國立中興大學科技管理研究所

Graduate Institute of Technology Management
National Chung Hsing University

碩士學位論文

A Thesis Submitted in Partial Fulfillment of the
Requirements for the degree of Master

以區塊鏈技術探討應用程式資料之運作

A Study on the use of Blockchain Technology for Supporting Application Data Operations

指導教授: 張樹之教授 ShuChih Ernest Chang

研究生: 郭姿吟 Tzu-Yin Kuo

中華民國一百零七年六月

Graduate Institute of Technology Management
National Chung Hsing University

A Thesis Submitted in Partial Fulfillment of the
Requirements for the degree of Master

A Study on the use of Blockchain Technology for Supporting Application Data Operations

Advisor: ShuChih Ernest Chang
Graduate Student: Tzu-Yin Kuo

June 2018

國立中興大學科技管理研究所

碩士學位論文

題目： 以區塊鏈技術探討應用程式資料之運作

A Study on the use of Blockchain Technology for Supporting Application Data Operations

姓名： 郭姿吟 學號： 7104026117

經 口 試 通 過 特 此 證 明

論文指導教授 張樹之

論文考試委員

宋中華

林君維

中華民國 107 年 06 月 22 日

授權書編號：**nchu-107-7104026117-1**

中興大學博碩士論文授權書

本授權書所授權之論文為立書人在 中興大學管理學院 科技管理研究所，**106**學年度第**2**學期取得碩士學位之論文。

論文題目：以區塊鏈技術探討應用程式資料之運作
指導教授：張樹之

授權事項：

一、 立書人 郭姿吟 同意無償授權 中興大學 將上列論文全文資料之以微縮、數位化或其他方式進行重製作為典藏及網際網路公開傳輸之用。 中興大學 在上述範圍內得再授權第三者進行重製或網際網路公開傳輸等其他利用。

二、 立書人 郭姿吟 同意無償授權教育部指定送繳之圖書館及 中興大學將上列論文全文資料之紙本為學術研究之目的予以重製， 不同意全本重製(允許部分重製)，惟每人以一份為限。

三、 立書人郭姿吟 同意中興大學以有償方式再授權資料庫廠商 將前條典藏之資料收錄於廠商之資料庫，並以電子形式透過單機、網際網路、無線網路或其他公開傳輸方式授權用戶進行檢索、瀏覽、下載、傳輸、列印等。立書人可選擇之權利金方案如下列二種擇一：

※由立書人收取有償授權之權利金，其權利金額度由中興大學與再授權之廠商議定之。
※由立書人將有償授權之權利金捐贈中興大學校務基金。

四、 中興大學得將第三條之權利再授權予其他第三人進行網際網路公開傳輸等其他加值利用。

五、 前四條授權均為非專屬授權，立書人仍擁有上述授權著作之著作權。立書人擔保本著作為立書人所創作之著作，有權依本授權書內容進行各項授權，且未侵害任何第三人之智慧財產權。如有侵害他人權益及觸犯法律之情事，立書人願自行負責一切法律責任，被授權人一概無涉。

論文紙本於中興大學圖書館內公開陳列上架時間： 三年後公開

有償授權條件：享有權利金的回饋，權利金通知本人領取。
論文全文上載網路公開時間： 三年後公開
立書人： 郭姿吟

簽名： 郭姿吟　　中華民國 **107** 年 **7** 月 **9** 日

---------- 請沿此虛線對折前下，上半部為授權書，下半部請自行留存 ----------

摘要

在新興科技之中，區塊鏈應用在儲存管理系統方面具有很大的優勢。許多雲端服務供應商使用集中式伺服器提供客戶儲存資料的服務，然而，集中式伺服器仍存在單點故障的風險。我們將區塊鏈加入雲端儲存服務之中，提出新的架構來管理資料。以區塊鏈為基礎作為雲端儲存的連接器來降低資料被竊取的風險。我們利用區塊鏈獨特的屬性，如：不可篡改、可追蹤、資料共享等特性，區塊鏈能確保交易資料完整且妥善的保存在網絡中的各節點，有效防止被篡改的可能。在這項研究中，我們將資料直接寫入架設在本機端的區塊鏈，並把資料存放至雲端後再寫入區塊鏈，將兩者作了比較並分析所提出架構的性能以及可行性。本研究以股票模擬交易應用程式為例，說明透過以區塊鏈為基礎的服務來分散資料儲存，達成減少中介成本的目標，進而有效降低雲端儲存服務發生單點故障的風險。

