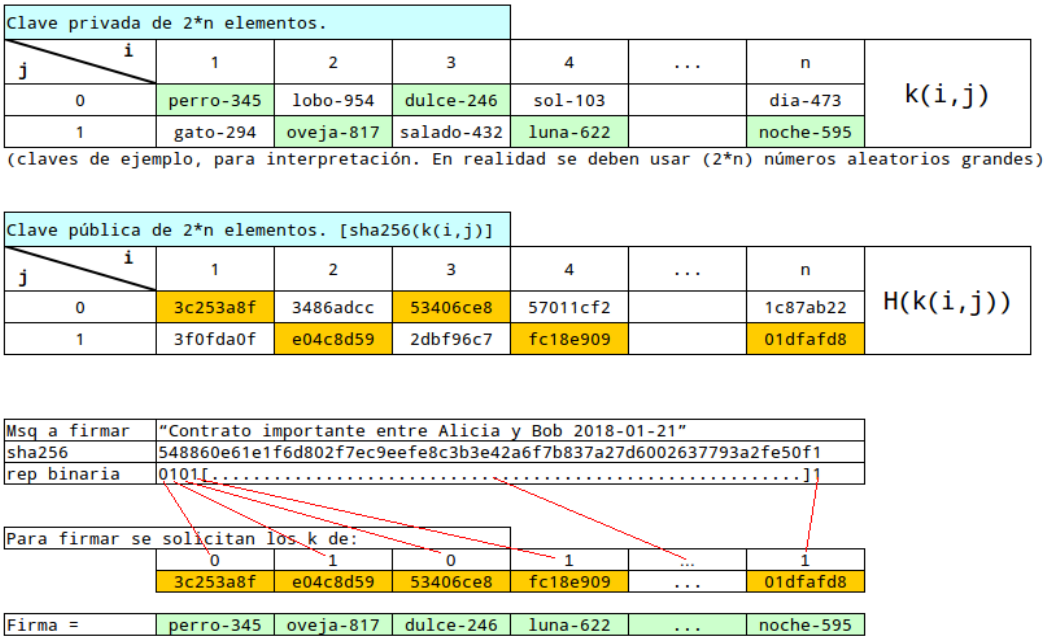

Ilustración 18: Firmas Lamport. Fuente: [66]

4- Ataques

La seguridad de las criptomonedas es directamente proporcional a la seguridad de su *blockchain*, mientras las primitivas criptográficas estén bien implementadas al mismo tiempo que sean fuertes y los nodos honestos tengan la mayor capacidad computacional de la red, se volverá muy difícil para un atacante llevar a cabo actividades maliciosas.

Se denomina actividad maliciosa a toda actividad que de alguna forma ocasione daños a una persona física o jurídica, entre algunas de ellas se puede nombrar el robo de identidad, fraude e intrusión a la red o el sistema, ya sea mediante virus, hackeo o malwares. El fraude, robo de identidad y hacking son particularmente relevantes para los *blockchain*[67].

Si bien la tecnología del *blockchain* está diseñada para ser anónima, transparente y resistente al robo, la popularidad creciente de las criptomonedas invita a cibercriminales a buscar maneras de explotar tanto aquellas vulnerabilidades presentes en el sistema como las de canal lateral.

A continuación se enumeran y describen las vulnerabilidades potenciales presentes tanto en Bitcoin como en Ethereum.

4.1- Doble gasto

El doble gasto se produce cuando un usuario de la red puede, simultáneamente, gastar dos veces el mismo conjunto de criptomonedas en dos diferentes transacciones. Por ejemplo; un cliente C_d crea una transacción $T_V^{C_d}$ siendo el tiempo t, enviando una cantidad de Bitcoins B_c a la dirección de un vendedor V para comprar algún producto. C_d transmite $T_V^{C_d}$ a la red. Siendo el tiempo $t' \approx t$, C_d crea y transmite otra transacción $T_{C_d}^{C_d}$ usando los mismos Bitcoins B_c a la dirección de C_d o a una billetera que se encuentra bajo su control. El ataque de C_d es exitoso si: C_d logra que V acepte $T_V^{C_d}$ (al ser confirmado por algún nodo *miner* y entregando V los productos comprados a C_d) y luego la mayor cantidad de nodos que realizan el *mining* aceptan en su lugar la transacción $T_{C_d}^{C_d}$ como válida y la agregan al *blockchain*[68].

Existen otras variantes del ataque del doble gasto, como por ejemplo el ataque Finney, en el cual un atacante genera un bloque en un *fork* antes de realizar una transacción y luego de realizarla, transmite el bloque generado a la red, siendo tal *fork* más largo que el actual de la *blockchain*, los nodos *miner*s se trasladarán a trabajar sobre el *fork* generado, el concepto general es el mismo.

4.2- Ataques de *Mining Pools*

Los *Mining Pools* son un conjunto de nodos *miner*s trabajando en sincronía como si fueran un solo nodo *miner*, el objetivo de los pools es incrementar el potencial computacional lo cual afecta directamente al

tiempo de verificación de los bloques, por lo tanto, incrementa la probabilidad de encontrar el siguiente bloque y obtener la recompensa en criptomonedas. Generalmente son gobernados por un manager, el cual se encarga de distribuir el trabajo no resuelto a los miembros del pool (*miner*s). La red de Bitcoin actualmente se compone únicamente de *miner*s individuales y pools privados o cerrados, mientras que la de Ethereum se compone de *miner*s individuales, pools privados y pools públicos.

El principal tipo de ataque que involucra a los *Mining Pools* se denomina *Selfish Mining*, mediante el cual, los nodos maliciosos en la red utilizan diversas técnicas para disminuir la capacidad de mining del resto de los nodos del pool o directamente evitar que estos últimos descubran efectivamente un bloque. Se puede destacar a las siguientes estrategias:

Ataque de descarte de bloques: el nodo malicioso oculta información (por ejemplo sobre el descubrimiento de un bloque) o la revela de un modo muy selectivo con los siguientes objetivos: obtener una recompensa mayor de la que le corresponde por su poder de computación utilizado y confundir al resto de los nodos *miner*s conduciéndolos a que desperdicien su poder computacional.

