

# Bitcoin & Blockchains

[PDF] Bitcoin: A peer-to-peer electronic cash system

S Nakamoto - Decentralized business review, 2008 - assets.pubpub.org

... To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet ...

☆ 儲存 引用 被引用 26020 次 相關文章 全部共 1306 個版本

(as of 2023.05.16)

# Bitcoin



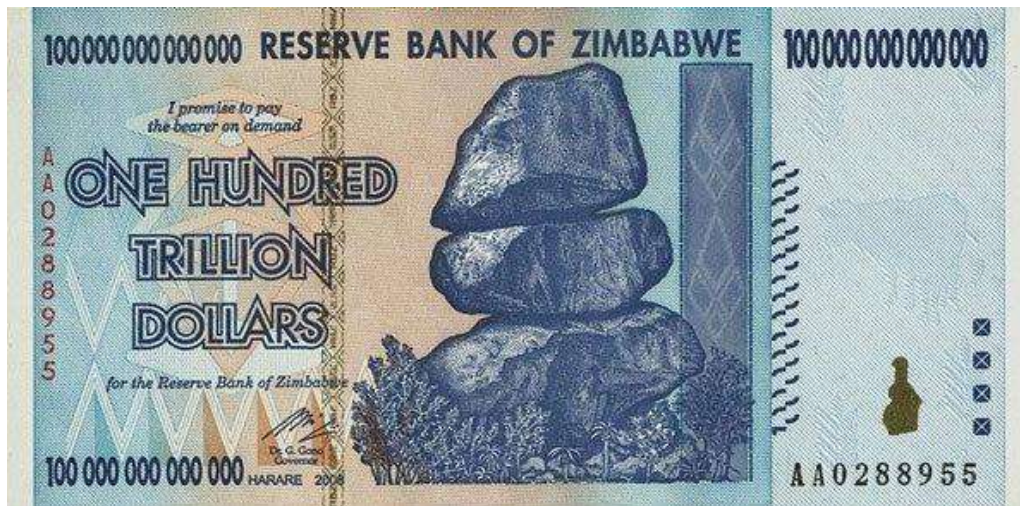
- ❑ A peer-to-peer payment **protocol, system, and digital currency** introduced as open source software in 2009 by Satoshi Nakamoto (中本聰), a pseudonym for a person or group of persons.
- ❑ A **cryptocurrency** (密碼貨幣) as it uses cryptography to control the creation and transfer of money.
- ❑ Property:
  - **Anonymous:** both money sender and receiver do not need to know each other; each party has just a sequence of meaningless number.
  - **Decentralized:** no central bank to control the issue and flow of BitCoins, nor does any centralized authority to monitor transactions.



# BitCoin 怎麼設計出來的？ 跟 Blockchains 有什麼關係？

# 法幣的問題

實體的錢 (現金) (法定貨幣, Fiat Money) 由國家的「**中央銀行**」發行並擔保，**維持貨幣穩定**。



Ca. 2015



舊臺幣 (source: [wiki](#)) (1946年發行，原本只發行1元、5元、10元的鈔券，但因大量發行造成惡性通膨，臺灣省政府在1949.06.15公布以舊臺幣4萬元折合新臺幣1元換發)

# BitCoin 設計的二個前提

- ❑ 不相信任何單一或少數的中央機制/中央銀行
- ❑ 需要具有現金特性的數位貨幣



# 實體貨幣(現金)特性

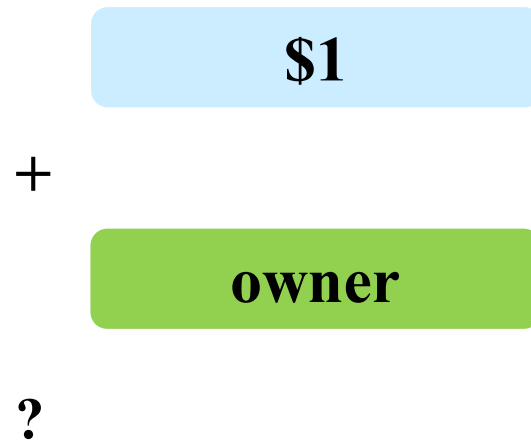
錢不認人，不能輕易偽造，一經交易，錢即易手。



虛擬貨幣如何能保有上述特性？



# 如何表示一個**虛擬貨幣** (Virtual Coin)?

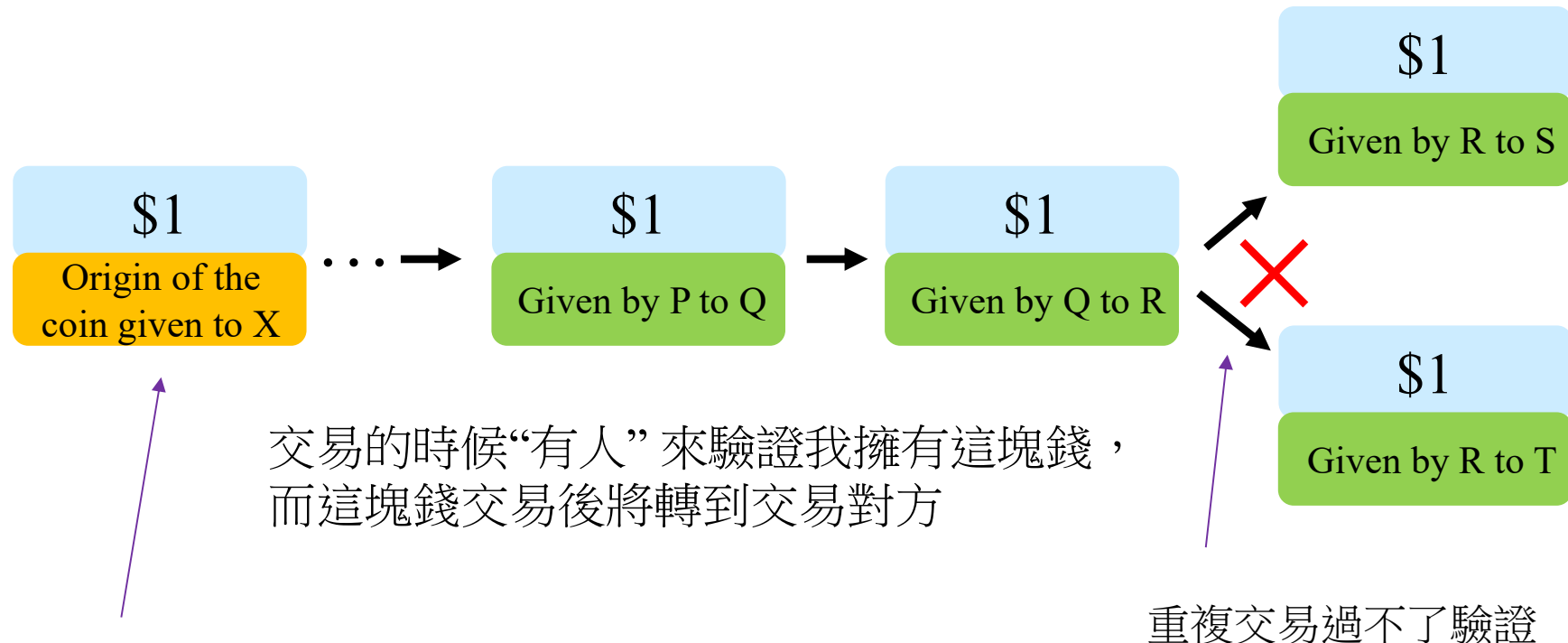


真正讓虛擬貨幣表示方式複雜化的主要因素在於如何能用這樣表示的虛擬貨幣來交易，同時還能讓整個虛擬貨幣系統維持法幣的主要特性：**無法輕易偽造(安全)**、**匿名**、**穩定的發行量**、**去中心化**

