Platzi

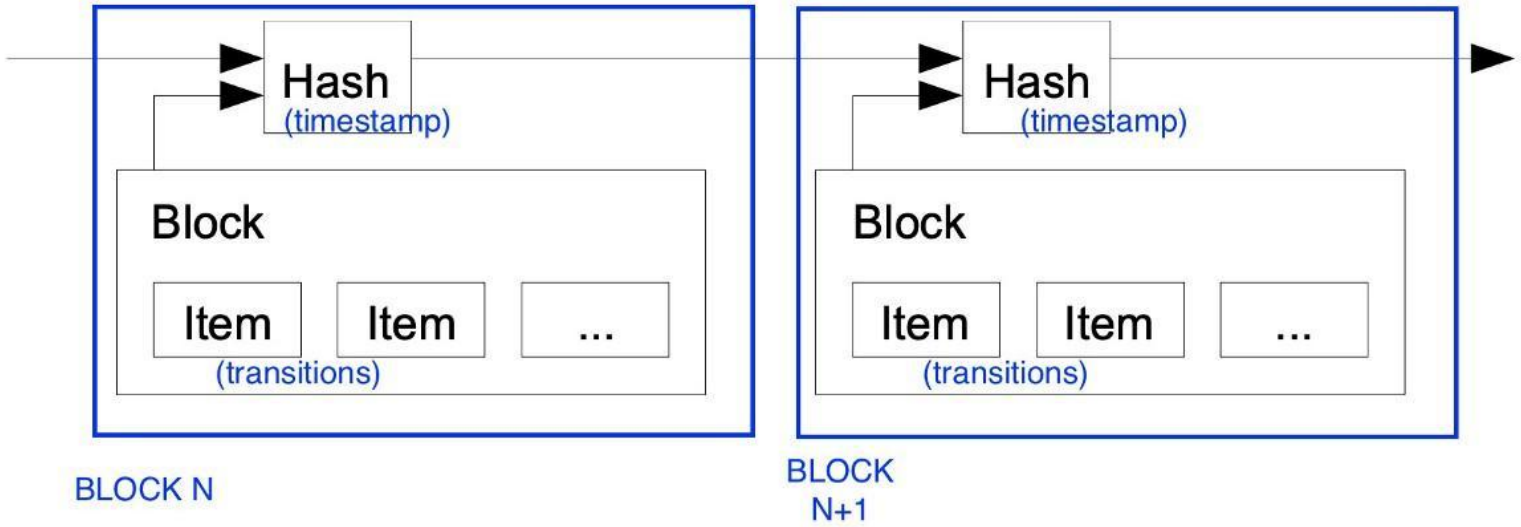
Curso de **Bitcoin** para Developers

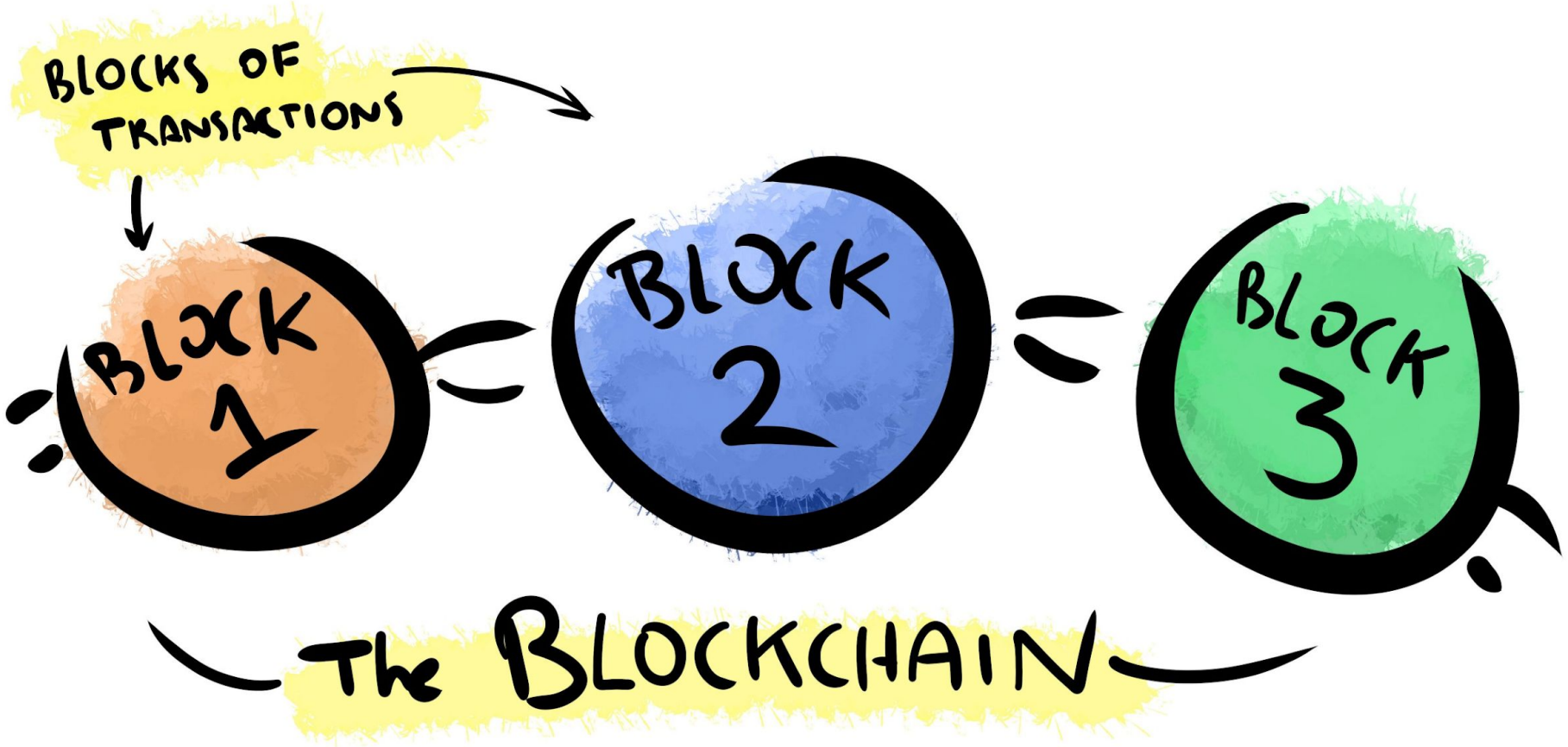
Módulo 1:
Blockchain

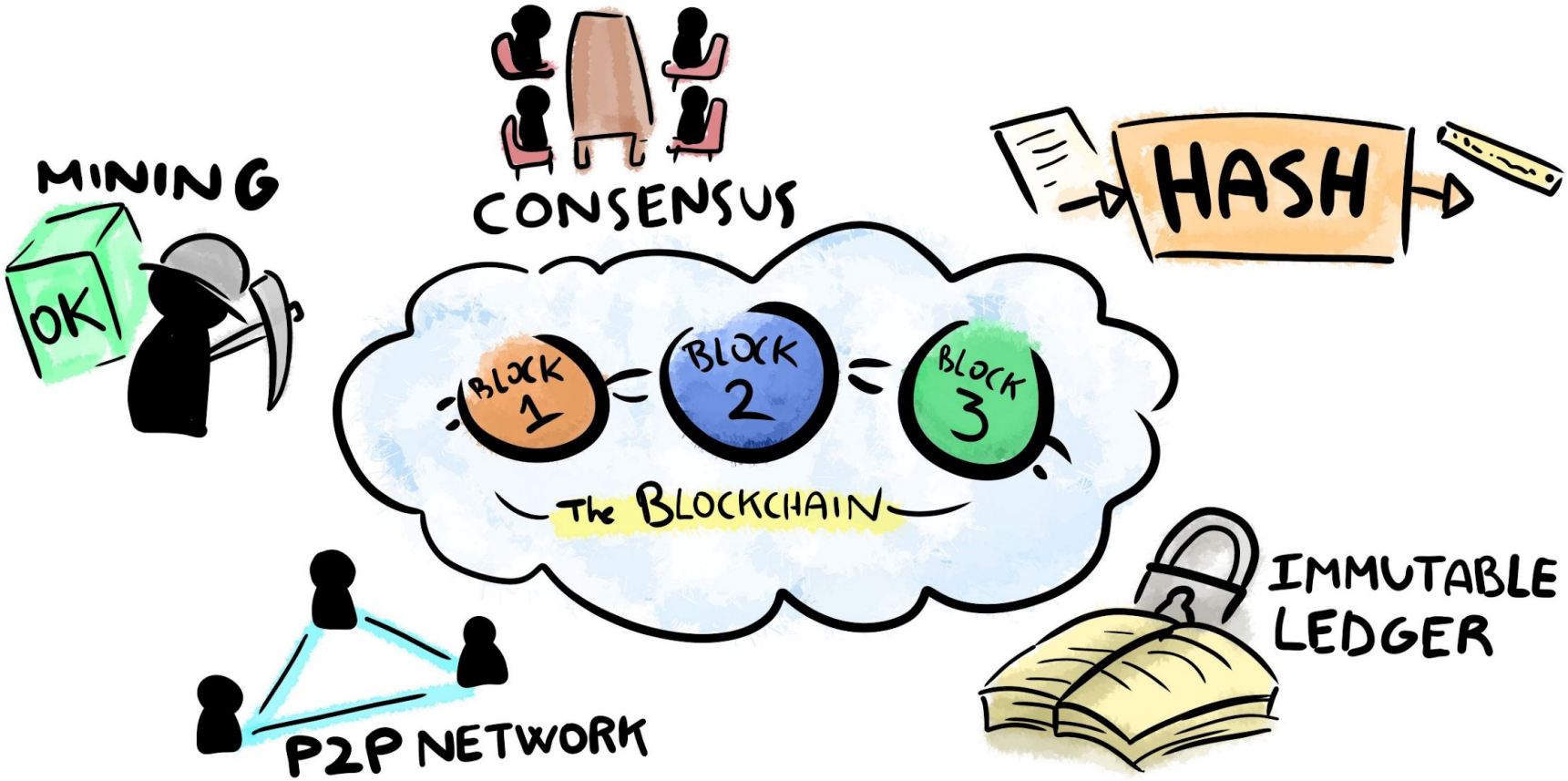
Bitcoin: Servidor de Marcas de Tiempo

- Solución propuesta en el whitepaper.
- Una marca de tiempo, o timestamp, prueba la existencia de información.
- Encadenamiento a través del hash de la marca de tiempo.