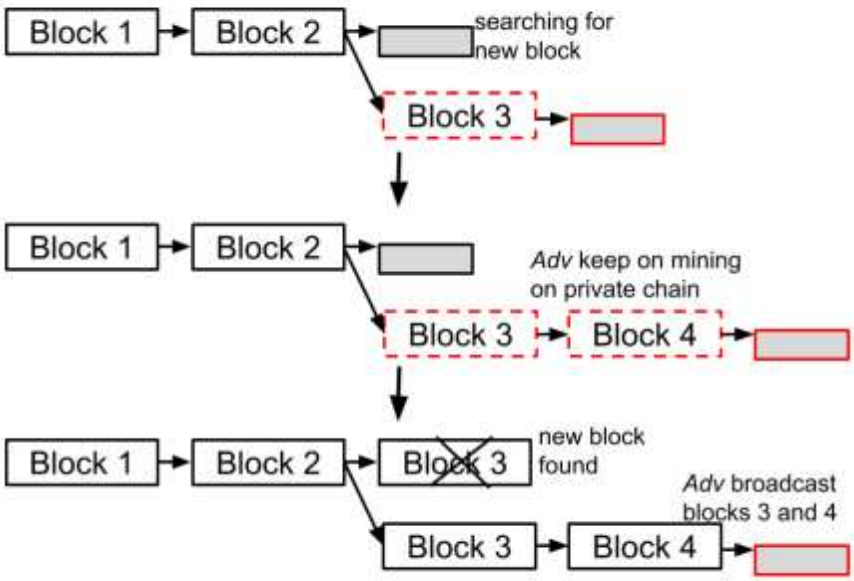

Ilustración 19: Block Discard Attack. **Fuente:** *[68]*

En la Ilustración 19 se puede observar que, en el *fork* de los nodos maliciosos, fue encontrado un bloque 3 pero no fue transmitido a la red, de este modo pueden empezar el *mining* del bloque 4 y obtienen la ventaja, si son capaces de encontrar el bloque 4 antes que los nodos honestos, en el momento en que estos últimos encuentren el bloque 3, los nodos maliciosos transmiten sus bloques minados, invalidando el bloque de los nodos honestos por la regla de la mayor longitud. La constante

ventaja en la adquisición de recompensas de los *selfish mining pools* atrae a nuevos nodos con la intención de obtener la mayor ganancia por su poder de computación, lo cual beneficia al *selfish mining pool*, porque implica que su poder computacional aumenta, eventualmente el *pool* podrá llevar a cabo el ataque del 51% y controlar el *blockchain* [69],[70].

Eclipse: en esta metodología, el atacante toma control de una cantidad de direcciones IP lo suficientemente grande como para monopolizar todas las conexiones salientes y entrantes de un nodo de la red. Posteriormente el atacante puede llevar a cabo otros tipos de ataques como el doble gasto y *Sefish Mining*. Este tipo de ataques puede ser de dos tipos: Ataque de Infraestructura, cuando el ataque se realiza sobre el ISP (*Internet Service Provider*) de la víctima, y de *Botnets* cuando el atacante puede manipular direcciones en un rango determinado de IPs.

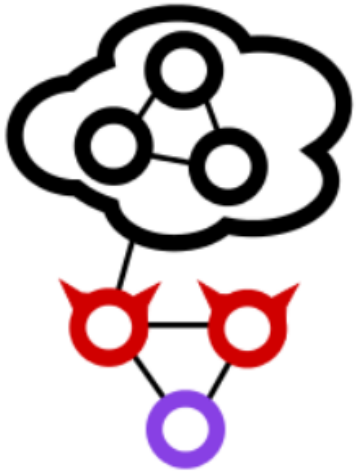

Ilustración 20: Eclipse Attack. Fuente: [71]

En la Ilustración 20 se puede observar que el nodo púrpura debe, obligatoriamente, conectarse a un nodo malicioso antes de poder llegar a la red. Lo que posibilita que los nodos maliciosos manipulen la información del nodo púrpura.

4.3- Amenazas de seguridad del lado del cliente

La creciente popularidad de las criptomonedas involucra un crecimiento en la cantidad de nuevos usuarios en la red, esto a su vez, fomenta la creación de aplicaciones de terceros que faciliten la gestión de las cuentas de los usuarios o sus billeteras. Como se mencionó anteriormente, las billeteras son archivos que contienen el par de claves/dirección, los cuales son utilizados para efectuar y recibir transacciones. En el caso de que dicho par de clave/dirección sean comprometidos, el usuario puede sufrir pérdidas inmediatas e irrevocables de dinero. Por lo tanto es esencial que el software de terceros implemente mecanismos eficaces que protejan contra hacking, bugs, uso incorrecto del software, etc.

La seguridad, tanto del protocolo de Bitcoin como del de Ethereum, dependen altamente de la seguridad de la criptografía de curvas elípticas, como se vio anteriormente, el algoritmo ECDSA se utiliza en conjunto con un número aleatorio para la generación de claves y firmas de las transacciones. Como se demuestra en [72]. El valor aleatorio debe ser diferente para cada firma, de lo contrario, un atacante podría derivar la clave privada del usuario.
Por otra parte, se presenta la amenaza de las computadoras cuánticas, las cuales, se estima que pueden quebrar en tiempos razonables las claves de los algoritmos de clave pública-privada actuales. [73]

ECDLP in $E(\mathrm{F}_p)$			RSA mod N	
$\lceil \log_2(p) \rceil$ bits	#Qubits	Time (sec)	$\lceil \log_2(N) \rceil$ bits	#Qubits
110	1014	273	512	1026
160	1466	711	1024	1024
192	1754	1149	-	-
224	2042	1881	2048	4098
256	2330	3848	3072	6146
384	3484	17003	7680	15362
521	4719	42888	15360	30722

Tabla 5 Comparativa entre los recursos necesarios para resolver el problema del logaritmo discreto de las curvas elípticas y RSA con módulo N. Fuente: [73]

Esta amenaza obliga a las diferentes implementaciones de *blockchain* a migrar sus actuales sistemas de cifrado a sistemas que puedan resistir los ataques de computadoras cuánticas.
En el caso de Ethereum, se implementarán las Firmas Lamport como nuevo sistema de firmas en la versión Constantinople, la cual es la segunda parte de la etapa Metrópolis de Ethereum[41].

