

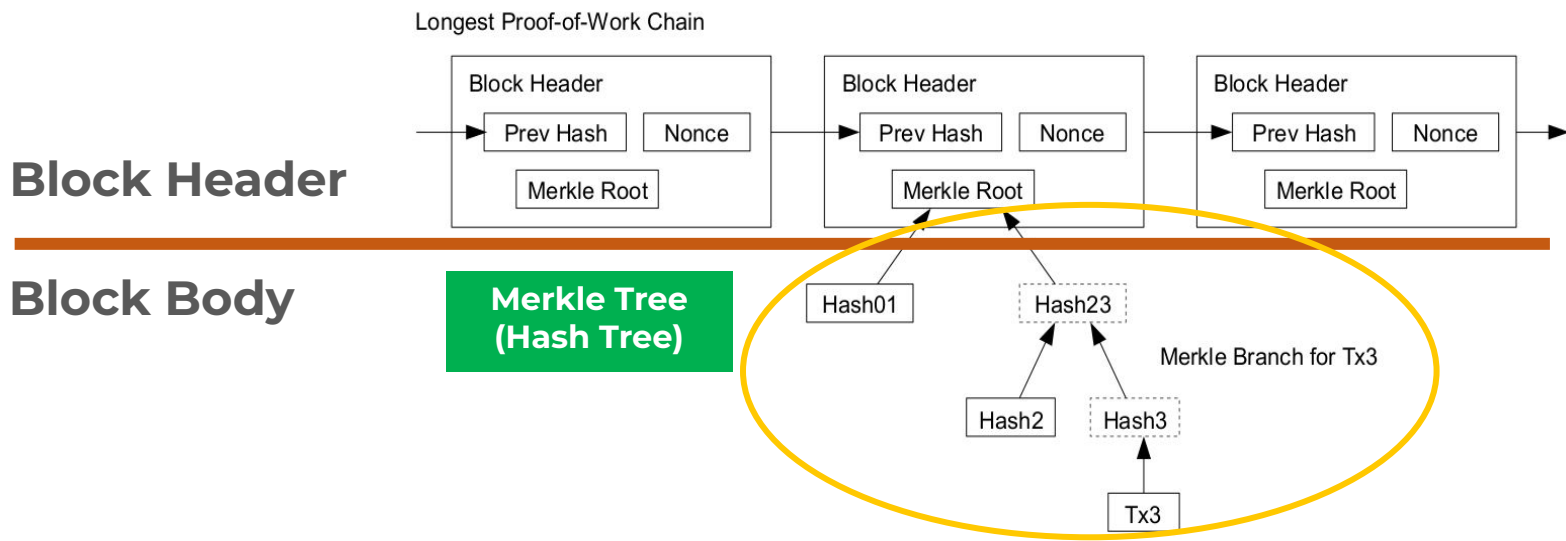
240419

Block Chain Demo

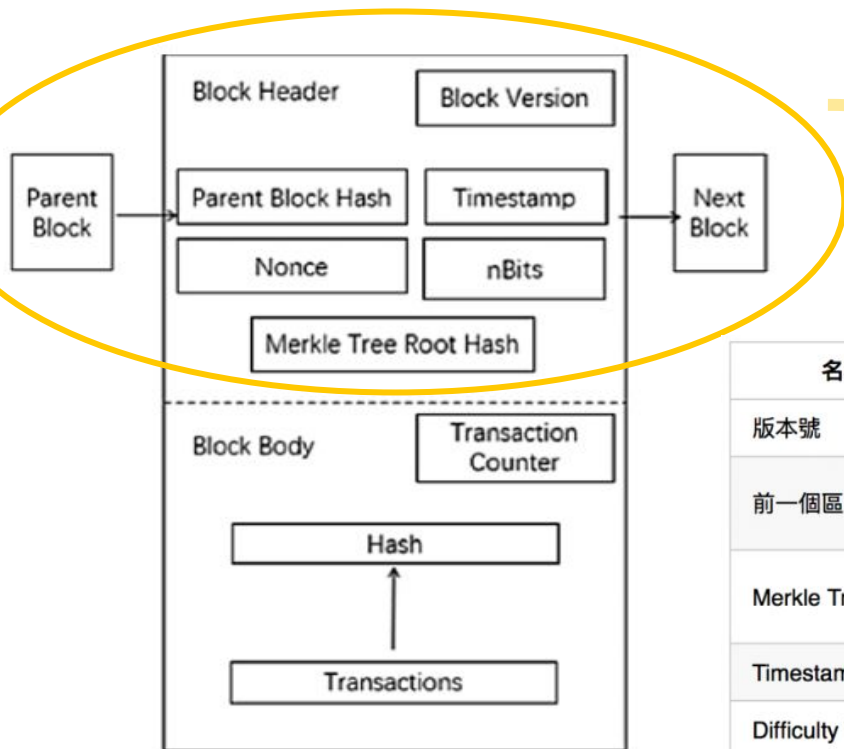
ISC Yu-Jen

Block Chain

- 區塊鏈本質上為一個去中心化的資料庫



Block Header

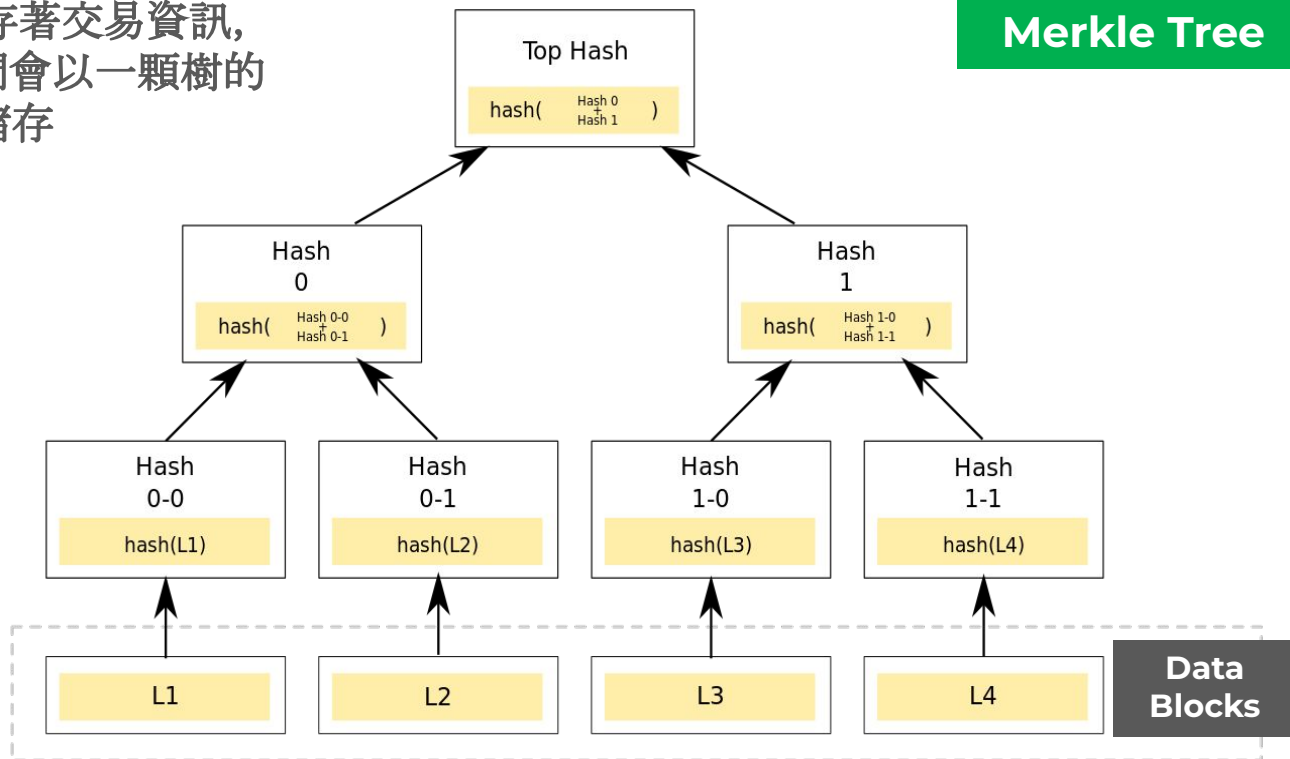


| 名稱 | 說明 | 大小 |
|-------------------|---|----------|
| 版本號 | 區塊數據的版本號，主要用於跟踪軟件或協議升級 | 4 Bytes |
| 前一個區塊的記錄 | 從上一個區塊的區塊頭所計算出來的Hash值 (Hash Of Previous Block Header) | 32 Bytes |
| Merkle Tree Root | 記錄了當前區塊中所有交易經由Merkle Tree演算法所算出來的Merkle樹根節點的Hash值 | 32 Bytes |
