Guía Comprensiva de Blockchain

Andres Felipe Ramirez Montaña

28 de julio de 2025

Índice general

Capítulo 1

Blockchain Definido: Bitcoin y Blockchain

1.1. ¿Qué es Blockchain y por qué es Importante?

Blockchain es una tecnología que permite la **transferencia de activos digitales de igual a igual (peer-to-peer) sin necesidad de un intermediario central**. Originalmente creada para soportar la criptomoneda Bitcoin, su potencial ha trascendido para transformar industrias como finanzas, cadena de suministro, salud y gobierno.

1.1.1. El Problema de la Centralización

Pensemos en una compra con tarjeta de crédito. Para que el dinero pase del cliente al comerciante, intervienen múltiples intermediarios: el banco del cliente, la red de la tarjeta (Visa, Mastercard), la bolsa, el banco del comerciante. Este es un sistema **centralizado**.

Sistema Centralizado: Cliente $\rightarrow$ Banco $\rightarrow$ Red $\rightarrow$ Banco $\rightarrow$ Comerciante

1.1.2. La Solución de la Descentralización

Blockchain propone un modelo **descentralizado**, donde los participantes pueden realizar transacciones directamente entre sí. Las funciones de los intermediarios se distribuyen entre los propios participantes de la red.

Sistema Descentralizado (Blockchain): Par A $\longleftrightarrow$ Red de Pares $\longleftrightarrow$ Par B

1.2. Los Tres Pilares de la Blockchain

Para que un sistema descentralizado funcione, especialmente entre pares que no se conocen ni confían entre sí, la blockchain se basa en tres conceptos fundamentales:

1. **Red Descentralizada entre Pares:** Permite transacciones directas, eliminando intermediarios.
2. **Establecimiento de Confianza Colectiva:** Como no hay una autoridad central, la confianza se logra a través de un proceso colectivo de **validación, verificación y consenso**.

3. **Libro Mayor Distribuido e Inmutable:** Las transacciones se registran en un libro contable que se copia y distribuye a todos los participantes. Una vez que un registro se añade, no se puede alterar ni eliminar.

1.2.1. Construyendo Confianza sin Autoridad Central

Imaginemos un escenario simple para entender cómo se crea la confianza:

- **La Transacción:** Yo te presto $10,000. Ambos lo anotamos en nuestros cuadernos.
- **El Problema:** ¿Qué pasa si yo cambio mi anotación a $11,000 o tú la tuya a $1,000? Hay una violación de confianza.
- **La Solución (Libro Distribuido):** Para evitarlo, le damos una copia del registro original a varios testigos (Lisa, Allison, Francis). Ahora, si alguien intenta cambiar su copia, los demás pueden ver la discrepancia y saber cuál es el registro correcto. Este es el principio de un **libro mayor distribuido**.
- **La Verificación:** Si un participante (Amy) intenta hacer una transacción fraudulenta, otro participante (Kevin) puede verificar los datos. Al descubrir que son incorrectos, **rechaza la transacción** y evita que contamine la red.

La blockchain escala este concepto a millones de transacciones entre participantes anónimos en todo el mundo, utilizando criptografía y algoritmos de consenso para automatizar la validación y la verificación.

1.3. La Contribución de Bitcoin

El protocolo Bitcoin, introducido en 2008-2009 por la figura anónima de **Satoshi Nakamoto**, fue revolucionario por dos razones principales:

1. **Creó el primer sistema de moneda digital funcional y descentralizado**, operando continuamente sin una autoridad central.
2. **Introdujo el modelo tecnológico para la blockchain**, una infraestructura autónoma que establece confianza y seguridad a través de software (validación, verificación, consenso).

Mientras que Bitcoin se centró en la transferencia de valor, sentó las bases para que innovaciones posteriores (como Ethereum) añadieran elementos de **computación** a la blockchain, abriendo un mundo de posibilidades para la Web 3.0.

1.4. Potencial y Aplicaciones de Blockchain

La capacidad de crear confianza y transferir valor sin intermediarios tiene aplicaciones que van mucho más allá de las criptomonedas:

- **Cadena de Suministro:** Trazabilidad de productos desde el origen hasta el consumidor.

- **Gestión de Identidad:** Una identidad digital única y soberana controlada por el usuario.
- **Voto Electrónico:** Sistemas de votación transparentes y a prueba de fraude.
- **Finanzas Descentralizadas (DeFi):** Préstamos, intercambios y otros servicios financieros sin bancos.
- **Registros Públicos:** Títulos de propiedad, certificados educativos y otros registros de forma inmutable.

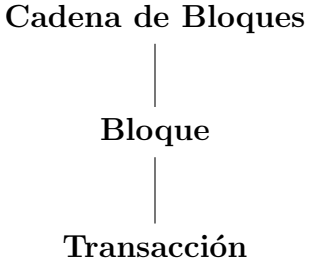
La tecnología blockchain está preparada para redefinir cómo interactuamos, hacemos negocios y construimos sistemas en un mundo cada vez más digital y globalizado.

Capítulo 2

Estructura de la Cadena de Bloques (Bitcoin)

2.1. Jerarquía de la Estructura

La estructura de la blockchain de Bitcoin se puede entender como una jerarquía de elementos, donde cada nivel agrupa al anterior.



- **Transacción:** Es el elemento más básico. Representa una transferencia de valor.
- **Bloque:** Es un contenedor que agrupa un conjunto de transacciones validadas.
- **Cadena de Bloques (Blockchain):** Es una secuencia de bloques enlazados criptográficamente.

El proceso de añadir un nuevo bloque a la cadena requiere un **consenso** entre los nodos especiales llamados **mineros**.

2.2. La Transacción y el Concepto de UTXO

Para entender cómo funciona una transacción en Bitcoin, es crucial comprender el concepto de **UTXO (Unspent Transaction Output)**, o Salida de Transacción No Gastada.

2.2.1. ¿Qué es una UTXO?

En lugar de tener "saldos.en cuentas como en un banco, Bitcoin funciona con un sistema de "monedas.o "fichas"digitales. Una UTXO es como una de esas monedas.

- El estado completo de la red Bitcoin se define por el conjunto de todas las UTXOs que existen en un momento dado.
- Una transacción **consume** una o más UTXOs (las gasta) y **crea** una o más UTXOs nuevas.

2.2.2. Estructura de una UTXO

Cada UTXO tiene:

1. Un **identificador único** de la transacción que la creó.
2. Un **índice** que indica su posición en la lista de salidas de esa transacción.
3. Un **valor** (la cantidad de satoshis/bitcoins que çontiene").
4. Un **script** (scriptPubKey) que define las condiciones para poder gastarla (normalmente, que se presente una firma digital válida).

2.2.3. Estructura de una Transacción

Una transacción se compone de:

- **Entradas (Inputs):** Referencias a las UTXOs que se van a gastar.
- **Salidas (Outputs):** Las nuevas UTXOs que se van a crear.

Regla fundamental: La suma del valor de las entradas debe ser mayor o igual a la suma del valor de las salidas. La diferencia, si la hay, es la **comisión del minero**.

Ejemplo de una Transacción Real: Una transacción con 3 entradas y 2 salidas significa que se "juntaron"tres monedas (UTXOs) antiguas para crear dos monedas (UTXOs) nuevas.

2.3. El Bloque: Contenedor de Transacciones

Un bloque agrupa las transacciones y añade metadatos importantes.

2.3.1. Estructura de un Bloque

1. **Cabecera del Bloque (Block Header):** Contiene metadatos cruciales:
   - **Hash del bloque anterior:** El enlace criptográfico que forma la cadena.
   - **Raíz de Merkle:** El hash que resume todas las transacciones del bloque.
   - **Marca de tiempo (Timestamp):** Cuándo fue minado el bloque.
   - **Nonce:** El número que resolvió el puzzle de la Prueba de Trabajo.
   - **Bits de Dificultad:** El objetivo de dificultad para el puzzle.
2. **Lista de Transacciones:** El conjunto de transacciones incluidas en el bloque.

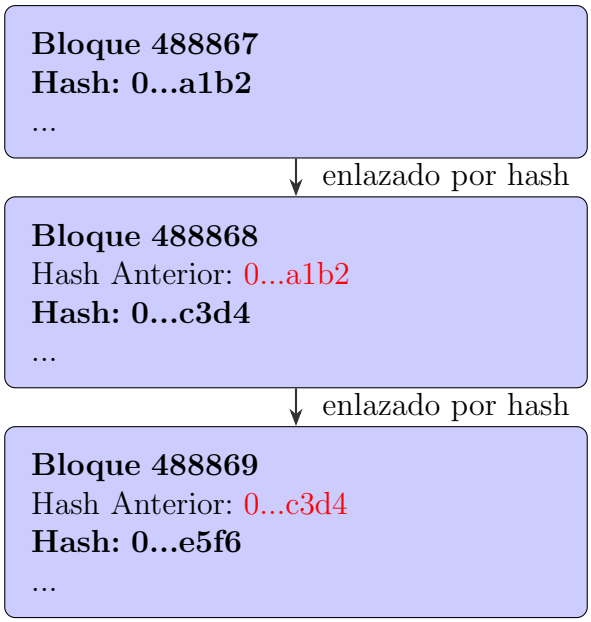
2.3.2. Ejemplo: El Bloque Génesis

El **bloque 0**, creado por Satoshi Nakamoto el 3 de enero de 2009, es el primer bloque de la cadena.

- Contenía una única transacción de 50 BTC.
- Su campo "hash del bloque anterior.[es] todo ceros, ya que no tenía predecesor.
- Estableció la recompensa inicial por bloque en 50 BTC.

2.4. La Cadena: Enlace Criptográfico

La verdadera fortaleza de la blockchain proviene de cómo se encadenan los bloques.



El **hash del bloque anterior** se incluye en la cabecera del bloque actual. Esto significa que si alguien intentara modificar una transacción en el bloque 488867, su hash (0...a1b2) cambiaría. Inmediatamente, el bloque 488868 se volvería inválido porque su campo "Hash Anteriorza no coincidiría. Esta dependencia crea una cadena inmutable y a prueba de manipulaciones.