En el mercado existen varias implementaciones de billeteras que buscan incrementar la seguridad del almacenado de claves, incluyendo software, hardware, en papel y de cerebro.

Cabe destacar que la mayor cantidad de ataques exitosos se llevaron a cabo teniendo como vector de ataque al software de terceros, en lugar de a defectos de los protocolos, pero se tratará este tema en detalle en el punto de amenazas materializadas.

4.4- Ataques a la red

Los ataques a la red se enfocan en explotar vulnerabilidades en la implementación y el diseño de los protocolos de las criptomonedas.

[41] La etapa actual al momento de redactar este trabajo es Byzantium.

En esta categoría de ataques se destaca (por ser la más común) a la denominada Denegación de Servicio Distribuido (DDoS[42]), las cuales tienen como objetivo a los centros de intercambio de divisas, los *mining pools* y los proveedores de billeteras electrónicas.

A diferencia de un ataque de denegación de servicio DoS, en el cual el ataque es llevado a cabo por un solo atacante, en la DDoS, múltiples atacantes lo hacen simultáneamente. Los nodos *miner*s maliciosos pueden llevar a cabo un ataque de DDoS (accediendo a algún tipo de *Botnet*[43]) hacia sus competidores, removiendo la efectividad de estos últimos en la red e incrementando la velocidad efectiva de los primeros. En estos ataques, los adversarios tratan de agotar los recursos de la red con el objetivo de interrumpir el acceso a los usuarios honestos, un ejemplo de esta estrategia puede consistir en que los nodos maliciosos congestionen con solicitudes de un gran número de clientes (como transacciones falsas) a un nodo honesto, transcurrido cierto tiempo, el nodo honesto empezará a descartar todas las solicitudes incluyendo aquellas de clientes honestos. [74]

Se puede deducir entonces que uno de los principales objetivos de los ataques del tipo DDoS es desmotivar a los nodos honestos y por lo tanto obligarlos a retirarse del proceso de *mining* debido a que sus chances de ganar la competencia de encontrar el bloque son mucho menores. De esta forma, los nodos maliciosos, al remover a los *miner*s individuales como también a *mining pools* pequeños, imponiéndose en la red.

El otro tipo de ataque a la red que debe ser considerado es el denominado *Malleability Attack*, el cual, si bien no se ha registrado en la red de Ethereum debido a que las transacciones en esta red referencian a cuentas en lugar de a otras transacciones como se describió con anterioridad en este trabajo [75], afectó de sobremanera al que en su momento fue el mayor Exchange de bitcoins del mundo, Mt. Gox [76], y fue la base de uno de los mayores ataques registrados a la red de Bitcoin [77].

Los ataques consisten en explotar una vulnerabilidad en el protocolo de Bitcoin, lo cual permite al atacante obstruir la cola de transacciones. Esta cola de transacciones contiene todas las transacciones pendientes que están a punto de ser transmitidas a la red.

Mientras ocurre la obstrucción, el atacante puede incluir transacciones fraudulentas con alta prioridad, convirtiéndose en el mayor pagador para los *miner*s. Cuando los *miner*s tratan de verificar estas transacciones, se encontrarán con que son falsas y habrán desperdiciado una considerable cantidad de tiempo en el proceso[44].

[42] Distributed Denial of Service

[43] El término Botnet hace referencia a una red formada por un conjunto de ordenadores bajo el control centralizado de un tercero, denominado Bot Herder. El control se consigue mediante la instalación de un software llamado Bot en cada host.

[44] Se dice que una primitiva criptográfica es maleable si dado un texto cifrado c_1, es posible encontrar otro texto cifrado c_2 tal que el resultado de descifrar $c_1(d_k(c_1))$ y $c_2(d_k(c_2))$ tienen similitud.

Como variante de los *malleability attack*, se presenta el ataque denominado *transaction malleability*, el mecanismo del ataque es el siguiente:

1- Dada una transacción $T^{n}_{A\to B}$, la cual transfiere n bitcoins desde la dirección A a la dirección B.
2- Es posible crear otra transacción T' la cual es sintácticamente diferente de $T^{n}_{A\to B}$ (por ejemplo, T y T' tienen distinto TxID), pero por otro lado, es idénta semánticamente (por ejemplo, T' también transfiere n bitcoins desde A a B).

Un punto a destacar, es que el atacante ni siquiera necesita conocer la clave privada de A para llevar a cabo el ataque. La principal razón del éxito de este ataque es que cada transacción esta unívocamente identificada por su TxID, por lo tanto, en algunos casos, T' será considerada una transacción diferente a $T^{n}_{A\to B}$.

Si bien la implementación de referencia de Bitcoin es inmune al ataque, algunos centros de intercambio de divisas como Mt. Gox, utilizaban implementaciones propias, lo que las hacían aparentemente vulnerables. Se detallará el acontecimiento en la siguiente sección.

4.5- Amenazas materializadas

En esta sección se discutirá brevemente sobre aquellos ataques que lograron explotar vulnerabilidades tanto en Bitcoin como en Ethereum.

Uno de los mayores ataques registrados en la historia de Bitcoin[45] fue el que tuvo como objetivo al que en su tiempo fue el mayor centro de intercambio de divisas, Mt. Gox. Fue un ataque masivo que ocasionó una pérdida de 744,408 Bitcoins[46]. La metodología del ataque fue la de *transaction malleability*, como se describió en la sección anterior. Sin embargo, la legitimidad del ataque no ha sido completamente confirmada aun, pero fue suficiente como para que Mt. Gox cese sus actividades y que el valor de los Bitcoins en el mercado baje rotundamente.