Bitcoin: Servidor de Marcas de Tiempo

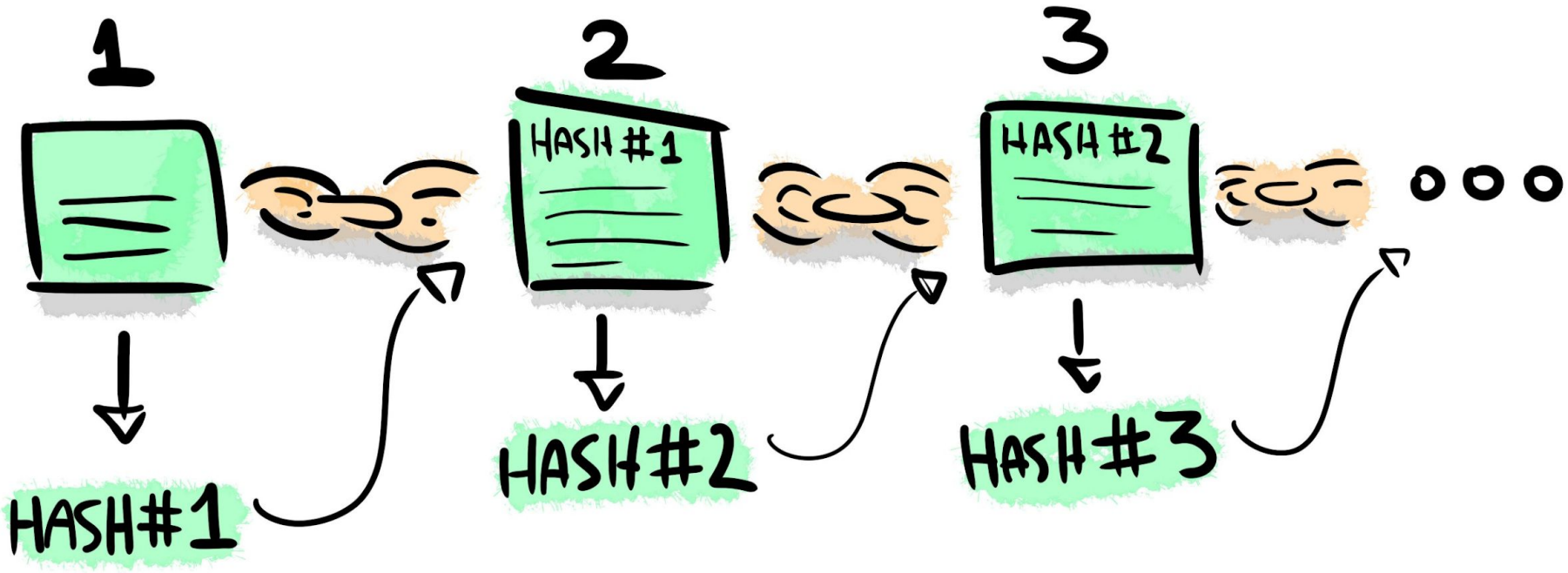


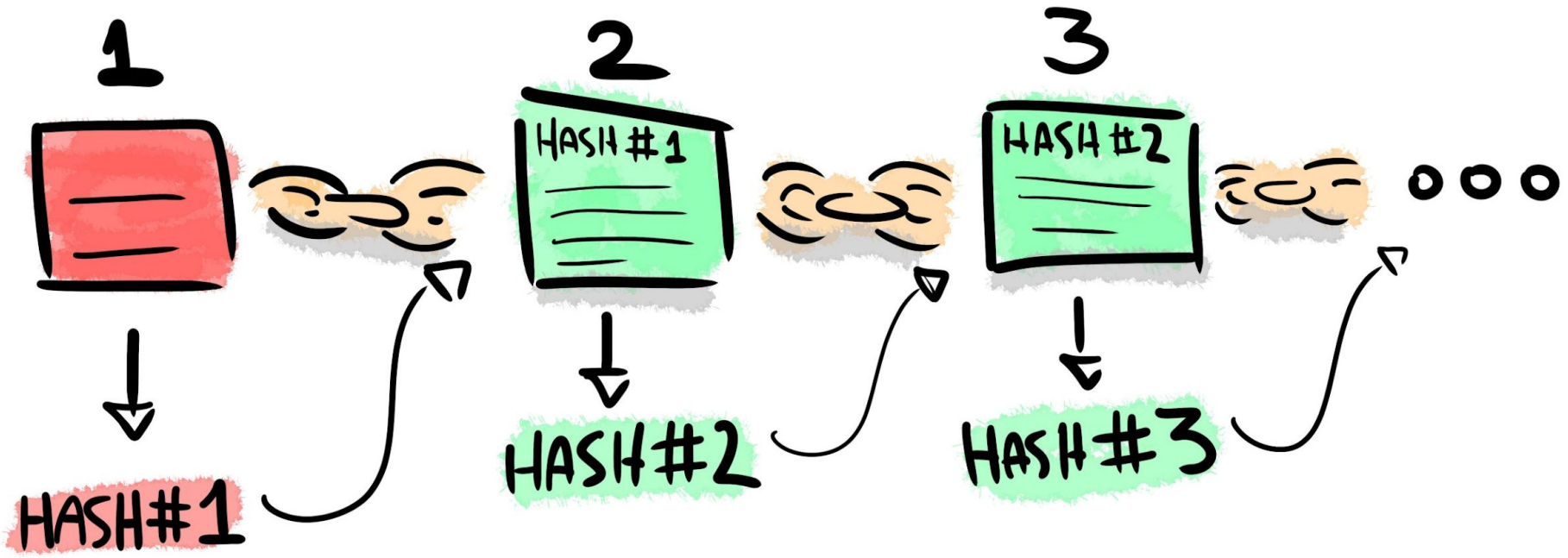

BLOCKS OF TRANSACTIONS
BLOCK 1
BLOCK 2
BLOCK 3
The BLOCKCHAIN

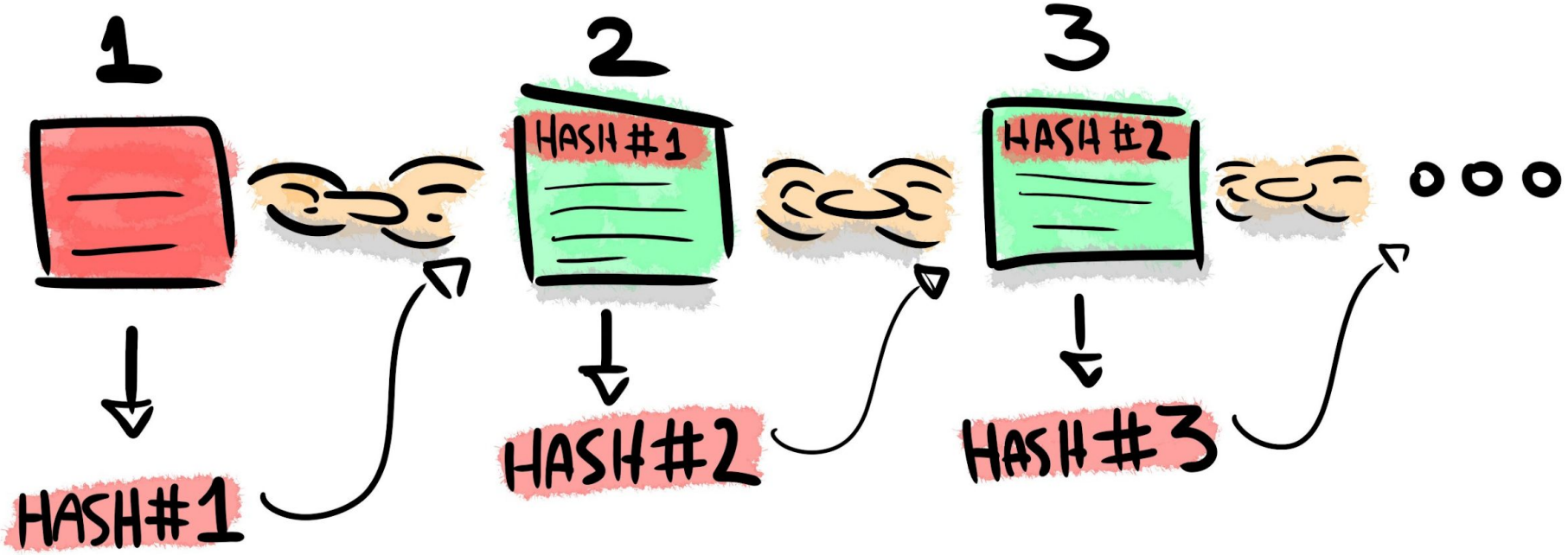

MINING
OK
CONSENSUS
HASH
BLOCK 1
BLOCK 2
BLOCK 3
The BLOCKCHAIN
IMMUTABLE LEDGER
P2P NETWORK

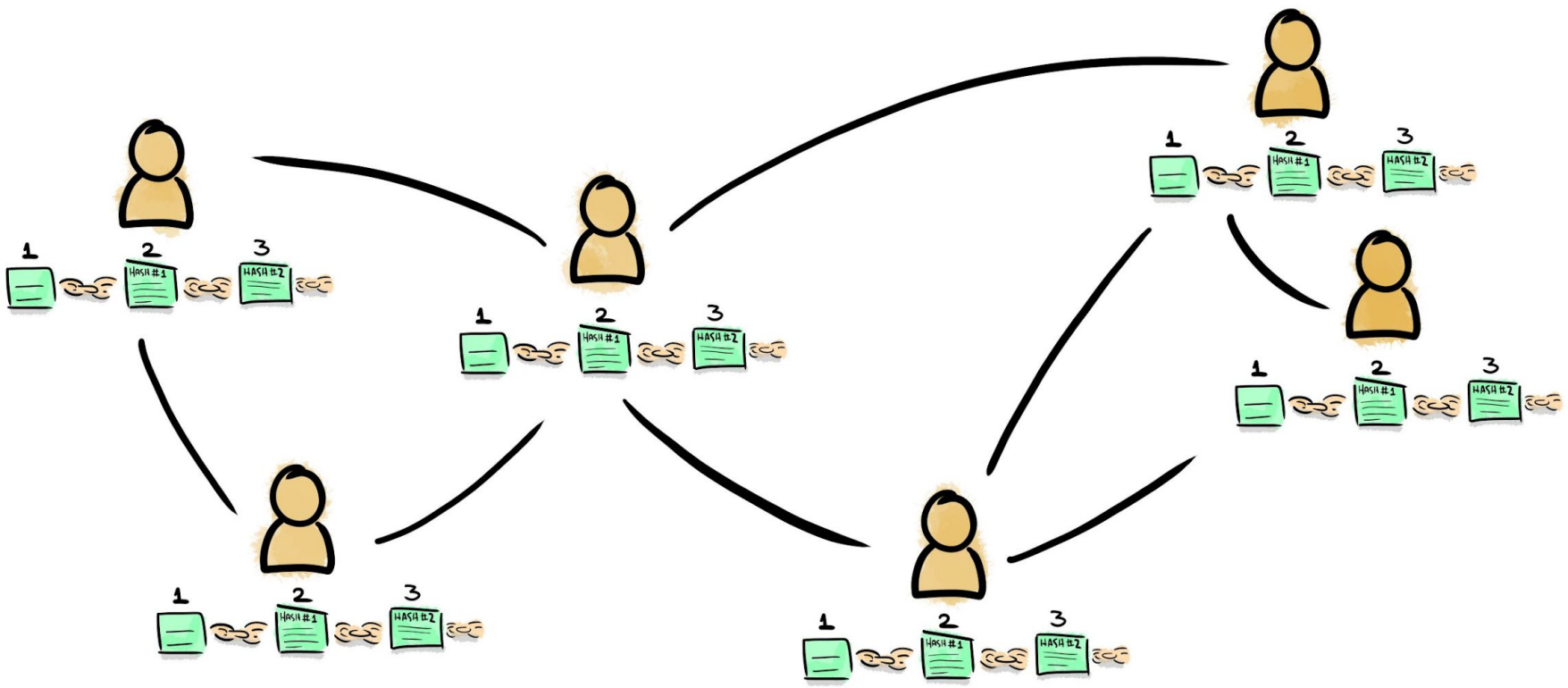
Hello

05ade08fcfb104f40b2536a14dfcd6e916d643f5cf8044b19028b607ae8f4908


1
2
HASH #1
3
HASH #2
HASH#1
HASH#2
HASH#3


1
2
3
HASH #1
HASH #2
HASH#1
HASH#2
HASH#3


1
2
HASH #1
3
HASH #2
HASH#1
HASH#2
HASH#3


1
2
HASH #1
3
HASH #2

Bitcoin: blockchains

- Mainnet: Blockchain principal
- Testnet: Blockchain de pruebas
- Regtest: Blockchain local

Reto 1:

¿Cómo explicarías a tu familia *(adultos y niños)* de la manera más simple el concepto de libro mayor descentralizado?

Módulo 1.1:
Bloques

Estructura de un bloque

Campo	Descripción	Tamaño (En bytes)
Block Header	Múltiples campos (metadata) conforman el encabezado, ver Estructura un encabezado	80
Block Size	El tamaño del bloque, en bytes	4
Transaction Counter	Cantidad de transacciones	1-9 Varint
Transactions	Las transacciones almacenadas en este bloque	Variable

Estructura de un encabezado

Campo	Descripción	Tamaño (En bytes)
Version	Número de versión de software, si se actualiza el software, se especifica una nueva versión.	4
Previous Block Hash	Referencia al hash del bloque previo en la cadena	32
Merkle Root	Hash de la raíz del *merkle tree* de las transacciones asociadas a este bloque	32
Timestamp	Timestamp actual del bloque en segundos desde 1970-01-01T00:00 UTC	4
Difficulty Target	La dificultad asociada al algoritmo de *proof-of-work* para este bloque	4
Nonce	Contador usado para el algoritmo de *proof-of-work*	4

Identificadores: Block Header Hash y Block Height

- El *block hash* se calcula al pasar 2 veces el encabezado través del algoritmo SHA256.
- El *block hash* identifica un bloque de manera única.
- *Block height* es la posición del bloque dentro de la cadena.

El Bloque génesis

- Primer bloque en la cadena de bloques.
- Ancestro en común de todos los bloques.
- Identificador bloque génesis en Bitcoin: 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

The Times 03/Jan/2009

Chancellor on brink of second bailout for banks.

Cómo se unen los bloques

- La copia local de la cadena de bloques se actualiza constantemente cuando nuevos bloques se añaden.
- **Clave**: cabecera y el campo *previous block hash*

Cómo se unen los bloques


```
{
    "size" : 43560,
    "version" : 2,
    "previousblockhash" :
        "000000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249",
    "merkleroot" :
        "5e049f4030e0ab2debb92378f53c0a6e09548aea083f3ab25e1d94ea1155e29d",
    "time" : 1388185038,
    "difficulty" : 1180923195.25802612,
    "nonce" : 4215469401,
    "tx" : [
```

Block Header Hash:
0000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4
Block Height: 277316

Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
CC55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54;
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root:
c910008c26e50763e9f548bb8b2fc323735f735
77effbc55502c51eb4cc7cf2e

Transacciones

Block Header Hash:
0000000000000002a7bbd25a417c0374
CC55261021e8a9ca74442b01284f0569
Block Height: 277315

Previous Block Header Hash:
00000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Timestamp: 2013-12-27 22:57:18
Difficulty: 1180923195.26
Nonce:4215469401
Merkle Root: 5e049f4030e0ab2debb92378f5
3c0a6e09548aea083f3ab25e1d94ea1155e29d

Transacciones

Módulo 1.2:
Merkle Trees

Merkle Trees

- Estructura de datos usada para resumir y verificar grandes conjuntos de datos.
- Algoritmo usado por Bitcoin: double-SHA256
- Huella digital del conjunto de transacciones dentro del bloque.

Block Header Hash:
0000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4
Block Height: 277316

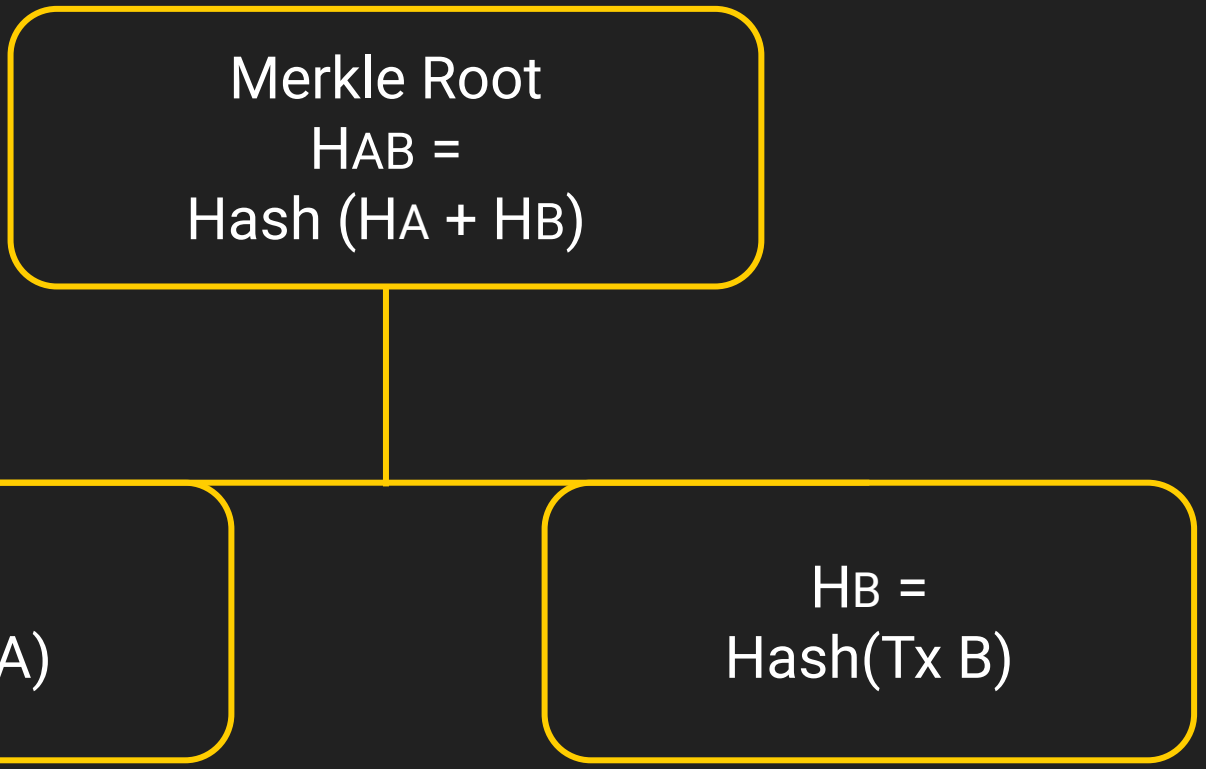
Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
CC55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54;
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root:
c910008c26e50763e9f548bb8b2fc323735f735
77effbc55502c51eb4cc7cf2e

