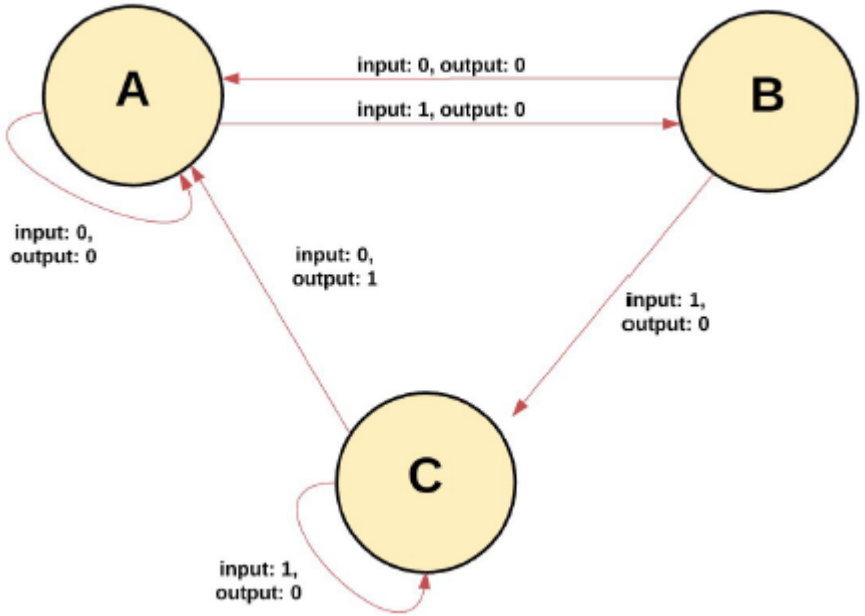
La Blockchain de Ethereum

La Blockchain de Ethereum (1)

- Es diferente a la estructura de Bitcoin
  - Se parecen → Cada nodo almacena una copia de la Blockchain
- Diferencias
  - Transaction-based State Machine
    - Los nodos de Ethereum almacenan el estado (state) más reciente de cada Smart Contract, además de las transacciones de ether.
  - La estructura interna
  - El algoritmo de minado (Ethash)

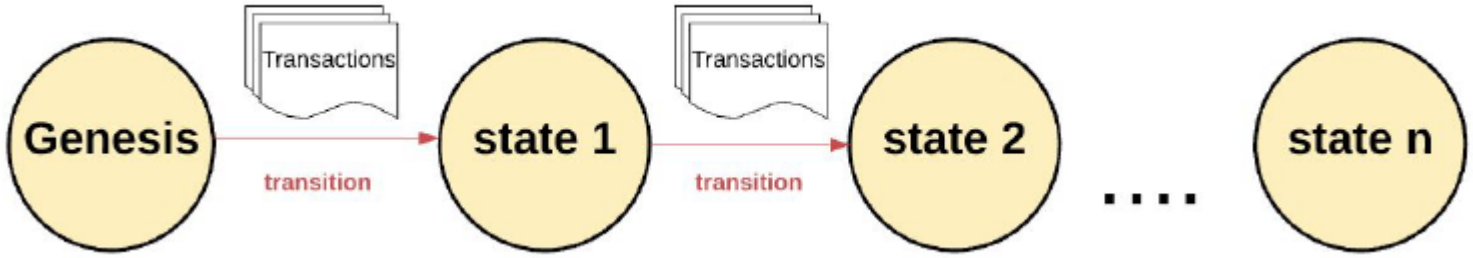
La Blockchain de Ethereum (2)

- ¿Transaction-based State Machine? (1)
  - Funciona como una máquina de estados
  - Es decir una máquina que lee una serie de inputs y que, en base a dichos inputs, pasa a otro estado.



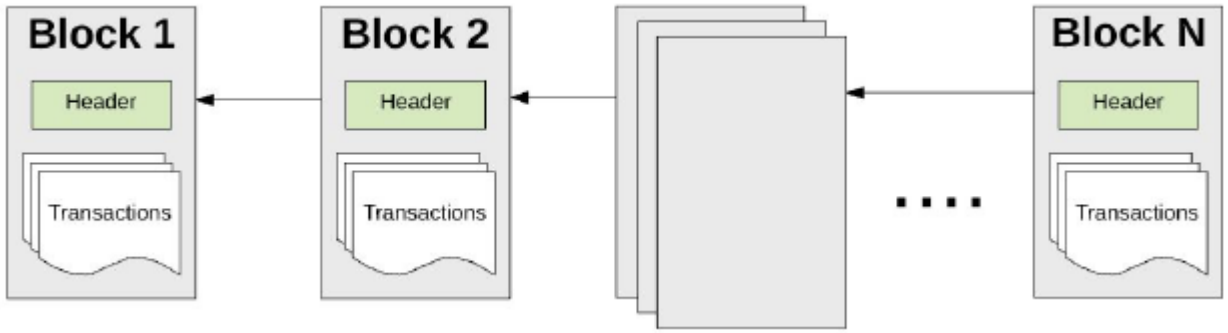
La Blockchain de Ethereum (3)