Entre otras víctimas de la misma metodología se encuentran Silk Road[47], en el anuncio oficial de la compañía se escribió lo siguiente: “Las investigaciones iniciales indican que un proveedor ha explotado una vulnerabilidad recientemente descubierta en el protocolo Bitcoin conocida como *transaction malleability* para retirar repetidamente Bitcoins de nuestro sistema hasta que estuvo completamente vacío.” [78]

Para eliminar esta vulnerabilidad, se implementó el BIP66 en el bloque 363724[79], un *soft fork* que obliga a los siguientes bloques a implementar el estándar ASN.1 codificado con DER. Dicho estándar provee una

[45] El mayor ataque registrado al momento de escribir este trabajo es el que afectó a Coincheck

[46] Alrededor de 400 millones de dólares al momento del ataque.

[47] El mayor sitio de comercio electrónico anónimo, famoso por su gran oferta de drogas ilegales y el uso de Tor y Bitcoin para proteger la privacidad de los usuarios.

manera única de representación de cualquier valor del tipo ASN.1 como una cadena de octetos. [80]
Por otra parte, dentro del ecosistema de Ethereum, en el año 2016, una organización autónoma descentralizada llamada The DAO, fue víctima de un ataque en el cual el responsable logró drenar más de 3,6 millones de Ether[48], alrededor del 14% del Ether total de ese momento (Si bien el ataque no explotó una vulnerabilidad de la red, vale la pena describirlo por el impacto del mismo).
Una DAO tiene como meta traducir las reglas de negocio y toma de decisiones de una organización al código fuente de un contrato inteligente, eliminando la necesidad de documentos y personas gobernando, por lo tanto, creando una estructura de control descentralizado [81]. The DAO es el nombre de una DAO en particular. El contrato inteligente que regía la organización poseía una vulnerabilidad en el código que permitía al atacante realizar llamadas recursivas sin límites. En la Ilustración 21 se puede observar la metodología del ataque de una manera simplificada.

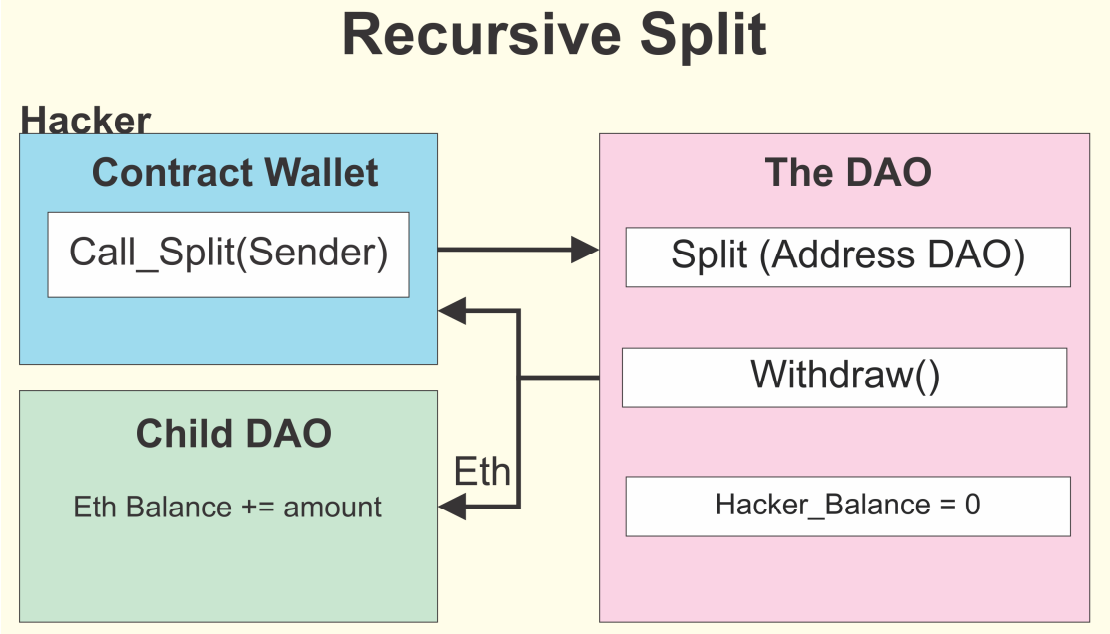

Ilustración 21: Recursive Split Attack. Fuente: Elaboración propia con información de [82]

La solución a este ataque fue un *hard fork*, lo que anuló los efectos de las transacciones involucradas en el ataque. Sin embargo, aquellos que no aceptaron el *hard fork* siguieron trabajando sobre el *blockchain* original bajo la siguiente premisa "el valor principal de un *blockchain* es la inmutabilidad; las transacciones válidas nunca pueden ser borradas u olvidadas. Código es Ley" creando así, la red de Ethereum Classic.

[48] Alrededor de 50 millones de dólares al momento del ataque.

La decisión de llevar a cabo el hard fork fue difícil de tomar, pero uno de los factores clave que influyeron fue que en el futuro, cuando Ethereum cambie las reglas de consenso a PoS, el poseedor de mayor cantidad de Ether tendría mayor poder de decisión, por lo tanto, dejar en manos de un criminal el poder de la toma de decisiones de la red no era la mejor opción.

Posteriormente han sido reportados otros ataques pero la gran mayoría han sido llevadas a cabo explotando vulnerabilidades relacionadas a software de terceros, por ejemplo, hacia finales del 2017, Cisco reportó una metodología de ataques utilizada por un grupo de hackers ucranianos, en la cual, se utilizaban los anuncios de Google para llevar a los usuarios a sitios maliciosos que tomaban información de las billeteras de los mismos. Los hackers pudieron robar en el transcurso de tres años alrededor de 50 millones de dólares[83].

5- Evolución a Futuro

Las criptomonedas, en algunos países, tienden a crecer en aceptación, mientras que en otros, se están estableciendo reglamentaciones para su uso legal o están siendo directamente prohibidas (en China han sido prohibidas, se apunta como principal causa al alto consumo energético que requiere el proceso de *mining*) [84].

Sin embargo sean o no aceptadas, las mayores barreras a superar por las criptomonedas están relacionadas al cibercrimen. El hecho de que un usuario pueda perder todo su balance de forma irreversible y no rastreable podría en un futuro ser resuelto por nuevas tecnologías o algoritmos que mejoren el protocolo, pero actualmente es la mayor limitante para el ingreso de nuevos usuarios al sistema.