實體貨幣無法**同一時間重複使用(double spending)**，虛擬貨幣如何防範？

# 作法：讓每個幣(coin)可以追朔到源頭

且每筆交易需要驗證，紀錄保存下來，同時避免重複交易

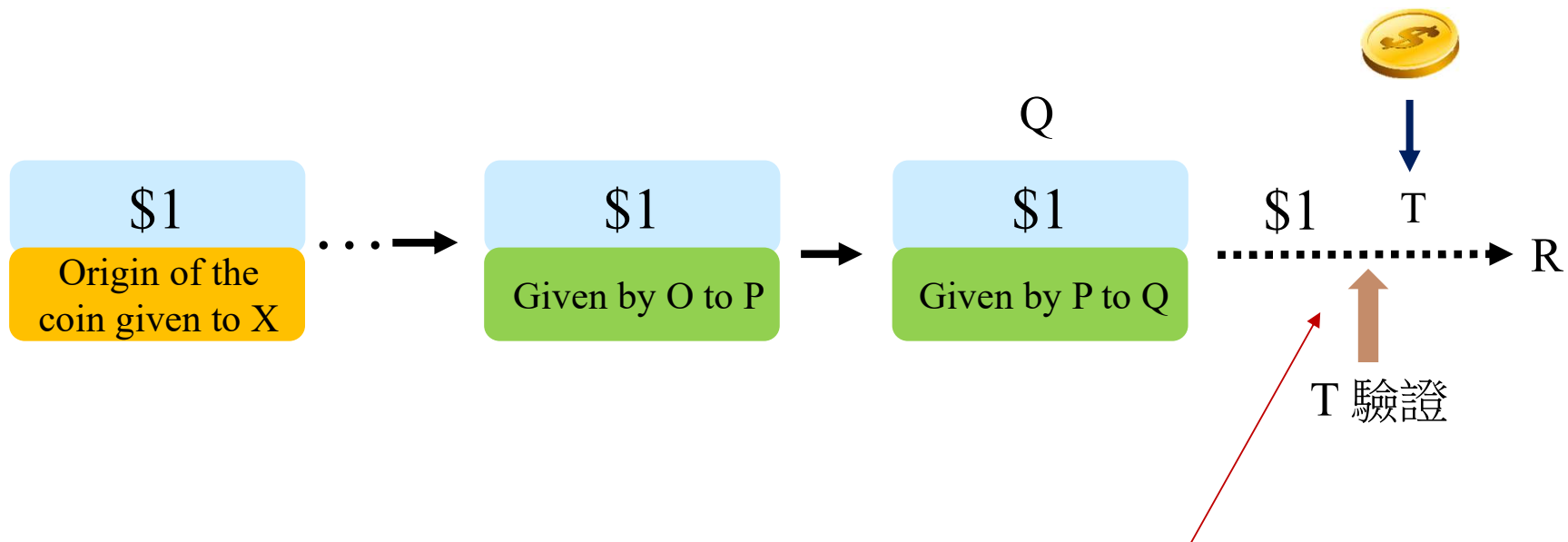


大家必須認同初始的幣，一般是系統啟用時置入的

誰來驗證？

交易紀錄如何保存？





- Q想要將\$1給R，必須有人來驗證，並且保存交易紀錄
- 為了鼓勵大家一起來參與，第一個驗證成功的人將獲得獎勵。這獎勵可以是新鑄造的幣，也可從交易抽傭
- 驗證可獲新幣，因此俗稱「挖礦 (mining)」

# Bitcoin 的第一個區塊

```
Bitcoin Genesis Block
Raw Hex Version

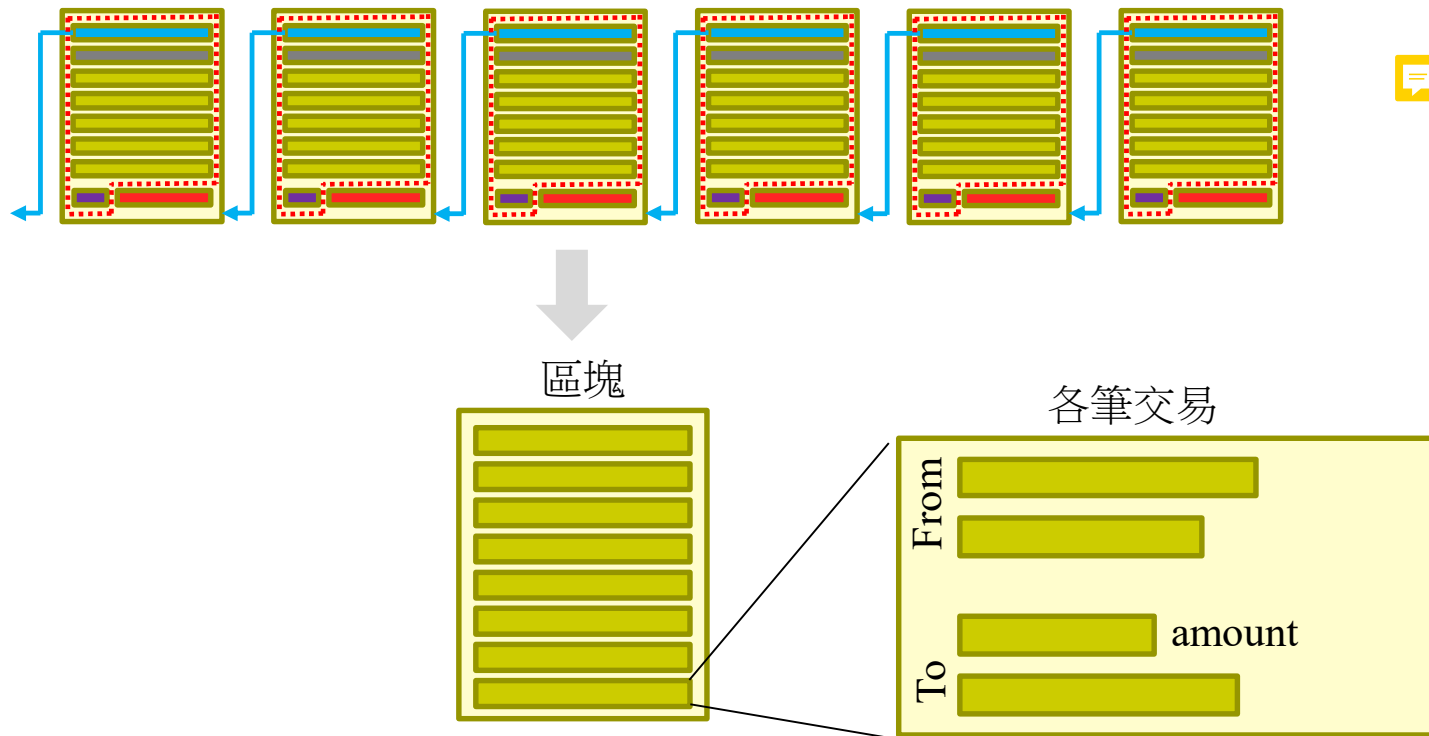
00000000 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000020 00 00 00 00 3B A3 ED FD 7A 7B 12 B2 7A C7 2C 3E ....;fiyz{.²zÇ,>
00000030 67 76 8F 61 7F C8 1B C3 88 8A 51 32 3A 9F B8 AA gv.a.È.Ã^ŠQ2:Ÿ,¢
00000040 4B 1E 5E 4A 29 AB 5F 49 FF FF 00 1D 1D AC 2B 7C K.^J)="_Iÿÿ...~+|
00000050 01 01 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000070 00 00 00 00 00 00 FF FF FF FF 4D 04 FF FF 00 1D .....ÿÿÿÿM.ÿÿ..
00000080 01 04 45 54 68 65 20 54 69 6D 65 73 20 30 33 2F ..EThe Times 03/
00000090 4A 61 6E 2F 32 30 30 39 20 43 68 61 6E 63 65 6C Jan/2009 Chancel
000000A0 6C 6F 72 20 6F 6E 20 62 72 69 6E 6B 20 6F 66 20 lor on brink of
000000B0 73 65 63 6F 6E 64 20 62 61 69 6C 6F 75 74 20 66 second bailout f
000000C0 6F 72 20 62 61 6E 6B 73 FF FF FF FF 01 00 F2 05 or banksÿÿÿÿ..ð.
000000D0 2A 01 00 00 00 43 41 04 67 8A FD B0 FE 55 48 27 *....CA.gŠŸ"bUH'
000000E0 19 67 F1 A6 71 30 B7 10 5C D6 A8 28 B0 39 09 A6 .gñ!q0+.\\Ö"(à9.¡
000000F0 79 62 E0 EA 1F 61 DE B6 49 F6 BC 3F 4C EF 38 C4 ybàè.abŸIÖ¿?LY8Ã
00000100 F3 55 04 E5 1E C1 12 DE 5C 38 4D F7 BA 0B BD 57 óU.Á.À.›\8x+9..W
00000110 8A 4C 70 2B 6B F1 1D 5F AC 00 00 00 00  ŠLp+kñ._~....
```

Source: [wiki](#)

“The Times 03/Jan/2009 Chancellor on brink  
of second bailout for banks”

# BitCoin的底層是一個Blockchain

- ❑ 每個區塊封存一段時間內的所有交易
- ❑ 這些區塊形成一個公共帳本 **Public Ledger**

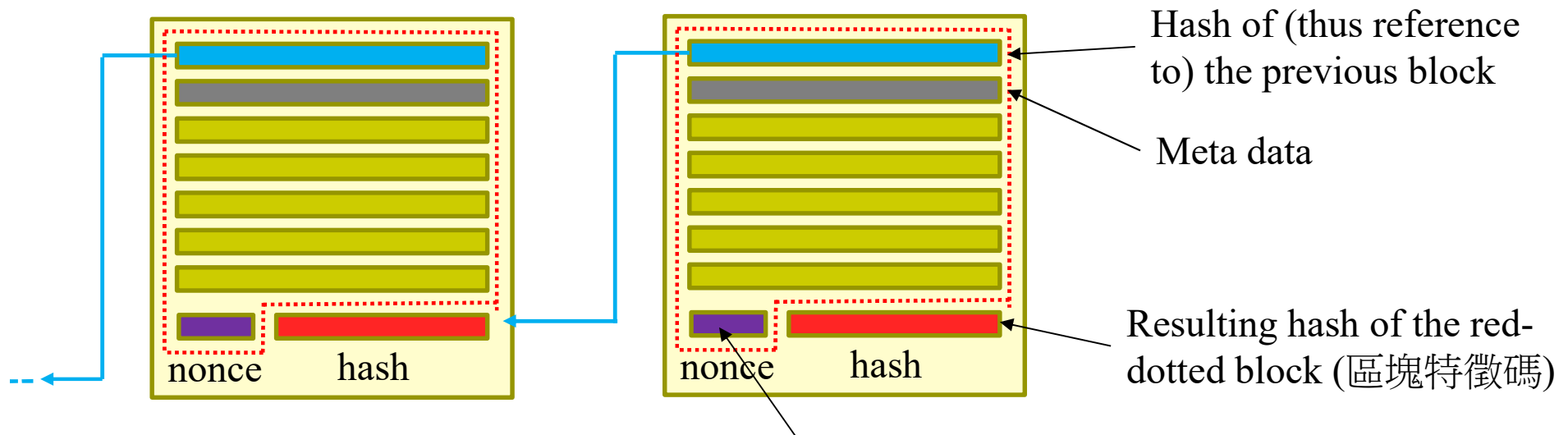


公共帳本存在每個節點，系統設計必須確保所有節點對於帳本的看法是一致的 (**consensus problem**)

# 相關技術挑戰

- ❑ 防止竄改
- ❑ 去中心化，但各節點對於系統的交易歷程 (帳本, ledger) 要有共識 (distributed consensus)

# 區塊之間的串連與防竄改



- Miners are trying to find a nonce such that SHA-256 of (“content in red area” + nonce (proof-of-work)) results a hash with some leading zeros (e.g., 20 bits of zeros)
- Whoever gets the first one wins the prize and attaches its nonce to form a new block and seals the transaction.
- The number of leading zeros required can be used to adjust the speed of new bitcoin creation. For Bitcoin, it is set such that on average about 10min is needed to solve the puzzle (append a new block), and is adjusted for every 2016 blocks (approximately 14 days.)