關鍵詞：區塊鏈、數據管理服務、雲端服務、分散式系統

Abstract

Among the emerging technologies, blockchain has a promising advantage in designing distributed storage management system. There are several companies that provide cloud storage services on a centralized server to protect customer data, which could be a risk of being a single point of failure. We propose a new architecture that uses the blockchain as a cloud-based data management service connector to reduce such risk. Leveraging unique blockchain properties, such as managing authentication, confidentiality, accountability and data sharing, blockchain guarantees that all the transactions are immutable and thus it prevents them from being tampered. Blockchain establishes a new generation of transactional process which is trustable, transparent and accountable. In this study, we build a stock trading simulation game to evaluate the feasibility and performance of the proposed architecture which can reduce intermediary costs by decentralizing data storage through blockchain-based services.

Keywords: Blockchain, Architecture connector, Cloud-based, Data management Service, Distributed.

目錄

圖目錄

表目錄

1. 緒論

當今的企業環境迅速變化，企業組織不斷成長，變得日趨龐大複雜。此外，伴隨著網際網路的新興服務和應用快速發展，軟硬體技術不斷地提升，使用者對於資訊服務的需求日益增加，雲端運算服務的各種應用已逐漸在日常生活中實現。隨著雲端運算的蓬勃發展，各項雲端服務的應用也日漸成熟，使用者透過無遠弗屆的網路可隨時隨地存取雲端衍生的各項創新服務，如圖 1.1。藉由雲端服務，將公司內部的設備維護成本轉換為服務運作的成本 (Gibson, Rondeau, Eveleigh, & Tan, 2012)，用戶端不需要支付龐大的 IT 基礎建設和維護費用，雲端運算的應用更不需要安裝在每個用戶的電腦，系統的維護變得更簡單，維護成本皆由雲端服務供應商負責，強化擴充性更增加了靈活性，有效縮短用戶端對軟體的適應時間，享受雲端服務供應商所提供的強大資訊科技運算能力。

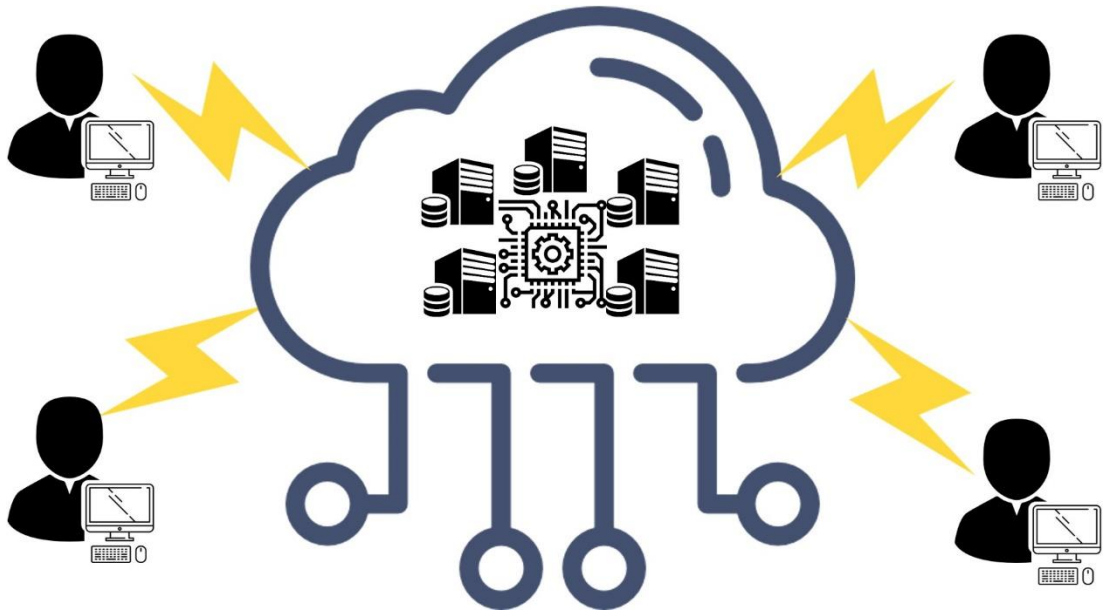


圖 1.1 雲端示意圖

然而企業組織在享受雲端服務便利性的同時，不可避免地要透過網際網路將資料從其主機移轉到雲端，並在雲端上進行資料管理 (data management)，如圖 1.2。對於企業而言，資料數據是其營運的命脈，若企業內部資料，甚至高敏感性資料都儲存在第三方雲端服務供應商的資料中心時，維護客戶資料的安全就變成雲端服務供應商的首要任務。如果沒有採取足夠的安全措施，也將面臨資料洩漏和被篡改