La realidad es que los sistemas actuales proveen mecanismos garantizan verificación de identidad, reversibilidad de transacciones, estándares de auditoría y sistemas de investigación en caso de que haya ocurrido algún tipo de fraude u otro crimen. Mientras que los fundamentos de las criptomonedas, al promover la privacidad de las transacciones y usuarios, como efecto secundario no permiten realizar lo escrito anteriormente.

Otra de las limitantes es la velocidad de las transacciones, sistemas como los de VISA o MasterCard pueden manejar más de veinticuatro mil transacciones por segundo, mientras que en la red de Bitcoin se llega apenas a 7 y la de Ethereum a 15.

Por otra parte, algunos bancos han comenzado a trabajar en un proyecto denominado Ripple [85] el cual tiene como meta desarrollar un sistema de crédito basado en el paradigma punto a punto. Esto es una señal de que los bancos están perdiendo el escepticismo hacia las criptomonedas. En un futuro, es posible que cada banco o sistema de pagos administre su propia red con su propia criptomoneda y el hecho de que en un reporte, la red de

Ripple ha procesado cincuenta mil transacciones por segundo[86], superando a lo visto anteriormente, mejora sus probabilidades de adopción.

En [14] se describen varios proyectos de implementación de la tecnología del *blockchain* a través de los contratos inteligentes proveídos por la red de Ethereum para mejorar ciertos aspectos de integración entre pacientes, proveedores de seguros, instituciones médicas y el gobierno. La problemática de las historias clínicas posee requerimientos de seguridad que pueden ser satisfechos por medio de esta tecnología.

Finalmente, el *blockchain* no se limita a las criptomonedas, han sido propuestas varias implementaciones para el *blockchain* como almacenamiento de documentos legales, sistemas de votación, servidores de DNS, libros de tomas de decisiones, antecedentes médicos. Pero aún no se han visto casos de éxito que hayan sido extendidos [87]–[90].

6- Conclusiones

Como se vio en el inicio del trabajo, el concepto de criptomonedas no es algo nuevo, es una idea que viene circulando hace décadas, sin embargo, gracias a la implementación del *blockchain* con Bitcoin es que fueron capaces de despegar como nueva tecnología.

Si bien Bitcoin presenta un protocolo robusto, posee la limitante de servir únicamente para la transferencia de valores. Limitante que no se presenta en Ethereum, la cual fue más allá implementando los contratos inteligentes mediante un lenguaje de programación propio, Solidity, un lenguaje Turing completo[49], mediante el cual no existen límites en cuanto a usos para la red: entre las opciones más resaltantes se encuentran:

Los mismos contratos inteligentes, los cuales, como están escritos con lógica del ordenador, no permiten ningún tipo de interpretación, se ejecutan tal y como fueron programados, no pueden ser censurados, estar inactivos, interferidos ni usarse para cometer fraude.

Las organizaciones autónomas descentralizadas, en las cuales todos los miembros pueden votar sobre la dirección que tomará la organización, los proyectos en los que invertirá, cómo reaccionará ante eventos, etc. Creando una democracia total y transparente para todos los usuarios.

Desarrollar aplicaciones descentralizadas, las cuales replican sus datos entre los nodos, de este modo, se brinda privacidad en cuanto a los datos de los usuarios y alta disponibilidad, evitando así afectar a usuarios y otras aplicaciones en caso de que una sola aplicación sea comprometida.

[49] Un lenguaje de programación se considera Turing Completo a aquel que imita a la máquina de Turing, es decir, puede ser utilizado para computar cualquier cosa considerando que posee memoria ilimitada y probabilidad cero de fallas[91].

En la tabla 6, se observa un resumen de la comparativa realizada en este trabajo. Los valores de la tabla corresponden a la fecha de redacción de este trabajo.

	Bitcoin	**Ethereum**
Fundador	Satoshi Nakamoto	Vitalik Buterin et al
Bloque génesis	01-09-2009	30-07-2015
Objetivo	Dinero Digital	Contratos Inteligentes
Transacción	Enviar BTC de Alice a Bob	Enviar ETH de Alice a Bob si se cumplen las condiciones del contrato.
Capitalización de mercado	≈164,5 Mil millones	≈70 Mil millones
Tamaño del blockchain	≈194 Gigabytes	≈456 Gigabytes
Tiempo de generación de bloques	10 minutos	15 segundos
Cantidad de bloques[50]	519.808	5.500.300
Recompensa por bloque	12,5 BTC	≈3,6 ETH
Direcciones	27 a 34 caracteres alfanuméricos	40 caracteres
Consenso	Prueba de trabajo	Prueba de trabajo, Prueba de participación
Algoritmo de Hash (PoW)	SHA-256	Ethash
Algoritmo de firma digital	ECDSA	ECDSA, Firmas Lamport
Algoritmo de hash (Direcciones)	SHA-256, RIPEMD-160	KECCAK-256
Mecanismo de integridad	Árbol de Merkle	Árboles de Patricia

Tabla 6 Comparativa de características de Bitcoin y Ethereum. Fuente: elaboración propia con información de [92]

Si bien las primitivas criptográficas implementadas en ambos protocolos son consideradas criptográficamente seguras al momento de redacción de este trabajo, las funciones de hash implementadas por Ethereum (Ethash, KECCAK-256) son mejores que las implementadas por Bitcoin (SHA-256, RIPEMD-160) al poseer mayor seguridad frente a colisiones y pre-imágenes.

El algoritmo utilizado para firmas digitales en ambos protocolos es el basado en curvas elípticas, dicho algoritmo basa su fortaleza en el problema del logaritmo discreto. En la actualizad no se conoce un algoritmo que pueda resolver dicho problema utilizando recursos computacionales tradicionales,

[50] Cantidad al momento de redacción de este trabajo.

sin embargo se presume que la computación cuántica podría ser capaz de eliminar dicha fortaleza.

Se destaca la iniciativa de Ethereum hacia el futuro del protocolo mediante la implementación de métodos de verificación personalizados, los cuales dejarán en manos del usuario la decisión sobre los algoritmos a utilizar al momento de verificar las firmas en las transacciones.