# Hash 雜湊函式



將不定長度訊息的輸入，透過演算  
運算，產生一個固定長度的亂碼，  
用來識別原資料，且從該亂碼無法  
反推得出原資料



**SHA-256**



Mapping of input to output  
is virtually 1-to-1 and  
computationally extremely  
hard to reverse!

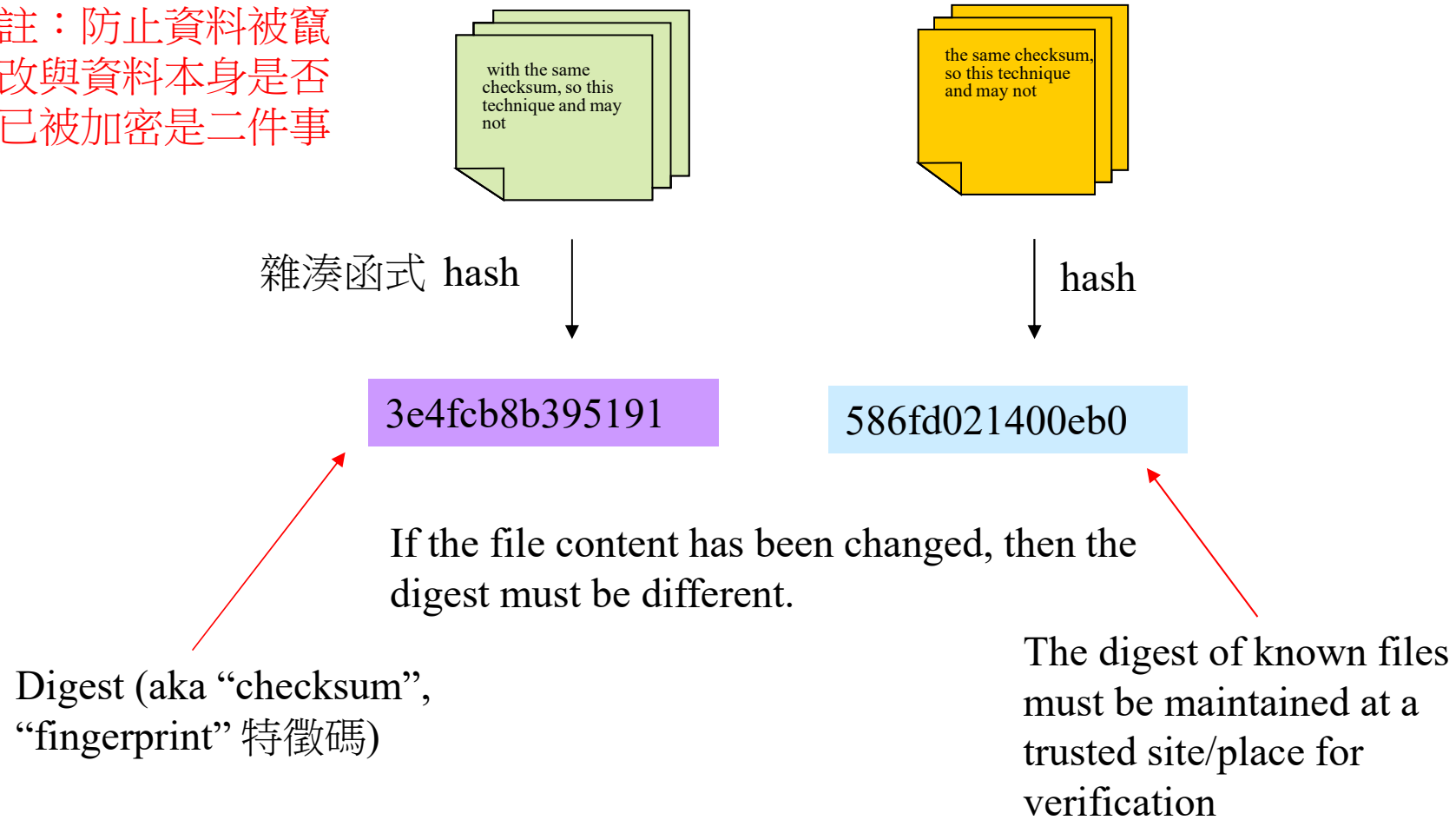


MCsbEHzdSasG7sRFPo7o....