| Timestamp | 當前區塊的生成時間戳 (Unix時間格式) | 4 Bytes |
| Difficulty Target | 該區塊的工作量證明算法的困難值 | 4 Bytes |
| Nonce | 表示工作量證明演算法進行的次數 | 4 Bytes |

Block Body

Block Body 儲存著交易資訊，
在 Block 中他們會以一顆樹的
資料結構進行儲存

Merkle Tree

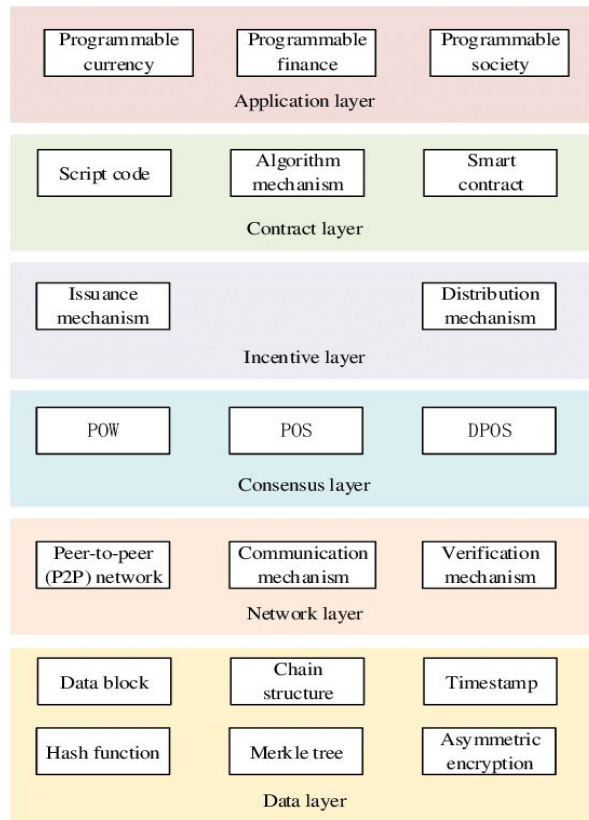


四筆交易

區塊鏈架構

區塊鏈上每一次的**區塊狀態變化**都稱為**交易**
(一個交易對應到一個雜湊 值、一段時間後會對交易進行打包)

- 由多個節點成一網路 (Peer-to-Peer)
- 當一個節點發起一筆交易時，會先將交易 **廣播** 給其它節點，透過 **共識演算法** 來決定節點們取得共識的方式
- 每次記帳由被 **共識演算法挑選** 的節點負責。該節點會將**驗證過**的交易寫進區塊鏈中，並廣播通知其他節點，其它節點則會依照 **預先定義的標準** 進行**校驗**
- 若**驗證成功**，則建立儲存池暫存資料，並 **轉發** 給附近的節點。若**驗證失敗**，則立即 **丟棄** 該資料 (確保無效資料不會繼續傳播)
- 當資料一旦被寫進區塊鏈中，便 **無法再被竄改**。這樣的機制讓 **所有節點共同來維護** 一帳本，即代表一但有人修改區塊鏈 內帳本記錄，也會被發現



區塊鏈共識

PoW

- 節點間公平競爭
- 達到完全去中心化
- 容易驗證
- 取得共識時間較長
- 浪費算力

PoS

- 耗能少
- 取得共識時間較短
- 攻擊成本高
- 持幣趨於集中化
- 幣的流動性差

DPoS

- 記帳效率更高
- 獎勵的分配更平均
- 存在一定的中心化控制

智能合約介紹

在區塊鏈上自動執行的腳本程式(Script)