的安全風險，近年來知名雲端服務供應商頻傳雲端漏洞事件造成用戶密碼和個人資料外洩 (Tianfield, 2012)，使用戶對於雲端資料儲存的安全性產生疑慮。

事實上，雲端服務仍有許多安全風險需要其供應商持續改善，尤其針對中小企業而言，許多中小企業用戶採用的是集中管理式的雲端服務供應商，當企業規模越來越龐大，集中式雲端服務資料庫的縱向擴張易發生單點故障之風險 (Jansen, 2011)，在單點故障或單一路徑斷線的情況下，會造成雲端服務供應商中斷網路服務，甚至癱瘓整條產線，這種情況可能會造成企業巨大的損失 (Kirar, Yadav & Maheswari, 2016)。此外，公司內部的機密資料存放在雲端服務供應商觸手可及的地方，若發生資料毀損或外洩，對組織的影響不僅止於實質的金錢損失，更會使企業形象受到損害(Saxena, & Pushkar, 2016)。綜觀以上所述，資料存取、虛擬環境安全等問題為重點考慮項目，必須在所有雲端服務的使用上建立適當的安全性控制，才能確保資料安全、可用且容易存取，更重要的是能讓資料得以妥善運用。

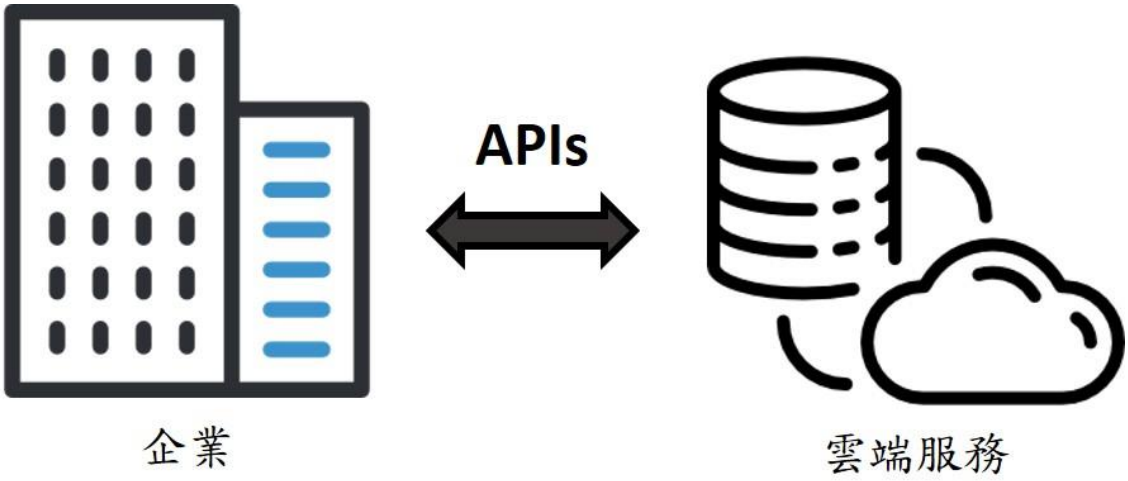


圖 1.2 資料儲存架構

針對企業和組織的操作系統、應用程式和資料庫的完全混合是技術快速發展的一部分，集中化的環境越來越無法適應需求，逐漸由原先的集中式資料庫發展向分散式資料庫，其中最明顯的好處在於增加分散在各地的使用者對資料的可使用程度，以及減少集中式系統軟硬體出錯時的整體成本。而分散式資料庫將資料及承載資料的硬體都散佈各地，允許區域的自主性，有效地降低網路的流量，增加處理的速度，可隨著企業的成長而做漸進式的擴充。即使發生了部份設施故障或錯誤，系統仍可維持工作，失效的部份可由其他部份遞補，這不但是分散式系統的特色，更提高了系統的可用性。

伴隨著比特幣風潮興起的區塊鏈技術，是一個巨大的分散式資料庫，區塊鏈的資料結構可以使網絡中的交易具可追蹤、永久性及不可篡改的特性，本研究將儲存在雲端上的資訊進行加密處理後，透過對等式網絡將使用者的數位資產及交易記錄，分散存放在網絡中不同的節點裡面，進而取代只存放在單一節點上的風險。以區塊鏈技術為基礎，將儲存在雲端的資料，加密處理後分散存放在網絡的區塊之中，使用者的檔案資料不會因為雲端伺服器故障而有外洩、被盜的風險 (Dai, Zhang, Wang & Jin, 2018)，使用者能安全地存取數據，隱私能有很好的保障。本研究分析雲端儲存可能面臨的問題，針對，希冀透過採用區塊鏈技術來降低並改善雲端儲存可能遇到的風險。