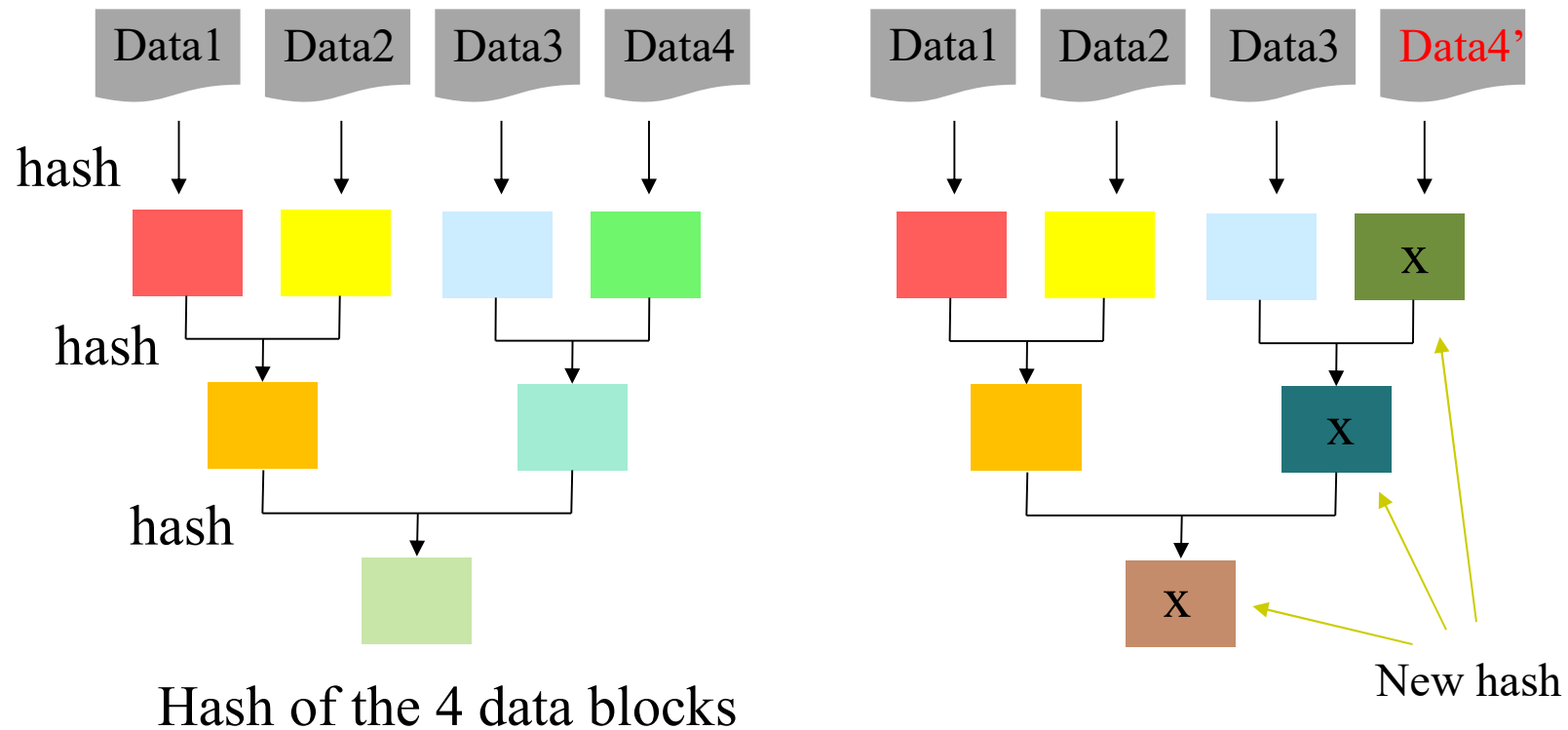


# 用 Hash 驗證資料是否被竄改

註：防止資料被竄改與資料本身是否已被加密是二件事

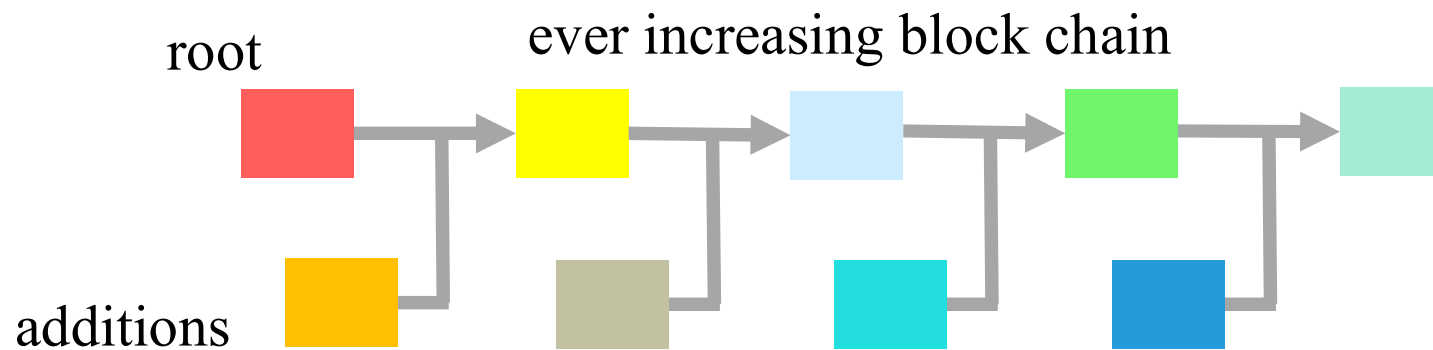


# Merkle Tree (Hash Tree)

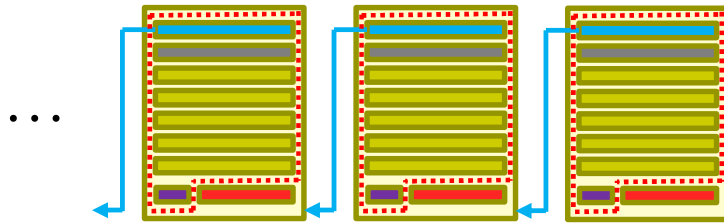


Hash trees allow efficient and secure verification of the contents of large data structures, as well as allow new data blocks to be added without affecting previous hashes (by keeping growing the tree).

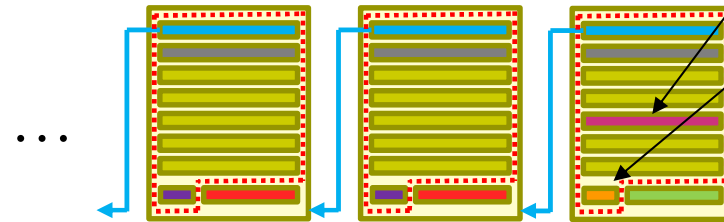
# blockchain的防竄改機制：Merkle Tree



# 光用 Hash 將區塊串起就很難竄改嗎？



竄改其中一筆資料，重新計算新區塊的hash，然後告訴所有人，這才是正確的區塊...

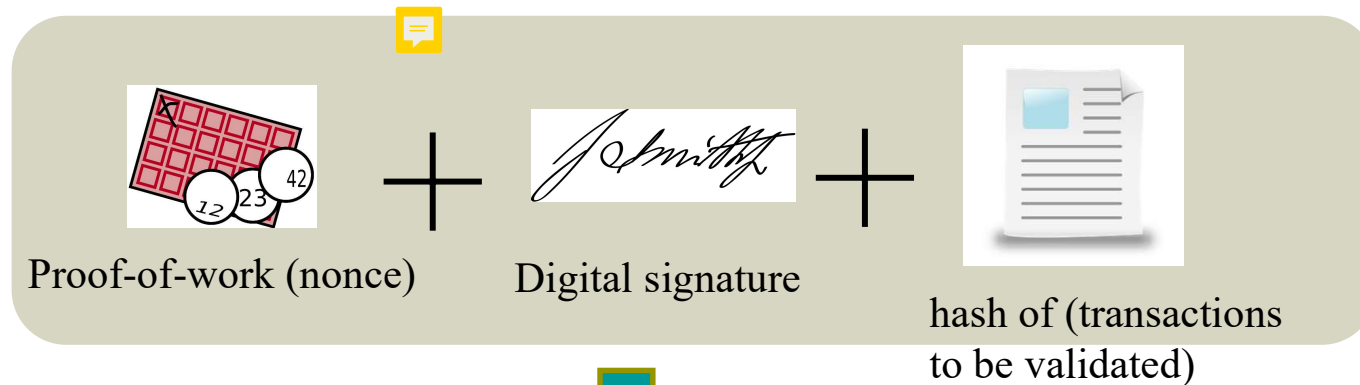


你需要讓多數的節點相信你的版本才是對的，所以，如果一個hacker真能操控多數的節點，這系統是不安全的



不過，在新區塊要封存進鏈時，可能在不同版本的新區塊，系統的節點怎麼取的共識要加哪一個版本進鏈？

# Proof-of-Work



SHA-256



Anyone who first “guesses” a nonce that produces some pattern of output (e.g., with 20 leading ‘0’) wins the prize. The hash nature makes the process look more like winning a slot machine

00000000000000000000xSasG7sRFPo7o....

Proof-of-work 增加新區塊加入的難度，也增強竄改區塊鏈的難度



# Proof-of-Work in Bitcoin

- 會給幫忙驗證的使用者一定的回饋，不僅提供誘因讓使用者參與驗證，同時提供一個分散式貨幣系統鑄造新幣的功能，因此“參與Bitcoin驗證”也稱“mining (挖礦)”。
  - 註：誘因機制不一定要產生新幣，可以用「手續費」方式讓交易雙方提供。

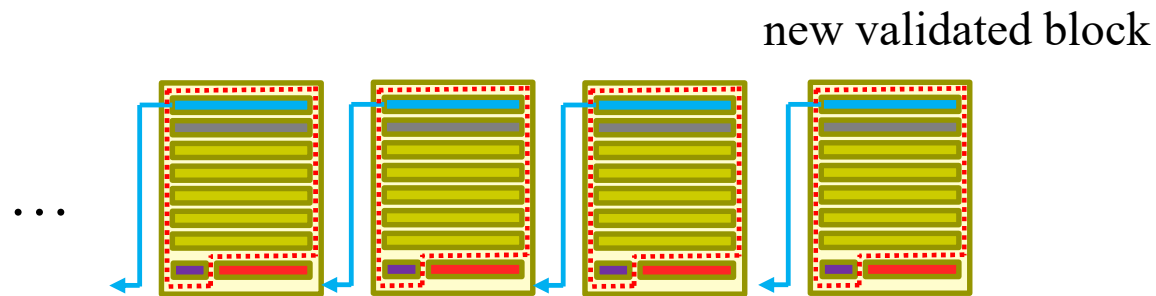


Bitcoin Mining (挖礦)



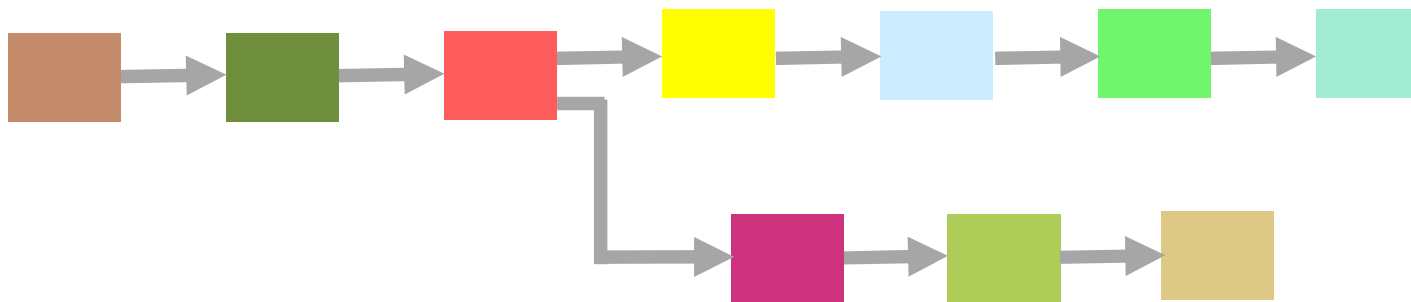
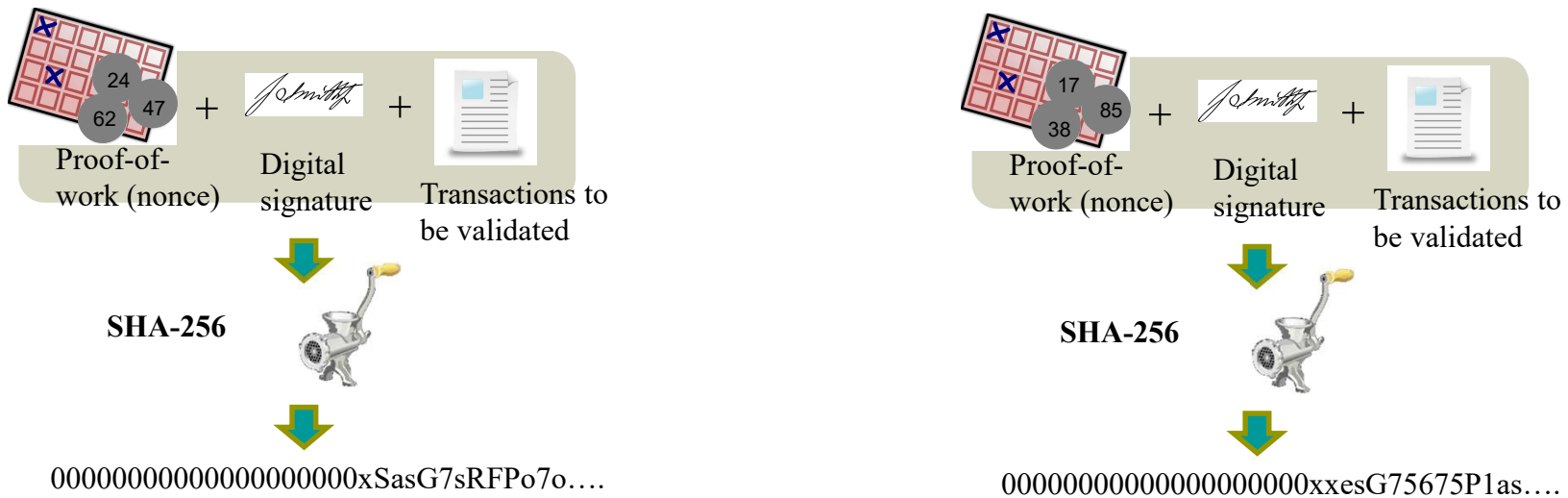
# Consensus in Blockchains