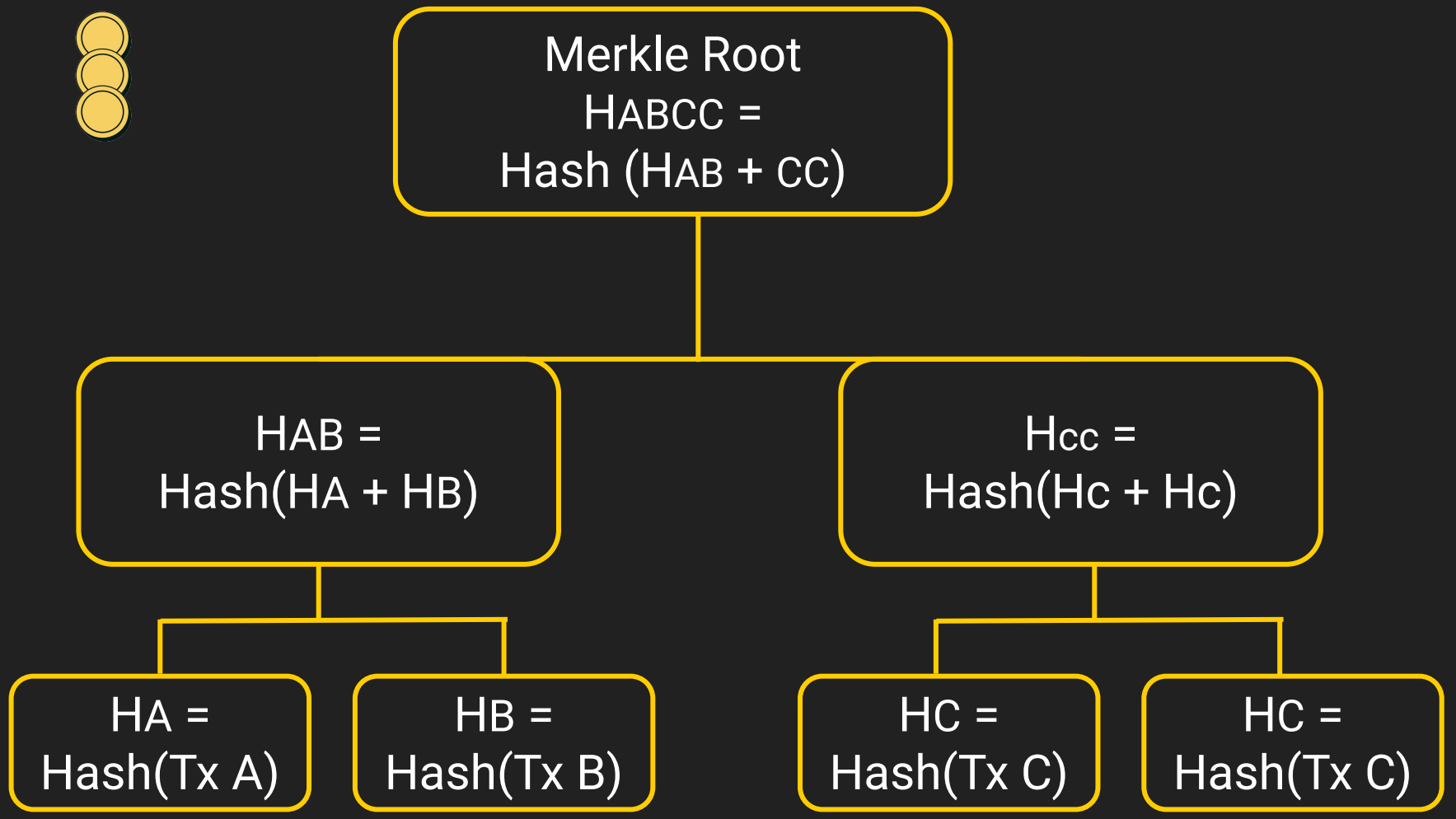
Transacciones

Block Header Hash:
0000000000000002a7bbd25a417c0374
CC55261021e8a9ca74442b01284f0569
Block Height: 277315

Previous Block Header Hash:
0000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Timestamp: 2013-12-27 22:57:18
Difficulty: 1180923195.26
Nonce:4215469401
Merkle Root: 5e049f4030e0ab2debb92378f5
3c0a6e09548aea083f3ab25e1d94ea1155e29d

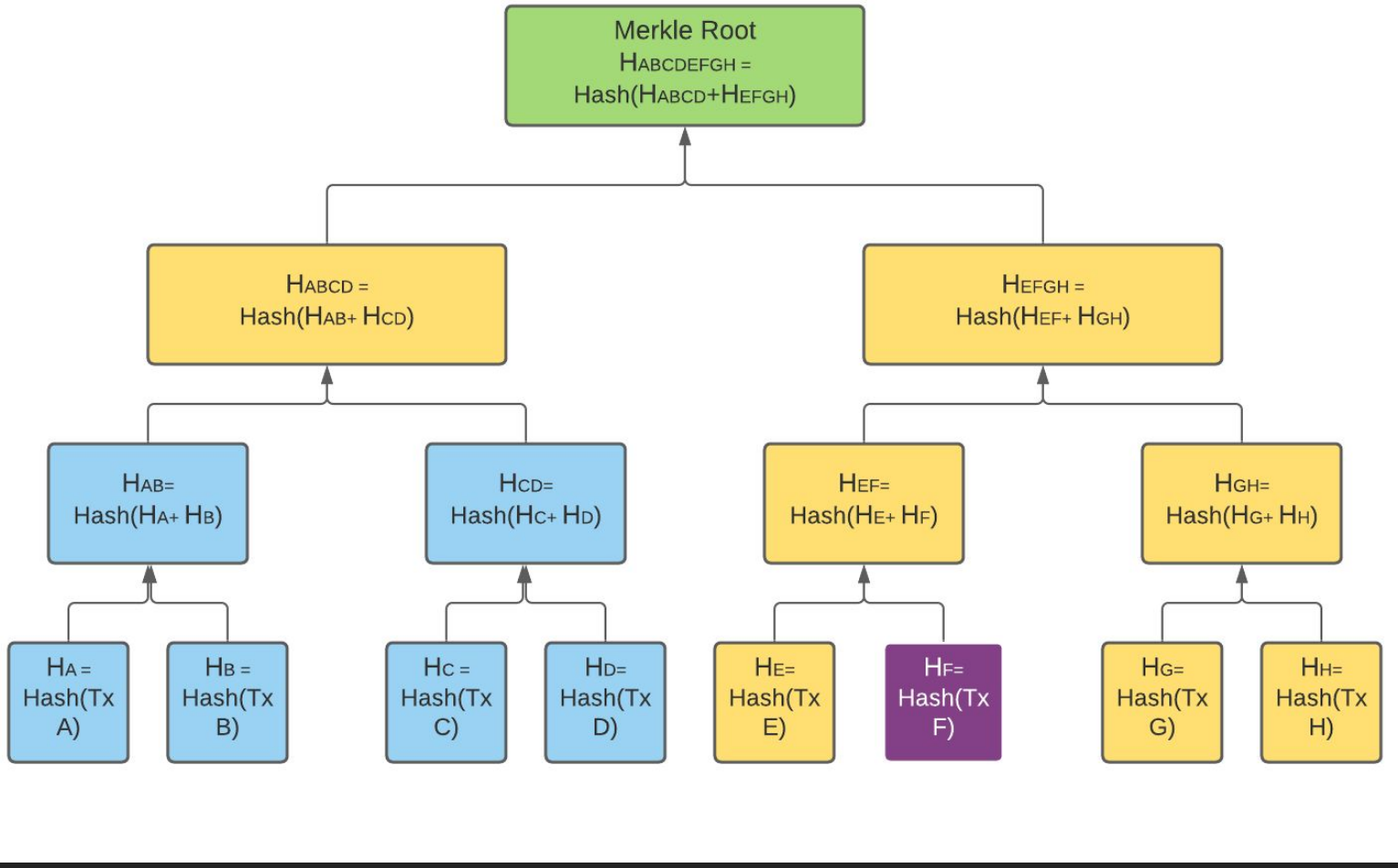
Transacciones


Merkle Root
HAB =
Hash (HA + HB)
HA =
Hash(Tx A)
HB =
Hash(Tx B)

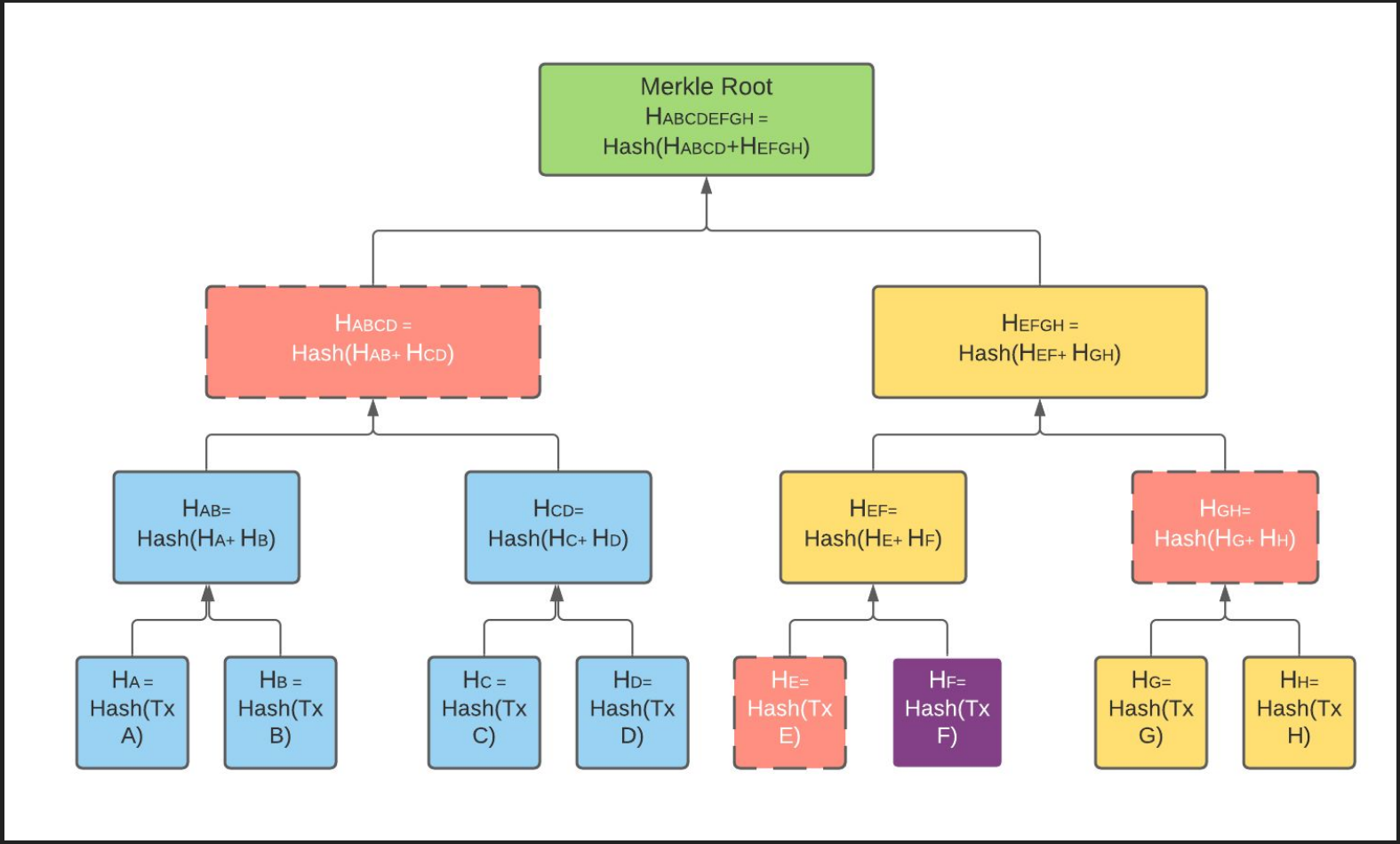

Merkle Root
HABCC =
Hash (HAB + CC)
HAB =
Hash(HA + HB)
Hcc =
Hash(Hc + Hc)
HA =
Hash(Tx A)
HB =
Hash(Tx B)
HC =
Hash(Tx C)
HC =
Hash(Tx C)

Reto 2:

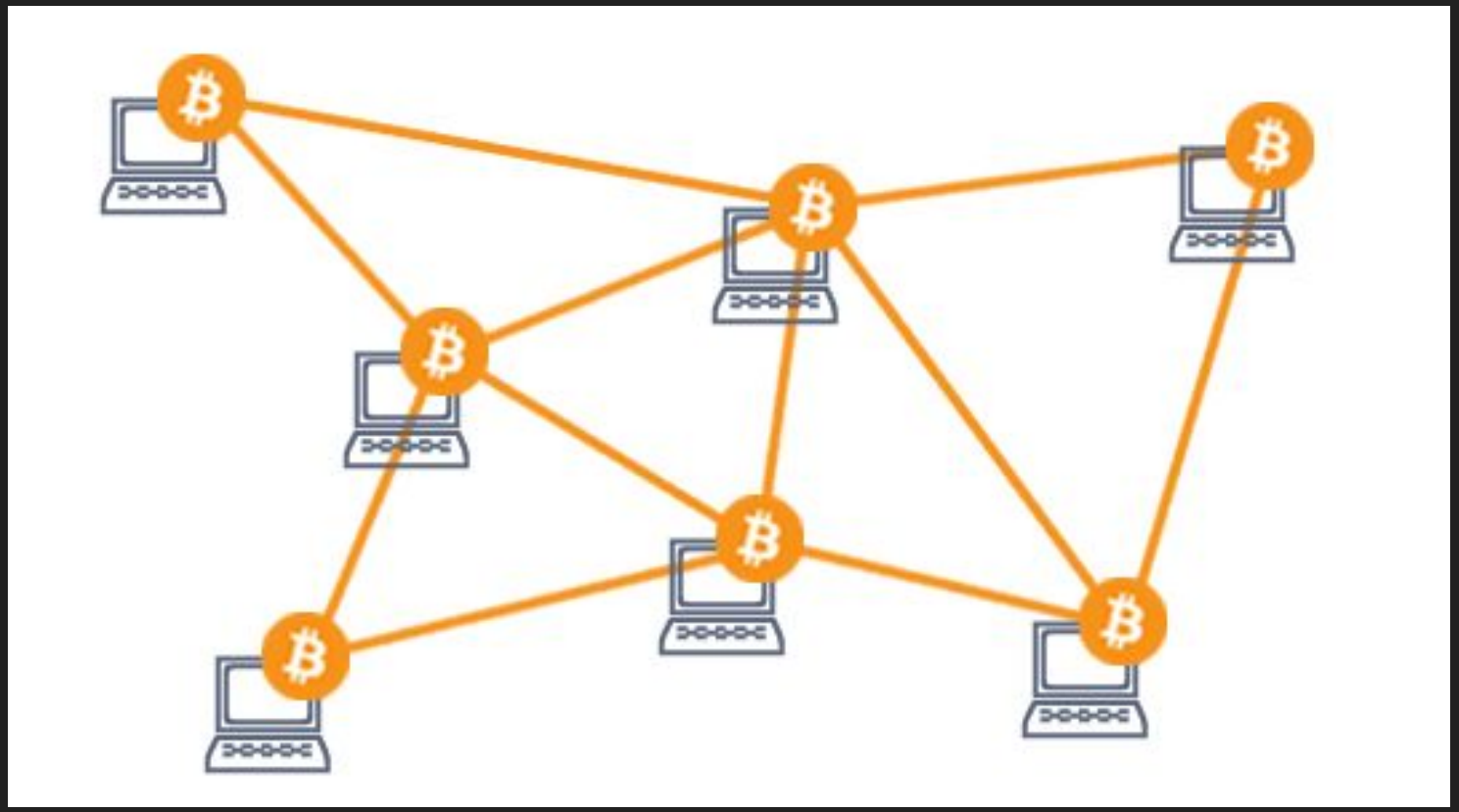
Investiga sobre Merkle Paths y cómo probar la existencia de un elemento sobre el siguiente árbol.


Merkle Root
HABCDEFGH =
Hash(HABCD+HEFGH)
HABCD =
Hash(HAB+ HCD)
HEFGH =
Hash(HEF+ HGH)
HAB=
Hash(HA+ HB)
HCD=
Hash(HC+ HD)
HEF=
Hash(HE+ HF)
HGH=
Hash(HG+ HH)
HA =
Hash(Tx
A)
HB =
Hash(Tx
B)
HC =
Hash(Tx
C)
HD=
Hash(Tx
D)
HE=
Hash(Tx
E)
HF=
Hash(Tx
F)
HG=
Hash(Tx
G)
HH=
Hash(Tx
H)

Reto 2 Solución:



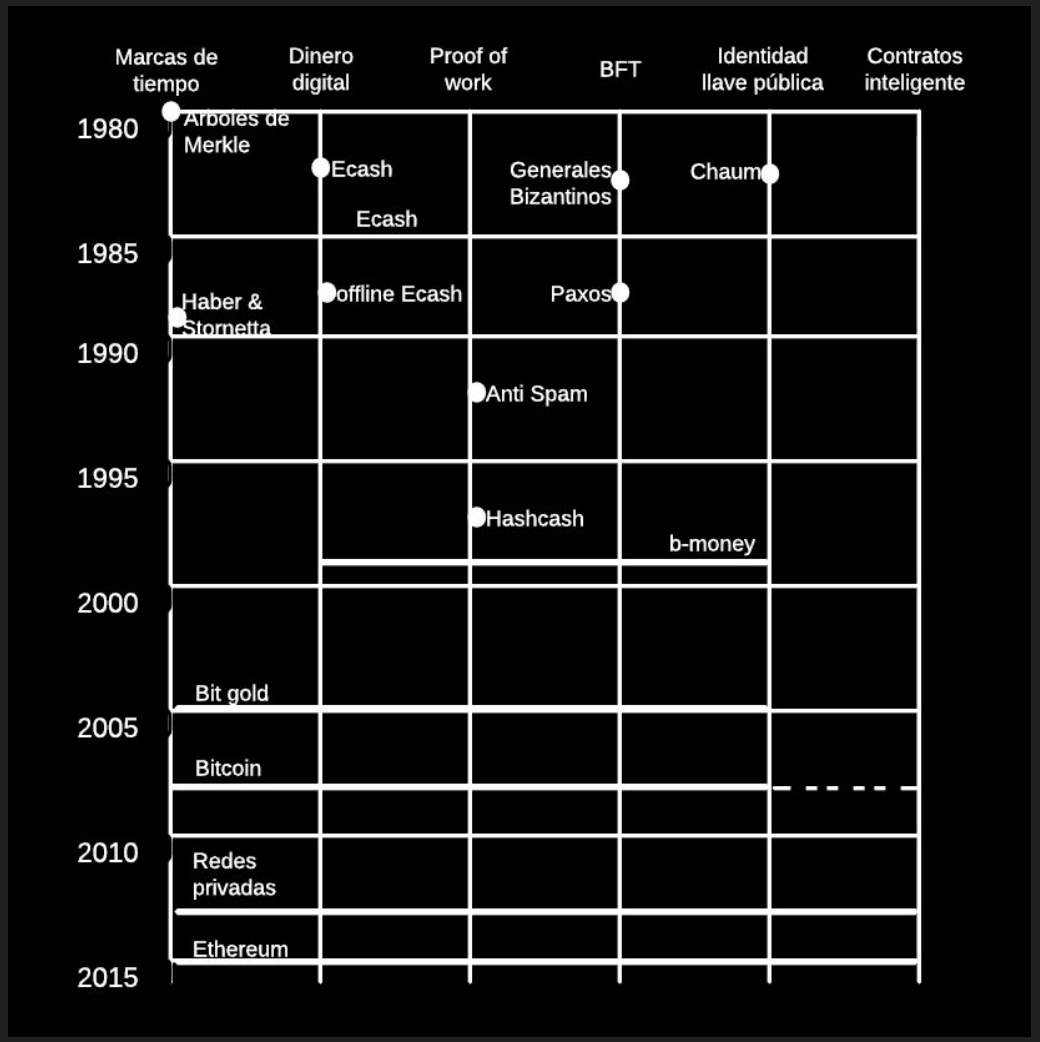
Módulo 2:
Bitcoin Network



La arquitectura de una red P2P

- Todos los participantes son pares entre sí, todos son iguales.
- Resiliente, descentralizada y abierta.
- Bitcoin, Napster, BitTorrent.

Antecedentes y Predecesores



Bitcoin Network

- Conjunto de nodos corriendo el protocolo bitcoin.
- Protocolos adicionales usan el P2P de bitcoin y extienden la red a otros nodos corriendo esos protocolos adicionales.
- Extended Bitcoin Network.

Tipos de nodos

- *Full node*
- *Pruning nodes*
- *Archive nodes*
- *Mining node*
- *Simplified payment verification (SPV)*

Full Node

- Nodo que valida la cadena completa y sus transacciones.
- Mantienen un estado actualizado, un conjunto de UTXO.

Pruning Node

- Nodo que descarga y procesa bloques para generar una base de datos para validación.
- Desecha bloques antiguos para ahorrar espacio.

Archive Node

- Contiene una copa de toda la historia de la cadena.
- Valida y consulta transacciones y bloques en cualquier punto del histórico.
- Entregan el historial a nuevos nodos para que se conviertan en *full nodes*.

Mining Node

- Compiten por crear nuevos bloques.
- Validan transacciones.
- Fundamentales para la seguridad y el consenso de la red.
- Mantienen un *mempool* de transacciones no confirmadas.
- Mining pools.

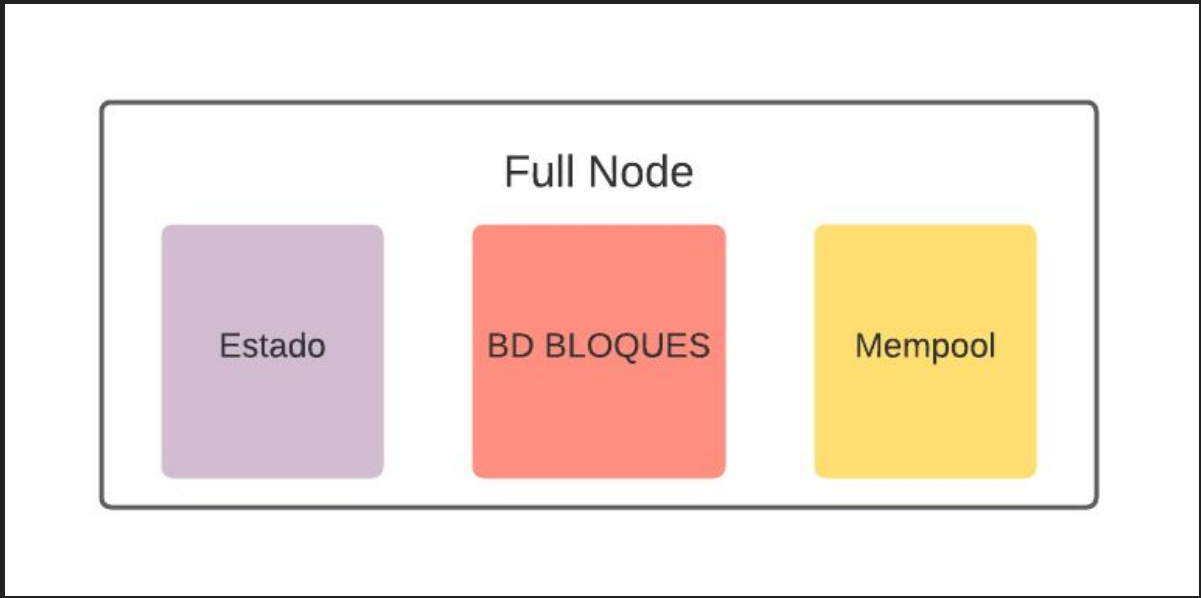
Clientes Ligeros (SPV)

- Operan sin almacenar la cadena completa.
- Descargan solo los encabezados.
- Confían en otros nodos para validar bloques y transacciones.
- Profundidad vs Altura.
- *Merkle Path*.

Otros conceptos

- Descarga de bloque inicial (IBD)
- Bitcoin Core
- Nodo como servicio

La información en un nodo



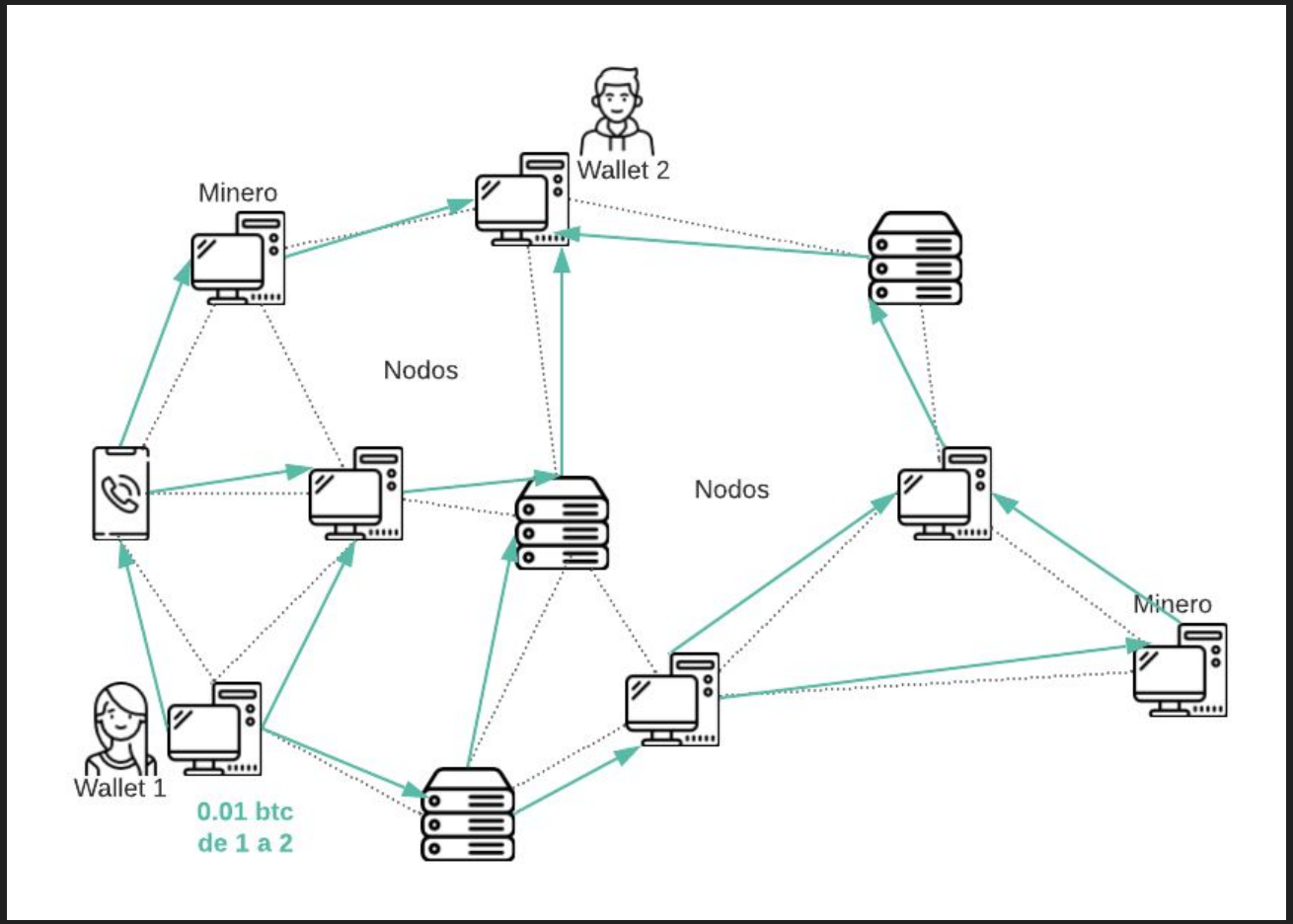
El estado actual es uno de los insumos

Con las transaccio
un nuevo bloque se
de UTXO, borrando
gastaron y agregar
creados por las tra