2. 研究背景

Wang 與 Chang (2016) 提出雲端服務暨股市交易實作服務，本章節以此研究為基礎，用區塊鏈的概念及技術加以改良，並建立以區塊鏈作為雲端資料管理服務的連接器。

2.1 股票交易模擬程式架構

本研究設計了一支以網路為基礎的股票交易模擬應用程式，如圖 2.1，我們首先在網頁伺服器上架設應用程式，瀏覽器和伺服器透過 HTTP 協定來溝通，使用者在自己的電腦裡面不需要安裝任何軟體，這支程式能夠運行在任何作業系統及平台上。

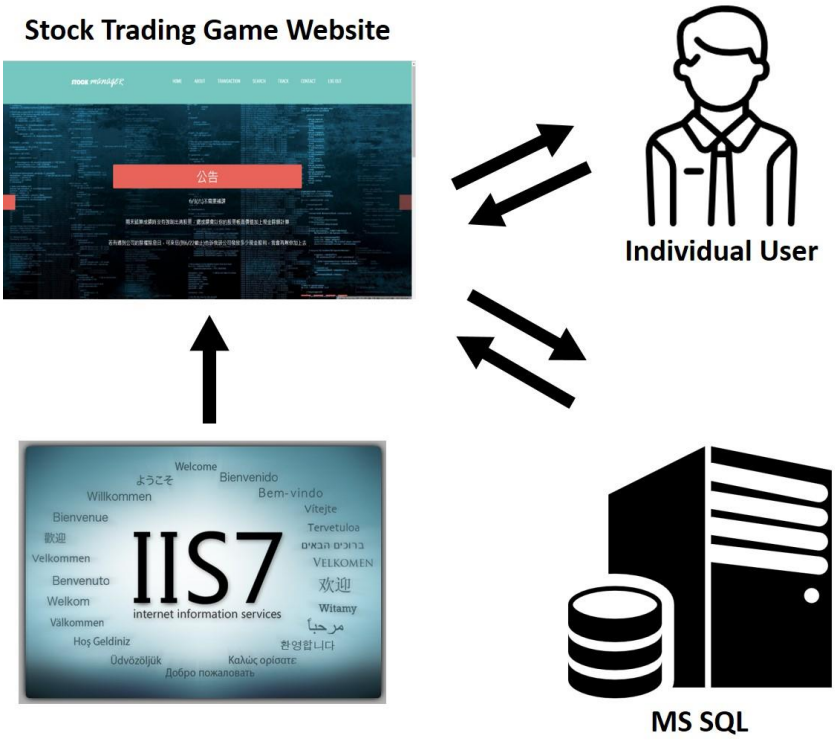


圖 2.1 架設在本機端之股票交易模擬應用程式架構

我們使用 ASP.NET 及 C#來開發這支應用程式，當中也包含了 HTML、CSS、Bootstrap 等網頁語法元素，而後端資料庫則是使用 Microsoft SQL Server。ASP.NET 內包含專門讓網頁應用程式開發的函式庫-Microsoft .NET framework，這個架構提供許多功能可以擴充及處理 ASP.NET 的網頁，這改善了網頁的可行性也讓連結更方便。

2.2 區塊鏈 (Blockchain)

中本聰在 2008 年發表的論文中提出了比特幣的概念，以及比特幣背後的技術——區塊鏈 (Nakamoto, 2008)。區塊鏈是一個巨大的全球性分散式帳本資料庫 (distributed ledger)，依靠複雜的密碼學加密資料，再透過巧妙的數學分散式演算法產生相關聯的區塊 (Swan, 2015)，每一個區塊都包含了前一個區塊的雜湊值 (hash value) 以及交易資訊，其中雜湊值可用來驗證前一個區塊內資訊的有效性並生成下一個區塊 (Tschorsch & Scheuermann, 2016) ，區塊鏈的資料結構使帳本具有分散性、不可篡改性、可追溯性等技術優勢。依據參與共識過程及應用部署類型，區塊鏈可區分為公有鏈、私有鏈以及聯盟鏈 (Buterin, 2015)。