- ❑ 一個節點(minor)找到 proof-of-work 後，通知其他節點，將此區塊的資料封存

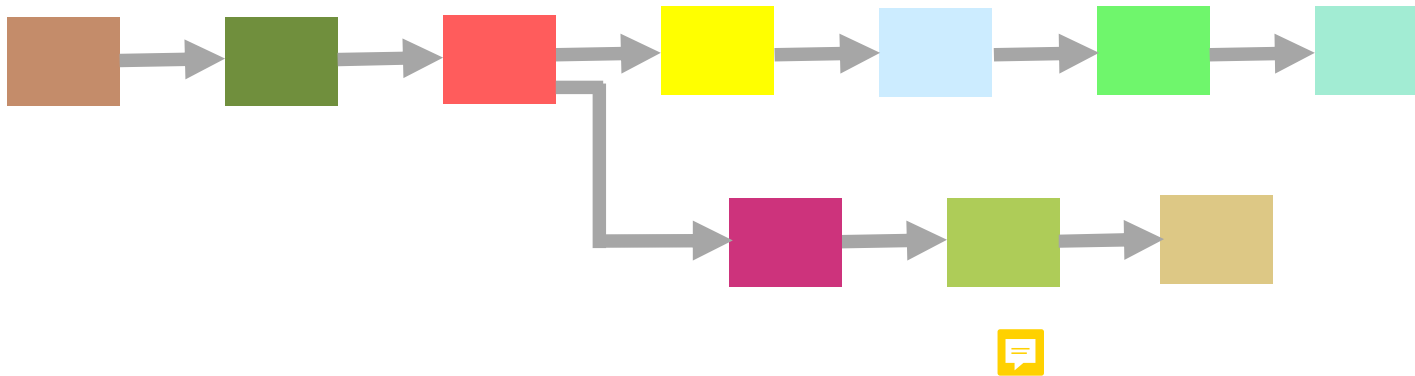


- ❑ 所有在“挖礦”的節點驗證新區塊proof-of-work的正確性後，更新其blockchains，然後嘗試驗證下一個區塊
- ❑ 所有挖礦節點也因此取得對於新區塊鏈的“共識 (consensus)”

萬一多人同時找到 **proof-of-work** 的答案時怎麼辦？

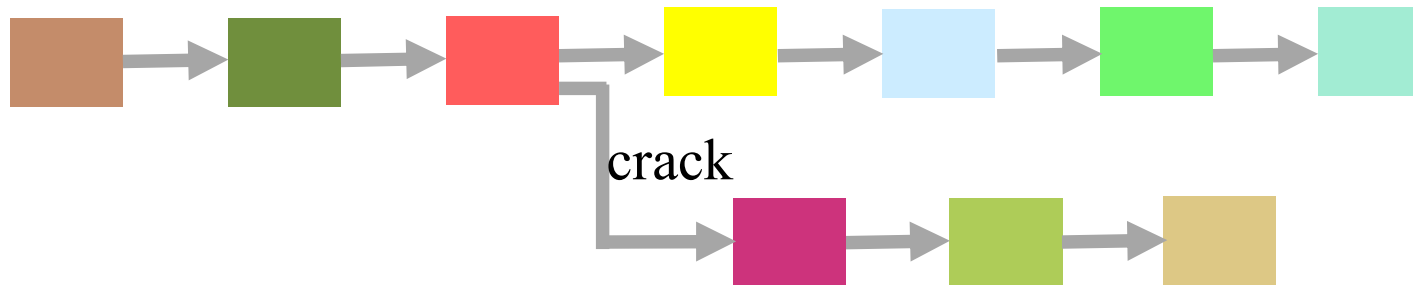


# Longest Transaction Block Chain



當一個節點發現分支情況時，它選擇長的，捨去短的，  
如果長度一樣，以系統預設的方式選擇其中之一

萬一有人破解其中一個block，  
竄改內容...



除非盜取者擁有的運算資源大於系統  
裡所有正常運作者的資源總和，否則  
新的分支不會長於原有的blockchain

# Bitcoin 的供應

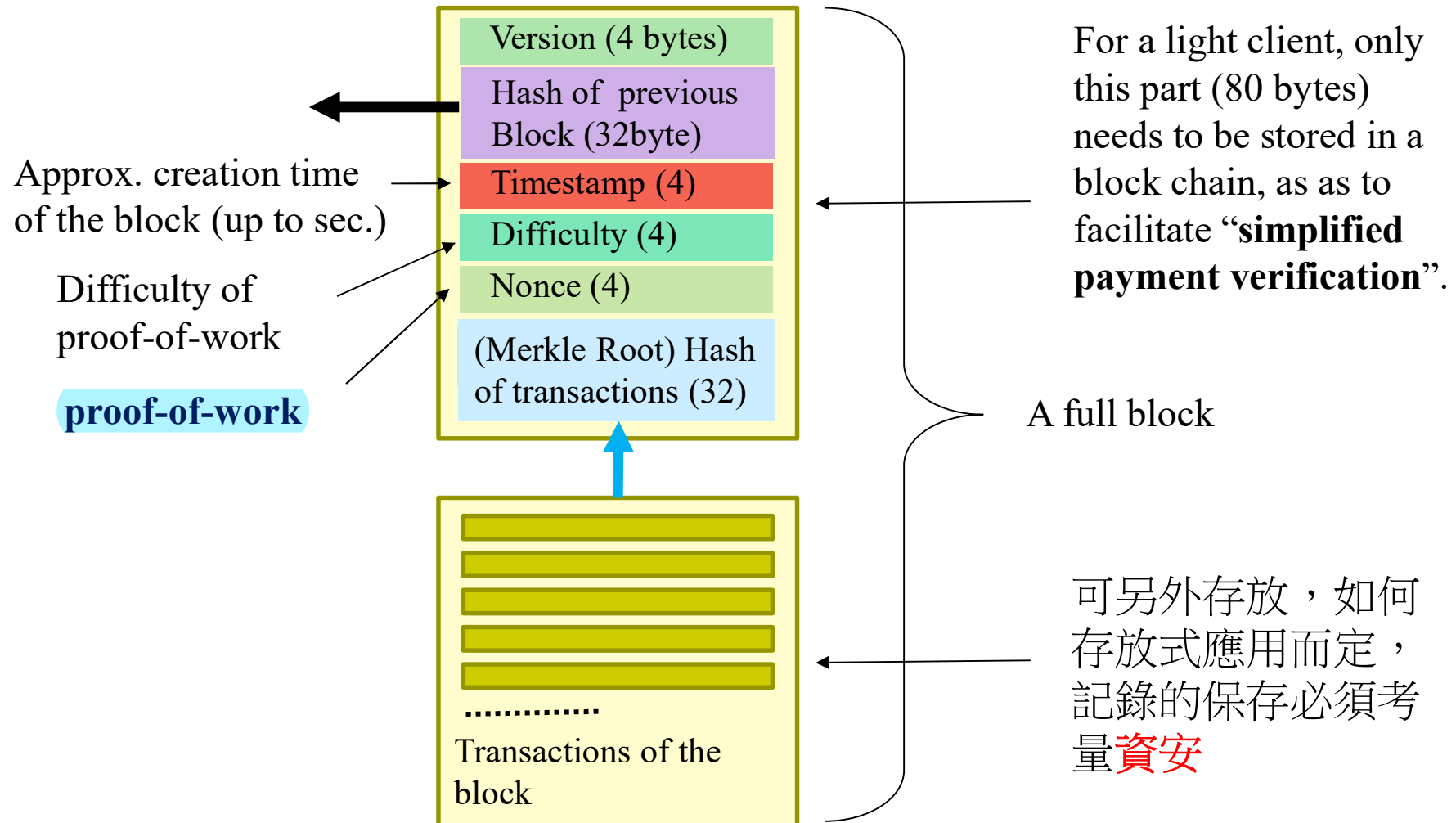
- ❑ New bitcoins are created through “**mining**”, a process to reward users to offer their computing power to verify and record payments into the public ledger (總帳).
- ❑ An arbitrary 21M BTCs have been capped, which is expected to be reached in 2140, after that, incentives for miners would be transaction fees.
- ❑ The bitcoin system adjusts the mining efficiency such that it will always take about ten minutes to add a new block regardless of the population of the miners and the sophistication the hardware they use.