Capítulo 3

Operaciones Básicas en la Blockchain (Enfoque Bitcoin)

3.1. Introducción a las Operaciones en la Red

Las operaciones en una red descentralizada como Bitcoin son realizadas por los **pares participantes** (nodos). Estas operaciones son esenciales para mantener la red funcional, segura y consistente. Las tareas clave incluyen:

- Validar transacciones.
- Agrupar transacciones en un bloque candidato.
- Competir para añadir el siguiente bloque a la cadena.
- Llegar a un consenso sobre el estado de la cadena.
- Encadenar los bloques para construir el registro inmutable.

3.2. Roles de los Participantes

En la red Bitcoin, los participantes pueden tener dos roles principales:

1. **Usuarios Estándar:**
    - Su función principal es iniciar la transferencia de valor (bitcoins) creando y firmando transacciones.
    - Operan con un software de "monedero"(wallet).
2. **Mineros (Nodos Mineros):**
    - Realizan todas las funciones de un usuario estándar.
    - **Además**, se encargan del trabajo pesado de la red:
        - Recogen y verifican las transacciones de la red.
        - Compiten por el derecho a crear el siguiente bloque.
        - Trabajan para alcanzar el consenso validando el bloque propuesto.
        - Difunden el nuevo bloque a toda la red.

¿Por qué alguien asumiría el trabajo extra de ser minero? Porque son **recompensados con bitcoins** por sus esfuerzos, a través de la recompensa de bloque y las tasas de transacción.

3.3. El Flujo de una Transacción hasta su Confirmación

1. **Validación de Transacciones:**
   - Cuando una transacción es creada, se difunde a la red.
   - Todos los mineros la reciben y la validan de forma independiente, comprobando más de 20 criterios.
   - **Criterios clave de validación:**
     - Que las entradas referenciadas (UTXOs) existan y no hayan sido gastadas previamente.
     - Que la suma de los valores de las entradas sea mayor o igual a la suma de los valores de las salidas.
     - Que la firma criptográfica sea válida.
   - Las transacciones inválidas se rechazan. Las válidas se añaden a un "área de espera"llamada **mempool**.
2. **Creación de un Bloque Candidato:**
   - Cada minero selecciona un conjunto de transacciones de su mempool (generalmente las que pagan mayores comisiones) para formar un nuevo bloque, llamado "bloque candidato".
3. **La Competición: Prueba de Trabajo (Proof-of-Work):**
   - Si cada minero pudiera añadir su bloque a la cadena sin más, se crearían innumerables ramas, llevando a un estado inconsistente. La red necesita un mecanismo para decidir quién tiene el derecho de añadir el siguiente bloque.
   - **La solución:** Los mineros compiten para resolver un complejo **puzzle criptográfico**. Este puzzle consiste en encontrar un número (llamado *Nonce*) que, al ser combinado con los datos del bloque y hasheado, produce un hash que empieza con un número determinado de ceros.
   - Este proceso requiere una enorme cantidad de poder computacional (CPU/ASIC), de ahí el nombre "Prueba de Trabajo".
4. **Consenso y Adición a la Cadena:**
   - El primer minero que encuentra la solución (el nonce válido) anuncia su bloque y la solución a toda la red.
   - Los demás nodos verifican rápidamente que la solución es correcta (verificar es muy fácil, encontrarlo es muy difícil).

- Si el bloque es válido, los demás nodos lo aceptan, lo añaden a su propia copia local de la blockchain y empiezan a trabajar en el siguiente bloque, construyendo sobre el que acaban de recibir.
- Las transacciones dentro de ese bloque se consideran **confirmadas**.

3.4. La Transacción Coinbase: Creación de Moneda

Dentro de cada bloque, la **primera transacción (índice cero)** es especial.

- Se llama **transacción coinbase**.
- Es creada por el minero que ganó la competición.
- **No tiene entradas (UTXOs de referencia)**, ya que crea moneda nueva "de la nada".
- Su salida contiene la **recompensa del bloque** (actualmente 6.25 BTC, antes era 12.5 BTC) más la suma de todas las **tasas de transacción** de las transacciones incluidas en el bloque.

Así es como se introducen nuevos bitcoins en el sistema y se recompensa a los mineros por asegurar la red.

Capítulo 4

Blockchain de Ethereum y Contratos Inteligentes

4.1. De Bitcoin a Ethereum: La Evolución

La blockchain de **Bitcoin** fue pionera, diseñada para un propósito específico y muy bien logrado: la transferencia de valor (moneda digital) de igual a igual (peer-to-peer).

Alrededor de 2013, los fundadores de Ethereum vieron un potencial mayor. Se preguntaron: ¿Y si la blockchain pudiera hacer más que solo transferir valor? ¿Y si pudiera **ejecutar código**? Esta idea transformó la blockchain de un simple libro contable a un **marco computacional descentralizado**.

Bitcoin	Ethereum
▪ Enfocada en transferencia de valor. ▪ Aplicación principal: Monedero digital. ▪ Libro contable de transacciones.	▪ Permite transferencia de valor **y ejecución de código**. ▪ Introduce los **Contratos Inteligentes**. ▪ Introduce la **Máquina Virtual de Ethereum (EVM)**. ▪ Permite **Aplicaciones Descentralizadas (dApps)**.

La pieza central de esta innovación es el **Contrato Inteligente**.

4.2. ¿Qué es un Contrato Inteligente?

Un contrato inteligente es, en esencia, **un programa de software que se ejecuta en la blockchain**. No está controlado por un usuario, sino que se ejecuta automáticamente cuando se cumplen ciertas condiciones predefinidas.

4.2.1. Características y Capacidades

A diferencia de una simple transferencia de dinero, un contrato inteligente puede ejecutar lógica compleja.

- **Transferencias Condicionales:** "Pagar X si se cumple Y".
- **Lógica de Negocios:** .ªceptar la oferta en una subasta solo si el postor es mayor de 18 y la oferta supera el mínimo".
- **Multi-Firma:** Requerir la aprobación de varias personas para transferir activos.
- **Eventos Temporales:** Ejecutar una acción en una fecha y hora específicas.

4.2.2. Estructura y Lenguaje de Programación

Estructuralmente, un contrato inteligente se asemeja a una **clase** en la programación orientada a objetos. Tiene:

- **Datos** (variables de estado).
- **Funciones** (métodos que pueden modificar los datos).
- **Modificadores** (que definen quién puede llamar a las funciones, ej. 'public', 'private').

El lenguaje más popular para escribir contratos inteligentes en Ethereum es **Solidity**.

Ejemplo de Contrato en Solidity

Este es un contrato muy simple que almacena y recupera un número entero.

```
// Especifica la version del compilador de Solidity
pragma solidity ^0.8.0;

// Definicion del contrato, similar a una clase
contract SimpleStorage {

    // Variable de estado para almacenar un numero
    // uint es un entero sin signo
    uint public storedData;

    // Funcion para cambiar el valor de storedData
    function set(uint x) public {
        storedData = x;
    }

    // Funcion para leer el valor de storedData
    // view indica que no modifica el estado del contrato
    function get() public view returns (uint) {
        return storedData;
    }
}
```

Listing 4.1: Contrato SimpleStorage en Solidity

4.3. ¿Dónde se Ejecuta el Código? La EVM

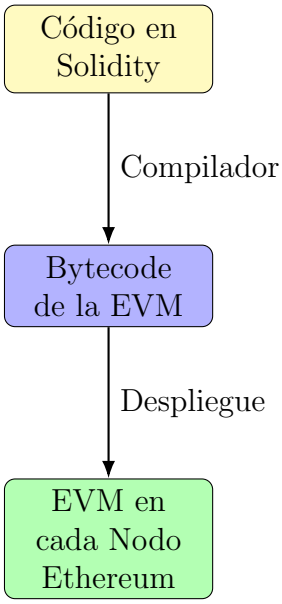
Para que el código de un contrato inteligente se pueda ejecutar en cualquier nodo de la red, sin importar el hardware o sistema operativo, Ethereum introdujo la **Ethereum Virtual Machine (EVM)**.

4.3.1. Definición de EVM

La EVM es una **máquina virtual** Turing completa que proporciona una capa de abstracción. Es un entorno de ejecución aislado que vive dentro de cada nodo completo de Ethereum.

4.3.2. Proceso de Despliegue y Ejecución

1. **Compilación:** El código de alto nivel (Solidity) se compila a un formato de bajo nivel que la EVM puede entender, llamado **bytecode**.
2. **Despliegue (Deploy):** Este bytecode se envía a la blockchain a través de una transacción especial. Al ser minado en un bloque, el contrato "vive.[en] la EVM de cada nodo.
3. **Ejecución:** Para interactuar con el contrato (ej. llamar a la función 'set'), un usuario envía una transacción a la dirección del contrato, la cual activa su ejecución en la EVM.



4.4. Resumen

Los contratos inteligentes son la gran innovación de Ethereum. Añaden una capa de **lógica y computación** sobre la infraestructura de confianza que proporciona la blockchain, permitiendo la creación de aplicaciones descentralizadas complejas que van mucho más allá de la simple transferencia de valor de Bitcoin.

Capítulo 5

Estructura de Ethereum

5.1. El Cambio de Paradigma: De UTXO a Cuentas

Mientras que Bitcoin define su estado a través de un conjunto de **Salidas de Transacción No Gastadas (UTXO)**, Ethereum introduce formalmente el concepto de **Cuenta** como parte fundamental de su protocolo.

Modelo de Bitcoin (UTXO)	Modelo de Ethereum (Cuentas)
▪ Basado en transacciones. ▪ El estado es la colección de "monedas"no gastadas (UTXOs). ▪ Una transacción consume UTXOs viejas y crea nuevas. ▪ No hay un concepto de "saldo"persistente.	▪ Basado en cuentas y transiciones de estado. ▪ El estado es la lista de todas las cuentas y sus saldos/datos. ▪ Una transacción actualiza directamente el saldo de las cuentas. ▪ Permite lógica compleja (contratos inteligentes).

En Ethereum, una transacción es un evento que puede transferir valor y/o mensajes, provocando una **transición de estado** en las cuentas.