公有鏈 (public blockchain) 是任何人皆可隨時進入系統中讀取資料、發送交易、參與競爭記帳，它就像帳本 (ledger) 一樣記錄所有的交易內容，這個帳本不是存放在銀行、政府機關、企業等集中式組織，它有無數份副本，分散存放在區塊鏈網絡中數以百萬計的節點 (node) 上，而節點是任何可以連接至網絡的設備 (Xu et al, 2017)。網絡中每個節點都儲存了一個區塊鏈資料庫，這代表它保存的記錄是真正公開並且容易驗證。區塊鏈的技術確保各節點執行相同的程式邏輯，使得網絡中的節點都持有相同一致的帳本，進而在交易雙方間建立信任，支援他們有效地直接交易，無須中間機構替他們對帳。下圖 2.2 說明區塊鏈產生區塊的步驟，使用者在節點上產生一筆新的交易時，該節點會產生一組雜湊值並將這組雜湊值發佈到網絡中的各節點，等待網絡中的各節點驗證通過後，即會將交易產生的區塊加到區塊鏈中。而公有鏈最大的特點為去中心化，網路盛行後，線上交易蓬勃發展，每一筆交易都包含了一定比例的手續費，因為每一筆交易中都需要金融機構擔任中間人來承擔交易雙方的信任風險，所以需要收取一筆手續費作為風險溢酬 (Buterin, 2017)。當發生交易糾紛時，此時金融機構就需負責協調及理賠，因此金融機構會在進行交易前，要求交易雙方提供個人資訊，以建立信任基礎，在區塊鏈中則是利用網絡上的所有節點幫忙驗證，讓需要交易的雙方直接進行點對點的交易，取代原本金融機構的職責，少了中間人的角色後，便能大幅降低交易手續費，提升交易的利潤及效率。

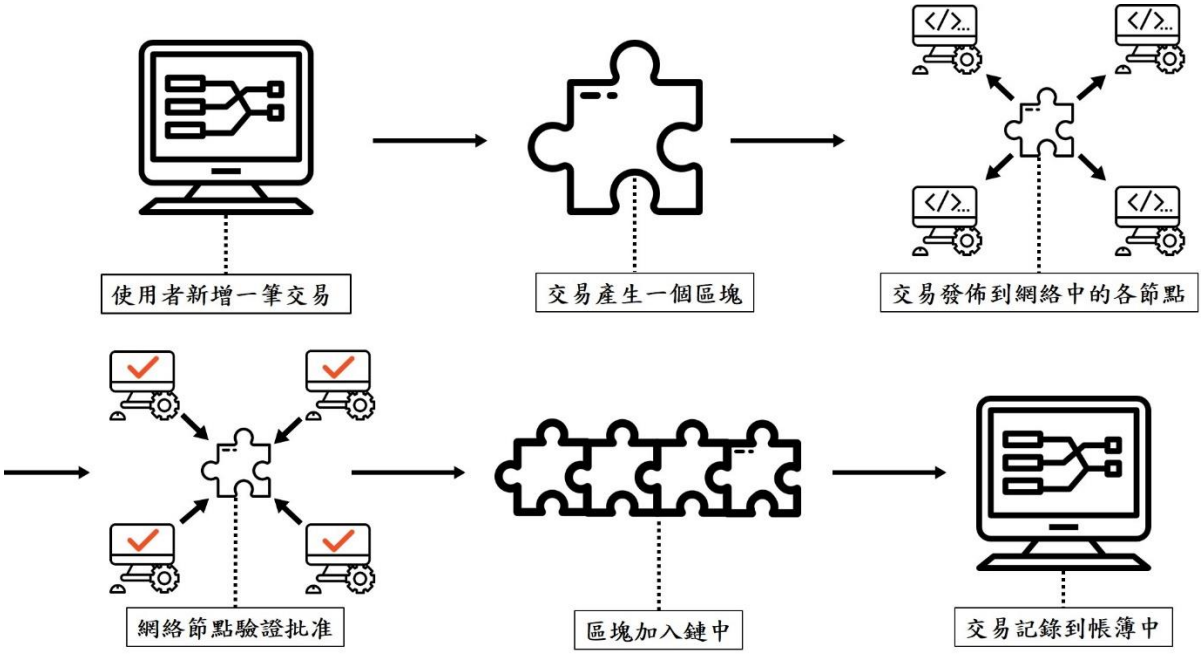


圖 2.2 區塊鏈產生之示意圖

私有鏈 (private blockchain) 是指其許可權由某個組織或機構控制的區塊鏈，節點的參與資格會被嚴格限制。由於參與的節點是備受限制的，因此私有鏈和公有鏈比較，其擁有較快的交易速度、更好的隱私保護、更低的交易成本且不容易被惡意攻擊 (Rouhani, & Deters, 2017)。相對中心化的資料庫，私有鏈能夠防止機構內單一節點故意隱瞞或者篡改數據，即便網絡中發生錯誤，也能迅速發現錯誤來源，因此許多大型金融機構更傾向於使用私有鏈技術。