## BitCoin的供應 (2/2)

- ❑ 50 BTC were injected into the BitCoin network when it was created in 2009.
- ❑ The reward for generating a block was 50BTC initially.
- ❑ The reward is halved for every 210,000 blocks generated.
- ❑ On average, generating a block takes about 10min.
- ❑ So about every 4 years the reward is halved.
  - 2020.05.12 起獎勵減半變成 6.25
- ❑ So the max number of BTCs that can be created is
$$210,000 \times (50 + 25 + 12.5 + 6.25 + \dots) \approx 21\text{M}$$
which are expected to be generated by 2140
- ❑ Min. unit of BTC (aka. a “Satoshi”) :  $10^{-8}$  BTC



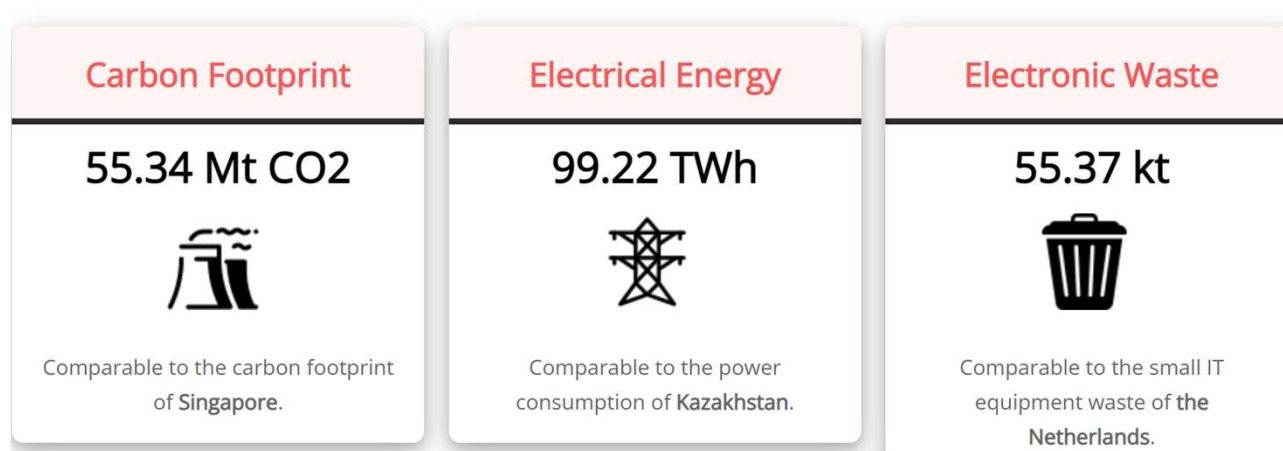
# A Typical Block



# Proof-of-Work


- ❑ 每個參與者都需要付出一些成本 (CPU運算資源)
- ❑ 參與者之間的競爭基本上是公平的，但擁有的運算資源愈多，奪標機會也愈大。
- ❑ 浪費能源

## Annualized Total Bitcoin Footprints



source: [digiconomist.net](https://digiconomist.net), captured 2023.05.17

# Proof-of-Stake

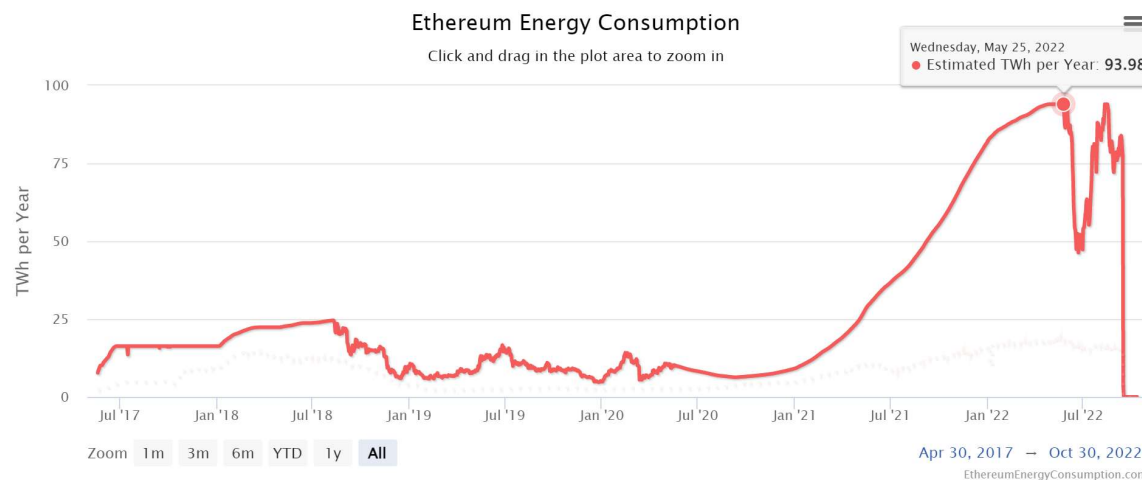
□ 以擁有之「資本」來決定誰能驗證，如 

■ Coin-age (used in, e.g., Peercoin):

- “number of coins” × “number of days they are held”
- Coin-age must be reset after being used, and must also be reset after certain amount of time to prevent some very old or very large collection of stakes from dominating the mechanism at any particular point.

# Ethereum: the “Merge”


- ❑ Completed on 2022.09.15, transitioning the network from Proof-of-Work (PoW) to Proof-of-Stake (PoS) consensus
- ❑ cuts energy use by over 99%
  - 以太幣每筆交易約需付出84kWh的能源成本（比特幣每筆交易需1,135kWh），轉到PoS狀態後，能源成本將縮小至每筆0.035kWh（約0.035度電）([source](#), 2022.08)



## Annualized Total Ethereum Footprints

(source: [digiconomist.net](#), 2022.10)

# Operations of BitCoin Network

- ❑ New transactions broadcast to all nodes.
- ❑ Each node collects new transactions into a block. 
- ❑ Each node works on finding a proof-of-work for the block.
- ❑ If some node succeeds, it broadcasts the proof and the block to all nodes.
- ❑ All nodes verify if the claim is true.
- ❑ If the claim is true, all nodes accept the block by appending the block to the chain.

# Ethereum

[PDF] [A next-generation smart contract and decentralized application platform](#)

[V Buterin](#) - white paper, 2014 - finpedia.vn

... of [applications](#) would be too small to warrant their own blockchain, and we note that there exist large classes of [decentralized applications](#), particularly [decentralized](#) autonomous ...

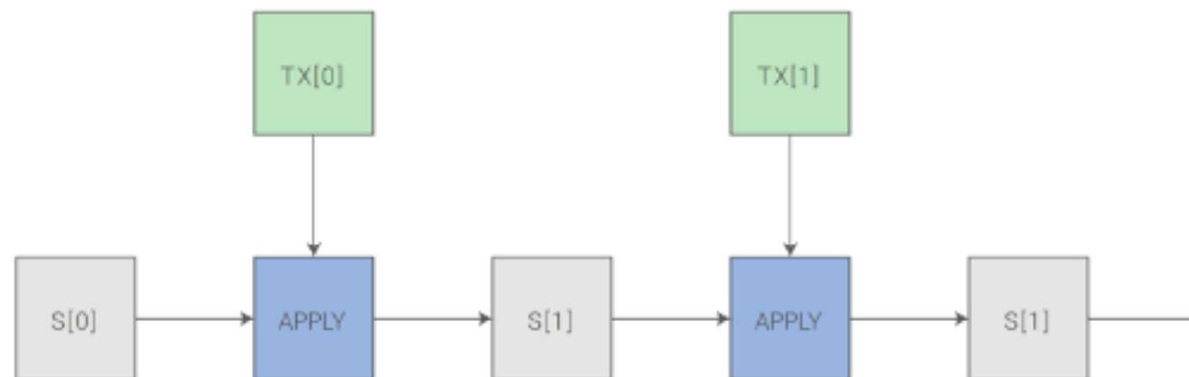
☆ 儲存 ↀ 引用 被引用 4254 次 相關文章 全部共 65 個版本 ↀ

(as of 2023.05.17)



# Ethereum (以太坊)


- ❑ Introduced on 2015.07.30 through ICO by Vitalik Buterin and team.
- ❑ A **decentralized platform** designed to run “**smart contracts**”
  - In Ethereum, the state is made up of objects called “**accounts**”
  - A block of transactions represents a state transition function transferring value and information between accounts.
  - It's native asset is called ‘ether’ (ETB, 以太幣)




source: Ethereum white paper, 2014

# Ethereum Accounts

❑ An Ethereum account (20-byte address) contains four fields:

- The **nonce**, a counter used to make sure each transaction can only be processed once
- The account's current **ether balance**
- The account's **contract code**, if present 
- The account's **storage** (empty by default)


❑ Two types of accounts:

- **Externally owned account:**  controlled by private key and with only Ether balance;
- **Smart contract account:** that has executable code
  - its code activated upon receiving a message, allowing it to read and write to internal storage and send other messages or create contracts in turn.