5.2. Tipos de Cuentas en Ethereum

Existen dos tipos de cuentas, cada una con un propósito diferente:

1. **Cuentas de Propiedad Externa (Externally Owned Accounts - EOA):**
   - **Controladas por:** Un par de claves pública-privada.
   - **Función:** Son las cuentas que pertenecen a los usuarios. Para participar en la red (enviar Ether, interactuar con contratos), necesitas una EOA.
   - **Capacidad:** Pueden iniciar transacciones.
2. **Cuentas de Contrato (Contract Accounts - CA):**

- **Controladas por:** El código del contrato inteligente que representan.
- **Función:** Representan a los contratos inteligentes desplegados en la blockchain.
- **Capacidad:** No pueden iniciar transacciones por sí mismas. Solo se **activan** cuando reciben una transacción de una EOA o de otro contrato.

Ambos tipos de cuentas pueden tener un **saldo en Ether** y pueden enviar y recibir Ether.

5.3. La Transacción en Ethereum

Una transacción en Ethereum es más rica en información que una de Bitcoin, ya que debe soportar la ejecución de código.

5.3.1. Campos de una Transacción Ethereum

- **Recipient (Destinatario):** La dirección de la cuenta receptora (puede ser una EOA o una CA).
- **Signature (Firma):** La firma digital del remitente, que autoriza la transacción.
- **Value (Valor):** La cantidad de Wei (la unidad más pequeña de Ether) a transferir.
- **Data (Datos/Payload):** Un campo opcional que contiene datos. Si la transacción va a un contrato, aquí se especifica la función a llamar y sus parámetros.
- **Gas Limit (Límite de Gas):** Máximo de unidades de gas que la transacción puede consumir.
- **Gas Price (Precio del Gas):** El precio que el remitente paga por unidad de gas.
- **Nonce:** Un contador que previene la repetición de transacciones desde la misma cuenta.

5.3.2. Ejemplo de Transacción Real

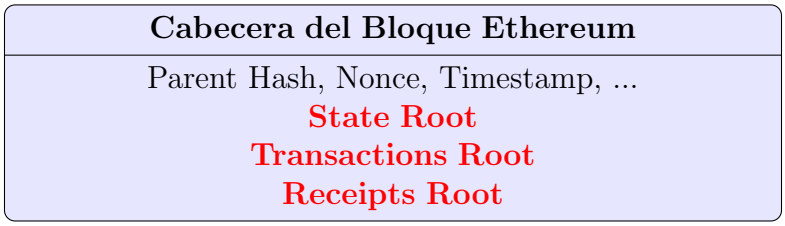
Una transacción real en un explorador de bloques mostrará todos estos campos, junto con información adicional como el hash de la transacción, el número del bloque en que fue incluida (altura), la marca de tiempo, y el estado del recibo de la transacción (éxito o fallo).

5.4. La Estructura del Bloque Ethereum

El bloque de Ethereum es más complejo que el de Bitcoin porque debe dar cuenta del estado de los contratos inteligentes.

5.4.1. Componentes de un Bloque Ethereum

1. **Cabecera del Bloque (Header):**
   - **Parent Hash:** Hash del bloque anterior.
   - **Nonce:** Solución a la Prueba de Trabajo.
   - **Timestamp, Difficulty, Gas Limit, Gas Used, etc.**
   - **Tres Raíces de Merkle cruciales:**
     - **Transactions Root:** Raíz del árbol de Merkle de todas las transacciones del bloque. Garantiza la integridad de las transacciones.
     - **State Root:** Raíz de un árbol Merkle (Patricia Trie) que representa el **estado completo** de todas las cuentas (saldos, código de contratos, almacenamiento de contratos) *después* de ejecutar las transacciones del bloque. Garantiza la integridad del estado.
     - **Receipts Root:** Raíz del árbol de Merkle de los recibos de cada transacción (que indican el resultado, como el gas usado y los logs generados). Garantiza la integridad de los resultados de la ejecución.
2. **Cuerpo del Bloque:**
   - La lista de transacciones incluidas.
   - (En versiones antiguas) La lista de cabeceras de bloques Ommer (tíos).



5.5. Resumen

La estructura de Ethereum, con su modelo basado en cuentas y las tres raíces en la cabecera del bloque, está diseñada para soportar no solo transferencias de valor, sino un sistema computacional global y descentralizado donde el estado del .ºrdenador mundial"se actualiza con cada bloque.

Capítulo 6

Operaciones en Ethereum

6.1. El Nodo Ethereum: Un Sistema Completo

Un nodo completo en la red Ethereum no es solo un simple participante, sino un sistema computacional que aloja todo el software necesario para el funcionamiento de la red. Sus responsabilidades incluyen:

- Iniciación de transacciones.
- Validación de transacciones y bloques.
- Minería y creación de bloques.
- Ejecución de contratos inteligentes a través de la **Máquina Virtual de Ethereum (EVM)**.

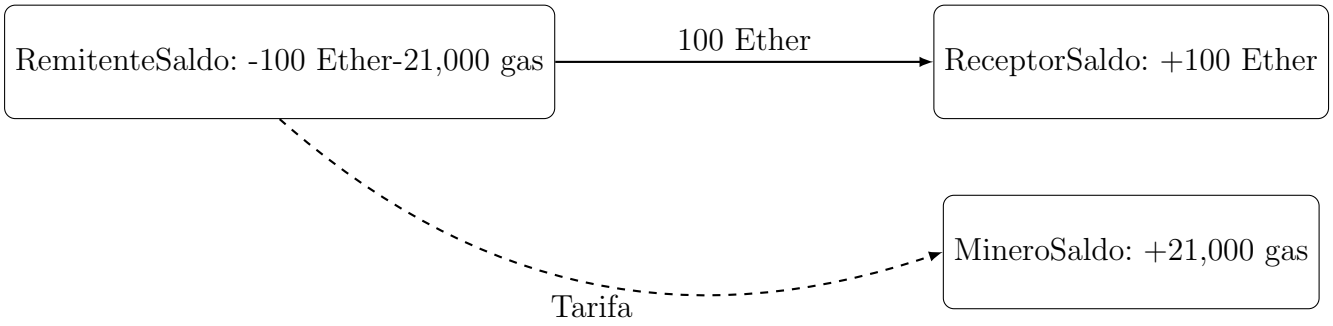
6.2. Tipos de Transacciones en Ethereum

Existen principalmente dos tipos de transacciones, dependiendo de la cuenta de destino.

6.2.1. Transferencia Simple de Ether

Esta es la operación más básica, análoga a una transacción en Bitcoin.

- **Objetivo:** Transferir Ether de una cuenta a otra.
- **Proceso:** Se especifica la cantidad de Ether a transferir, la dirección del destinatario y una tarifa (*gas*).
- **Ejemplo:** Una transacción transfiere 100 Ether. Además del monto, se pagan 21,000 puntos de gas como tarifa al minero que incluye la transacción en un bloque.



6.2.2. Invocación de un Contrato Inteligente

Esta operación es la que dota a Ethereum de su poder computacional.

- **Objetivo:** Ejecutar una función de un contrato inteligente que ya ha sido desplegado en la blockchain.
- **Proceso:**
  1. Se crea una transacción donde la **dirección de destino es la dirección del contrato inteligente**.
  2. La entrada necesaria para la ejecución (los parámetros de la función) se incluye en el campo **payload (o data)** de la transacción.
  3. Cuando los mineros procesan esta transacción, el código del contrato se activa y se ejecuta en su EVM.
  4. La ejecución puede **actualizar el estado** del contrato (modificar sus variables internas).
  5. Los resultados de la ejecución se registran en una estructura de datos llamada **recibo (receipt)**. La blockchain almacena un hash de todos los recibos.

6.3. El Ciclo de Vida de una Transacción

Desde su creación hasta su confirmación, una transacción pasa por varias etapas:

1. **Generación:** Un usuario crea y firma una transacción.
2. **Validación Inicial:** La transacción se difunde a la red. Los nodos que la reciben realizan una validación preliminar:
   - ¿La firma es correcta?
   - ¿La marca de tiempo y el nonce son válidos?
   - ¿El remitente tiene fondos suficientes para cubrir el monto y las **tasas de gas** especificadas?
3. **Agrupación (Mempool):** Las transacciones validadas se añaden a un "área de espera.º *mempool* en cada nodo minero.
4. **Ejecución y Creación de Bloque:**

- Un minero selecciona un conjunto de transacciones de su mempool para incluirlas en un nuevo bloque.
- **Importante:** Si hay transacciones que invocan contratos, el código de esos contratos es ejecutado por **todos los mineros** que intentan crear el bloque.
- El minero compite para resolver el rompecabezas de **Prueba de Trabajo (Proof-of-Work)**. El PoW de Ethereum está diseñado para ser resistente a hardware especializado (ASIC), basándose en el uso intensivo de memoria.

5. **Difusión y Consenso:** El minero que resuelve el puzzle difunde su nuevo bloque a la red. Los demás nodos lo verifican y, si es válido, lo añaden a su copia de la cadena.

6.4. La Pregunta Clave: ¿Quién Paga por Todo Esto?

Todas estas operaciones (validación, ejecución de código, consenso, almacenamiento) consumen recursos computacionales. La red necesita una forma de compensar a los nodos (especialmente a los mineros) por su trabajo. Esto se resuelve a través del **Modelo de Incentivos**, que se basa en las tarifas de transacción (gas).

Capítulo 7

El Modelo de Incentivos en Ethereum

7.1. Introducción: El Combustible de Ethereum

La minería es el proceso que asegura la red: valida cómputos, forma bloques y los añade a la cadena. Para que los participantes (mineros) dediquen sus recursos computacionales a esta tarea, debe existir un **incentivo económico**. En Ethereum, este sistema de incentivos se gestiona a través de un concepto fundamental: el **Gas**.

7.2. El Concepto de Gas

El gas es la unidad que mide la cantidad de trabajo computacional requerido para ejecutar una operación en Ethereum. Cada acción, desde una simple transferencia hasta la ejecución de un complejo contrato inteligente, tiene un costo fijo medido en puntos de gas.