聯盟鏈 (Consortium Blockchain) 是指有若干個機構共同參與管理的區塊鏈，每個機構都運行著一個或多個節點，其中數據只允許系統內的機構進行讀寫和發送交易，並且共同來記錄交易數據。聯盟鏈設計的隱私許可權較私有鏈複雜。由銀行間組成

2.2.1 雜湊函數 (Hash Function)

雜湊函數又稱雜湊演算法，為組成區塊鏈技術中不可或缺的要素。在區塊鏈的運行中，許多操作都會使用雜湊函數，輸入任何長度的字元都會計算出一組獨特且固定大小的「雜湊值」，只要輸入的文本有些微的改變，就會使雜湊值有所不同 (Yaga, Mell, Roby, & Scarfone, 2018)，如表 2.1 所示。

表 2.1 輸入值和 SHA-256 雜湊值範例

輸入文本 (Input Text)	SHA-256 雜湊值 (SHA-256 Hash Value)
0	5feceb66ffc86f38d952786c6d696c79c2dbc239dd4e91b46729d73a27fb57e9
1	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
Hello NCHU!	0b8f7072df27d464107afe6d1c6fb77c9bf16d4d72a39da2d85453ea7e02efa1

雜湊演算法具有下列 3 種特性：

- 不可逆性 (Irreversible)：雜湊的設計為單向 (one-way)，原始資料經由雜湊演算法計算出雜湊值，無法由運算後的雜湊值反推運算前的資料內容。
- 抗碰撞性 (Collision resistance)：不同的原始資料會運算出不同雜湊值，不可能找到兩個不同的資料具有相同的雜湊值，這個特性就像人類的指紋一樣，因此稱為數位指紋 (Digital fingerprint) (Lemieux, 2016)。
- 擴散性 (Diffusion)：原始資料中變動任何一個小地方都會影響到密文的各部份，即使只有些微資料不同，計算出來的雜湊值就會改變，當發現雜湊值不同，代表原始資料已被篡改過了。

2.2.2 地址 (Address)

使用者的地址由簡短的字母和數字組合而成，是將使用者的公鑰利用雜湊函數加密後加上檢查碼，再編碼為地址，地址的長度比公鑰還短。每一個私鑰/位址都會被編碼為 keyfile，它是可以透過文字編輯器編輯的明文 JSON 格式。關鍵的金鑰儲存在 keyfile 中，它被使用者用於創建帳戶時所輸入之密碼加密，而 Keyfiles 被儲存在用戶節點的子目錄 keystore 中。地址主要用於發送和接受數位資產 (digital assets)，可作為使用者在區塊鏈中的「身份」。，區塊鏈系統會利用地址作為交易的端點 (Yaga et al., 2018)，如圖 2.3。

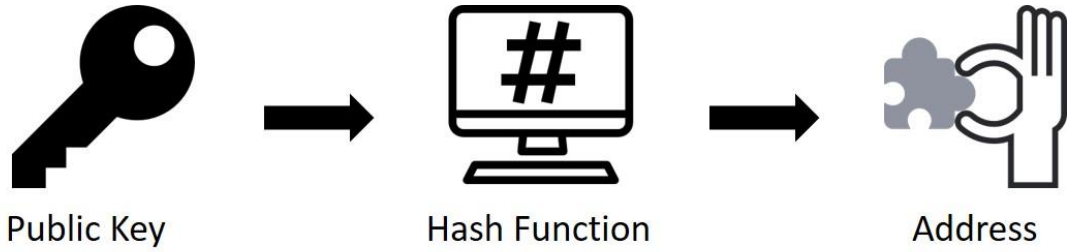


圖 2.3 地址的產生方式

當區塊鏈在交易數位資產時，會指定一個特定的地址來實現。使用者欲使用該數位資產時，必須證明自己擁有該地址的對應私鑰 (private key)，透過私鑰對交易進行數位簽章，而使用公鑰 (public key) 可對交易進行驗證。

2.2.3 區塊 (Blocks)

