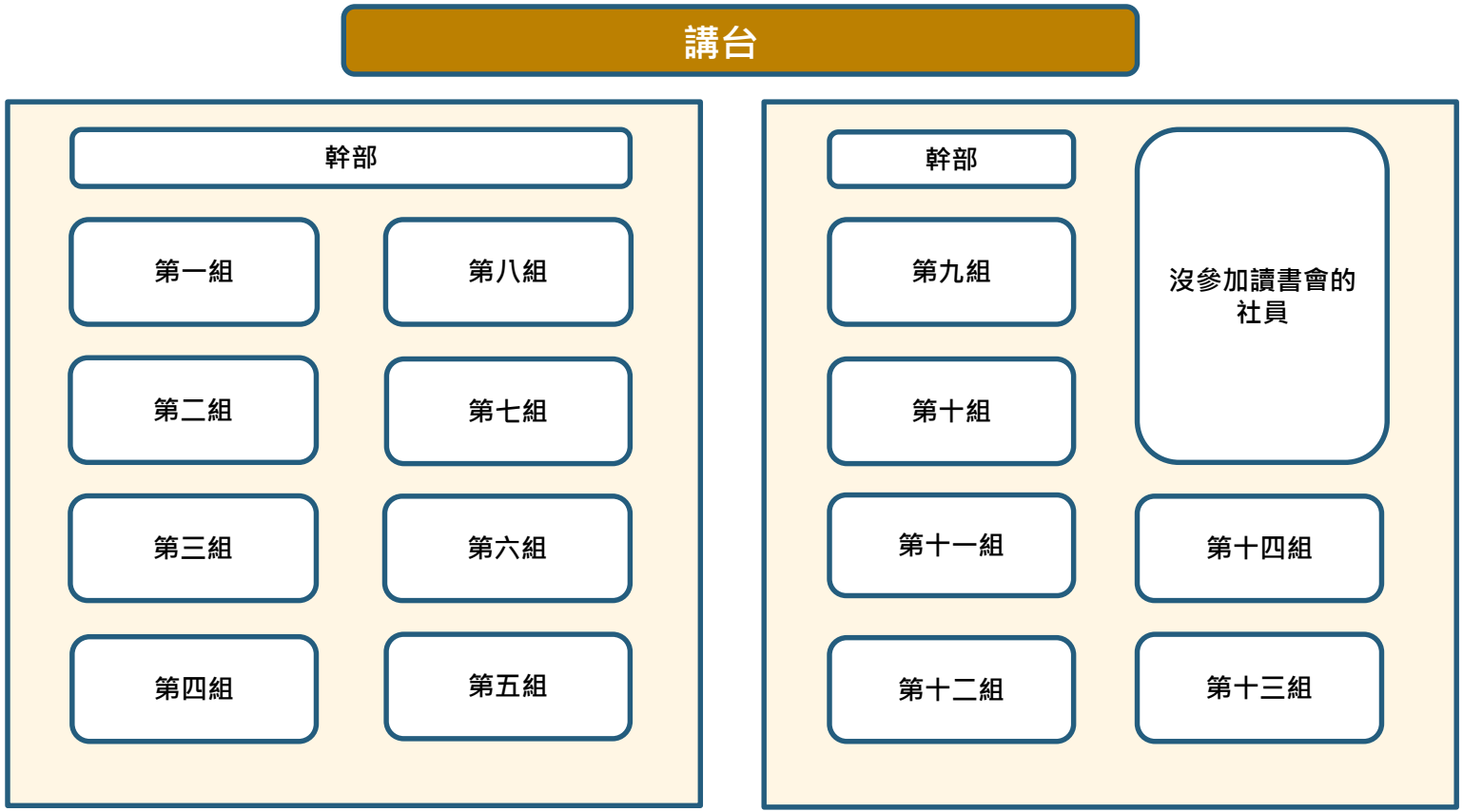
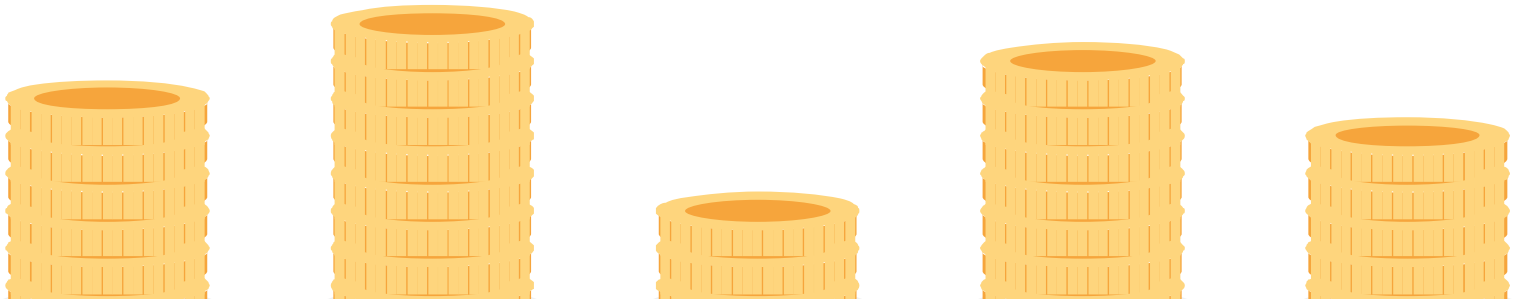
各小組區域分配，大家請先跟自己的小組坐在一起