7.3. Gas en Ethereum (EVM)

Cada operación que puede realizarse mediante una transacción o contrato en la plataforma Ethereum cuesta una cierta cantidad de gas ; las operaciones que requieren más recursos computacionales cuestan más gas que las que requieren menos recursos computacionales.

La importancia del gas radica en que garantiza el pago de la tarifa adecuada por las transacciones enviadas a la red. Al exigir esta cantidad, se garantiza que la red no se sature con gran cantidad de trabajo intensivo que no aporta valor a nadie **¿Esto por que sucede?** Ethereum permite la ejecución de código de cualquier complejidad dentro del ecosistema utilizando conceptos como **EVM**. Es importante medir directamente el trabajo realizado en lugar de simplemente establecer una tarifa en función de la longitud de una transacción o un contrato.

Esto nos puede dar a la siguiente duda: **¿Entonces, si el gas es básicamente un comisión por la transacción computacionalmente, como se paga o que valor monetario se toma dentro de esta?** El gas no se mide por un token en especifico, es decir, no se puede poseer 1000 unidades de gas.

El gas solo esta presente dentro de la maquina virtual de Ethereum. A la hora de pagar el gas, la comisión por transacción se cobra en una cantidad determinada de ether, el token integrado en la red de Ethereum y el token con el que se recompensa a los mineros por producir los bloques.

Se sabe que los token dentro del ecosistema de las tecnologías descentralizadas como Bitcóin y Ethereum, el valor que adquiere no es fijo y cambia con respecto valores del mercado global, entonces **¿Por que las operaciones tienen un costo medido directamente en ether?** La respuesta se debe a que es muy útil y eficiente separar el costo computacional con el costo de los token de la blockchain, es decir, el costo computacional no sube ni baja solo porque el precio del ether cambie.

En Ethereum, llegamos a una terminología un poco confusa: las operaciones en la EVM (Ethereum Virtual Machine) tienen un **costo de gas**, pero el **gas en sí también tiene un precio medido en ether**.

¿Por qué sucede esto? Cuando se realizan transacciones o se ejecuta código dentro de contratos inteligentes, se consume cierta cantidad de *gas*, que es simplemente una **unidad de medida del trabajo computacional** realizado por la red. Sin embargo, el gas **no es una moneda**, por lo tanto, debe **convertirse a una cantidad de ether (ETH)** para poder ser pagado a los validadores de la red.

Cada transacción en Ethereum especifica el **precio del gas** (también llamado *gas price*) que está dispuesto a pagar el usuario, expresado normalmente en **Gwei**, que es una fracción de ether:

$$1 \text{ Gwei} = 10^{-9} \text{ ETH}$$

Mientras mayor sea el precio del gas ofrecido, mayor será la prioridad que los mineros (o validadores) le den a la transacción, ya que obtendrán una recompensa más alta. Esta dinámica crea un **mercado competitivo** de precios por transacción, en donde los usuarios compiten para que sus operaciones sean procesadas más rápidamente.

Entonces, el costo total que paga una transacción está dado por la siguiente fórmula:

$$\text{Tarifa total (en ETH)} = \text{Cantidad de gas usada} \times \text{Precio por unidad de gas (en ETH)}$$

Ejemplo:

- Gas utilizado: 21,000 unidades
- Precio del gas: 50 Gwei = 0.00000005 ETH

$$\text{Tarifa total} = 21,000 \times 0{,}00000005 \text{ ETH} = 0{,}00105 \text{ ETH}$$

Esta cantidad es deducida automáticamente del saldo del remitente y entregada al validador que incluye la transacción en un bloque.

Aunque sea un poco confuso y es que **existe a diferencia entre que una transacción se quede sin gas y que no tenga una comisión lo suficientemente alta**. Acá van algunas cosas importantes que se debe considerar a la hora de hacer algunas operaciones dentro del ecosistema Ethereum:

1. Si el precio del gas que establezco en mi transacción es demasiado bajo, nadie se molestará en ejecutarla. Simplemente, los mineros no la incluirán en la blockchain.

2. Si proporciono un precio de gas aceptable , y mi transacción requiere tanto trabajo computacional que los costos combinados de gas superan la cantidad que asigné como comisión, ese gas se considera "gastadoz no lo recupero. El minero dejará de procesar la transacción, revertirá los cambios realizados, pero la incluirá en la blockchain como una "transacción fallidaz cobrará las comisiones correspondientes.

3. Ofrecer una tarifa demasiado alta también es diferente a ofrecer demasiado ether. Si estableces un precio de gas muy alto, terminarás pagando mucho ether por solo unas pocas operaciones , al igual que establecer una tarifa de transacción altísima en bitcoin. Definitivamente tendrás prioridad, pero perderás tu dinero.

4. Si se establece un precio de gas normal y simplemente adquiriste más ether del necesario para pagar el gas consumido por tu transacción, se te reembolsará el exceso . Los mineros solo te cobran por el trabajo que realmente realizan. **Puedes pensar en el precio del gas como el salario por hora del minero, y el costo del gas como su hoja de tiempo de trabajo realizado.**

El gas es el mecanismo clave que hace que los cálculos complejos en Ethereum sean "seguros"para el funcionamiento de la red , ya que cualquier programa que se descontrole solo durará mientras el dinero proporcionado por quienes lo solicitaron se ejecute. Cuando el dinero se detiene, los mineros dejan de trabajar en él. Y los errores que cometas en tu programa solo afectarán a quienes pagan por usarlo ; el resto de la red no puede sufrir problemas de rendimiento debido a tu error.

7.3.1. ¿Por qué usar Gas y no directamente Ether?

Aunque esto se explico anteriormente, toma esto como un resumen mas para comprender.

El valor de las criptomonedas como el Ether (ETH) es volátil y fluctúa con el mercado. Si las tarifas se fijaran en ETH, el costo real de una operación cambiaría constantemente. El gas resuelve este problema:

- **Costo de la Operación (en Gas):** Es fijo y estable. (Ej: una transferencia siempre cuesta 21,000 gas).
- **Precio del Gas (en Ether):** Es variable y lo decide el usuario. Se expresa en una unidad pequeña de Ether llamada *Gwei*. El usuario ofrece un precio por cada unidad de gas, creando un mercado de tarifas.

Costo Total de la Transacción (en Ether) = Gas Usado × Precio del Gas

7.3.2. El Gas en una Transacción

Cuando un usuario envía una transacción, especifica dos valores relacionados con el gas:

1. **Límite de Gas (Gas Limit):** La cantidad **máxima** de gas que el usuario está dispuesto a gastar en esa transacción. Esto previene que un contrato con errores entre en un bucle infinito y consuma todos los fondos del usuario.

2. **Precio del Gas (Gas Price):** El precio que el usuario pagará por cada unidad de gas consumida.

Si la transacción no tiene suficiente gas para completarse, es rechazada (falla), pero **el gas consumido hasta el punto del fallo se cobra igualmente**. Esto es como enviar una carta con franqueo insuficiente; el trabajo realizado no se devuelve. Si sobra gas al final de una ejecución exitosa, se devuelve al remitente.

7.4. El Gas a Nivel de Bloque

El concepto de gas también se aplica a los bloques:

- **Límite de Gas del Bloque:** Cantidad máxima de gas que todas las transacciones de un bloque pueden consumir en total. Esto limita el tamaño computacional de un bloque y asegura que los bloques puedan procesarse en un tiempo razonable.

- **Gas Gastado en el Bloque:** La cantidad real de gas que consumieron todas las transacciones incluidas en el bloque.

Ejemplo: Si un bloque tiene un límite de 1,500,000 gas y una transacción simple cuesta 21,000 gas, en ese bloque cabrían teóricamente unas 71 transacciones simples (1,500,000 / 21,000). Si se incluyen contratos inteligentes (que gastan más gas), el número de transacciones por bloque será menor.

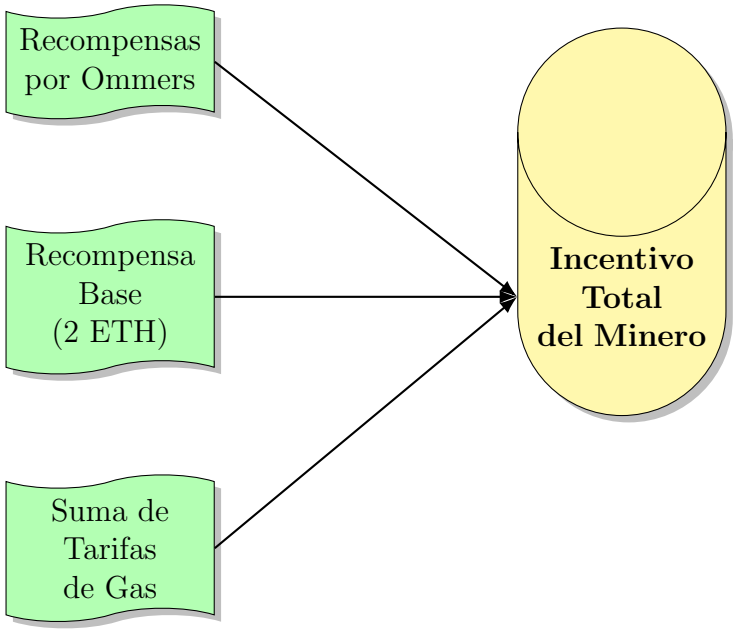
7.5. El Modelo de Incentivos para Mineros

Las recompensas que recibe un minero por crear un bloque provienen de tres fuentes:

1. **Recompensa Base por Bloque (Block Reward):** Una cantidad fija de Ether creada "de la nadaz otorgada al minero por haber minado el bloque con éxito. Históricamente ha sido de 5, luego 3, y actualmente 2 ETH (antes del cambio a Proof of Stake).

2. **Tasas de Transacción (Gas Fees):** El minero se queda con la suma de todas las tarifas de gas de las transacciones que incluyó en su bloque. Esto incentiva a los mineros a incluir transacciones con precios de gas más altos.

3. **Recompensas por Ommers:** A veces, más de un minero resuelve el puzzle casi al mismo tiempo, creando bloques válidos competidores. El bloque que se propaga más rápido por la red se convierte en el canónico. Los bloques de los otros mineros, que se quedan "huérfanos", se llaman **Ommers** (o bloques tío). Para no desperdiciar ese trabajo y mejorar la seguridad de la red, la cadena principal puede referenciar a estos Ommers. El minero del bloque Ommer recibe una recompensa de consolación, y el minero del bloque principal que lo incluye también recibe una pequeña bonificación.