為了將交易訊息排出順序，因此將訊息打包成一組，稱為區塊，每個區塊所包含的資訊有 (Drescher, 2017)：區塊的大小 (block size)、區塊頭 (block header)、交易資訊，每個區塊包含若干個交易訊息和前一個區塊的雜湊值，如圖 2.4 (Yaga et al., 2018)，可以讓每個區塊與前一個區塊資料產生無形的連結，再透過時間戳 (time stamp) 來確保區塊序列及歷史記錄的正確性。每個節點都可以將若干個交易訊息打包成區塊發送到網絡上，區塊鏈系統使用加密雜湊函數設計了一道複雜的數學題，最快猜出正確答案的節點，就可以將最新區塊寫入區塊鏈。

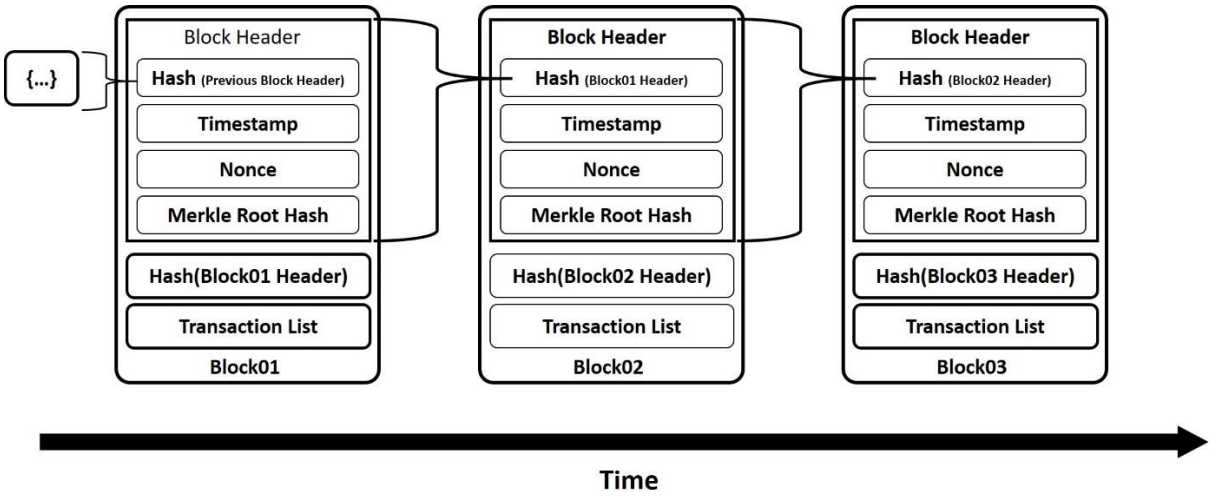


圖 2.4 區塊鏈內容與結構

區塊鏈透過共識演算法讓每個擁有交易紀錄的節點，以多數決的方式取得資

料的共識。共識決機制牽涉到每個節點的存放資料，就結果而言降低了中央控管單位因資安事故導致金融詐欺事件的風險。因此區塊鏈最主要的特性為以下幾點：

- 不可篡改性
- 可追蹤性
- 分散式帳本

2.3 智能合約 (Smart Contract)

智能合約是在區塊鏈上運行的程式之一 (Xu et al., 2016)。智能合約的概念最初是由 Nick Szabo 在 1994 年提出 (Szabo, 1994)，當年提出時尚未有合適的環境來實現，直到在比特幣白皮書中提出區塊鏈的概念後智能合約才逐步實現 (Nakamoto, 2008)。它是區塊鏈設計和應用中最關鍵的因素，由電腦程式語言編寫而成，能夠自動在區塊鏈網路節點之中執行起初所設定好的規則，負責處理與轉移具有實際價值的數位資產。開發者用程式碼撰寫合約內容並部署到區塊鏈平台上，在條件達成時，立即觸發合約內設定好的功能，自動執行合約條款。智能合約能與其他合約進行互動、依照合約內容做出決策、儲存資料與傳送以太幣 (Ether)。其運作以「事件驅動」的方式進行，藉由區塊鏈技術確保合約不會被惡意修改或偽造。

本研究中，我們設計了一種智能合約，其可將儲存在雲端上的重要資料位置加密，而這些資料能透過區塊鏈技術加密後發佈到區塊鏈上，有效降低資料被篡改的可能性。

2.4 以太坊 (Ethereum)

以太坊是一個能讓使用者自由開發智能合約的開源區塊鏈平台 (Yaga et al., 2018)，其提供開發者開發 DApp (Decentralized app) 時所需的資源、工具及基礎設備，降低了 DApp 開發的成本和門檻。除了使開發者方便創建 DApp 之外，同時也能讓各種 DApp 直接運行在以太坊平台上。