# Ethereum State Transition

- State transition  $\text{APPLY}(S, \text{TX}) \rightarrow S'$  is defined as follows:
1. Check if the transaction is well-formed
  2. Calculate the transaction fee as  $\text{STARTGAS} * \text{GASPRICE}$ , and determine the sending address from the signature. Subtract the fee from the sender's account balance and increment the sender's nonce.
  3. Initialize  $\text{GAS} = \text{STARTGAS}$ , and take off a certain quantity of **gas per byte** to pay for the bytes in the transaction
  4. Transfer the transaction value from the sender's account to the receiving account. If the **receiving account is a contract**, run the contract's code **either to completion or until the execution runs out of gas**.
  5. If the value **transfer failed** because the sender did not have enough money, or the code execution ran out of gas, **revert all state changes except the payment of the fees**, and add the fees to the **miner's account**.
  6. Otherwise, refund the fees for all remaining gas to the sender, and send the fees paid for gas consumed to the miner.

# Code Execution

- ❑ The code in Ethereum contracts is written in a low-level, **stack-based** bytecode language, referred to as “Ethereum Virtual Machine (EVM) code”.
- The code consists of a series of bytes, where each byte represents an operation.
- Operations access to three types of storage space:
  - The **stack**, a last-in-first-out container to which 32-byte values can be pushed and popped
  - **Memory**, an infinitely expandable byte array
  - The contract’s long-term **storage**, a key/value store where keys and values are both 32 bytes
    - ◆ Unlike stack and memory, which reset after computation ends, storage persists for the long term.
  - The code can also access the value, sender and data of the incoming message, as well as block header data, and can also return a byte array of data as an output.

# Ethereum Virtual Machine (EVM)

❑ A full computational state of EVM:

(block\_state, transaction, message, code, memory, stack, pc, gas)

- block\_state: global state containing all accounts and includes balances and storage.
- In each round of execution, the current instruction is found by taking the pc-th byte of code, and each instruction has its own definition in terms of how it affects the tuple.



Contract code is executed in the local nodes' EVM and in all the nodes that confirm the block and the transactions contained it.

# Writing Smart Contracts

- ❑ Programming Languages for smart contracts in Ethereum:
  - Serpent
  - Solidity
  - Viper
  - Rust
- ❑ Compiled into the EVM bytecode

# Solidity Language

- ❑ an object-oriented, high-level language for implementing smart contracts
- ❑ statically typed, supports inheritance, libraries, and complex user-defined types
- ❑ design influenced by C++, Python, and JavaScript
- ❑ programs executed in Ethereum Virtual Machine (EVM)
- ❑ For details, see <https://docs.soliditylang.org/en/v0.8.20/>

The Serpent language

# A Smart Contract Example

```
pragma solidity ^0.8.0;
contract Coin {
    uint256 public totalSupply;
    address public owner;
    event CoinCreated(address indexed to, uint256 amount);
    constructor(uint256 initialSupply) public {
        totalSupply = initialSupply;
        owner = msg.sender;
        CoinCreated(owner, initialSupply);
    }
    function transfer(address to, uint256 amount) public {
        require(to != address(0));
        require(amount <= balanceOf(msg.sender));
        // Update the state
        totalSupply -= amount;
        balances[msg.sender] -= amount;
        balances[to] += amount;
        // Emit an event
        Transfer(msg.sender, to, amount);
    }
    function balanceOf(address who) public view returns (uint256) {
        return balances[who];
    }
}
```





# A Smart Contract Example (2)

A contract maintains a mapping of balances for each address, allowing users to deposit and withdraw funds. The contract owner has additional privileges defined by the onlyOwner modifier, which restricts certain functions to be called only by the owner.

The contract emits an event Transfer whenever a transfer of funds occurs, providing a convenient way to track transactions on the blockchain.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SampleContract {
    address public owner;
    mapping(address => uint256) public balances;
    event Transfer(address indexed from, address indexed to, uint256 value);
    constructor() {
        owner = msg.sender;
    }
    modifier onlyOwner() {
        require(msg.sender == owner, "Only the contract owner can call this function");
        _;
    }
}
```

generated by ChatGPT, 2023.05.17

# A Smart Contract Example (2) (contd.)

```
function deposit() public payable {  
    balances[msg.sender] += msg.value;  
}
```

```
function withdraw(uint256 amount) public {  
    require(amount <= balances[msg.sender], "Insufficient balance");  
    balances[msg.sender] -= amount;  
    payable(msg.sender).transfer(amount);  
    emit Transfer(address(this), msg.sender, amount);  
}
```

```
function transfer(address to, uint256 amount) public {  
    require(amount <= balances[msg.sender], "Insufficient balance");  
    balances[msg.sender] -= amount;  
    balances[to] += amount;  
    emit Transfer(msg.sender, to, amount);  
}  
}
```

generated by ChatGPT, 2023.05.17

# 非同質化代幣, **NFT** (Non-Fungible Token)

- ❑ ERC (Ethereum Request for Comments): 以太坊開發者公開徵求意見，已建立共同可以遵循的標準
- ❑ ERC20: 以Ethereum平台發行同質性代幣之標準協議
  - 在 ERC20 中，每個帳戶地址紀錄的是有多少代幣
- ❑ ERC721: 以Ethereum平台發行非同質性代幣之標準協議
  - 在 ERC721 中，每個帳戶需要額外單獨記錄每枚代幣的 ID

# ERC20

1. 代幣的全名
2. 代幣的縮寫
3. 代幣的最小單位數值
4. 總代幣發行量