- ¿Transaction-based State Machine? (2)
  - Se empieza en un “genesis state” antes de cualquier transacción.
  - Cuando hay transacciones se “pasa” a otro estado final.
  - El estado final representa el “estado actual” de Ethereum.



La Blockchain de Ethereum (4)

- ¿Transaction-based State Machine? (3)
  - El “estado actual” tiene millones de transacciones que se agrupan en bloques.
  - Cada bloque tiene una serie de transacciones y está encadenado al bloque anterior.



La Blockchain de Ethereum (5)

- ¿Transaction-based State Machine? (y 4)
  - Para pasar de un estado a otro, la transacción debe ser validada.
  - Para ser validada la transacción pasa el proceso de minado.
  - Cualquier nodo de la red que se declare como minero puede intentar crear y validar un bloque (proof-of-work). El minero es recompensado con Ether.
  - Cada vez que se añade un nuevo bloque, se generan nuevos token Ethers.

La Blockchain de Ethereum (6)

- Estructura Interna (1)
  - Accounts
  - States
  - Transactions

La Blockchain de Ethereum (7)

- Estructura Interna (1) – Accounts (1)
  - Representan el estado actual
  - Son objetos que se comunican a través de un framework de mensajeria.
  - Se identifican con un address de 160 bits.
  - Cada account tiene un estado asociado (state).
  - Existen dos tipos:
    - Externally Owned Accounts (EOA).
    - Contract Accounts (CA).

La Blockchain de Ethereum (8)

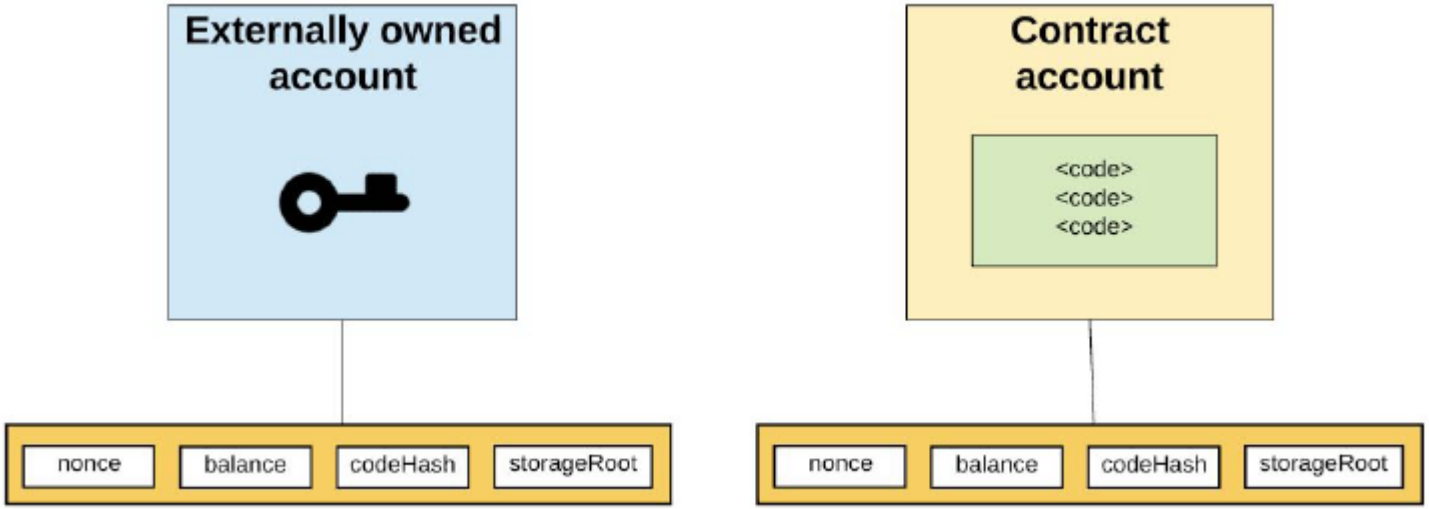
- Estructura Interna (2) – Accounts (y 2)
  - Las EOA puede enviar mensajes a otra EOA (transferencia de valor) o CA (activa / invoca el código del contrato) firmando una transacción con un private key.
  - CA no pueden iniciar nuevas transacciones por su cuenta.