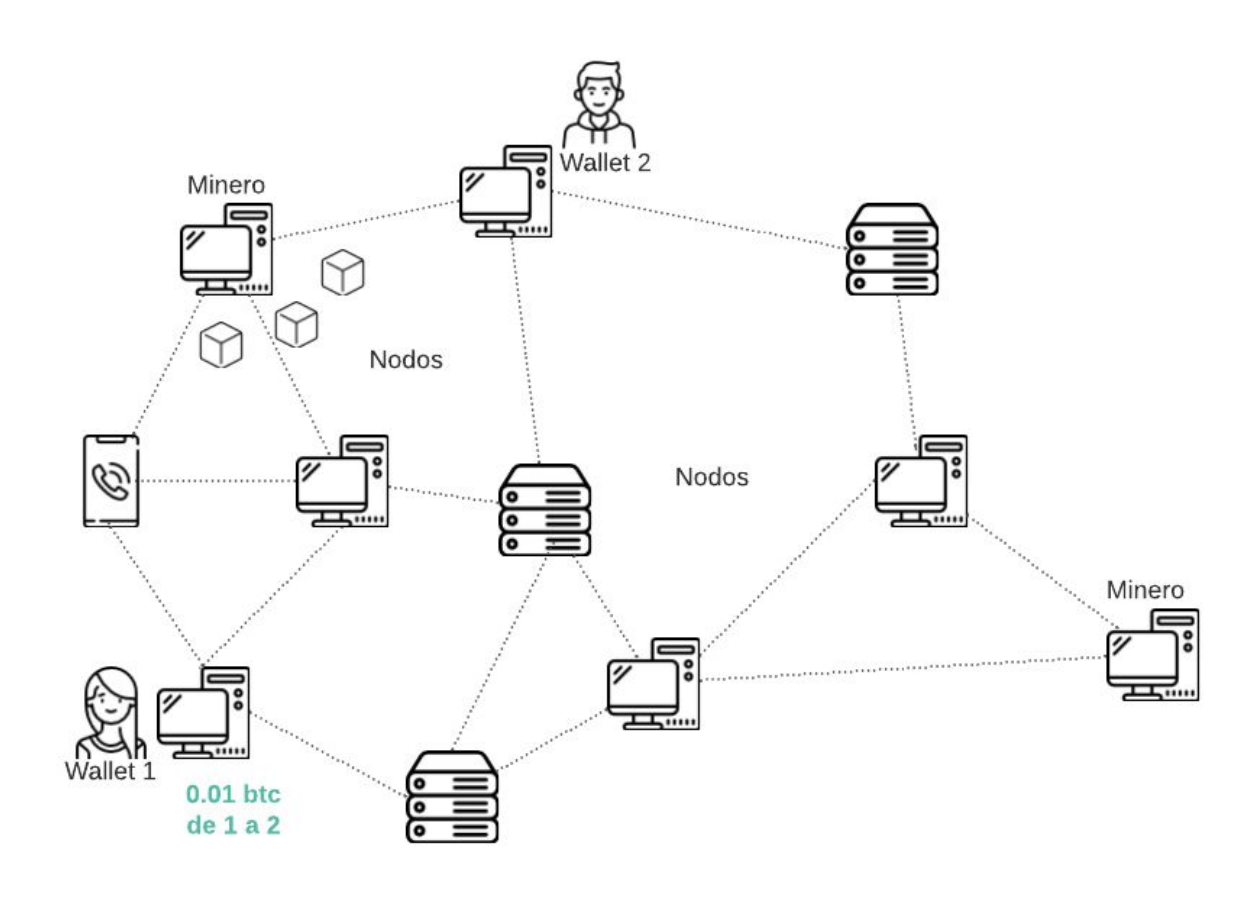
Mempool

- Estructura que contiene los mejores candidatos (transacciones) para minar (agregar un nuevo bloque)
- Políticas
  - Ejemplo: transacción muy grande o muy pequeña

Flujo Transaccional

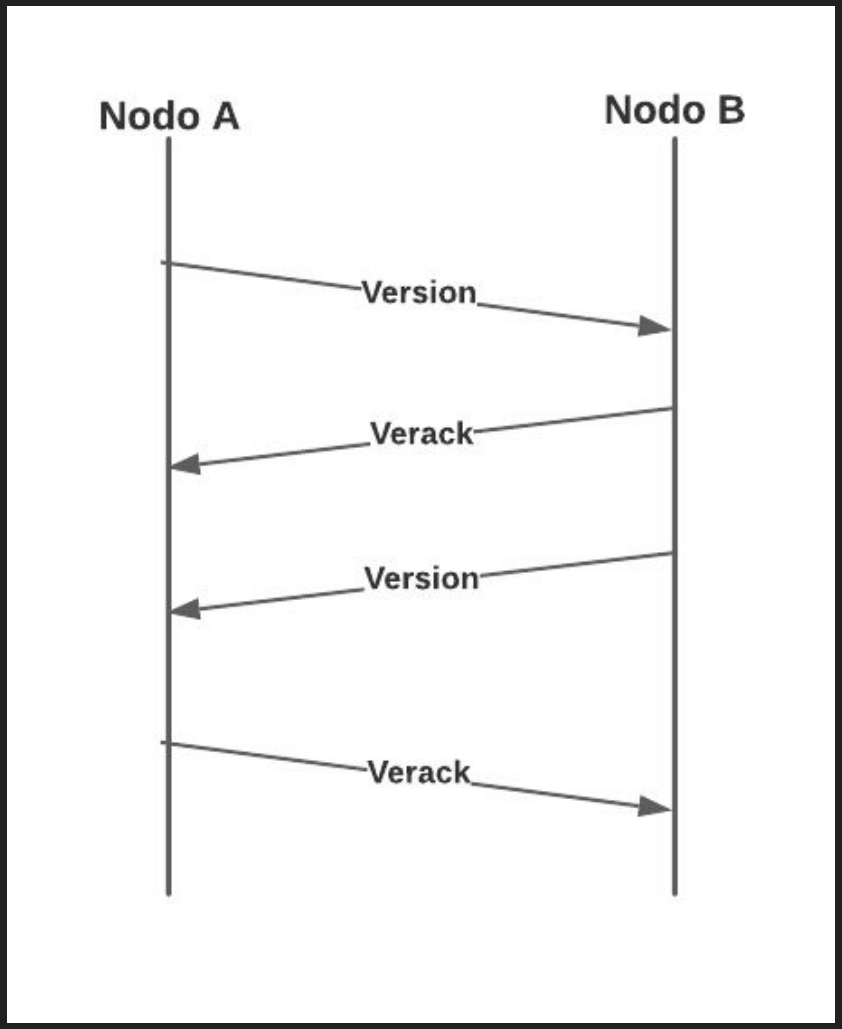


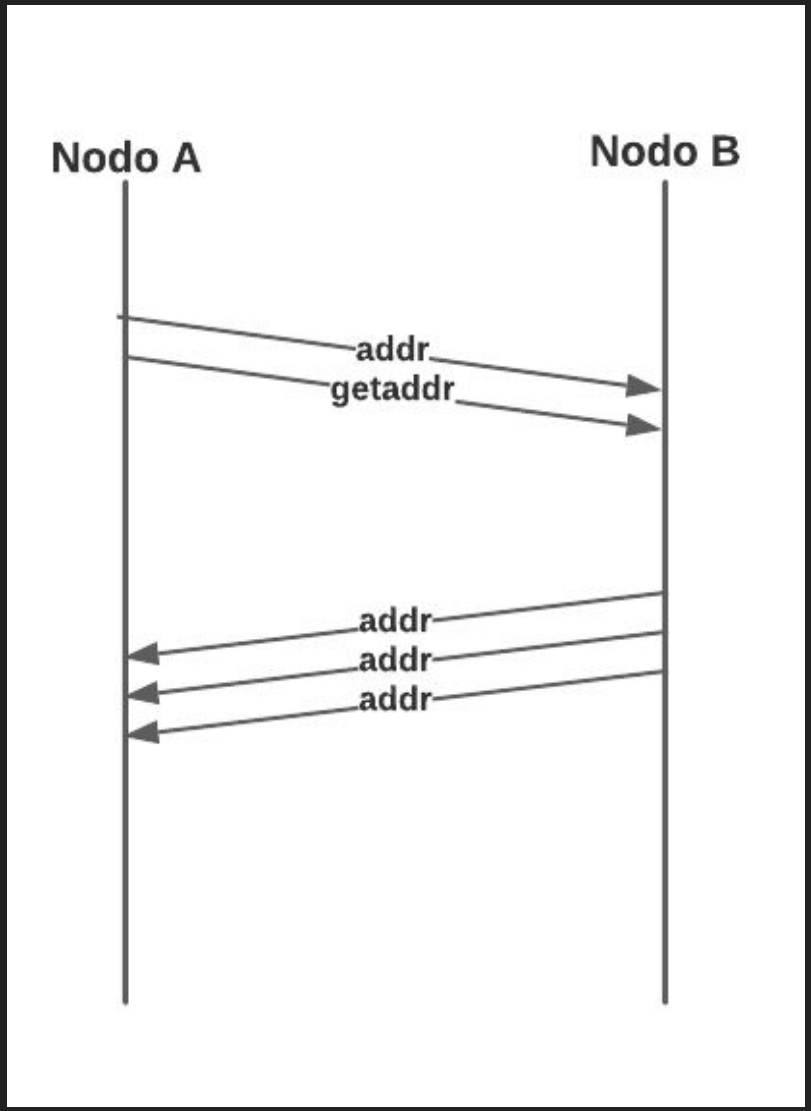
Flujo Transaccional

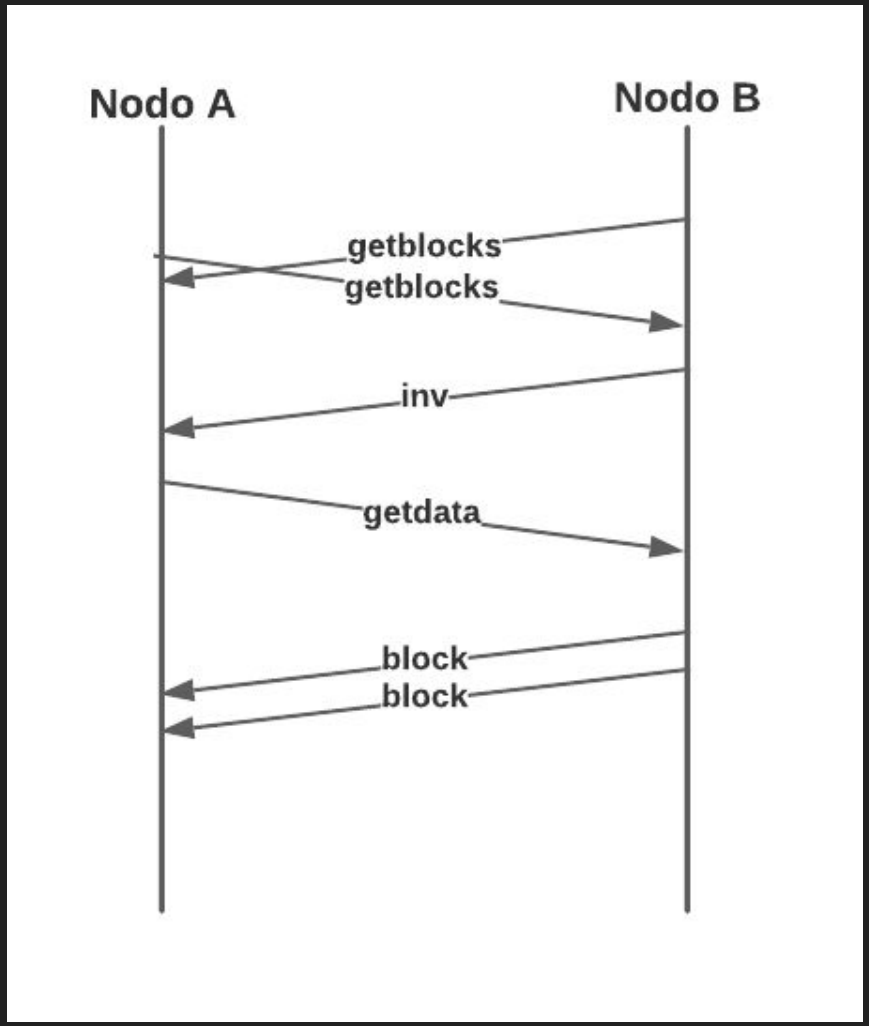


Network Discovery

- Localización geográfica no es relevante.
- *Handshake version message.*
- *DNS Seeds.*
- *Sync.*


Nodo A
Nodo B
Version
Verack
Version
Verack


Nodo A
Nodo B
addr
getaddr
addr
addr
addr


Nodo A
Nodo B
getblocks
getblocks
inv
getdata
block
block

Módulo 3:
Minería y Consenso

Minería

- Asegura la cadena y habilita el consenso sin una autoridad central.
- Incentivo: Nuevas monedas acuñadas y tarifas de transacciones.
- Un minero valida transacciones y las registra en un bloque.
- Los mineros compiten para resolver un problema matemático basado en un algoritmo criptográfico.
- *Proof of work*.

Minería

- Suministro de bitcoin es creado a través de la minería.
- *Halving.*
- Ganancias: *fees* vs bitcoin acuñado.

Minería: Economía Bitcoin

- Un bloque se genera aproximadamente cada 10 minutos.
- 210000 bloques o 4 años aprox para que el proceso de *halving* se haga efectivo (Reducción al 50% de la emisión de bitcoin por bloque).
- Bitcoin: Economía deflacionaria.

Minería

- Validación independiente de transacciones.
- Agregación independiente de transacciones en nuevos bloques.
- Verificación de nuevos bloques.
- Selección independiente de la cadena con más computación acumulada.