Este conjunto de técnicas criptográficas fuertes junto a una infraestructura descentralizada, proveen a ambos protocolos la capacidad de realizar transacciones punto a punto de forma anónima. Este anonimato puede tener ventajas y desventajas. Se presenta la situación en la que una persona no desea que el gobierno posea todos los datos de las transacciones que realiza, pero por el contrario, desea que el gobierno posea todos los datos de las transacciones que realizan los pedófilos.

Por otro lado, es importante destacar el papel que juegan los diferentes tipos de ataques, teóricos y materializados que afectaron y aún afectan a los protocolos, en la aceptación de las criptomonedas. La gran cantidad de pérdidas monetarias irreversibles que han ocurrido, afectan su valor en el mercado, sumado a esto, la dificultad de inserción de nuevos usuarios a este mundo, afecta negativamente en su popularidad. Se habla de dificultad porque es necesario que el usuario comprenda el protocolo y los riesgos asociados al mismo.

También es importante destacar que las legislaciones actuales están poco o nada preparadas para reglamentar el uso y aceptación de las criptomonedas, en [1] la autora presenta el estado de las legislaciones en la República Argentina y en [14] el autor estudia ciertas legislaciones internacionales. Este aspecto del ecosistema de las criptomonedas quedó fuera del alcance de este trabajo para enfocarlo en el aspecto técnico de las mismas.

Es considerable el esfuerzo que han puesto los investigadores en desarrollar implementaciones para la tecnología del *blockchain*, en [14] se presentan implementaciones sumamente interesantes para resolver problemáticas relacionadas a historiales clínicos. Sin embargo, estas iniciativas así como la del voto electrónico entre otras, han fallado en ser aceptadas ampliamente. Esto puede deberse simplemente al miedo de las entidades a migrar a una tecnología aún inmadura. Como trabajo futuro se podrían investigar las dificultades que se presentan ante los proyectos implementados sobre el *blockchain*.

Otra desventaja a ser superada para que la tecnología pueda evolucionar y desplazar los sistemas tradicionales está relacionada al consumo energético, el *mining* utilizando como método de consenso el POW consume cantidades de energía exorbitantes, pero esto puede ser superado si se decide cambiar el método de consenso.

Se concluye que la tecnología de las criptomonedas aún tiene un largo camino por delante con varias barreras que superar para alcanzar el nivel de

adopción que posee actualmente el dinero real. Sin embargo el panorama es prometedor debido a las comunidades que apoyan las diferentes redes, desarrollando cada vez mejores productos para la administración del balance e interacción del usuario con las criptomonedas.

Al mismo tiempo, mientras mayor sea la popularidad de las mismas, mayor será el interés de los cibercriminales para encontrar una manera de obtener beneficios mediante la explotación de vulnerabilidades. Motivo por el cual es necesario mantener las primitivas criptográficas que sostienen el protocolo actualizadas y estandarizadas para evitar posibles brechas de seguridad.

Sobre la tecnología del *blockchain*, es necesario desarrollar plataformas que consuman menos recursos que las existentes actualmente, teniendo en cuenta el espacio en disco y el procesamiento.

Se decide finalmente, que Ethereum es actualmente un mejor protocolo que Bitcoin debido a la implementación de primitivas criptográficas más seguras, la posibilidad de usar el protocolo para mucho más que transferencia de valores y por la flexibilidad que prevé para cambiar la regla de consenso.

7- Bibliografía Específica

[1] N. Fernández, «Dinero Electrónico». abr-2017.

[2] D. G. Fernández Sánchez, «Criptodivisas - Criptocurrency». 2015.

[3] D. Chaum, «Blind Signatures for untraceable payments». 1982.

[4] A. J. Menezes, P. C. Van Oorschot, y S. A. Vanstone, *Handbook of applied cryptography*. Boca Raton: CRC Press, 1997.

[5] J. Pitta, «Requiem for a Bright Idea», 11-ene-1999. .

[6] B. Tedeschi, «E-Commerce Report; Seller of Online Currency May Have Been Victim of Fraud.», *The New York Times*, 27-ago-2001. .

[7] D. Wei, «B-money». 1998.

[8] Kyle Randolph, «Ethereum White Paper». .

[9] C. Mitchell y Institution of Electrical Engineers, Eds., *Trusted computing*. London: Institution of Electrical Engineers, 2005.

[10] B. Lim, H. Lee, y S. Kurnia, «Exploring the reasons for a failure of electronic payment systems: A case study of an Australian company», *J. Res. Pract. Inf. Technol.*, vol. 39, n.º 4, pp. 231–244, 2007.

[11] T. Churchouse, «Become a blockchain expert in 1,384 words», 13-oct-2017. .

[12] A. Narayanan, *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton: Princeton University Press, 2016.

[13] M. Arnold, P. Stafford, y J. Wild, «Technology: Banks seek the key to blockchain», *Financial Times*, 01-nov-2015. [En línea]. Disponible en: https://www.ft.com/content/eb1f8256-7b4b-11e5-a1fe-567b37f80b64. [Accedido: 23-nov-2017].

[14] J. Aguirre, «Cadena de Bloques: Potencial aplicación a historias clínicas electrónicas.» 2017.

[15] Hessiod Services LLC, «Bitcoin Mining», *Hessiod Services LLC.* .

[16] C. Pomerance y CRYPTO, Eds., *Advances in cryptology - CRYPTO '87: proceedings*. Berlin: Springer, 1988.

[17] R. Faride, «Is Bitcoin an invitation for money laundering?», *riazfaride.com*, 12-jul-2012. .

[18] «Consensus - Bitcoin Wiki». [En línea]. Disponible en: https://en.bitcoin.it/wiki/Consensus. [Accedido: 24-nov-2017].

[19] Investopedia, «Soft Fork, https://www.investopedia.com/terms/s/soft-fork.asp», *www.investopedia.com*. .

[20] A. H. CFA, «Soft Fork», *Investopedia*, 25-jul-2016. [En línea]. Disponible en: https://www.investopedia.com/terms/s/soft-fork.asp. [Accedido: 24-nov-2017].