La Blockchain de Ethereum (9)

- Estructura Interna (3) – States (1)
  - El estado (state) de la Account se compone de cuatro elementos:
    - Nonce:
      - Si EOA → representa el número de transacciones enviadas desde account address.
      - Si CA → número de contratos creados por la account.
    - Balance: Total de wei en propiedad del address. (1 ether == 1018 wei).
    - codeHash: hash del EVM code. (EOA == empty string).
    - storageRoot: hash del root node del Merkel Patricia Tree. Es en sí mismo un árbol.

La Blockchain de Ethereum (10)

- Estructura Interna (4) – States (2)
  - Gráficamente:



La Blockchain de Ethereum (11)

- Estructura Interna (5) – States (3)
  - El estado global consiste en mapear los accounts addresses y los account states en un Merkle Patricia Tree
    - Gran número de hojas al final con los datos
    - Nodos intermedios donde cada nodo es el hash de los nodos hijos
    - El root node se forma con el hash de los nodos hijos en la raíz del árbol

Ethereum Modified Merkle-Paricia-Trie System

An interpretation of the Ethereum Project Yellow Paper

G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.

Lee Thomas
Ver 0.0 2016-06-23

Block Header, H or B_H

stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied

Hash function:
KECCAK256()

Simplified World State, σ

Keys							Values
a	7	1	1	3	5	5	45.0 ETH
a	7	7	d	3	3	7	1.00 WEI
a	7	f	9	3	6	5	1.1 ETH
a	7	7	d	3	9	7	0.12 ETH

World State Trie

ROOT: Extension Node

prefix	*shared nibble(s)*	*next node*
0	a7	

Branch Node

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*a*	*b*	*c*	*d*	*e*	*f*	*value*
	●						●								●	

Leaf Node

prefix	*key-end*	*value*
2	1355	45.0ETH

Extension Node

prefix	*shared nibble(s)*	*next node*
0	d3	

Leaf Node

prefix	*key-end*	*value*
2	9365	1.1ETH

Branch Node

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*a*	*b*	*c*	*d*	*e*	*f*	*value*
			●						●							

Leaf Node

prefix	*key-end*	*value*
3□	7	1.00WEI

Leaf Node

prefix	*key-end*	*value*
3□	7	0.12ETH

Prefixes

0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

La Blockchain de Ethereum (13)

- Estructura Interna (7) – States (y 5)
  - La ventaja de usar Merkle Patricia Tree es
    - root node es criptográficamente dependiente de la información almacenada en el árbol
    - por tanto, el hash del root node puede ser utilizado como una prueba de identidad de los datos.
  - Para validar un dato, se puede realizar lo que se denomina Merkle Proof. Requiere:
    - El dato a validar
    - El root node hash
    - El branch (que son los hashes desde el dato al root node).

La Blockchain de Ethereum (13)

- Estructura Interna (8) – Transacciones
  - nonce: cuenta del número de transacciones del sender.
  - gasPrice: número de wei que el sender está dispuesto a pagar.
  - gasLimit: cantidad máxima de gas que el sender pagaría.
  - to: recipient account address.
  - value: cantidad de wei que sender envía al recipient.
  - v, r, s: usados para generar la firma del sender.