Políticas

- Conjunto de reglas de validación que se aplican adicional al consenso sobre transacciones no confirmadas
- No afecta las transacciones en bloques.
- Uso de recursos, heurísticas, reglas

Combiance Transaction

- Transacción especial que contiene la recompensa por el proceso de minado.
- Solo tiene un input y un output.

Recompensas y Fees

- Existen 2 tipos de recompensas:
  - Añadir un nuevo bloque a la cadena principal
  - Sumando las tarifas(fees) de las transacciones añadidas en el bloque
    - Tarifas totales = Suma(salidas) + Suma(entradas)

Estructura Coinbase Transaction Input

Campo	Descripción	Tamaño (En bytes)
Transaction Hash	Todos sus bits son cero, no tiene referencia a un hash de transacción	32
Output index	Todos los bits son unos: 0xFFFFFFFF	4
Coinbase Data Size	Longitud de la información, de 2 a 100 bytes	1-9 bytes
Coinbase Data	Información usada para un nonce extra y etiquetas de minería	Variable
Sequence number	0xFFFFFFFF	4

Estructura Transaction Input

Campo	Descripción	Tamaño (En bytes)
Transaction Hash	Referencia a la transacción que contiene la UTXO a usarse	32
Output index	El índice de la UTXO	4
Unlocking-Script Size	Longitud del script, de 2 a 100 bytes	1-9 bytes
Unlocking-Script	Script que satisface con las condiciones del script que bloquea la UTXO	Variable
Sequence number	0xFFFFFFFF	4

Construyendo el encabezado

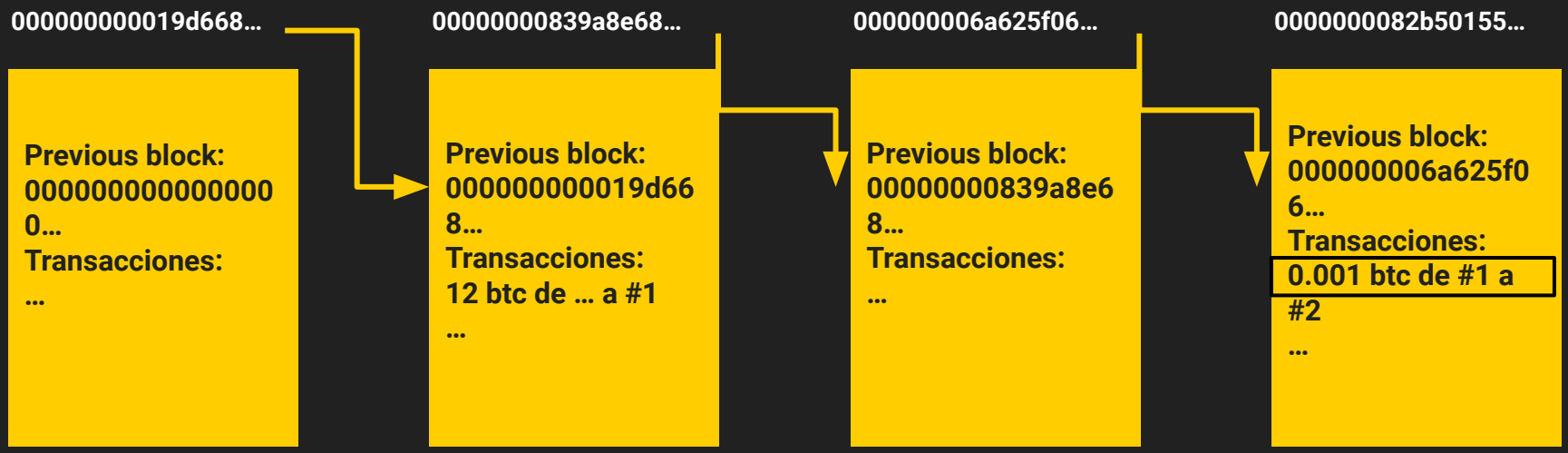
- Version
- Previous block hash
- Merkle Tree
- Timestamp
- Target
- Nonce

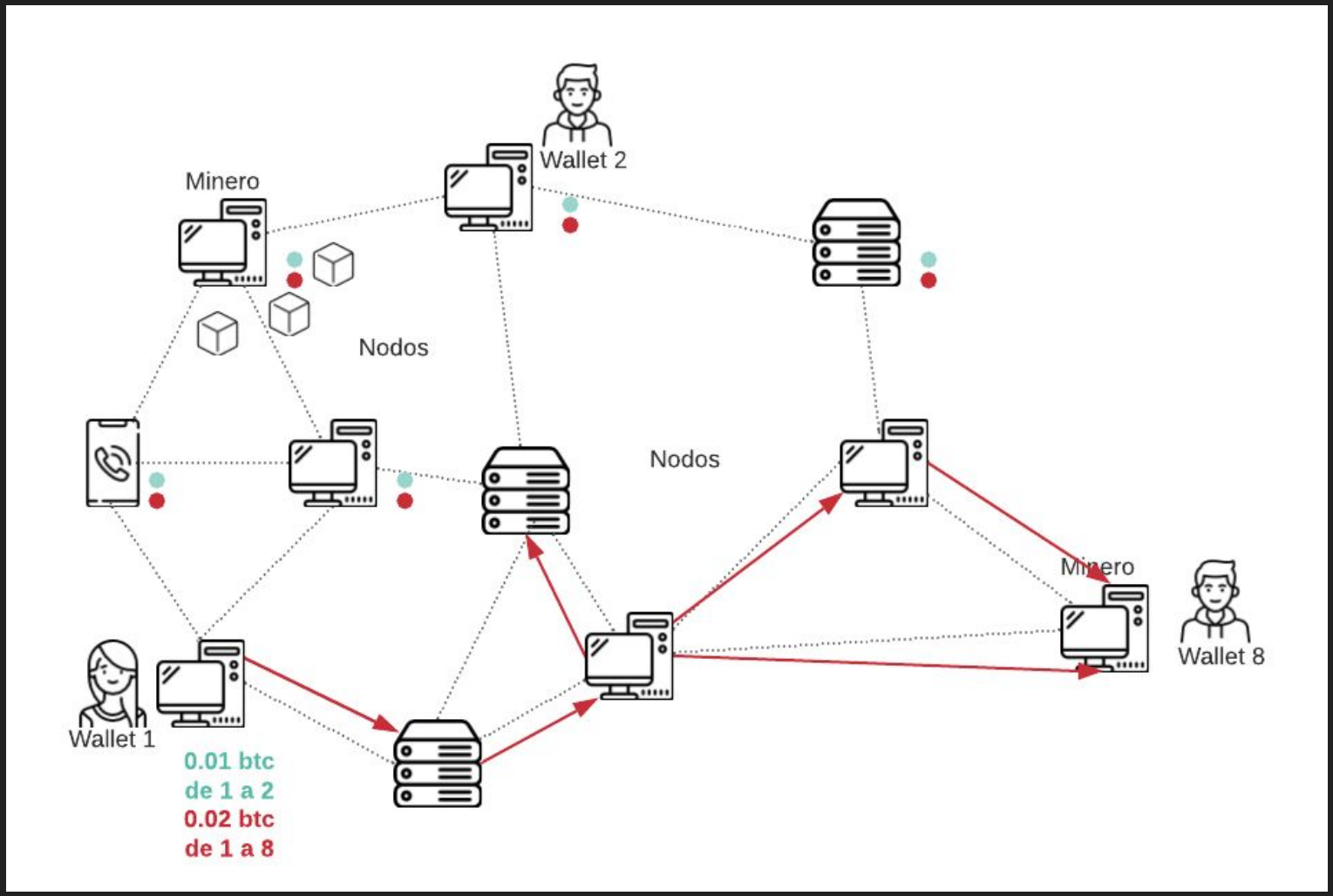
Minando un bloque

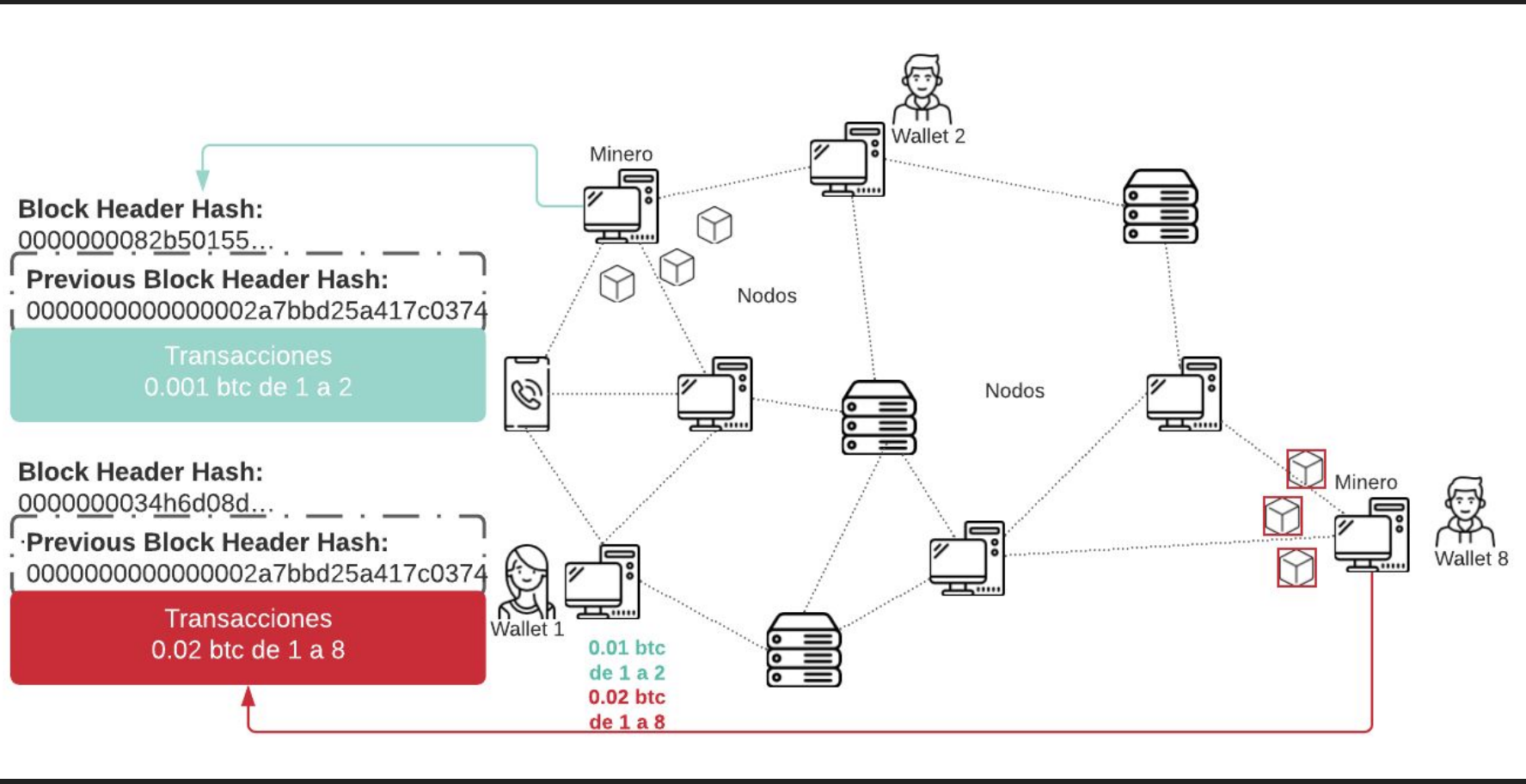
- Pasar la cabecera de un bloque a través de una función de hashing hasta encontrar un valor válido.
- Proof-of-work:
  - SHA 256.
  - Producir una salida que sea igual o menor que el objetivo.
  - Objetivo alto, menor dificultad.
  - Objetivo bajo, mayor dificultad.
  - ¿Cuántos de los bits iniciales deben ser cero?