[21] Investopedia, «Hard Fork», *www.investopedia.com*. .

[22] A. H. CFA, «Hard Fork», *Investopedia*, 25-jul-2016. [En línea]. Disponible en: https://www.investopedia.com/terms/h/hard-fork.asp. [Accedido: 24-nov-2017].

[23] «blockchain - Ethereum block architecture - Ethereum Stack Exchange». [En línea]. Disponible en: https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture. [Accedido: 23-nov-2017].

[24] NIST, «NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition», 02-oct-2012. .

[25] H. Jameson, «Which cryptographic hash function does Ethereum use?», *stackexchange*. .

[26] B. H. Bloom, «Space/time trade-offs in hash coding with allowable errors», *Commun. ACM*, vol. 13, n.º 7, pp. 422–426, 1970.

[27] IEEE Spectrum, «Bitcoin Infographic», *Visually*. .

[28] Bitcoin Wiki, «Dirección», *Bitcoin Wiki*, 12-dic-2012. .

[29] Bitcoin Wiki, «Technical background of version 1 Bitcoin addresses», *Bitcoin Wiki*, 28-oct-2017. .

[30] «Bitcoin Wallet». .

[31] Bitcoin Wiki, «Transaction», *Bitcoin Wiki*, 01-ene-2018. .

[32] «Bitcoin Developer Guide», 01-ene-2017. .

[33] Bitcoin Wiki, «Satoshi», *Bitcoin Wiki*, 15-feb-2017. .

[34] Bitcoin Wiki, «Protocol Rules», *Bitcoin Wiki*, 25-ago-2017. .

[35] E. B. Barker y J. M. Kelsey, «Recommendation for Random Number Generation Using Deterministic Random Bit Generators», National Institute of Standards and Technology, NIST SP 800-90Ar1, jun. 2015.

[36] Y. Malhotra, «Bitcoin Protocol: Model of "Cryptographic Proof" Based Global Crypto-Currency & Electronic Payments System», 04-dic-2013. .

[37] «Account Management», 16-mar-2016. .

[38] «Introducción a JSON», 2015. .

[39] V. Kobel, «Generating a usable Ethereum wallet and its corresponding keys», 15-feb-2017. .

[40] V. Pradeep, «Ethereum's Memory Hardness Explained, and the Road to Mining It with Custom Hardware», 28-abr-2017. .

[41] J. Coleman, «What proof of work function does Ethereum use?», 23-ene-2016. .

[42] Z. Ramzan, «Bitcoin: Proof of Work», 2013. .

[43] M. Payeras, A. Isern, y M. Puigserver, «Sistemas de Pago Electrónico», 03-nov-2014. .

[44] K. Bjoernsen, «Koblitz Curves and its practical uses in Bitcoin security», *Order E GF 2k*, vol. 2, n.º 1, p. 7, 2009.

[45] H. Mayer, «ECDSA security in bitcoin and ethereum: a research survey», *CoinFaabrik June*, vol. 28, 2016.

[46] Bitcoin Wiki, «Codificación Base58Check», *Bitcoin Wiki*, 12-dic-2012. .

[47] Medium, «Bitcoin Address Generation», *Medium*. .

[48] G. Wood, «Ethereum: A secure decentralised generalised transaction ledger», *Ethereum Proj. Yellow Pap.*, vol. 151, pp. 1–32, 2014.

[49] B. Preneel, A. Bosselaers, y H. Dobbertin, *The cryptographic hash function RIPEMD-160*. Citeseer, 1997.

[50] G. Bertoni, J. Daemen, P. Michaël, G. Van Assche, y R. Van Keer, «The sponge and duplex constructions». 2017.

[51] Wikipedia, «SHA-3», *Wikipedia*, 2012. .

[52] B. Guido, D. Joan, P. Michaël, y V. A. Gilles, «The K reference», 2011.

[53] *[FOSDEM 2013] Keccak, More Than Just SHA3SUM*. 2014.

[54] *Píldora formativa 46: ¿Qué son SHA-2 y SHA-3?* 2017.

[55] cseberino, «Why Ethereum Classic Uses An Incorrect SHA3 Implementation», 2017. .

[56] V. Buterin, «Ethash», *Github*, 21-ene-2016. .

[57] V. Buterin, «Dagger-Hashimoto.md», 07-dic-2014. .

[58] Bitcoin Wiki, «Protocol Documentation», *Bitcoin Wiki*, 2017. .

[59] Wikipedia, «Radix Tree», *Wikipedia*. .

[60] S. Tual, «Patricia Tree», 16-feb-2014. .

[61] E. Buchman, «Understanding the ethereum trie». .

[62] V. Buterin, «Merkling in Ethereum», 15-nov-2015. .

[63] *DEVCON1 Understanding the Ethereum Blockchain Protocol - Vitalik Buterin*. 2016.

[64] L. Lamport, «Constructing digital signatures from a one-way function», Technical Report CSL-98, SRI International Palo Alto, 1979.

[65] Wikipedia, «Lamport Signature», *Wikipedia*, 2017. .

[66] Wikimedia, «Lamport Signature», *Wikimedia*. .

[67] J. J. Xu, «Are blockchains immune to all malicious attacks?», *Financ. Innov.*, vol. 2, n.º 1, dic. 2016.

[68] M. Conti, S. Kumar E, C. Lal, y S. Ruj, «A Survey on Security and Privacy Issues of Bitcoin». 03-jun-2017.

[69] I. Eyal y E. G. Sirer, «Majority is not enough: Bitcoin mining is vulnerable», en *International conference on financial cryptography and data security*, 2014, pp. 436–454.

[70] P. Camacho, «Analyzing Bitcoin Security». 20-jun-2016.

[71] E. Heilman, A. Kendler, A. Zohart, y S. Goldberg, «Eclipse Attacks on Bitcoin's Peer-to-Peer Network», *Boston University*, oct-2015. .

[72] Howgrave-Graham y Smart, «Lattice Attacks on Digital Signature Schemes», *Kluwer Acad. Publ.*, vol. 23, n.º Designs, Codes and Cryptography, pp. 283-290, ago. 2001.