上課的時候如果想坐前面可以先把東西放在自己的區域，坐到前面來聽，實作討論的時候再坐回去



工作坊一



去中心化，如何顛覆世界？

區塊鏈演化三部曲

0

沒有區塊鏈之前

中心化的世界

1

區塊鏈1.0

比特幣：去中心化的開始

2

區塊鏈2.0

以太坊：智慧合約認證

3

區塊鏈3.0

IOTA：連接實體生活、物聯網

工作坊流程圖

19:00~19:05

開場

講一下要幹嘛

19:10~19:50

上課

複習重點概念

手把手教學

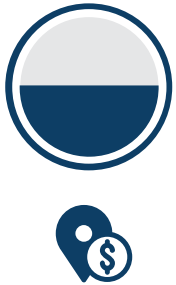
19:50~20:05

前置準備

各小組確定一切

可以正常執行

20:05~20:50

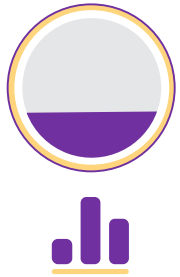


小組實作任務

公布題目

小組解任務

20:50~21:00

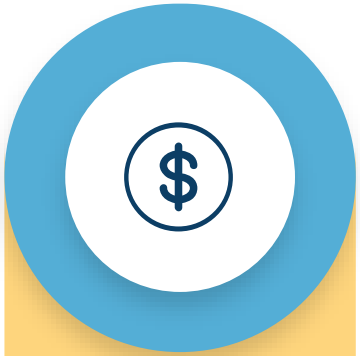


頒獎時間

收表單

結尾 + 講解

希望大家學到的東西



區塊鏈的結構

區塊鏈工作原理

區塊鏈
會遇到什麼問題

動手實作更瞭解

區塊鏈是什麼？

區塊鏈是去中心化的分散式資料庫

1

去中心化

分散式帳本

分散式的資料庫
全民皆可參與的記事本
每筆資料都可被記錄

2

不可篡改

不變數據源

Hash
Ox2d6a7b0f6adeff3842
3d4c62cd8b6ccb708da
d85da5d3d06756ad4d8
a04a6a2

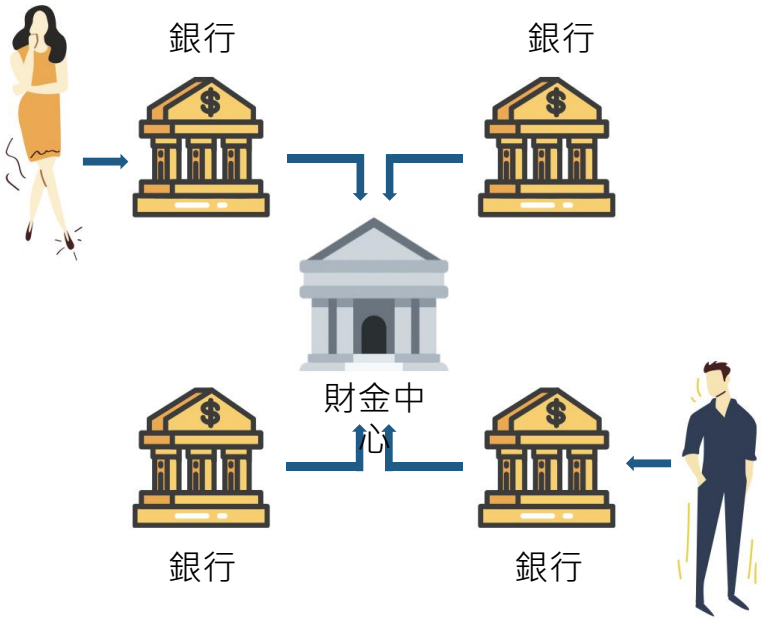
3

安全信任

公開透明性

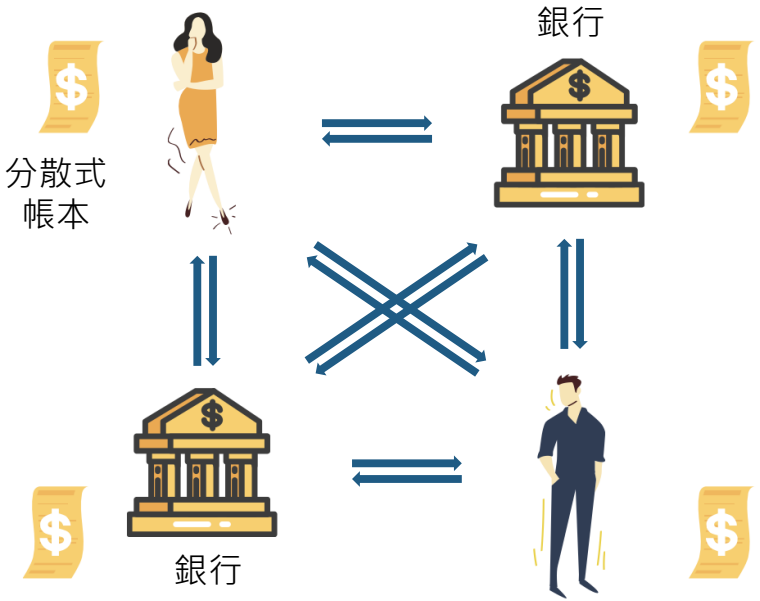
任何人可以透過網路看到
資訊的來源與歷程

沒有區塊鏈 — 中心化的世界



所有的交易必須有一個中介機構、交易所在中心做媒合，中心保存所有交易紀錄，讓全球經濟、金融體系可以運轉。

區塊鏈 1.0 — 點對點的開始

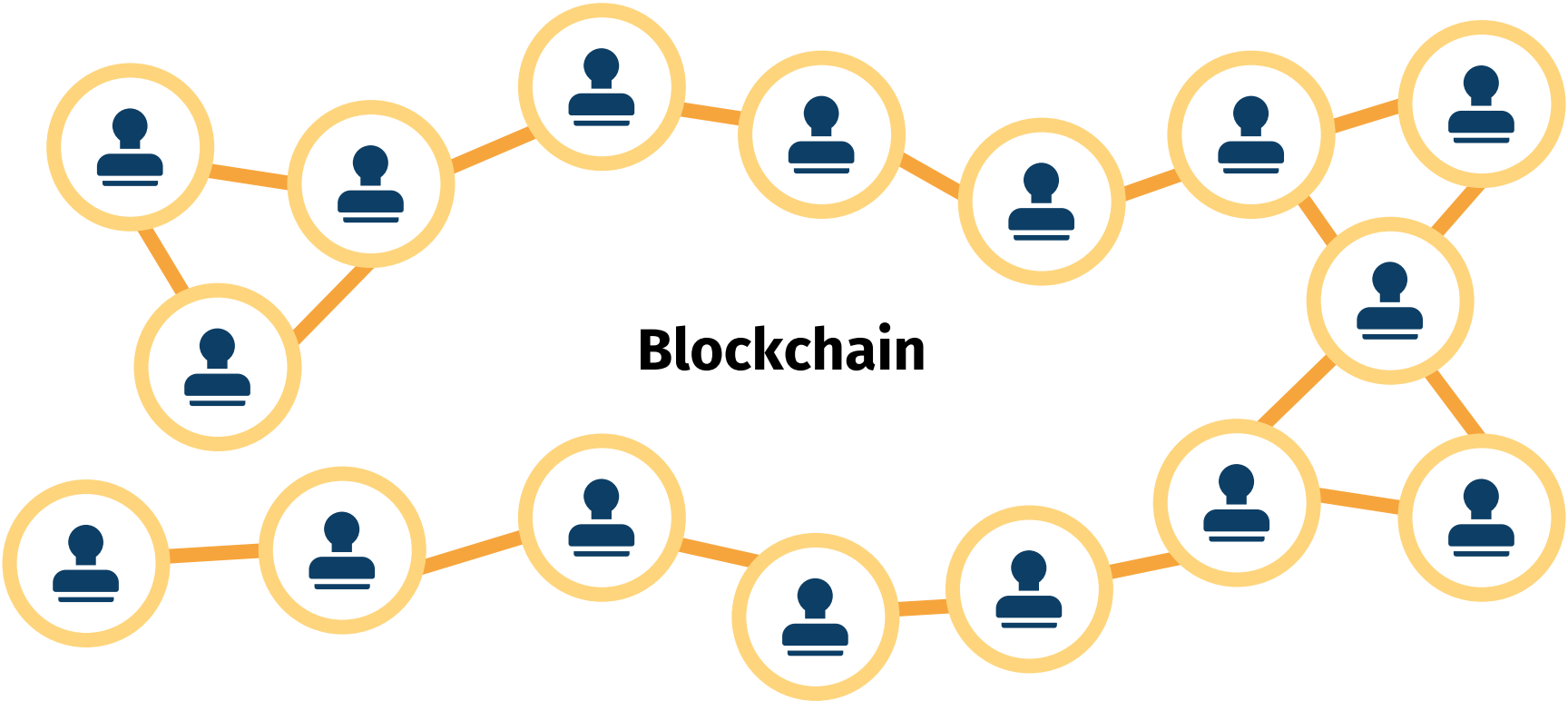


無論是個人對個人、銀行對銀行，彼此都能互相轉帳，再也不用透過中介機構；交易帳本經過加密，分散儲存，比以往更安全、交易紀錄更難被竄改。

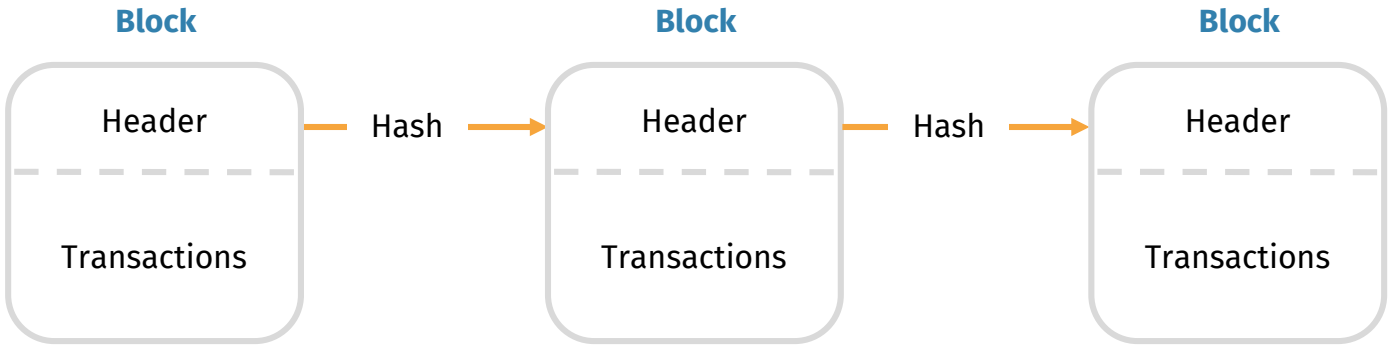