Recompensas
por Ommers
Recompensa
Base
(2 ETH)
Suma de
Tarifas
de Gas
Incentivo
Total
del Minero

Capítulo 8

Algoritmos y Técnicas: Criptografía de Clave Pública

8.1. Introducción a la Seguridad en Blockchain

Para asegurar la integridad y la eficiencia de una cadena de bloques (blockchain), se utilizan predominantemente dos técnicas criptográficas:

- **Hashing (o Funciones de Resumen):** Garantiza la integridad de los datos.
- **Criptografía de Clave Asimétrica (o de Clave Pública):** Permite la autenticación, autorización y confidencialidad en una red donde los participantes no se conocen ni confían entre sí.

En un entorno descentralizado, es imposible verificar la identidad con métodos tradicionales (como un carnet de conducir). La criptografía de clave pública resuelve problemas fundamentales como: ¿cómo identificar a los participantes?, ¿cómo autorizar transacciones? y ¿cómo detectar falsificaciones?

8.2. Criptografía de Clave Simétrica: El Predecesor

Antes de entender la clave pública, es útil conocer su contraparte, la clave simétrica.

8.2.1. Definición

Se utiliza **la misma clave** tanto para cifrar (encriptar) como para descifrar la información.

- **Ejemplo simple (Cifrado César):** Se desplaza cada letra de un mensaje un número fijo de posiciones en el alfabeto. Ese número es la çlave".
- **Mensaje original:** "Nos vemos en el cine"
- **Clave:** 3
- **Proceso:** Cada letra se desplaza 3 posiciones hacia adelante ('N' se convierte en 'Q', 'o' en 'r', etc.).
- **Descifrado:** El receptor usa la misma clave (3) para desplazar las letras 3 posiciones hacia atrás.

8.2.2. Problemas de la Clave Simétrica

Este método, aunque simple de entender, presenta dos grandes debilidades, especialmente en una red descentralizada:

1. **Vulnerabilidad:** Es relativamente fácil deducir la clave secreta a partir del análisis de los datos cifrados.
2. **Distribución de la Clave:** ¿Cómo se entrega la clave secreta al receptor de forma segura? Si un canal es seguro para enviar la clave, ¿por qué no usar ese mismo canal para enviar el mensaje original?

8.3. Criptografía de Clave Pública: La Solución de Blockchain

Para resolver los problemas anteriores, la criptografía asimétrica no usa una, sino **dos claves vinculadas matemáticamente**.

8.3.1. El Par de Claves

Cada participante en la red genera un par de claves único:

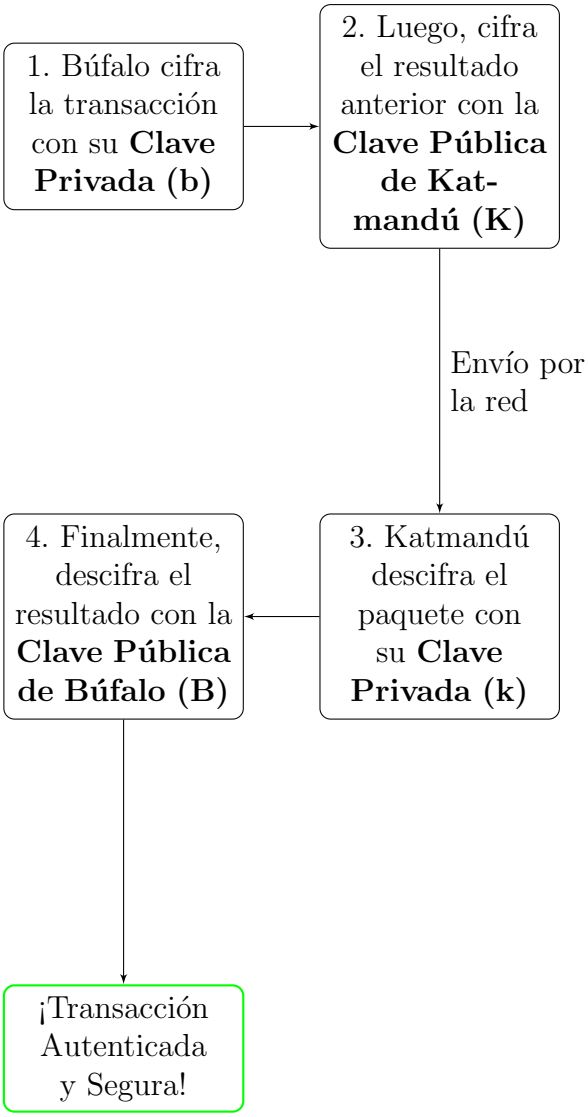
- **Clave Privada (b, k):** Es secreta y solo el propietario la conoce. Se guarda de forma segura, a menudo protegida por una contraseña o frase de paso.
- **Clave Pública (B, K):** Se comparte libremente con cualquiera en la red.

La magia de este par de claves reside en su funcionamiento: lo que se cifra con una de las claves, **solo puede ser descifrado con la otra clave del par**.

8.3.2. Proceso de Autenticación y Confidencialidad

Veamos un ejemplo práctico para asegurar que el remitente es quien dice ser (autenticación) y que solo el destinatario puede leer el mensaje (confidencialidad).

Escenario: Un participante de Búfalo (B) quiere enviar datos de una transacción a un participante en Katmandú (K).



Garantías de este proceso:

- Solo **Katmandú** pudo descifrar el mensaje inicial, porque solo él posee la clave privada 'k'.
- Solo **Búfalo** pudo haber enviado el mensaje, porque la clave pública 'B' solo puede descifrar algo firmado por la clave privada 'b'.

8.4. Algoritmos: RSA vs. ECC

Existen diferentes algoritmos para implementar la criptografía de clave pública.

8.4.1. RSA (Rivest-Shamir-Adleman)

Es un algoritmo muy popular y utilizado en muchas aplicaciones, como la autenticación sin contraseña para acceder a servidores en la nube (ej. Amazon Web Services). Sin embargo, para las blockchains, se necesita algo más eficiente.

8.4.2. Criptografía de Curva Elíptica (ECC)

ECC es la familia de algoritmos utilizada tanto en Bitcoin como en Ethereum. La razón principal es su **eficiencia y fortaleza**.

- **Eficiencia:** Las operaciones criptográficas (firmar, verificar) son constantes en una blockchain. ECC requiere menos poder computacional.
- **Fortaleza:** ECC ofrece la misma seguridad que RSA pero con claves mucho más cortas.

Comparación de Fortaleza:

Un par de claves **ECC de 256 bits** es tan seguro como un par de claves **RSA de 3072 bits**.

Esta diferencia es crucial para la velocidad y escalabilidad de la red blockchain.

Capítulo 9

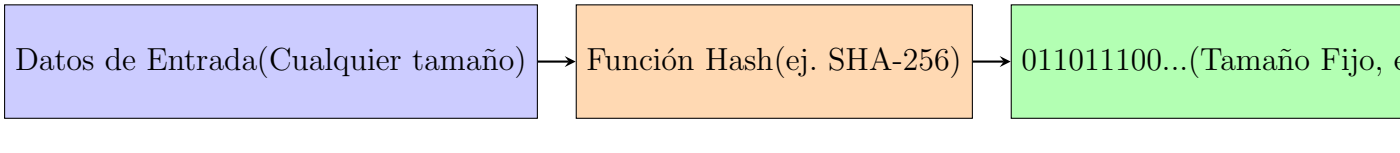
Hashing

9.1. ¿Por qué aprender sobre Hashing?

El hashing es un pilar fundamental de la tecnología blockchain. Conceptos como "tasa de hash.º "poder de hash"son omnipresentes. Entender el hashing es tan crucial como saber proteger tu tarjeta de crédito, ya que es la base para la seguridad de los activos, la integridad de las transacciones y la confidencialidad de los datos en la blockchain.

9.2. ¿Qué es una Función Hash?

Una función hash es un algoritmo matemático que toma una entrada de **longitud arbitraria** (puede ser una letra, un documento, un bloque entero de datos) y la transforma en una salida de **longitud fija**, llamada **valor hash** o simplemente **hash**.



Una característica esencial es que un cambio mínimo en la entrada (ej. cambiar una coma por un punto) produce un valor hash de salida **completamente diferente**.

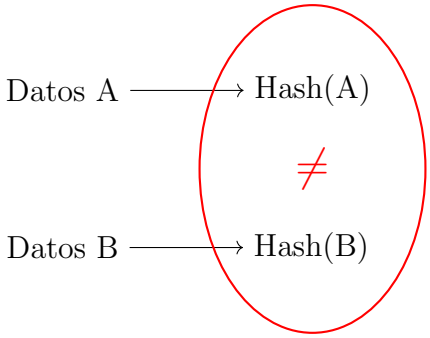
9.2.1. Requisitos Fundamentales de una Función Hash Criptográfica

Para que una función hash sea segura y útil en blockchain, debe cumplir dos requisitos:

1. **Ser Unidireccional (One-Way):** Debe ser computacionalmente imposible derivar los datos de entrada a partir del valor hash de salida.

   Analogía: ¿Puedes reconstruir las patatas originales a partir de un puré de patatas? No.

2. **Resistente a Colisiones (Collision-Free):** Debe ser extremadamente improbable que dos conjuntos de datos de entrada diferentes produzcan el mismo valor hash de salida.



9.2.2. El Poder de un Hash de 256 bits

Los algoritmos comunes en blockchain, como **SHA-256** y **Keccak-256**, producen un hash de 256 bits. Esto crea un espacio de valores posibles inmenso: $2^{256} \approx 1{,}15 \times 10^{77}$ (un 1 seguido de 77 ceros).

Es más probable que un meteorito golpee tu casa a que se genere una colisión (dos hashes idénticos) con un algoritmo de 256 bits.

9.3. Técnicas de Hashing: Plana vs. Árbol

Dependiendo de la necesidad, los datos se pueden hashear de diferentes maneras. Usemos el operador 'ADD' como una función hash simple para ilustrar.