[73] M. Roetteler, M. Naehrig, K. M. Svore, y K. Lauter, «Quantum resource estimates for computing elliptic curve discrete logarithms», en *International Conference on the Theory and Application of Cryptology and Information Security*, 2017, pp. 241–270.

[74] M. Vasek, M. Thornton, y T. Moore, «Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem», en *International Conference on Financial Cryptography and Data Security*, 2014, pp. 57–71.

[75] B. Vitalik, «Transaction Malleability: Does Ethereum have this issue?», 2016. .

[76] Wikipedia, «Mt. Gox», *Wikipedia*, 2017. .

[77] E. Spaven, «Bitcoin Exchanges Under "Massive and Concerted Attack"», 13-feb-2014. .

[78] A. Greenberg, «Silk Road 2.0 "Hack" Blamed On Bitcoin Bug, All Funds Stolen», 13-feb-2014. .

[79] StephenM347, «When did BIP66 switch from activation to enforcement?», 06-jul-2015. .

[80] B. Kaliski, «A Layman's Guide to a Subset of ASN.1, BER, and DER», 01-nov-1993. .

[81] D. Siegel, «Understanding The DAO Attack», 25-jun-2016. .

[82] *The DAO explicado. Meetup Ethereum Madrid.* 2016.

[83] J. Wieczner, «Hackers Stole $50 Million in Cryptocurrency Using "Poison" Google Ads», 14-feb-2018. .

[84] S. Hsu, «China's Shutdown Of Bitcoin Miners Isn't Just About Electricity», 15-ene-2018. .

[85] «Ripple», 2018. .

[86] Kaustav, «How Ripple (XRP) Outdid the Transaction Speed of VISA», 28-mar-2018. .

[87] P. Schrodt, «Cryptocurrency Will Replace National Currencies By 2030, According to This Futurist», 01-mar-2018. .

[88] A. Extance, «The future of cryptocurrencies: Bitcoin and beyond», 30-sep-2015. .

[89] «The Future Of Cryptocurrency», 2018. .

[90] K. Stinchcombe, «Ten years in, nobody has come up with a use for blockchain», 22-dic-2017. .

[91] Brailsford, *Turing Complete - Computerphile*. 2016.

[92] Bitinfocharts, «Cryptocurrency statistics», *Cryptocurrency statistics*. .

8- Bibliografía general

[1] I.-C. Lin y T.-C. Liao, «A Survey of Blockchain Security Issues and Challenges.», *IJ Network Security*, vol. 19, n.º 5, pp. 653–659, 2017.

[2] F. Tschorsch y B. Scheuermann, «Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies», *IEEE Communications Surveys & Tutorials*, vol. 18, n.º 3, pp. 2084-2123, 2016.

[3] J. Teutsch, S. Jain, y P. Saxena, «When cryptocurrencies mine their own business», en *International Conference on Financial Cryptography and Data Security*, 2016, pp. 499–514.

[4] N. Atzei, M. Bartoletti, y T. Cimoli, «A survey of attacks on Ethereum smart contracts (SoK)», en *International Conference on Principles of Security and Trust*, 2017, pp. 164–186.

[5] M. Swan, *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc., 2015.

[6] E. Wall y G. Malm, «Using blockchain technology and smart contracts to create a distributed securities depository», 2016.

[7] A. Guadamuz y C. Marsden, «Blockchains and Bitcoin: Regulatory responses to cryptocurrencies», 2015.

[8] M. Crosby, P. Pattanayak, S. Verma, y V. Kalyanaraman, «Blockchain technology: Beyond bitcoin», *Applied Innovation*, vol. 2, pp. 6–10, 2016.

[9] K. Weiss, «Digital Currency: The Transition to a Cashless Society», 2016.

[10] A. Back *et al.*, «Enabling blockchain innovations with pegged sidechains», *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.

[11] S. Adamsson y M. Tahir, *From One to Many-The Impact of Individual's Beliefs in the Development of Cryptocurrency*. 2015.

[12] M. Al-Laham, H. Al-Tarawneh, y N. Abdallat, «Development of electronic money and its impact on the central bank role and monetary policy», *Issues in Informing Science and Information Technology*, vol. 6, n.º unknown, pp. 339–349, 2009.

[13] A. Cisneros Campos, «Estudio de la red Bitcoin».

[14] I. Bashir, *Mastering blockchain: distributed ledgers, decentralization and smart contracts explained*. 2017.

[15] N. Koblitz, «CM-curves with good cryptographic properties», en *Annual International Cryptology Conference*, 1991, pp. 279–287.

[16] T. Lange, «Koblitz curve cryptosystems», *Finite Fields and Their Applications*, vol. 11, n.º 2, pp. 200-229, abr. 2005.

[17] G. Zyskind, O. Nathan, y A. «Sandy» Pentland, «Decentralizing Privacy: Using Blockchain to Protect Personal Data», 2015, pp. 180-184.

[18] C. K. Lok, «The Octopus in Hong Kong: the success of a smart card-based e-payment system and beyond», *Communications of the IIMA*, vol. 4, n.º 4, p. 8, 2004.

[19] S. Goldwasser y M. Bellare, «Lecture notes on cryptography», *Summer course "Cryptography and computer security" at MIT*, vol. 1999, p. 1999, 1996.

[20] R. Bellovin, *Cryptography: Authentication, blind signature, and digital cash*. 2009.

[21] D. G. Hileman y M. Rauchs, «2017 Global Cryptocurrency Benchmarking Study», 2017.

[22] V. Malik, «The history and the future of Bitcoin».

[23] E. P. E. Deepika y E. R. Kaur, «Cryptocurrency: Trends, Perspectives and Challenges».

[24] S. Ahamad, M. Nair, y B. Varghese, «A survey on crypto currencies», en *4th International Conference on Advances in Computer Science, AETACS*, 2013, pp. 42–48.

[25] R. Merkle, «A Digital Signature based on conventional encryption function». Springer-Verlag, 1998.

[26] E. Piscini, D. Dalton, y lory Kehoe, «Blockchain & Cyber Security. Let's Discuss». Deloitte EMEA Grid Blockchain Lab.