區塊鏈的結構

區塊裡到底裝了些什麼 ！？

區塊鏈示意圖



區塊的基本結構



Header 的資料

名稱	說明	Size
版本號	區塊數據的版本號	4 Bytes
Previous Hash	指向前一個區塊的 Hash	32 Bytes
Merkle Root	區塊內所有交易計算得出的 Hash	32 Bytes
Timestamp	區塊產生時間	4 Bytes
Difficulty	區塊產生難度	4 Bytes
Nonce	PoW 演算法執行次數	4 Bytes

Transactions 的資料

名稱	Size
版本號	4 Bytes
交易輸入總額	1-9 Bytes
交易輸入地址	1-9 Bytes
交易輸出總額	1-9 Bytes
交易輸出地址	1-9 Bytes
時戳	4 Bytes

Block Header — Previous Hash

版本號 | Previous Hash | Merkle Root | Timestamp | Difficulty | Nonce

什麼是 Hash Function？

將輸入值帶入雜湊函數獲得的值稱「雜湊值」

不定長度

訊息 → Hash Function 雜湊函數 → 雜湊值

固定長度

Hello World	SHA-256	a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b57b277d9ad9f146e
Hello World	SHA-256	a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b57b277d9ad9f146e

網頁密碼

網頁儲存的是「 經過 Hash 的密碼」，而非使用者輸入時的明碼。因此忘記密碼時，需要重設密碼，而不是給你原先的密碼！

Header
Transactions

Previous Hash 用於連接前一個 **Block**

Header
Transactions

Block Header — Previous Hash

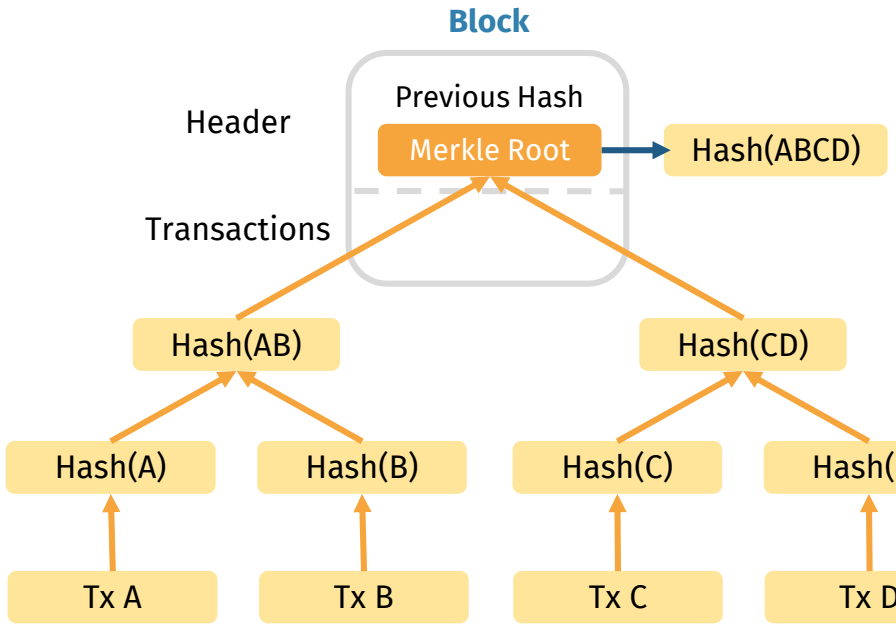
版本號 — **Previous Hash** — Merkle Root — Timestamp — Difficulty — Nonce