Datos de entrada: [10, 4, 6, 21, 19]

9.3.1. Hashing Simple (o Lineal)

Se ordenan todos los elementos y se hashean juntos.

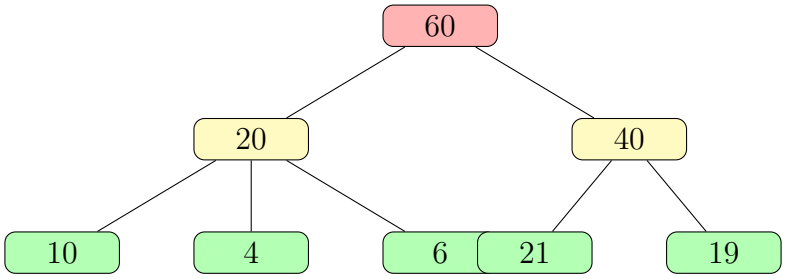
$$\text{Hash} = \text{ADD}(10, 4, 6, 21, 19) = 60$$

¿Cuándo se usa? Cuando el número de elementos es fijo y pequeño, y solo nos interesa la integridad del conjunto completo. **Ejemplo: La cabecera (header) de un bloque.**

9.3.2. Árbol de Merkle

Los datos se organizan en un árbol binario.

- Hojas: Los datos individuales.
- Nodos padres: El hash de sus hijos.



¿Cuándo se usa? Cuando el número de elementos es variable y grande, y necesitamos verificar elementos individuales de forma eficiente. **Ejemplos: transacciones, estados y recibos en un bloque.** La ventaja es la eficiencia (O(log N) vs. O(N)).

9.4. Usos del Hashing en Ethereum

En resumen, las funciones hash son omnipresentes y se utilizan para:

- Generar direcciones de cuentas.
- Crear firmas digitales.
- Calcular el hash de transacciones (y la Raíz de Merkle de transacciones).
- Calcular el hash de estados (y la Raíz de Merkle de estados).
- Calcular el hash de recibos.
- Calcular el hash de la cabecera del bloque.

Capítulo 10

Integridad de las Transacciones

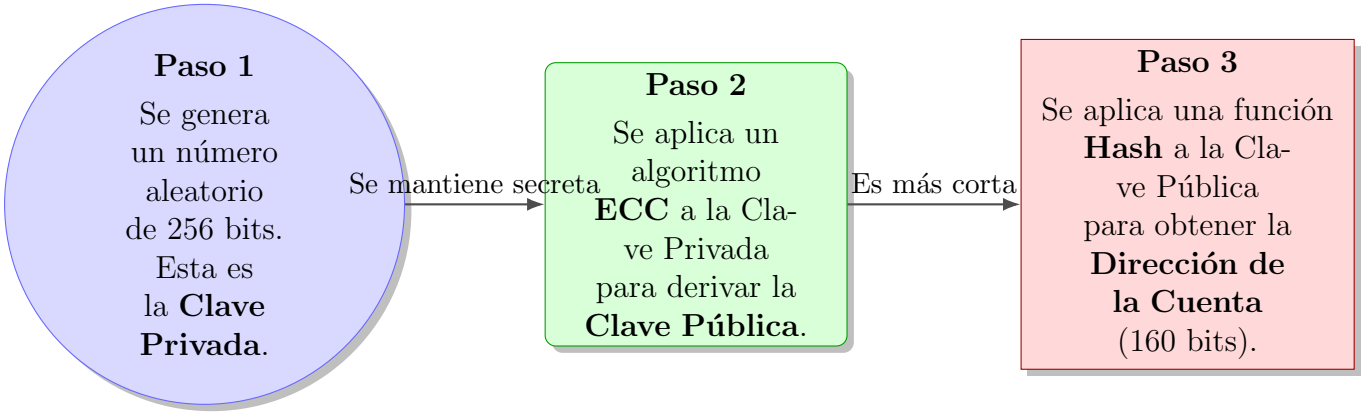
10.1. Pilares de la Integridad de una Transacción

Para garantizar que una transacción en la blockchain sea válida y segura, se deben cumplir tres condiciones fundamentales. Estas se logran combinando el **hashing** y la **criptografía de clave pública**.

1. **Dirección de Cuenta Única:** Un método estándar para identificar a cada participante de forma inequívoca.
2. **Autorización del Remitente:** El remitente debe autorizar la transacción de una manera que no pueda negar haberlo hecho (no repudio), lo cual se logra con una **firma digital**.
3. **Inmutabilidad del Contenido:** Verificación de que los datos de la transacción no han sido alterados después de su creación.

10.2. Generación de la Dirección de Cuenta

La dirección de una cuenta (similar a un número de cuenta bancaria) se deriva del par de claves pública y privada del usuario. Este proceso asegura que solo el propietario de la clave privada pueda controlar los fondos de esa dirección.



Nota Importante: La clave privada se debe mantener absolutamente segura. La clave pública se comparte, y la dirección de la cuenta es lo que se le da a otros para recibir fondos.

10.3. Firma Digital: Autorización y Verificación

La firma digital es el mecanismo que prueba la autenticidad e integridad de una transacción.

10.3.1. Definición de Firma Digital

Una firma digital no es más que el **hash de los datos, cifrado con la clave privada del remitente**.

$$\text{Firma Digital} = \text{Cifrar}(\textit{Hash(Datos)}, \textit{Clave Privada})$$

10.3.2. Proceso de Firma y Verificación de una Transacción

Proceso de Firma (Remitente)

1. **Calcular el Hash:** Se aplica una función hash (ej. SHA-256) a todos los campos de la transacción (monto, destinatario, etc.) para obtener un resumen único.
2. **Cifrar el Hash:** Se cifra el hash obtenido en el paso anterior utilizando la **clave privada** del remitente. El resultado es la firma digital.
3. **Adjuntar Firma:** Se agrega esta firma digital a la transacción. Ahora la transacción está lista para ser enviada a la red.

Proceso de Verificación (Mineros / Nodos)

1. **Descifrar la Firma:** El verificador toma la firma digital de la transacción y la descifra usando la **clave pública** del remitente. El resultado es el hash original (Hash A).
2. **Recalcular el Hash:** El verificador toma los datos de la transacción (sin la firma) y calcula su hash usando la misma función hash que usó el remitente (Hash B).
3. **Comparar:** Se comparan ambos hashes.
   - **Si Hash A == Hash B:** La transacción es **válida**. El remitente es auténtico y los datos no han sido modificados. Se acepta.
   - **Si Hash A != Hash B:** La transacción es **inválida**. Ha sido manipulada o la firma es falsa. Se rechaza.

Nota sobre la Verificación Completa: Además de la firma, los nodos de la red también verifican otros detalles importantes para una validación completa, como:

- La marca de tiempo de la transacción.
- El *nonce* (un número para prevenir la repetición de transacciones).
- Que el remitente tenga saldo suficiente en su cuenta.

- Que las tasas de transacción (gas) sean adecuadas.

Capítulo 11

Proteger la Cadena de Bloques

11.1. La Integridad del Bloque

Un bloque en una blockchain (como Ethereum) es un contenedor de información. Su integridad es crucial, ya que la cadena completa se basa en la validez de cada uno de sus eslabones (bloques). Proteger un bloque significa asegurar que:

- El contenido de su **cabecera (header)** no sea manipulado.
- Las **transacciones** que contiene no sean alteradas.
- Las **transiciones de estado** (cambios en las cuentas o contratos) se calculen y verifiquen eficientemente.

El objetivo es mantener la **inmutabilidad**, la característica principal de una blockchain.

11.2. Componentes Principales de un Bloque Ethereum

Un bloque Ethereum está compuesto por:

1. **La Cabecera (Header):** Contiene metadatos sobre el bloque.
2. **Las Transacciones:** La lista de transacciones incluidas en el bloque.
3. **Componentes Adicionales:** Como los hashes de las raíces de estado y de transacción.

11.2.1. El Hash del Bloque

El **hash del bloque** es una huella digital única que identifica todo el bloque. Se calcula aplicando un algoritmo de hash (en Ethereum, una variante de SHA-3 llamada **Keccak-256**) a **todos los elementos de la cabecera del bloque**.

$$\text{Hash del Bloque} = \text{Keccak-256}(\text{Todos los elementos de la cabecera})$$

La cabecera incluye, a su vez, los hashes raíz de las transacciones y del estado, lo que significa que cualquier cambio, por mínimo que sea, en cualquier parte del bloque, cambiará drásticamente el hash final del bloque.

11.3. Manejo Eficiente de Datos con Árboles de Merkle

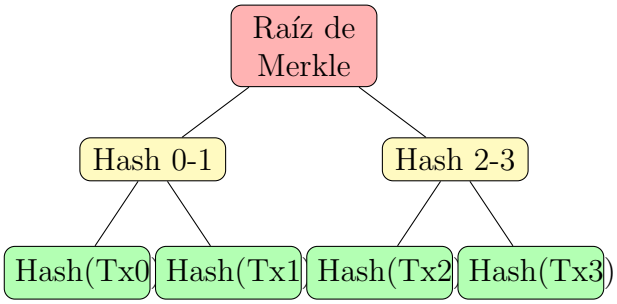
Un bloque puede contener miles de transacciones (en Bitcoin, 2000; en Ethereum, 100). Validar cada una de ellas linealmente sería muy ineficiente. La solución es organizar los hashes de las transacciones en una estructura de datos llamada **Árbol de Merkle**.

11.3.1. ¿Qué es un Árbol de Merkle?

Es una estructura de árbol donde:

- Las **hojas** son los hashes de las transacciones individuales.
- Los **nodos intermedios** son el hash de la concatenación de sus nodos hijos.
- El **nodo superior o raíz** se conoce como la **Raíz de Merkle** o **Hash Raíz**.

Esta Raíz de Merkle (por ejemplo, la raíz de transacción) es lo que se incluye en la cabecera del bloque.