- 去中心化 (Decentralization)

- ➡ 智慧合約在區塊鏈網路中運行，不依賴單一的節點或第三方公證機構

- 不可篡改 (Immutability)

- ➡ 一旦部署到區塊鏈上，智慧合約的規則 (邏輯)就沒辦法被修改

- 透明性 (Public)

- ➡ 智慧合約的規則和交易紀錄對所有網路的節點皆為公開透明

- 自動執行 (Automatic)

- ➡ 一旦智慧合約的預設條件被觸發，合約將會自動執行，無需人工干預


Gas vs Wei

- Ethereum 網絡中, **wei** 和 **gas** 是兩個不同類型的單位
- Wei
 - ➡ Ethereum 的基本貨幣單位, 通常用來表示資產或價值
 - ➡ $1 \text{ Ether} = 10^{18} \text{ wei}$
- Gas
 - ➡ 用於衡量交易或是智慧合約執行所需的工作量
 - ➡ 每種操作, 都會有一個預先定義好的 Gas 成本
- 發起一個交易或調用一個智慧合約中的方法時, 會先指定一個 **gas price**, 這個價格是用 **wei** 表示的, 換句話說就是「我願意為每個 **gas** 單位支付多少 **wei**」

ABI (Application Binary Interface)

- ABI 是智慧合約與外界交互的一個界面
- 定義了如何調用智慧合約的函數, 以及這些函數會如何返回 Data
- 內容包括合約的參數名稱、參數類型、傳回類型等等
- 通常都是以 JSON 格式儲存

```
1  pragma solidity >=0.8.2 <0.9.0;
2
3  contract Storage {
4
5      uint256 number;
6
7      function store(uint256 num) public { 22520 gas
8          number = num;
9      }
10
11     function retrieve() public view returns (uint256){ 2415 gas
12         return number;
13     }
14 }
15
```




```
[
  {
    "inputs": [],
    "name": "retrieve",
    "outputs": [
      {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "uint256",
        "name": "num",
        "type": "uint256"
      }
    ],
    "name": "store",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  }
]
```

Binary Code

- 一串二進位的代碼，非常低階，基本上是給電腦看的，用以實現合約中的所有邏輯與函數
- 代表合約的實際邏輯和規則，會被寫入到區塊鏈上
- 一旦寫入，除非合約設計有自毀或升級機制，否則這個代碼是不可改變的

```
1  pragma solidity >=0.8.2 <0.9.0;
2
3  contract Storage {
4
5      uint256 number;
6
7      function store(uint256 num) public { 22520 gas
8          number = num;
9      }
10
11     function retrieve() public view returns (uint256){ 2415 gas
12         return number;
13     }
14 }
15
```



```
6080604052348015600e575f80fd5b506101438061001c5f395ff3fe6080604052
34801561000f575f80fd5b5060043610610034575f3560e01c80632e64cec11461
00385780636057361d14610056575b5f80fd5b610040610072565b60405161004d
919061009b565b60405180910390f35b610070600480360381019061006b919061
00e2565b61007a565b005b5f8054905090565b805f8190555050565b5f81905091
9050565b61009581610083565b82525050565b5f6020820190506100ae5f830184
61008c565b92915050565b5f80fd5b6100c181610083565b81146100cb575f80fd
5b50565b5f813590506100dc816100b8565b92915050565b5f6020828403121561
00f7576100f66100b4565b5b5f610104848285016100ce565b9150509291505056
fea26469706673582212201fbf53c09b44b94f77a29988279caf38059e4cfa72b1
fe0e5147a3e7dc63a4bd64736f6c63430008190033
```

genesis.json

- 用來配置生成我們私有鏈的創世區塊
- 代表合約的實際邏輯和規則，會被寫入到區塊鏈上

```
{
  "config": {
    "chainId": 666,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0
  },
  "alloc": {},
  "coinbase": "0x0000000000000000000000000000000000000000",
  "difficulty": "0x20000",
  "extraData": "",
  "gasLimit": "0xffffffff",
  "nonce": "0x0000000000000042",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp": "0x00"
}
```

THANKS

ISC Yu-Jen