Block Header — Merkle Root

版本號 | Previous Hash | **Merkle Root** | Timestamp | Difficulty | Nonce

什麼是 Merkle Tree？

Merkle Tree 是 Hash 值所組成的二元樹



1. 不可篡改性

一個 Hash 被改到，牽一髮而動全身，該 Hash 以上的 Node 都會變動，最後導致 Merkle Root 變動

→ Leaf 是各交易的雜湊

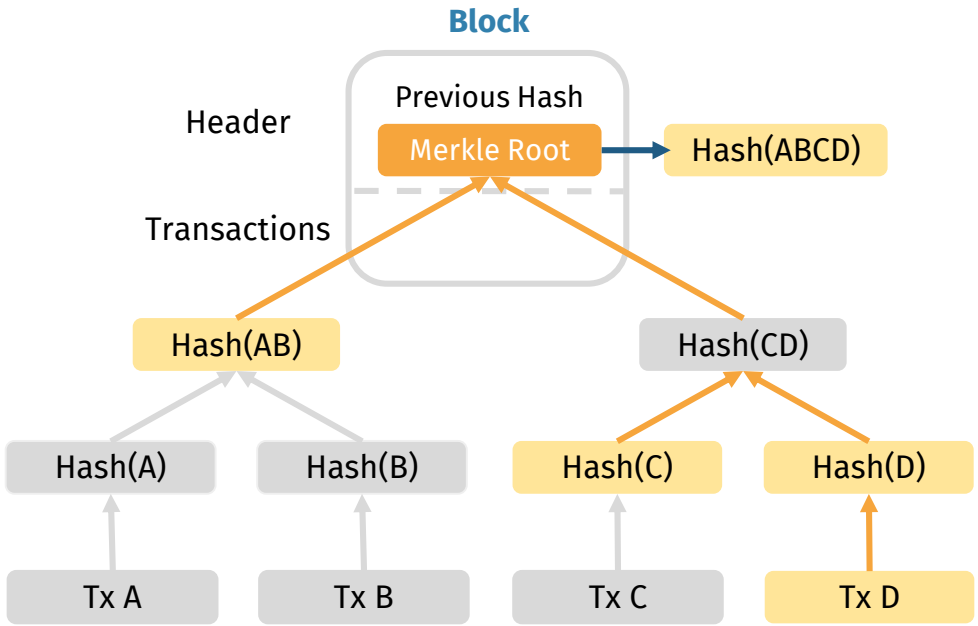
→ 不同的交易(transactions)

Block Header — Merkle Root

版本號 — Previous Hash — **Merkle Root** — Timestamp — Difficulty — Nonce

什麼是 Merkle Tree？

Merkle Tree 是 Hash 值所組成的二元樹



2. 驗證部分資訊就能驗證整體資訊

因資料量龐大，難以驗證所有資料，因此用 Merkle tree 的方法來達到「驗證部分資訊就能驗證整體資訊」的效果。

Block Header — Timestamp、Difficulty、Nonce

區塊產生的時間。
所對應的是每一次交易紀錄的認證，用於證明交易紀錄的真實性。

在給定目標下找到 Hash 值的難度，被運用於挖礦中。
Difficulty 代表 hash 值開頭需要滿足幾個 0。
如果設定難度是 5，代表 hash 值開頭需要為 5 個 0 開頭才算合格。

Nonce 為一隨機數，被運用於挖礦中，更動此值來為區塊重新計算 Hash 值。
直到找到包含所需數量的 Leading Zero 的 Hash 值，所需的零位數由 Difficulty 決定。

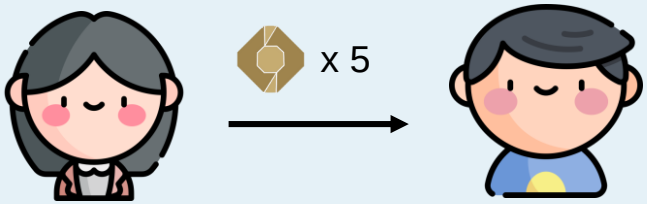
藉由重新發展數字貨幣來了解區塊鏈的工作原理

Point：從交易到區塊，再從區塊到區塊鏈

FCoin 1.0

-產生一個簡單的數字貨幣

形式：交易直接由Alice轉給Bob，
沒有任何驗證



帳戶裡有十塊錢

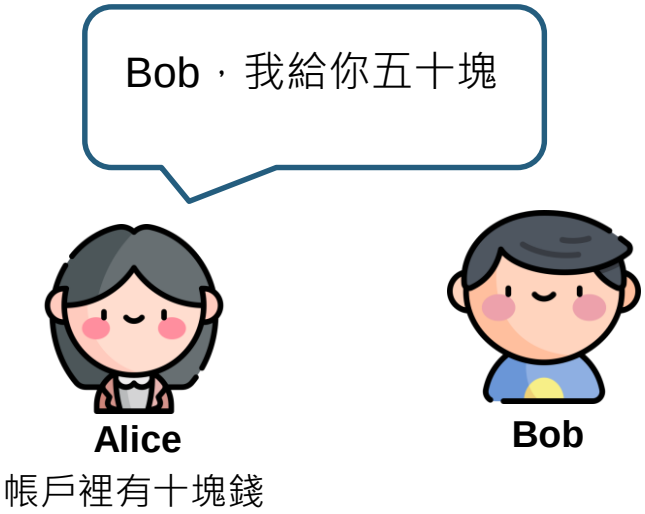
交易

隱憂

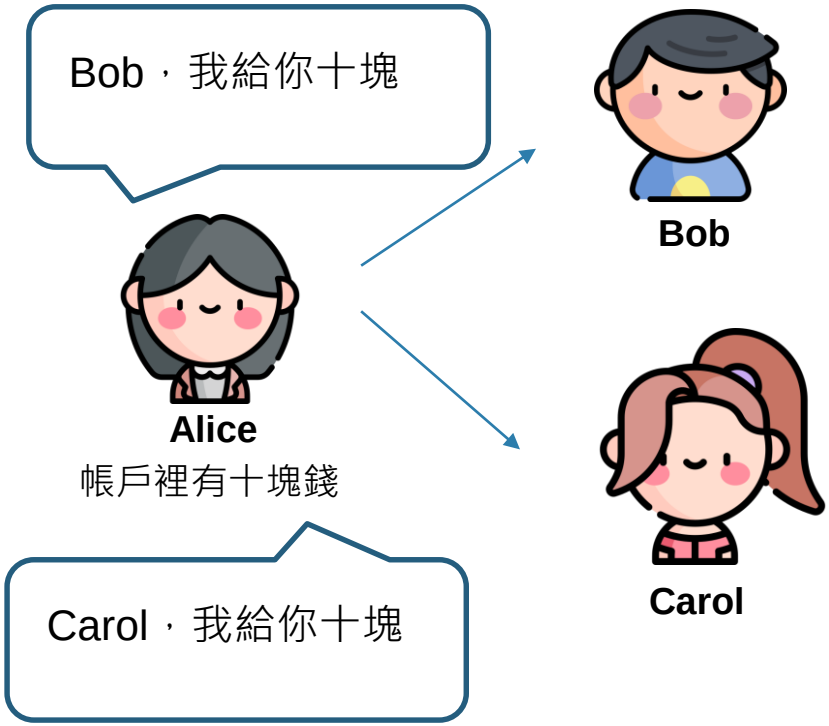
1. 不再有中心化機構（如：銀行）幫我們驗證貨幣來源與去向。
2. 因為是數字貨幣的形式（看不到實體），不像現實生活中一百塊是直接實體支付

偽造、雙重支付

情況一
偽造（Forge）



情況二
雙重支付（Double spending）



少了中心化機構，我們需要...