11.3.2. Ventajas del Árbol de Merkle

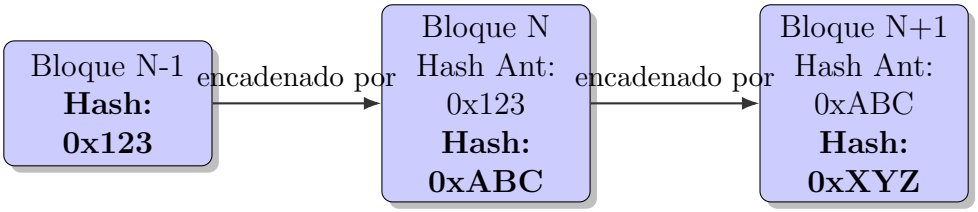
1. **Validación Eficiente:** Para verificar si una transacción (ej. Tx2) está en el bloque, no necesitas todas las transacciones. Solo necesitas Tx2, Hash(Tx3) y Hash(0-1). Con estos datos, puedes recalcular la Raíz de Merkle. Si coincide con la del bloque, la transacción es válida. Esto es mucho más rápido (eficiencia logarítmica, O(log N)).
2. **Recálculo Eficiente de Estados:** En Ethereum, los contratos inteligentes cambian el .estado"de la blockchain. Estos estados también se organizan en un Árbol de Merkle. Si solo un estado cambia, solo hay que recalcular el hash de la ruta desde esa hoja hasta la raíz, en lugar de todo el árbol.

11.4. El Rol del Hash del Bloque en la Seguridad de la Cadena

El hash del bloque cumple dos propósitos vitales para la seguridad:

1. **Verificación de la Integridad:** Como el hash depende de todo el contenido, cualquier manipulación en una transacción cambiaría la Raíz de Merkle, lo que a su vez cambiaría el hash del bloque. Esto es inmediatamente detectable.

2. **Encadenamiento de Bloques:** La cabecera de cada bloque nuevo **incluye el hash del bloque anterior**. Esto crea un eslabón criptográfico que une los bloques en una cadena.



Si un atacante modifica un dato en el Bloque N-1, su hash (0x123) cambiará. Esto romperá la cadena, porque el Bloque N seguirá apuntando al hash antiguo. Cualquier nodo de la red detectaría esta inconsistencia y rechazaría la cadena manipulada. Esto es lo que refuerza la **inmutabilidad** de la blockchain.

Capítulo 12

Fundamentos de la confianza: Sistemas descentralizados

Para explicar este concepto, vamos a tomar el siguiente ejemplo:

Digamos que vamos a volar desde un destino A a un destino B, la autoridad del aeropuerto hace un exhaustivo de protocolos de seguridad importantes, antes de subir a la avión. Esto establece la confianza. Luego hay confianza adicional una vez que han ingresado y sus pasaportes y documentos de viaje fueron verificados, validados y su equipaje ha sido revisado. Aún más confianza generan cuando el personal de la aerolínea verifica su boarding pass en la puerta de embarque y entran al avión para viajar.

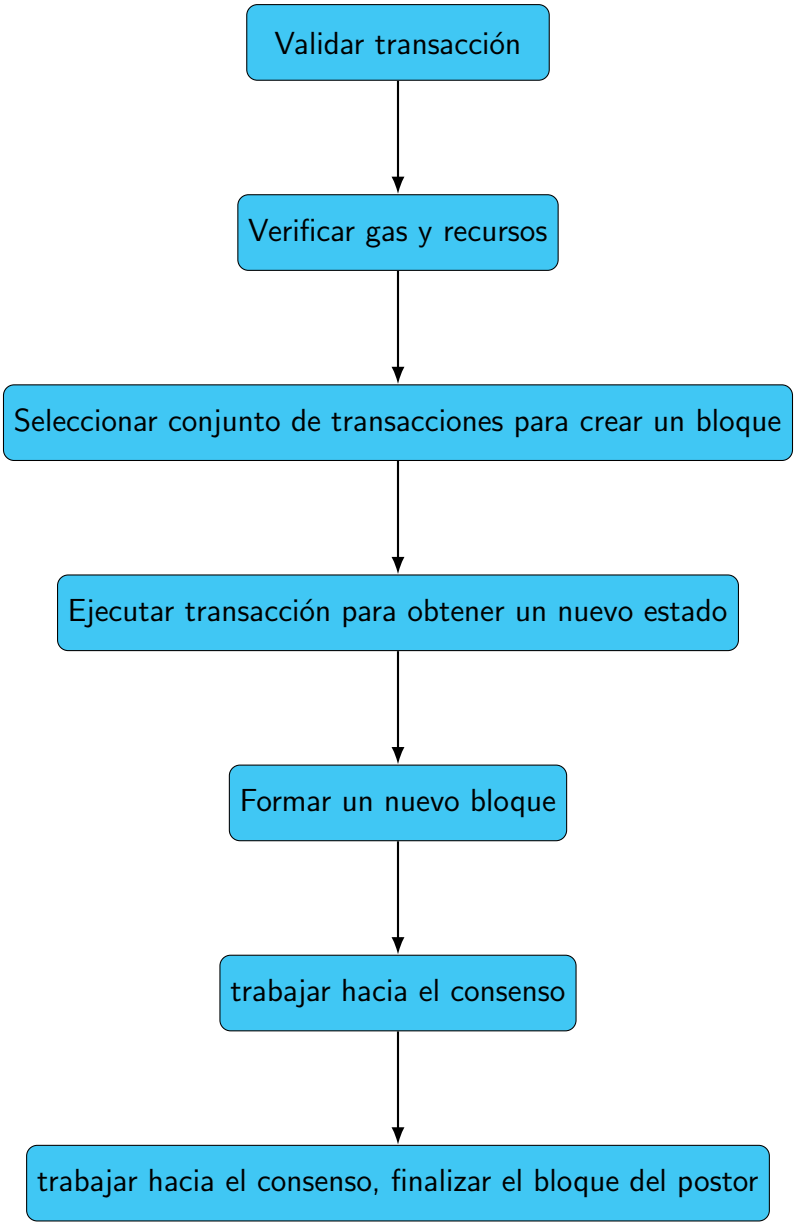
12.1. ¿Que es un sistema descentralizado?

En los sistemas descentralizados, todas las operaciones no hay hay alguien sea entidad corporativas, protocolos o sistemas de monitorio o personas de alto de poder u otro individuo que sea intermediario en algún proceso en especifico, esto viendo como la transparencias de datos en tipo información o valor monetario. Pero acá surge algo... **¿Como esto genera la confianza y quien hace el proceso similar al de personal de seguridad y protocolos del aeropuerto?**

Es sencillo, para certificar credenciales y hacer las validaciones de usuarios es importante recordar los algoritmos criptográficos de hash o como clave publica y clave privada, y de como estos sirven para generar las firmas de transacciones. **La confianza en una cadena de bloques descentralizar es también sobre asegurar, validar, verificar y asegurarse de que los recursos necesarios para la ejecución de la transacción estén disponibles. Esto se logra asegurando la cadena de bloques usando protocolos específicos, validando la transacción y bloque para protegerse de las falsificaciones, verificando la disponibilidad de los recursos para la transacción, y ejecutando y confirmando las transacciones.**

12.2. Ruta de confianza

La ruta de confianza esta definida por las siguientes operaciones:



Para entender cada paso, vamos a explicar cada uno de ellos y esto que significa:

El paso 1 valida la transacción y el paso 2 comprueba los recursos computacionales, en este caso el gas para hacer dicho trabajo. En el caso de un Bitcoin, Hay acerca de 20 criterios que tienen que verificarse antes de validar una transacción.

La sintaxis, la firma de transacción, la hora, el nonce, el límite de gas, y el saldo de la cuenta del remitente se validan antes de la ejecución. El gas, o puntos de gas, y otros recursos disponibles para la ejecución del contrato inteligente, también se validan. El hash y las firmas de transacción también son verificados. El paso número tres es ejecutar las transacciones. Se calcula el árbol de merkle de las transacciones validadas. Esto es en Ethereum. Esta es la raíz de la transacción del encabezado de bloque. Todos los mineros ejecutan la transacción para transferir, así como también para la ejecución de los contratos inteligentes. El estado resultante de la ejecución de la transacción es usado en calcular el árbol de merkle de los estados, la raíz del estado del encabezado de bloque. También se calcula la raíz receptora del encabezado de bloque.

Capítulo 13

Protocolo de consenso

Tomando lo anterior de los conceptos de confianza y como este cambia en sistemas descentralizados, una cadena segura es una cadena principal única con un estado coherente. Cada bloque válido que se añade a esta cadena, aumenta el nivel de confianza de la cadena. Los mineros compiten, compiten por añadir su bloque a la cadena. Entonces, surge la siguientes preguntas: **¿Qué ocurre si todos quieren añadir su bloque candidato a la cadena? Cada uno de los bloques candidatos es de un minero que compite. ¿Cuál es el siguiente bloque que se añade a la cadena? ¿Pueden ponerse de acuerdo sobre el siguiente bloque? ¿Existe un método o un protocolo para elegir el siguiente bloque?** La respuesta es si: **Proof of Work**.

13.1. Proof of Work (Prueba de trabajo)

Es el primer mecanismo de consenso distribuido, desarrollado por el creador seudónimo de bitcóin, **Satoshi Nakamoto**. En Proof of Work, ordenadores de la red encargados de mantener la seguridad de la blockchain (conocidos como mineros) trabajan en resolver un rompecabezas compuesto por una función matemática hash. Esta tarea es sencilla (para un ordenador) pero extremadamente repetitiva y, por lo tanto, computacionalmente costosa. Los ordenadores compiten para encontrar un hash con propiedades específicas. El ordenador que encuentra la respuesta primero (la prueba de que ha realizado el trabajo necesario) puede añadir un nuevo bloque de transacciones a la cadena de bloques.

Este proceso asegura que todos los bloques en la cadena sean válidos y confiables. Si un bloque es agregado, aumenta la confianza en la cadena, y los demás mineros actualizan su copia local de la cadena para incluir el nuevo bloque. Así, todos trabajan juntos para mantener la seguridad y la integridad de la información en la cadena de bloques.