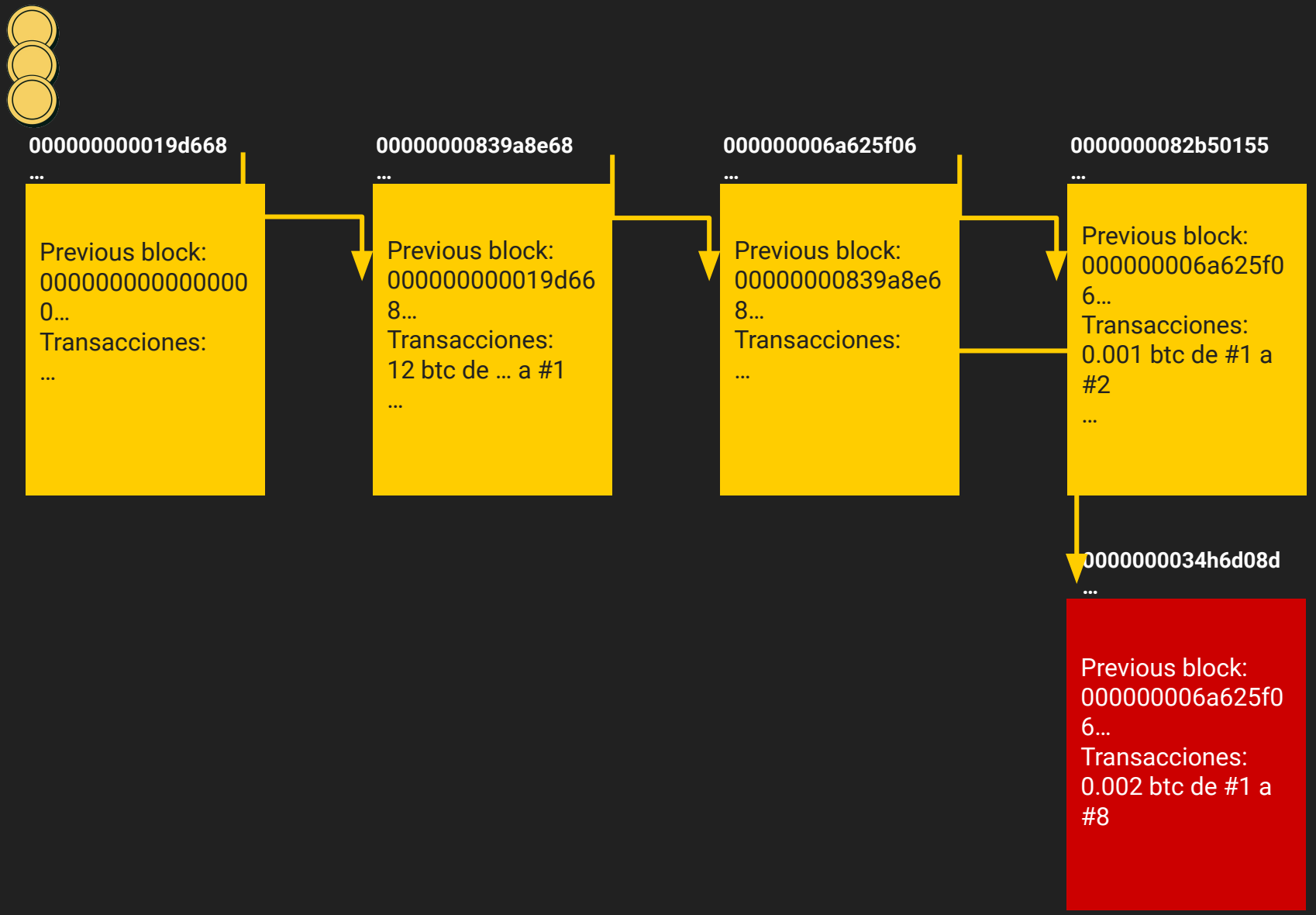
Proof of Work

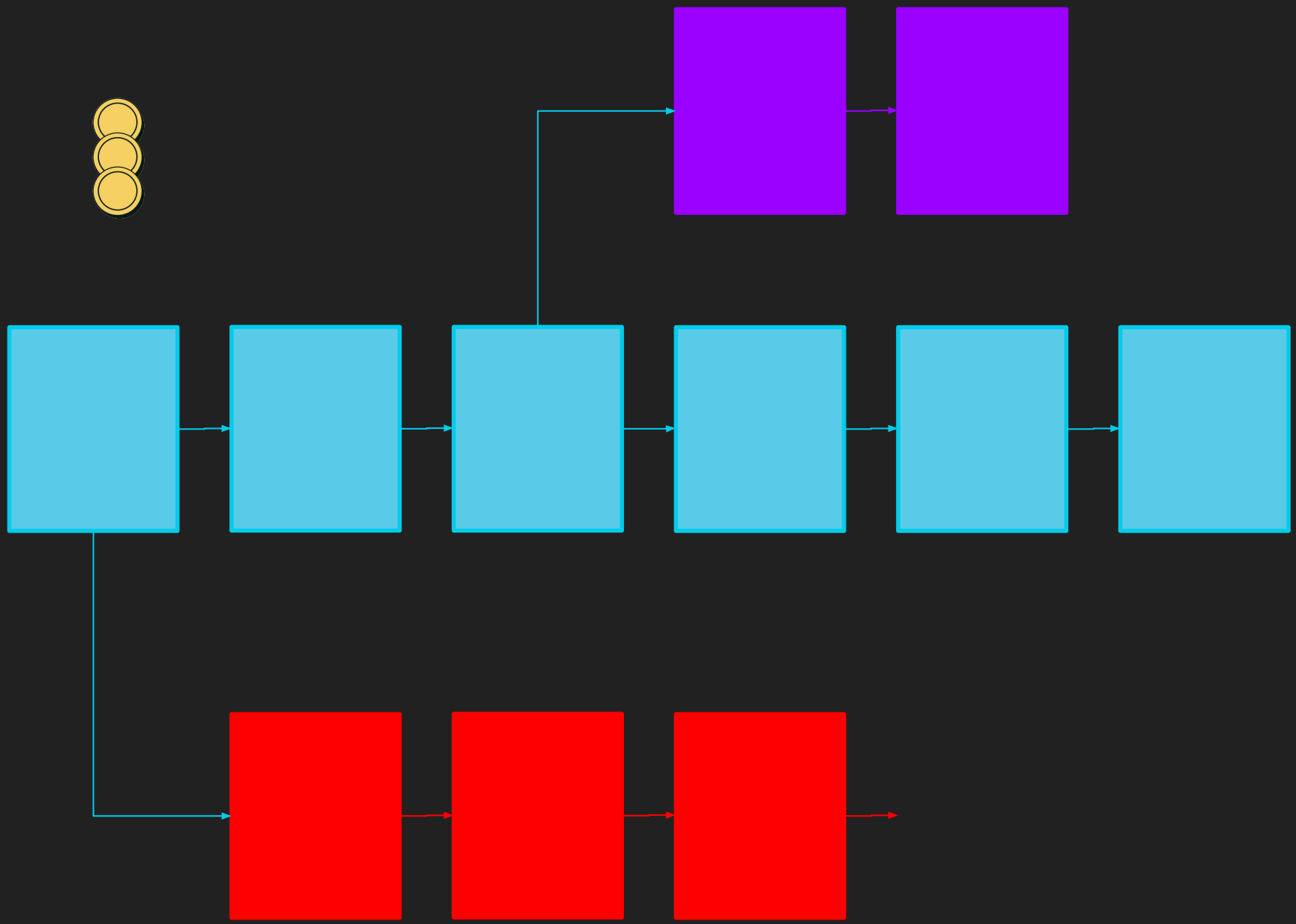
Forks


000000000019d668...
Previous block:
0000000000000000...
Transacciones:
...
00000000839a8e68...
Previous block:
000000000019d668...
Transacciones:
12 btc de ... a #1
...
000000006a625f06...
Previous block:
00000000839a8e68...
Transacciones:
...
0000000082b50155...
Previous block:
000000006a625f06...
Transacciones:
0.001 btc de #1 a #2
...


Wallet 2
Minero
Nodos
Nodos
Minero
Wallet 8
Wallet 1
0.01 btc
de 1 a 2
0.02 btc
de 1 a 8


Block Header Hash:
0000000082b50155...
Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
Transacciones
0.001 btc de 1 a 2
Block Header Hash:
0000000034h6d08d...
Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
Transacciones
0.02 btc de 1 a 8
Minero
Wallet 2
Nodos
Nodos
Minero
Wallet 8
Wallet 1
0.01 btc
de 1 a 2
0.02 btc
de 1 a 8


000000000019d668
...
Previous block:
0000000000000000...
Transacciones:
...
00000000839a8e68
...
Previous block:
000000000019d668...
Transacciones:
12 btc de ... a #1
...
000000006a625f06
...
Previous block:
00000000839a8e68...
Transacciones:
...
0000000082b50155
...
Previous block:
000000006a625f06...
Transacciones:
0.001 btc de #1 a #2
...
0000000034h6d08d
...
Previous block:
000000006a625f06...
Transacciones:
0.002 btc de #1 a #8



Modificando las reglas de consenso

- Hard forks
  - Cambio en las reglas de consenso.
  - Software, network, mining, chain.

- Soft forks
  - Cambio compatible con las reglas de consenso que permiten seguir operando.

Reto 3:

Investiga sobre el ataque del 51%, ¿crees que sería posible para un actor malicioso lograr esto sobre la red actual de Bitcoin?

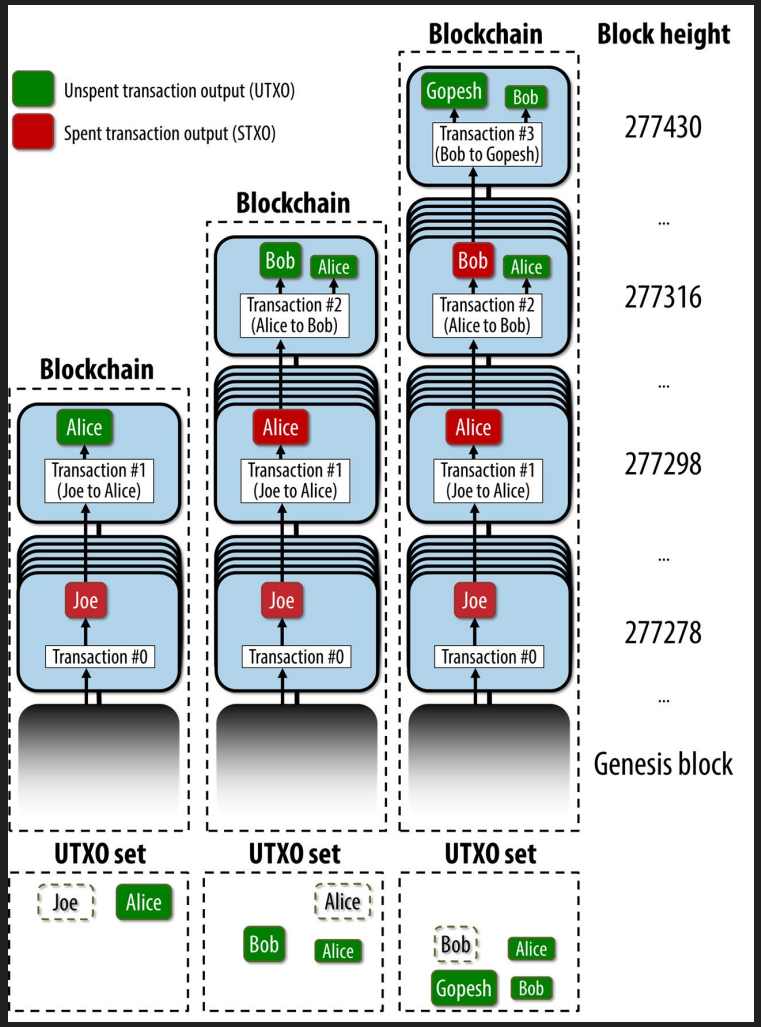
Módulo 4: Transacciones

¿Qué son?

- Parte más importante del sistema.
- Estructuras de datos que cifran/codifican la transferencia de valor entre participantes del sistema.

Inputs y Outputs

- Output:
  - Trozos/Porciones/Fracciones de bitcoin, registrados y reconocidos como válidos por toda la red.
  - UTXO.
- Input:
  - Identifican que UTXO se consumirá y provee una prueba la propiedad de la misma a través de un script.


Unspent transaction output (UTXO)
Spent transaction output (STXO)
Blockchain
Block height
Gopesh
Bob
Transaction #3
(Bob to Gopesh)
277430
...
Bob
Alice
Transaction #2
(Alice to Bob)
277316
...
Alice
Transaction #1
(Joe to Alice)
277298
...
Joe
Transaction #0
277278
...
Genesis block
UTXO set
Joe
Alice
Bob
Gopesh

Outputs

- Se componen de:
  - Una cantidad de bitcoin, representada en satoshis (100.000.000 sat = 1 btc).
  - Un enigma/acertijo/problema criptográfico que determina las condiciones para gastar/usar la salida.

Detalle output


```
  "vout": [
    {
      "value": 0.01500000,
      "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": 0.08450000,
      "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
    }
  ]
}
```

Inputs

- Se componen de:
  - ID de transacción, referenciando la transacción que contiene la UTXO que se va a usar.
  - Un índice de salidas (vout) identificando que UTXO de esa transacción está referenciada.
  - Un *scriptSig* que satisface las condiciones existentes en la UTXO para desbloquear y gastar.
  - Un número de secuencia.

Detalle Input


```
"vin": [
  {
    "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
    "vout": 0,
    "scriptSig" : "3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6ac
    "sequence": 4294967295
  }
],
```

```
{
  "value": 0.01500000,
  "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
},
```

Detalle Scriptsig

- Contiene:
  - Signature Size
  - Signature
  - Public Key Size
  - Public Key

Script

- Lenguaje de script para transacciones bitcoin.
- El script que bloquea una UTXO junto con el script que desbloquea se ejecuta para verificar si satisface las condiciones.
- Turing incompleto.
- Basado en pila.

Scripts de transacciones

- Script de bloqueo
  - Condición para consumir una salida.
  - scriptPubKey.
- Script de desbloqueo
  - Script que satisface las condiciones para consumir una salida.
  - scriptSig.

Fees

- Compensación a los mineros por asegurar la red.
- Calculadas basadas en el tamaño de la transacción, no su valor.
- Priorización.

Añadiendo *fees* a las transacciones

- Transacciones no tienen un campo para *fees*
- *Fees* están implícitas como la diferencia entre la sumatoria de inputs y la sumatoria de outputs
- Fees = Sum(inputs) – Sum(outputs)

Módulo 5:
Llaves y direcciones

Asymmetric-key cryptography

1. Plain text

2. Encryption
With public key (wand)

Hocus pocus

3. Cipher text

4. Decryption
with private key

Princess kiss

5. Plain text

In asymmetric key cryptography, encryption and decryption are done with different keys

Llaves

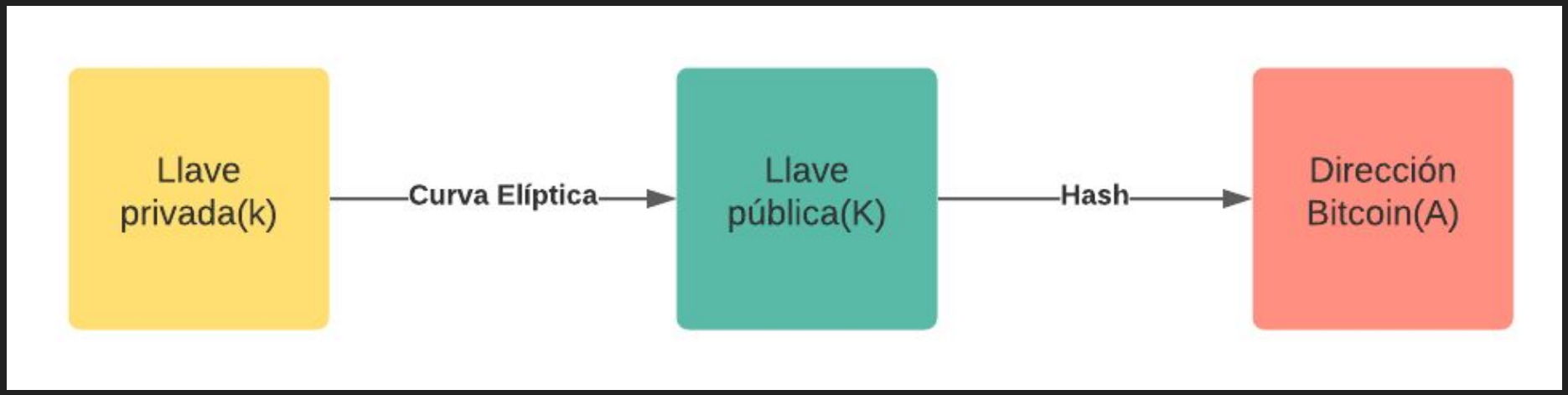
- Propiedad de bitcoin se establece a través de llaves, direcciones y firmas.
- Las llaves en una billetera son independientes del protocolo Bitcoin.
- Billeteras solo contienen llaves.
- Llaves vienen en pares: privada y pública.