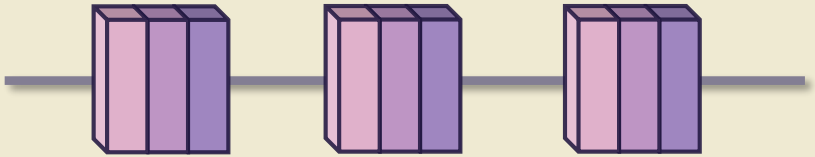
需求

1. 找人來幫忙驗證交易資料

2. 一個公開的記帳本（Public ledger），讓大家可以看所有過往交易資料

解法

礦工

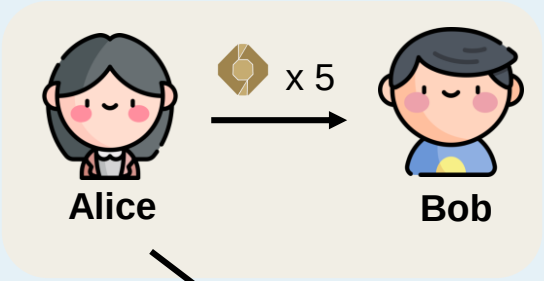


區塊鏈

FCoin 2.0

-加入礦工和數位簽章

Alice's 私鑰



交易

給礦工的交易手續費

1. Alice用私鑰，以數位簽章的方式將交易加密
   （目的相當於蓋印章）
2. 這個交易被廣播後，由礦工拿著公鑰解密來驗證。

雜湊函數

ce922519a3…

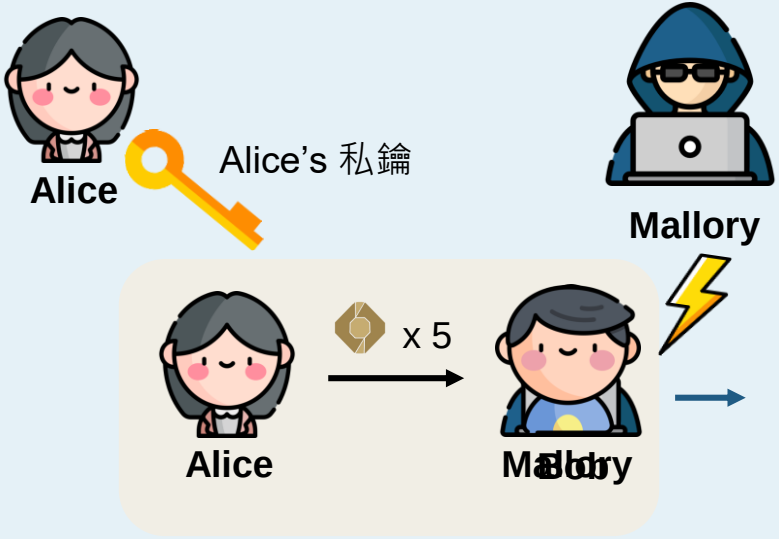
Miner

由Alice提供的公鑰
（每個人都有）

礦工將這筆交易放到
Merkle Tree中

FCoin 2.0

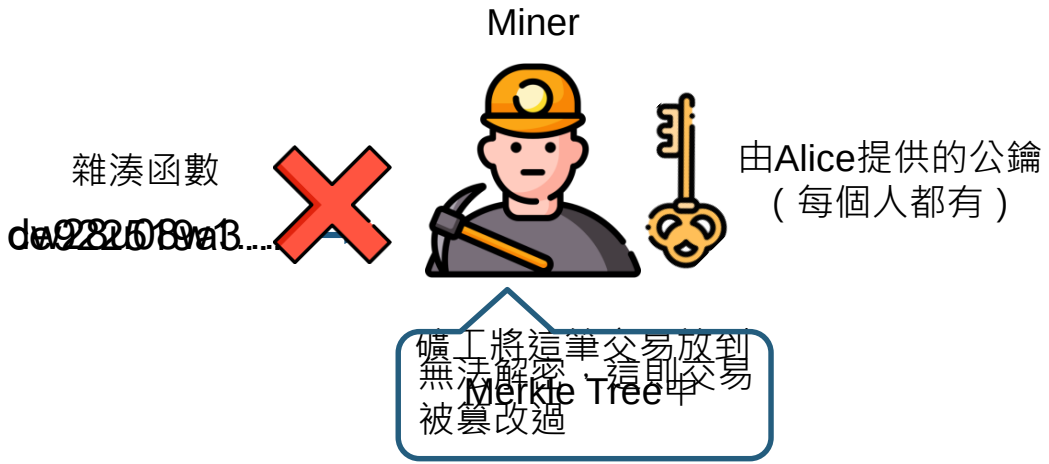
-數位簽章的重要



What if…

駭客 Mallory 竄改交易紀錄再打包？

Ans：他沒有Alice的私鑰進行數位簽章，因此礦工藉由手上Alice的公鑰會知道這筆交易有問題。



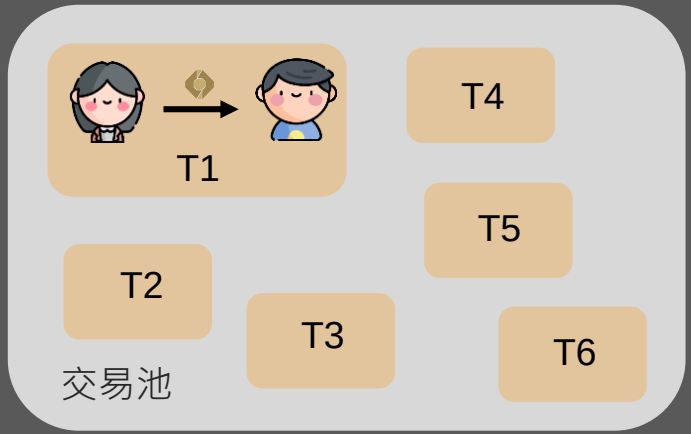
礦工的第一個工作-驗證交易

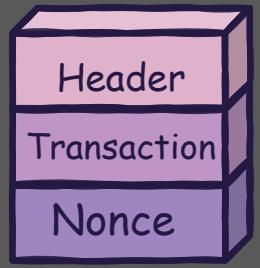
Miner

礦工幫忙驗證交易，
並將其一一放入Merkle Tree

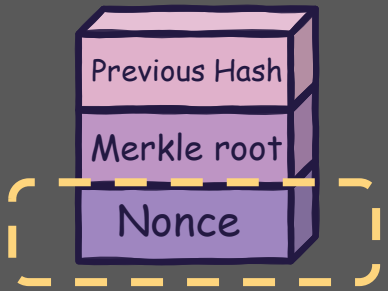
準則：手續費高的先幫忙驗證
（由程式自動執行）

交易池：放滿了等待確認的交易





礦工的第二個工作-建立區塊，開始挖礦

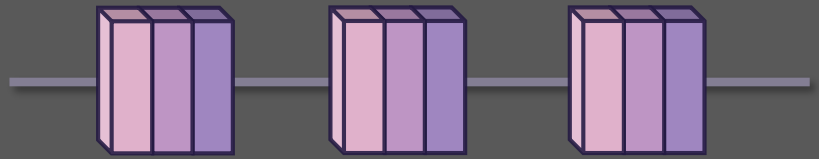


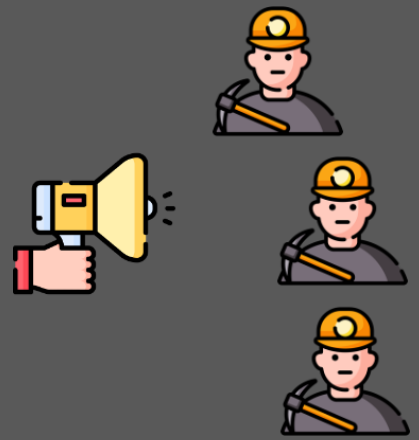
礦工電腦不斷改變Nonce的值，
直到「整個區塊」的Hash值
符合設定好的難度（Difficulty）

難度：每次改變Nonce值，整個區塊的Hash值會變得截然不同（雪崩效應），目標（Target）通常是一串前面有數個零的Hash值。比特幣將難度設定在每十分鐘出塊一次，這個難度每週都會依照礦工挖礦情形「動態調整」