Nota: La única forma en que un atacante podría cometer un fraude de este tipo es poseer una enorme potencia computacional, lo que le permitiría minar bloque tras bloque, ganando la competencia de prueba de trabajo una y otra vez. Esto se conoce como un .ªtaque del 51 %"debido a la necesidad de poseer más de la mitad de la tasa de hash total de la red.

13.2. Proof of Stake (Prueba de participacion)

Se sabe que Proof of Work es bastante computacionalmente costoso y se requiere de muchos recursos para ser parte del minado de lo bloques en la blockchain. Entonces se da **Proof of Stake** con un enfoque alternativo que ha ganado popularidad en los últimos años y que no requiere de hardware especializado.

En Proof of Work, la tasa de hash determina la probabilidad de que un participante añada el siguiente bloque de transacciones a la blockchain. En Proof of Stake, la participación en monedas del participante determina su probabilidad. Es decir, cada nodo de la red está vinculado a una dirección, y cuantas más monedas tenga esa dirección, mayor será la probabilidad de que mine (o .ªpueste", en este caso) el siguiente bloque. Es como una lotería: el ganador se determina al azar, pero cuantas más monedas (billetes de lotería) tenga, mayores serán las probabilidades.

Nota: Un atacante que intente realizar una transacción fraudulenta necesitaría más del 50 % de las monedas para procesar las transacciones requeridas de forma fiable; comprarlas elevaría el precio y encarecería la operación. Revisión de mecanismos de consenso

Capítulo 14

Robustez en la Blockchain

La robustez en blockchain se refiere a la capacidad del sistema para manejar situaciones excepcionales o problemas de manera efectiva y segura. Esto es especialmente importante en redes descentralizadas, donde no hay intermediarios que supervisen las transacciones. Aquí hay algunos puntos clave sobre la robustez en blockchain:

1. **Manejo de excepciones:** La robustez implica que el sistema puede gestionar situaciones inesperadas, como conflictos en las transacciones o problemas de consenso entre los mineros.

2. **Seguridad:** Un sistema robusto asegura que las transacciones sean válidas y que no se produzcan fraudes, como el doble gasto, donde un mismo activo digital se intenta usar en múltiples transacciones.

3. **Consolidación de cadenas:** En caso de que se generen múltiples cadenas debido a que varios mineros resuelven un bloque al mismo tiempo, un sistema robusto tiene métodos para consolidar estas cadenas en una sola, asegurando que solo una sea la válida.

4. **Confianza:** La robustez mejora la confianza en el sistema, ya que los usuarios pueden estar seguros de que las transacciones se manejarán de manera justa y segura.

Para este caso vamos hablar de 2 excepciones específicamente, lo primero que vamos a preguntar es:

¿Qué ocurre si más de un minero resuelve el rompecabezas de consenso muy cerca en el tiempo el uno del otro? Esta excepción tiene una probabilidad bastante baja pero aquí vamos a explicar en caso de que suceda. En primer lugar hay protocolos por ejemplo en el **Bitcóin** tiene métodos para consolidar estas cadenas en una sola, asegurando que solo una sea la válida. Es como si en una carrera, dos corredores llegaran a la meta al mismo tiempo, pero solo uno es declarado ganador después de revisar quién llegó primero. Obviamente, en caso de los otros mineros que estuvieron cerca de resolver el hash, se la un incentivo.

Por el lado de **Ethereum** utiliza un enfoque que permite la creación de bloques Runner-Up.

1. **Cadenas paralelas:** Cuando dos mineros encuentran un bloque válido al mismo tiempo, se pueden crear dos cadenas diferentes temporalmente. Esto es similar a tener dos versiones de un evento que suceden casi al mismo tiempo.

2. **Bloques Runner-Up:** Ethereum permite que ambos bloques sean aceptados inicialmente, pero solo uno se convierte en el bloque principal de la cadena. El bloque que no se convierte en el principal se considera un bloque Runner-Up".

3. **Incentivos:** Para mantener la seguridad de la red, Ethereum ofrece un pequeño incentivo a los mineros que crean estos bloques Runner-Up. Esto ayuda a motivar a los mineros a seguir participando en la red, incluso si su bloque no se convierte en el principal.

4. **Consolidación:** Después de que se añade un bloque a la cadena principal, los bloques Runner-Up se mantienen durante un tiempo (generalmente seis bloques) antes de ser descartados. Esto asegura que la red pueda consolidar la cadena principal de manera efectiva.

¿Qué ocurre si más de una transacción hace referencia como entrada al mismo activo digital?

Ahora, hablemos del doble gasto. Esto ocurre cuando se intenta usar el mismo activo digital en más de una transacción, similar a cuando una aerolínea vende el mismo asiento a dos personas. En blockchain, para evitar esto, se permite que solo la primera transacción que use un activo digital sea válida, mientras que las demás se rechazan. En Ethereum, se utiliza un número único llamado **nonce** para cada transacción, lo que ayuda a prevenir el doble gasto. Así, cada vez que se realiza una transacción, se asegura que no se pueda usar el mismo activo dos veces.

14.1. Nonce en Ethereum

En Ethereum, el término **nonce** se refiere a un número que cumple funciones esenciales en diferentes contextos dentro de la red. A continuación se detallan los distintos usos y propósitos del *nonce*.

1. Nonce de Cuenta

Cada cuenta externa (EOA, *Externally Owned Account*) en Ethereum posee un **nonce de cuenta**, que es un contador que indica cuántas transacciones ha enviado dicha cuenta.

- Cada vez que una cuenta externa envía una transacción, su **nonce** incrementa en uno.
- Este valor debe ser exacto en cada transacción; de lo contrario, la red la rechazará.
- El objetivo es evitar ataques de repetición (*replay attacks*) y garantizar el orden correcto de las transacciones emitidas por una misma cuenta.

Ejemplo: Si una cuenta ha enviado 5 transacciones, su nonce actual es 5. Para que la siguiente transacción sea válida, debe tener `nonce = 5`.

14.1.1. 2. Nonce en Creación de Contratos Inteligentes

Cuando se crea un contrato inteligente, Ethereum utiliza el nonce del creador para determinar la dirección del nuevo contrato. Esta dirección se calcula mediante el siguiente algoritmo:

$$\text{Dirección del contrato} = \text{Keccak256}(\text{RLP}(\text{remitente}, \text{nonce}))$$

- Esto garantiza que una misma cuenta no pueda crear dos contratos con la misma dirección.
- Si el creador ha desplegado 3 contratos anteriormente, el *nonce* usado será 3.

14.1.2. 3. Nonce en Minería (antes de Ethereum 2.0)

Antes de la transición a Ethereum 2.0 y la adopción de *Proof of Stake*, Ethereum utilizaba *Proof of Work* como mecanismo de consenso. En este contexto:

- El **nonce** era un número que los mineros modificaban en cada intento de minado.
- Su objetivo era encontrar un hash del bloque que cumpliera con los requisitos de dificultad de la red.
- Al encontrar un hash válido, el bloque era aceptado y el minero recibía la recompensa.

Nota: Desde "The Merge", Ethereum ya no utiliza Proof of Work, por lo tanto este tipo de nonce ya no se emplea.

Capítulo 15

Horquillas

Bifurcaciones en Blockchain

Las bifurcaciones (forks) son eventos importantes en la evolución de una blockchain. Representan cambios en el protocolo que pueden surgir de la necesidad de aplicar mejoras, corregir errores o gestionar crisis. En esta sección, explicamos de forma clara y concisa los conceptos fundamentales relacionados con las bifurcaciones en el contexto de Ethereum.

1. ¿Qué es una bifurcación?

Una **bifurcación** en blockchain es una desviación o división en la cadena de bloques cuando los nodos ya no están de acuerdo sobre la siguiente versión del software o del protocolo.

- Pueden ser **planificadas** o **no planificadas**.
- Son una parte natural del proceso evolutivo de la tecnología blockchain.
- Reflejan cómo se adaptan y mejoran las cadenas frente a nuevas necesidades o problemas críticos.

2. Tipos de bifurcaciones

- **Bifurcación blanda (soft fork):**
  - Es compatible con versiones anteriores.
  - No requiere que todos los nodos actualicen su software.
  - Se considera un *parche* o mejora menor.
  - Ejemplo: introducción del algoritmo `scrypt` en Bitcoin.
- **Bifurcación dura (hard fork):**
  - Implica un cambio importante en el protocolo.
  - Requiere que todos los nodos se actualicen.
  - Después de una hard fork, las cadenas resultantes pueden ser **incompatibles**.

- Ejemplo: bifurcación de Ethereum en 2016 que originó `Ethereum` y `Ethereum Classic`.

Resumen comparativo:

Característica	**Soft Fork**	**Hard Fork**
Compatibilidad	Compatible con versiones antiguas	Incompatible con versiones antiguas
Actualización de nodos	No es obligatoria	Obligatoria para participar
Ejemplo	Introducción de `scrypt`	Ethereum → Ethereum Classic
Naturaleza	Cambios menores	Cambios estructurales

3. Caso: Bifurcación dura del 17 de octubre de 2017

Esta fue una bifurcación **planificada** en la red Ethereum. Algunas de las **EIPs** (Ethereum Improvement Proposals) asociadas fueron:

- Introducción del **procesamiento paralelo** de transacciones.
- Evaluación del consenso **Proof of Stake (PoS)** cada 100 bloques, manteniendo mayoritariamente Proof of Work (PoW).
- Reducción del incentivo por bloque: de 5 éteres a 3 éteres.

4. Importancia de las bifurcaciones

- Son mecanismos que aportan **robustez** al sistema blockchain.
- Permiten corregir fallos, añadir funcionalidades y adaptarse a nuevos desafíos.
- Bien gestionadas, aumentan la **credibilidad** y la confianza en la red.
- Se pueden comparar con:
  - Soft fork → parche de software.
  - Hard fork → nueva versión de un sistema operativo.

5. Ejemplo emblemático: DAO Hack y Ethereum Classic

En 2016, una falla crítica en un contrato inteligente de la **DAO (Organización Autónoma Descentralizada)** resultó en un robo multimillonario. Para solucionar esto:

- Se ejecutó una **bifurcación dura no planificada**.
- Dio origen a dos redes: `Ethereum (ETH)` y `Ethereum Classic (ETC)`.