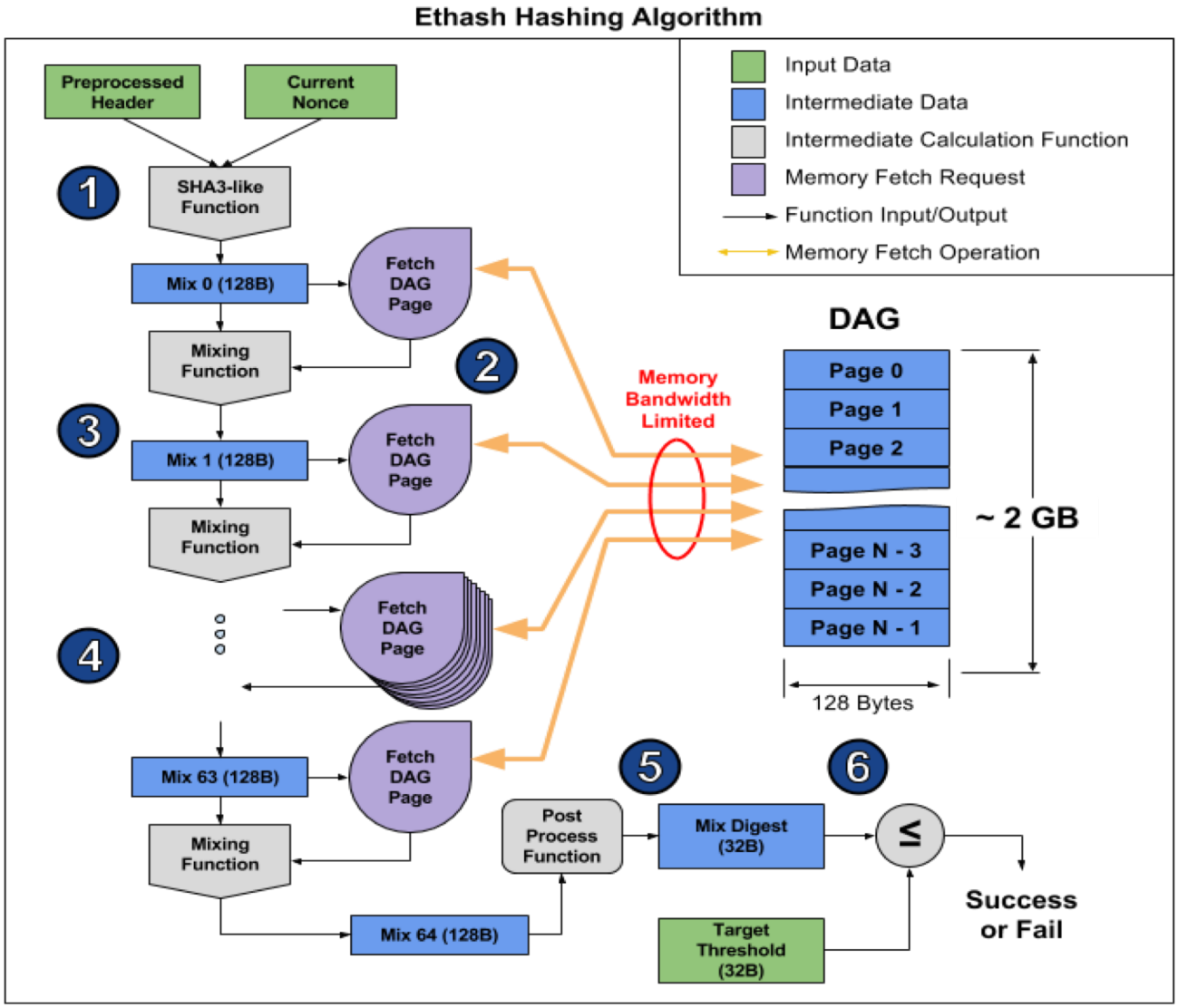
Ethash (1)

- Es un algoritmo de tipo PoW
- Es una variante del algoritmo Dagger (Vitalik Buterin) Hashimoto (Thaddeus Dryja)
- Necesita una estructura de datos de 1Gb para operar (el DAG)
  - Se genera al iniciar Ethereum por primera vez
  - Se actualiza cada N bloques (30000 bloques)
  - Tiene un número mágico al principio aleatorio
  - Es una tabla de n x 4 bytes (n = 16777186)

Ethash (2)

- Objetivos que persigue:
  - Ser resistente al uso de ASIC
  - Los bloques deben de ser verificables muy rápidamente por clientes ligeros
  - El objetivo es calcular un nonce usando:
    - la información de la cabecera del bloque
    - Subconjuntos aleatorios de la información del DAG (uso intensivo de E/S y de RAM) + SHA3

Ethash (y 3)


Ethash Hashing Algorithm
Input Data
Intermediate Data
Intermediate Calculation Function
Memory Fetch Request
Function Input/Output
Memory Fetch Operation
Preprocessed Header
Current Nonce
1
SHA3-like Function
Mix 0 (128B)
Fetch DAG Page
Mixing Function
2
3
Mix 1 (128B)
Fetch DAG Page
Mixing Function
4
Fetch DAG Page
Mix 63 (128B)
Fetch DAG Page
Mixing Function
Mix 64 (128B)
Memory Bandwidth Limited
DAG
Page 0
Page 1
Page 2
Page N - 3
Page N - 2
Page N - 1
~ 2 GB
128 Bytes
5
Post Process Function
Mix Digest (32B)
6
≥
Target Threshold (32B)
Success or Fail