礦工成功挖出區塊後

1. 將區塊放到區塊鏈上，廣播給所有礦工修正手上的區塊鏈（記帳本），提供永久透明的交易資料





2. 得到交易手續費、出塊獎勵

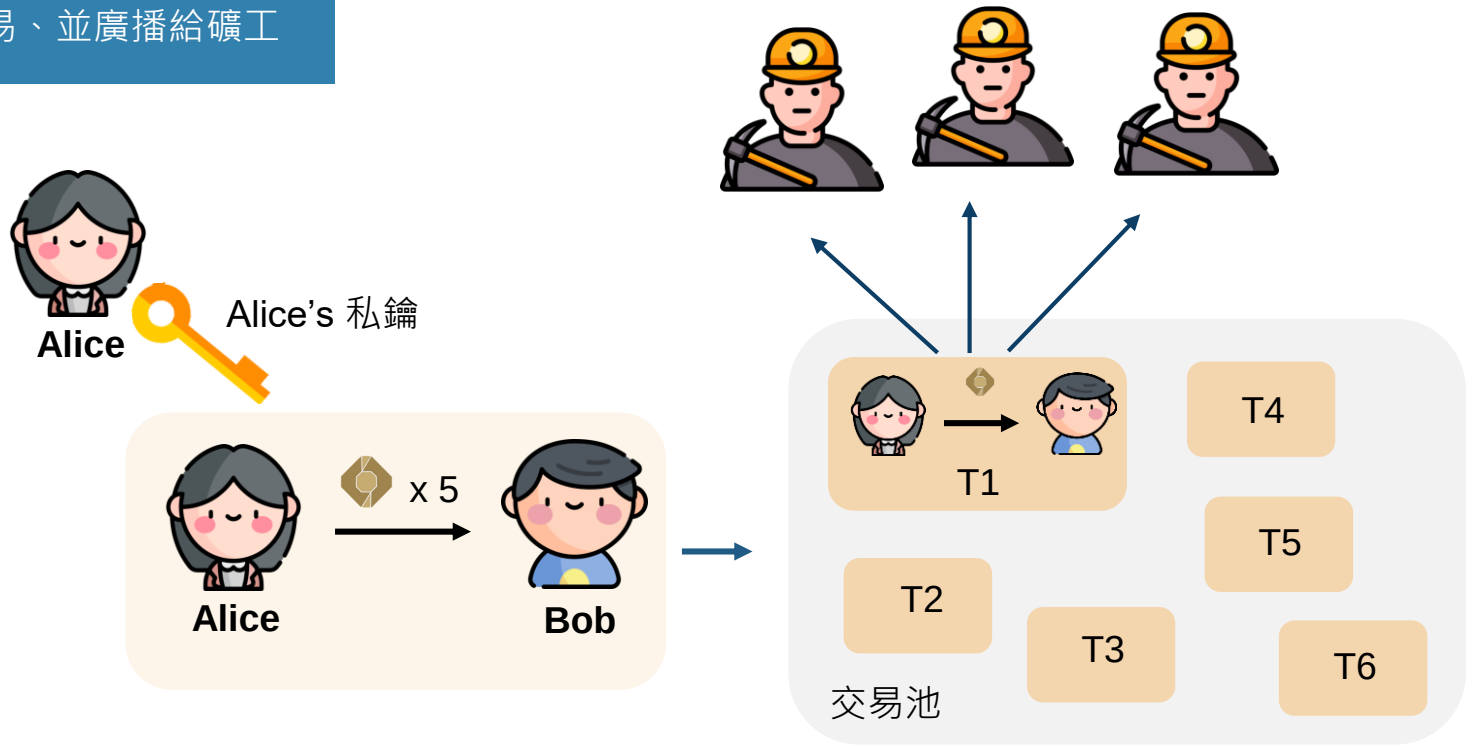
簡易區塊鏈運作流程

第一步 – 發起交易、並廣播給礦工

第二步

第三步

第四步



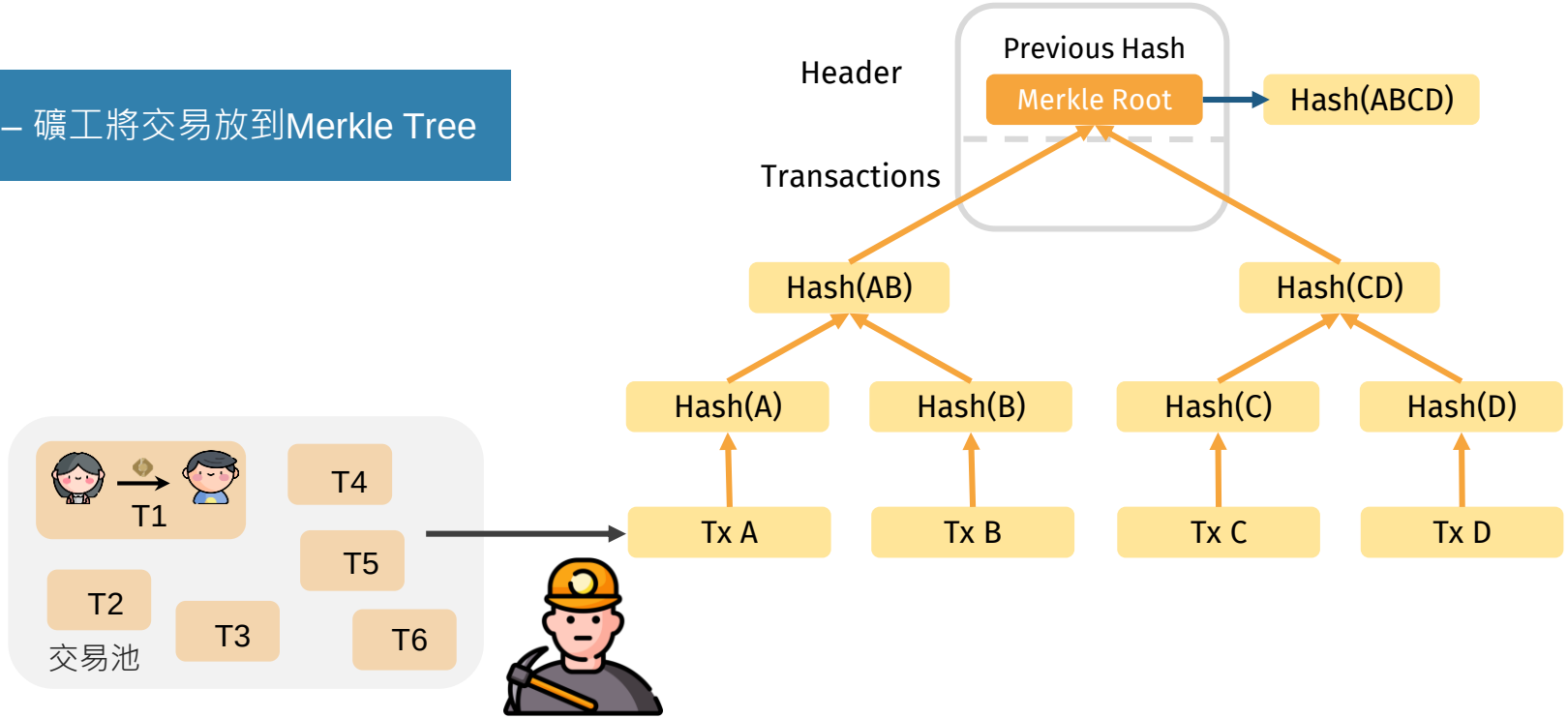
簡易區塊鏈運作流程

第一步

第二步 – 礦工將交易放到Merkle Tree

第三步

第四步



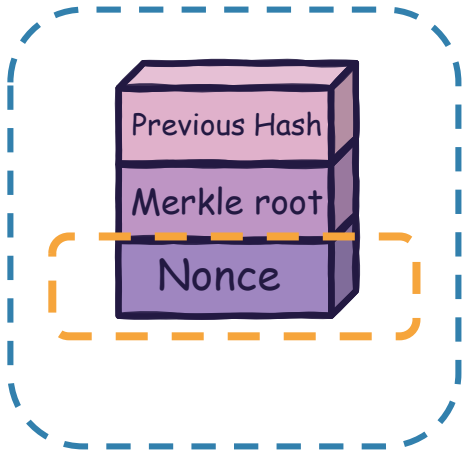
簡易區塊鏈運作流程

第一步

第二步

第三步 – 挖礦

第四步



不斷更動 Nonce值

直到 整個區塊的Hash 符合難度

Until

000000qk72m…

Difficulty：六個零

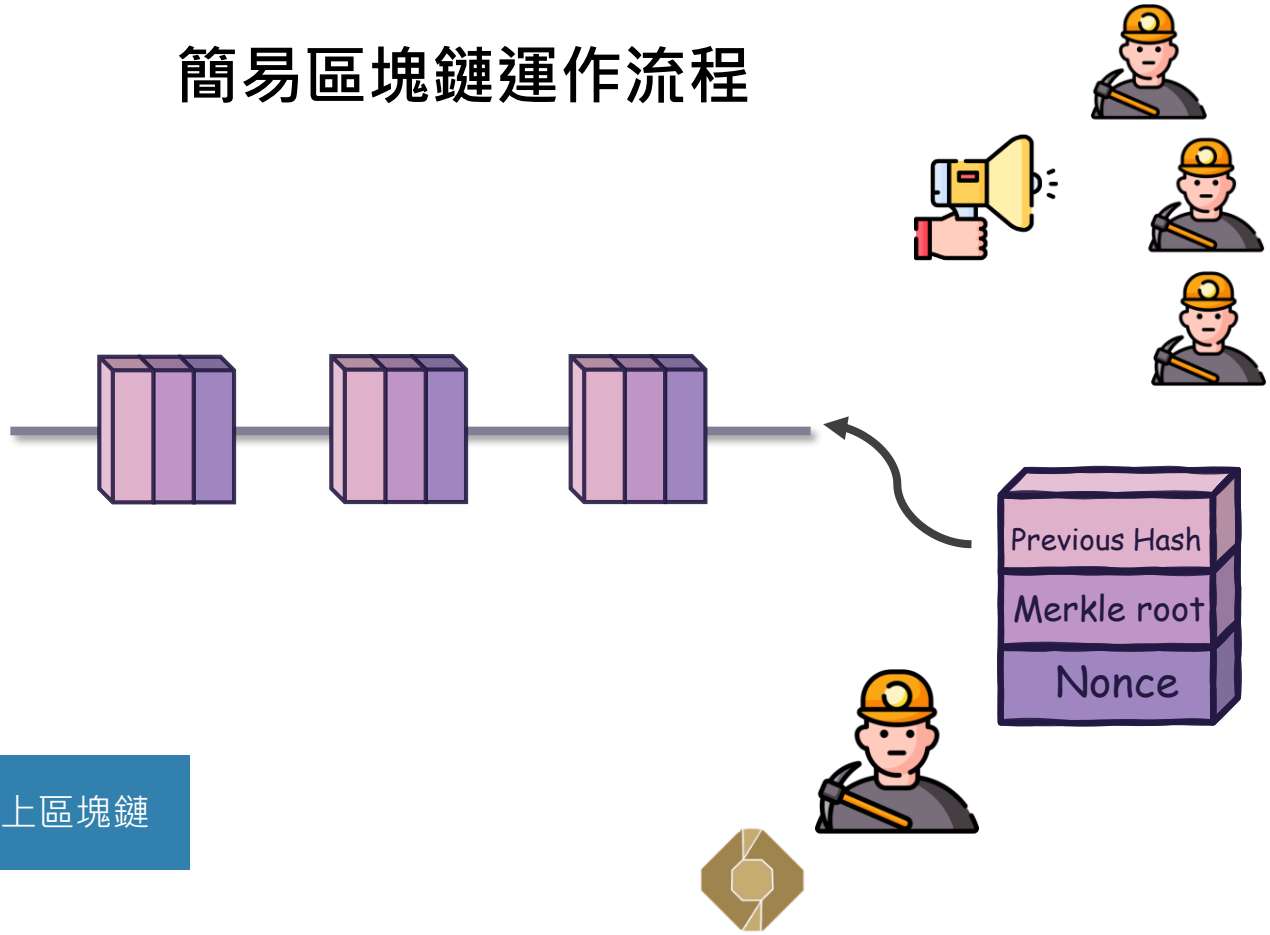
簡易區塊鏈運作流程

第一步

第二步

第三步

第四步 – 礦工將區塊放上區塊鏈



node.py

宣告一個 #475行
BlockChain物件

INIT( ): #51行
1. 初始化所有變數
2. 向別人拷貝 Clone_blockchain( )
3. 建立 Socket 連線 #51行、322行

Threading

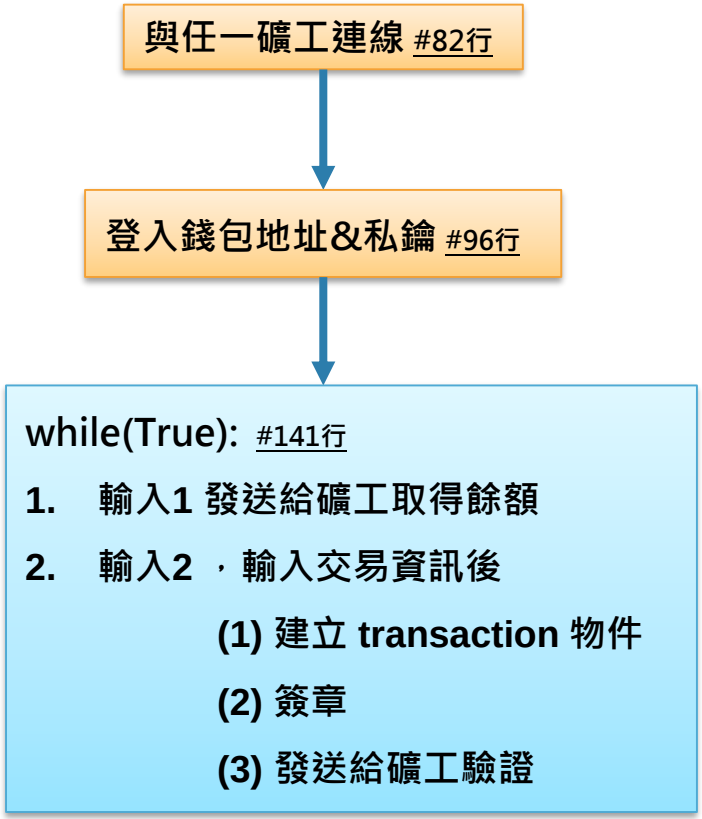