```
1  function name() public view returns (string)
2  function symbol() public view returns (string)
3  function decimals() public view returns (uint8)
4  function totalSupply() public view returns (uint256)
5  function balanceOf(address _owner) public view returns (uint256 balance)
6  function transfer(address _to, uint256 _value) public returns (bool success)
7  function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
8  function approve(address _spender, uint256 _value) public returns (bool success)
9  function allowance(address _owner, address _spender) public view returns (uint256 remaining)
10
```

Source: <https://ethereum.org/en/developers/docs/standards/tokens/erc-20/>

5. 查詢 `_owner` 帳戶代幣餘額
6. 傳送 `_value` 數量的代幣到 `_to` 錢包地址
7. 從 `address _from` 傳送 `_value` 數量的代幣到 `_to` 錢包地址
8. 容許 `_spender` 從 `_owner` 錢包中提取 `_value` 數量的代幣
9. 查詢 `_spender` 尚可從 `_owner` 錢包提取的代幣數量
10. 觸發事件 ...

# NFT (Non-Fungible Token)



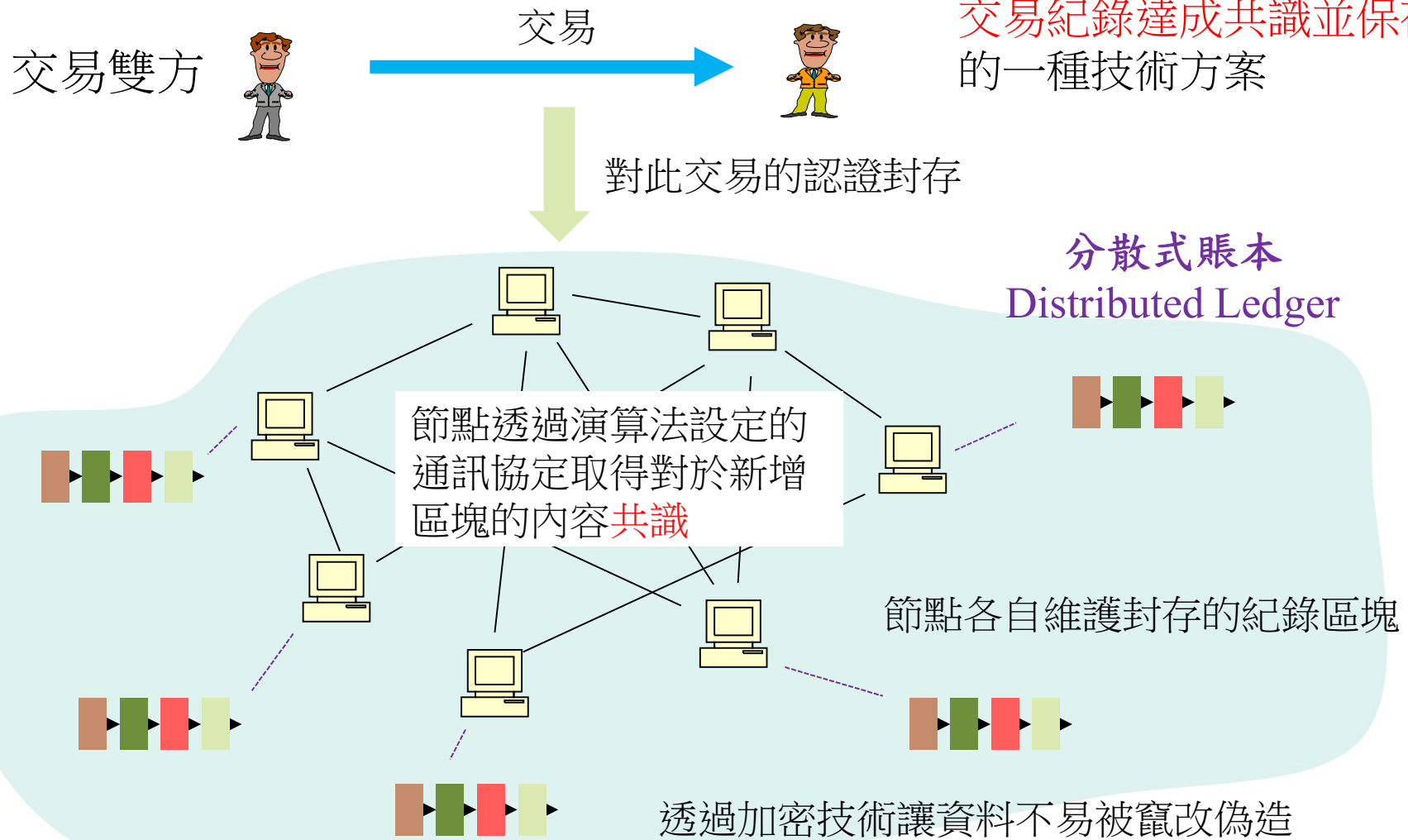
(Source: [wiki](#))

NFT 一般代表對某項數位資產的擁有權(ownership)，但不代表擁有它的版權，除非寫在合約裡

# BitCoin開創了Blockchain的 技術與新興應用熱題

# 區塊鏈

區塊鏈本質上是一個去中心化的資料庫，由參與節點透過演算法設計，來對交易紀錄達成共識並保存的一種技術方案



# 參與保存驗證的主機群



集中式  
centralized

由單一或少數機構完全負責

此機制違反區塊鏈設計初衷，但實務上這是最容易設計的機制



聯盟/會員式  
consortium

會員共同來維護資料，彼此存在一定程度信任感，但又無法完全信任單一或少數個體，此機制的安全性有賴於會員加入聯盟的規範



開放給任何人  
fully decentralized

越多人依系統設計的機制參與系統會越安全，但仍需假設成員三教九流，各懷鬼胎，彼此互不信任，因此系統設計重點在如何鼓勵節點參與，同時確保系統能安全運做，惟實務上這是最難設計的機制

極高

對主控權由少數節點掌控的接受度

極低

**Blockchain 技術是為第三種情境所設計，但目前多數應用屬第二種情境**



# Blockchain技術的實務意涵

可以實作出一數位的「交易/流通記錄」服務

1. 所有的記錄被完全的保留下來，無法輕易竄改偽造
  - a. 因為所有的記錄都被保存下來，新的交易/流通記錄可以被驗證是否允許(如 $A \rightarrow B$ , A是否有擁有要交易給B的東西)，一旦通過驗證，代表交易確認，也新增至記錄裡。
  - b. 交易雙方皆可匿名隱藏真實身分，但非區塊鏈重點
2. 服務的使用者不信任將上述工作交給任何特定機構 (trusted 3<sup>rd</sup> party) 去做

它的應用取決於在交易/流通記錄裡，記錄的是什麼？

數位文件

數位創作

虛擬貨幣

行智能合約 (Smart Contract)

但真實世界裡什麼應用同時需要上述二項特性 (特別是 2.)？

# 實體世界裡的交易如何應用Blockchain

□ 需將實體世界的資訊轉入Blockchain應用系統，但如何確保這個「轉入」過程是正確的？

■ 食品履歷：

– 養殖者輸入「今日餵食xxx」，怎麼確保他真的做了這些？

實務上，可以藉由「稽核」強化線下資料進入「區塊鏈」的可信度，但這實際上違反了區塊鏈的核心設計理念：不相信任一特定機構！

# 區塊鏈的驗證效率限制其應用規模

- ❑ Bitcoin的設計 (block/10min, 1MB block size, 380byte/transaction)，限制每秒能處理約5筆交易 (Transactions Per Second, TPS )
- ❑ Ethereum TPS約在10~15
- ❑ 傳統支付網路如Visa平均TPS約為1,700 (此為實際交易數，因其集中式機制(centralized)，理論上其TPS只受限於主機速度)
- ❑ 分散式運算無可避免的代價是效率/可擴展性，有些區塊鏈網路宣稱可達更高的TPS (如 Ethereum 2.0)，往往是在分散式與集中式運算之間折衷(如使用較少運算節點)



食安



Source: G子的漫畫生活, [pixnet](#)

# 食品履歷能解決食安議題嗎？

## 區塊鏈如何應用於食品履歷?



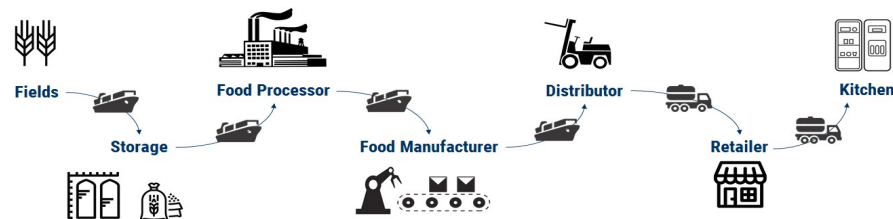
# 食品履歷



Source: 行政院 雲端運算與政府施政, 2012.09

# 建立食品履歷的挑戰

- ❑ 從原料到零售端的任一環節只要有廠商不願參與，就影響食品履歷的成效
  - 如何讓廠商願意提供資料？
    - 法令？誘因？
- ❑ 如何讓資料可以一路串接，形成鏈結的資訊
  - 公司編號、產品編號、生產單據編號、交易單據編號
- ❑ 如何確保資料真偽？
  - 輸入的資料是正確的
  - 過程中不被竄改
  - 稽核？
    - 誰來做？
- ❑ 資料保存在哪？



區塊鏈技術解決那些問題？

# 步步雞



source: 养殖“黑科技”催生“步步雞”， 爱奇艺

眾安科技推進區塊鏈技術 確保步步雞健康可靠, 2017.10.18

為何光二維條碼不足以防偽？

中国移动 上午11:01 73%  
中国移动 上午10:55 69%

< 返回 关闭

步步鸡

<

查询结果

鸡只编号 0000029222

农场直播

当前状态 已售出

166  
总日龄(天)

1.5  
出栏毛重(kg)

2186342  
总步数

基本信息

步数统计

健康状况

运动轨迹





# 京東“跑步雞”

京東推「扶貧跑步雞」 每隻雞跑夠百萬步 即3倍價錢收購, 2017.06.13



¥148.00

京東扶貧跑步雞 北京油雞品種 1.0kg 散養雞 土雞 公雞整雞肉 年貨禮盒裝 吃湯

3.3萬+條評價

京東生鮮京東自營旗艦店

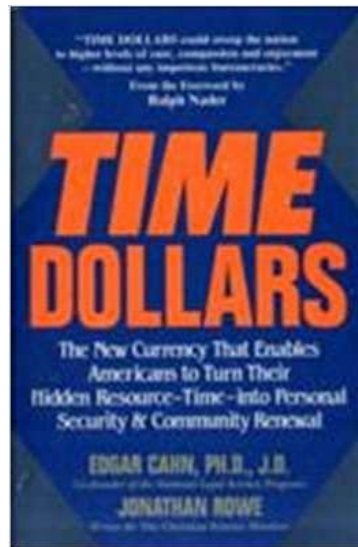
<https://item.jd.com/6379987.html>

為什麼京東“跑步雞” 168元賣到斷貨, 2018.02.22



# 時間銀行 (Time-based Currency)

以服務交換服務，將對別人的服務當作一種「點數」存入戶頭，以便在日後以儲存之點數換取別人的服務。

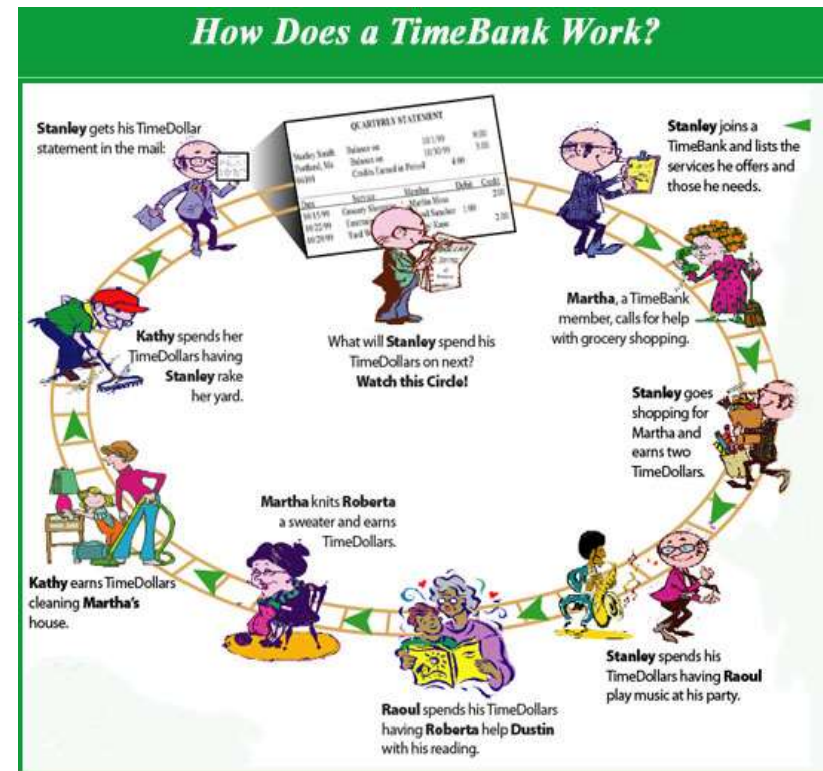


Jonathan Rowe & Edgar S. Cahn, 1992

Time Credits

Timebanks      Service Credits

台灣時間銀行協會



Source: [changemakers.com](http://changemakers.com)