Direcciones

- La llave pública del destinatario es representado por una Bitcoin *address*.
- Los destinos pueden ser flexibles, ej: Un script representado en una dirección.
- La dirección es parte de lo que se comparte al mundo.

Llaves públicas y privadas

- Bitcoin usa Curva Elíptica para criptografía.
- Llave pública: recibir fondos.
- Llave privada: firmar transacciones para utilizar fondos.
- Criptografía asimétrica.


Llave privada(k)
Curva Elíptica
Llave pública(K)
Hash
Dirección Bitcoin(A)

Llaves privadas

- Número escogido al azar.
- Debe permanecer secreta.
- La llave pública se genera de la llave privada.
- Es necesario que la generación del número sea realmente aleatoria.
- CSPRNG.

Llaves públicas

- Calculada de la llave privada usando multiplicación de curva elíptica.
- Criptografía de curva elíptica.
- Secp256k1.
- Una llave privada puede convertirse en una pública pero no en la dirección opuesta.

Direcciones

- String alfanumérico que puede ser compartido con cualquiera
- Representa al dueño de un par de llaves o un script de pago.
- Una dirección no es una llave pública, la primera se deriva de la segunda usando una función de una vía

Formatos

- Tanto llaves públicas como privadas se pueden representan en distintos formatos.
- Agregan facilidad para que las personas puedan leer y transcribir llaves.

Formatos: Llaves privadas

Tipo	Prefijo	Descripción	Ejemplo
Raw	N/A	32 bytes	0C28FCA386C7A227600B2FE50B7CAE11EC86D3BF1FBE471BE89827E19D72AA1D
Hex	N/A	64 dígitos Hexadecimales	1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd
WIF	5	Base58Check con prefijo 0x80	5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
WIF-compressed	K o L	Igual que el anterior + sufijo 0x01	KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ

Formatos: Llaves públicas

- Comprimida
- Sin comprimir

Reto 4:

Ejecuta el script que dejaremos en esta clase usando python, verás la implementación de llaves y direcciones, cuéntanos en los comentarios cómo te fue y qué diferencias ves entre las direcciones

https://github.com/bitcoinbook/bitcoinbook/blob/develop/code/key-to-address-ecc-example.py

Módulo 6: Billeteras

¿Qué son?

- Medio digital que permite recibir, enviar y usar un activo digital.
- Puede ser software o hardware.
- Billeteras solo contienen llaves.

- Generar par de llaves
  - Llave privada, llave pública

- Firma transacción con llave privada
  - Transacción, llave privada -> firma

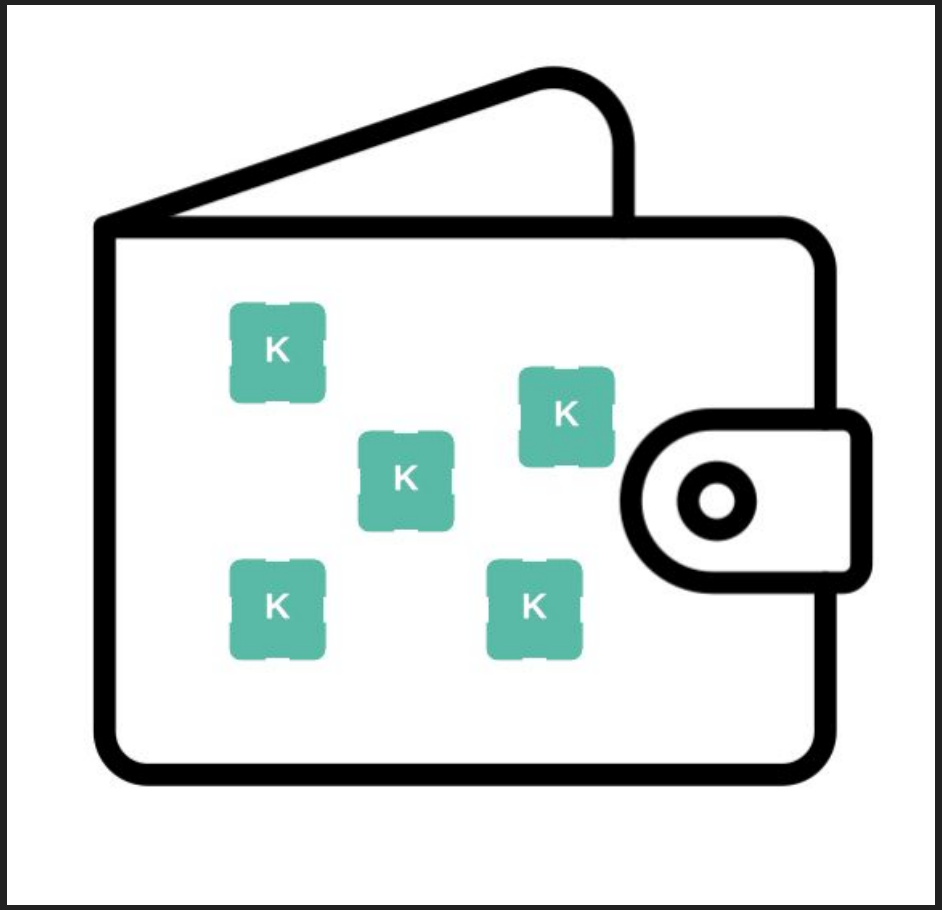
- Validar firma con llave pública
  - Transacción, llave pública, firma -> Válido/inválido

Tipos de billetera

- *Nondeterministic wallet*
- *Deterministic wallet*

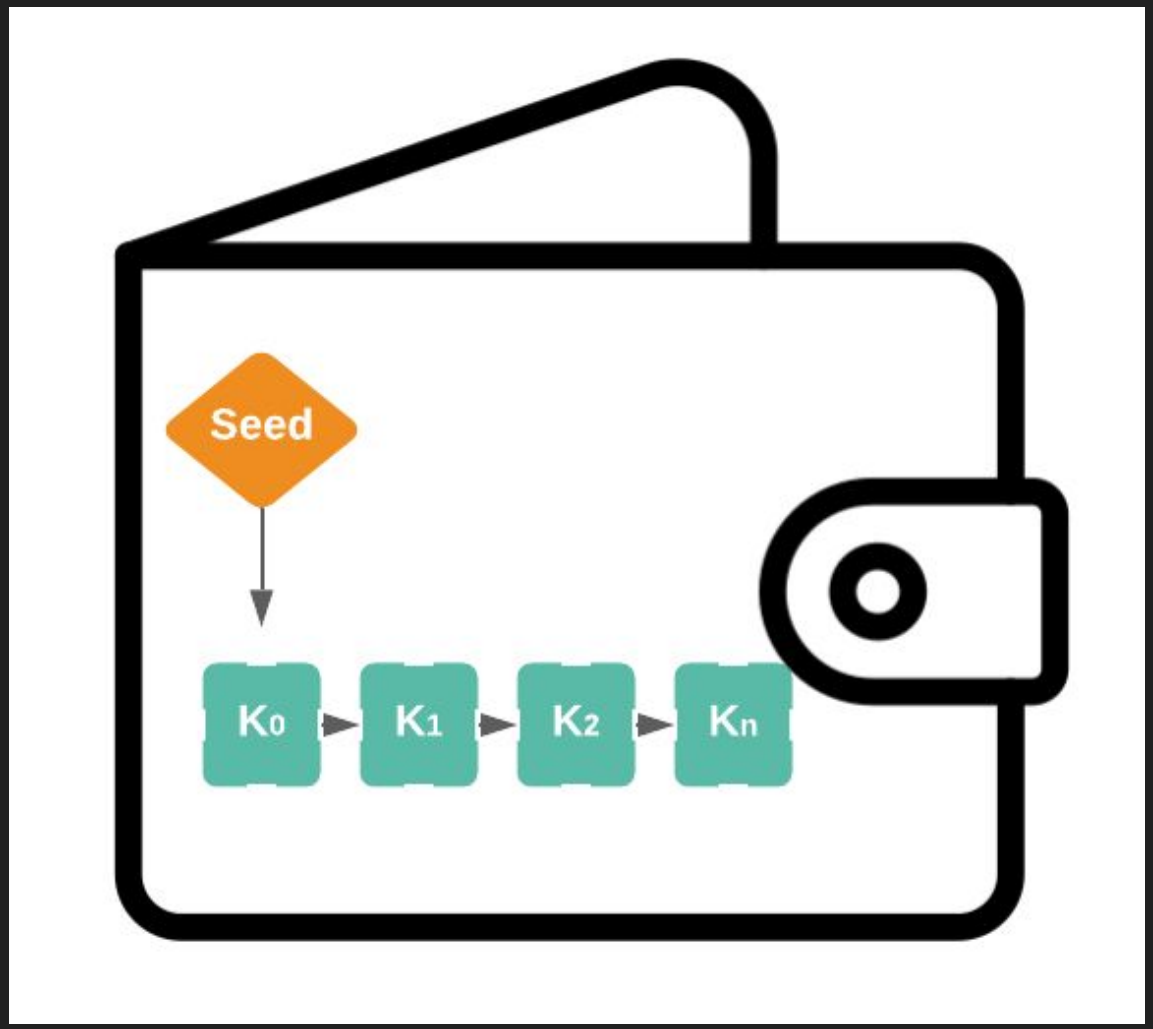
Nondeterministic Wallet

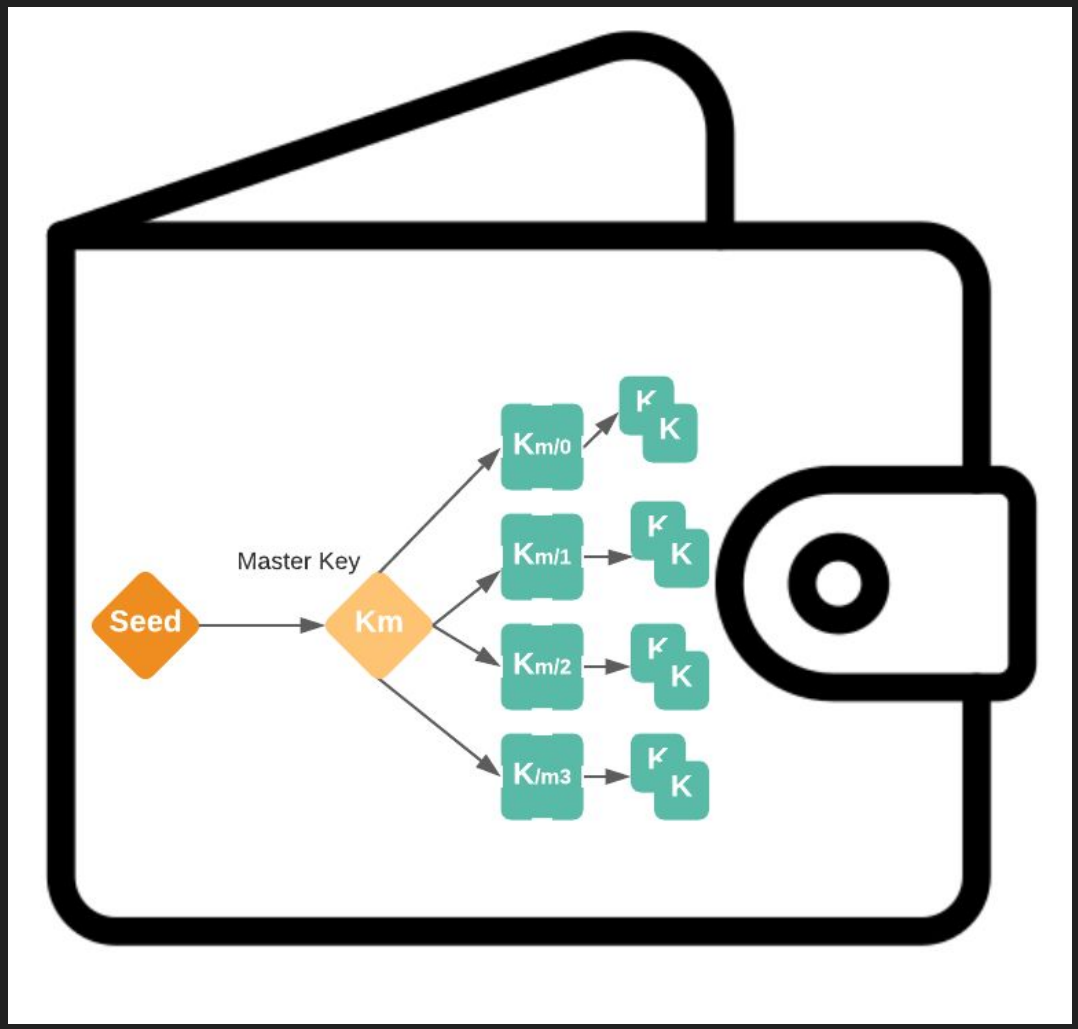
- Cada llave se genera de manera a través de un número aleatorio
- Las llaves no se relacionan una con otra


K
K
K
K
K

Deterministic Wallet

- Todas las llaves se generan a partir de una llave maestra, conocida como *seed*.
- Las llaves están relacionadas unas con otras y pueden ser generadas nuevamente con el *seed* original.
- *Mnemonic code words*.


Seed
K0
K1
K2
Kn


Seed
Master Key
Km
Km/0
Km/1
Km/2
K/m3
K
K
K
K
K
K
K
K

Códigos mnemónicos

- Creación de un seed a partir de una secuencia de palabras en inglés
- Facilidad para transcribir, exportar e importar.

Tipo	Ejemplo
Seed como hex	0C1E24E5917779D297E14D45F14E1A1A
Seed de un mnemónico de 12 palabras	army van defense carry jealous true garbage claim echo media make crunch

Reto 5:

Con todo lo aprendido hasta el momento, descarga Simple Bitcoin Wallet https://play.google.com/store/apps/details?id=com.btcontract.wallet&hl=es_CO&gl=US y crea tu primer billetera