start( ): #476行、262行
1. 一組地址&私鑰
2. while True:
   mine_block( )
   adjust_difficulty( )

Socket( ):
while True:
   掛著等待新連線 #326行
   新連線建立後，再開一個子執行緒receive_message( ) #338行

Threading → connection1
Threading → connection2
Threading → connection3
.....

mine_block( ): #136行
1. 宣告一個 new_block 並初始化
2. 把交易裝進來 add_transaction_to_block( )
3. while(還沒滿足difficulty要求):
   調整 nonce
   計算哈希 get_hash(new_block,nonce)
   如果先收到別人挖到礦的消息就return
4. 挖到礦了，要廣播給大家 broadcast_block( )
5. BlockChain.append(new_block)

client.py


與任一礦工連線 #82行
登入錢包地址&私鑰 #96行
while(True): #141行
1. 輸入1 發送給礦工取得餘額
2. 輸入2 ，輸入交易資訊後
(1) 建立 transaction 物件
(2) 簽章
(3) 發送給礦工驗證

實作模擬任務（初階）

1

Difficulty 的影響

Difficulty 每 +1 變難多少

感受一下 Difficulty 每 +1 同樣算力下挖到的時間變久多少？嘗試用理論去解釋你的觀察結果？

1分 / 2分

2

轉帳測試

A 發錢給 B

A 發錢給 B ，觀察一下 A 和 B 帳戶的餘額變化？(回答合理就有算對)

1分

3

挖礦與交易間的關係

挖礦與交易

Client 端不斷的發起交易，會影響 Node 端礦工挖礦的速度嗎？為甚麼？

1分 / 2分

實作模擬任務（進階）

4 Difficulty、Block_time 的限制

調整參數

Difficulty、Block_time

調太低會發生甚麼？

為甚麼？

怎麼解決？

兩個變數二選一就好

2分 / 3分 / 3分

5 調整nonce的方式

礦工怎麼調整 nonce？

不同的調整方法，會影響挖到礦的機率嗎？

會的話，最好的方法是？

不會的話，為甚麼？

2分 / 3分

6 嘗試竄改會發生什麼事?

不可竄改性

有人先把 difficulty 等參數偷竄改後再連線，會發生什麼事情？為甚麼？

(注意：沒發生事情說明你沒真的完成竄改，想一下為甚麼)

2分 / 3分

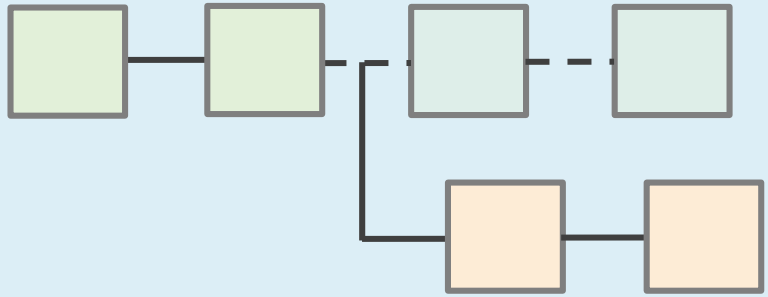
實作模擬任務（進階）

任何有趣的實驗或是發現！

51 % attack

情況：某個節點擁有50%以上的算力

會引發什麼問題？



Miner

Mallory