以太坊提供一種名為 Solidity 的程式語言 (Komhar, 2017)，專門用來撰寫智能合約，讓開發人員能夠建立和發佈下一代分散式應用，如：電子投票、電子商務、小額付款、拍賣等智能合約，開發者與使用者依據使用量繳交以太坊上的乙太幣作為費用。以太坊包含兩種類型的帳戶：外部帳戶和合約帳戶。本研究以合約帳戶為主，合約帳戶由程式碼控制，帳戶之間可相互傳遞訊息以實現圖靈完備 (Turing Complete) 運算。合約帳戶用以太坊上特定的二進制碼 (Binary Code)，其透過以太坊虛擬機器 (Ethereum Virtual Machine, EVM) 運行於區塊鏈上。當合約帳戶受到觸發，合約內部的程式碼將被啟動，使合約對內部執行資料讀取、寫入或發送其他消息等相關功能。

2.4.1 Geth

Geth 是 Go-ethereum 的簡稱，以太坊的官方客戶端 (Client)，用 go 語言編寫而成，是目前最廣泛使用的客戶端，透過 geth 可以實現以太坊的各式功能 (Antonio Madeira, 2017)，如：創建新的帳戶以及帳戶的編輯和刪除、讓節點開啟挖礦的工作、以太幣的轉移、智能合約的部署和執行等功能，其支援 Windows、Linux、Mac OS 作業系統。為了因應不同的需求，以太坊客戶端還有用 C++, Ruby, Python, Java 等其他多種語言編寫而成的，本研究主要以 geth 使用為主。

2.4.2 web3.js

Web3.js 是以太坊提供的一個 Javascript 的函式庫 (Vogelsteller, Kotewicz, Wilcke, & Oance, 2018)，允許使用者透過 HTTP 或 IPC 與本機端或遠端的以太坊節點進行連接，程式碼如下：

```
// in node.js use: var Web3 = require('web3');
var web3 = new Web3(Web3.givenProvider || ws://localhost:8545");
```


2.5 資料直接存放區塊鏈

本研究的測試資料是我們所做的股票交易模擬應用程式，首先在伺服器上架設私有鏈網絡，並使用權威證明 (Proof of Authority, PoA) 做為網絡中的共識演算法，有授權的節點才能產生新的區塊。之後，再此私有鏈網絡中創立使用者帳戶並得到地址，最後將原本經由網頁伺服器 (web server) 下單，進行股票交易的相關資訊存放到後端 MSSQL 資料庫的部份，替換成透過發佈智能合約將資料存放到區塊鏈上面，如圖 2.6。

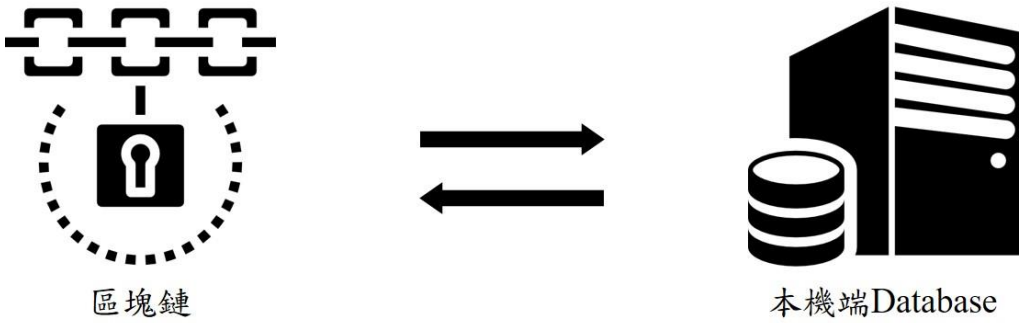


圖 2.5 資料直接存放區塊鏈示意圖

2.6 將存放在雲端的資料位址寫入區塊鏈

我們基於 2.5 節所述的方法建立第二種方法，在雲端上建立股票交易模擬程式的使用者介面 (UI)，並架設網頁伺服器和資料庫。首先使用者創立帳號，將帳號、密碼、下單時間加密後寫入區塊鏈並作為指標 (pointer)，指向資料儲存在雲端的位置 (Xu et al., 2016)。使用者需要拿到完整的交易資料時，再透過區塊鏈作為連接器去解密資料儲存在雲端上的位置，取得位置後，使用者可直接運用之前將帳號、密碼及下單資料加密後取得的雜湊值 (Transaction Hash)，登入股票模擬交易系統下載交易資料。如圖 2.7。

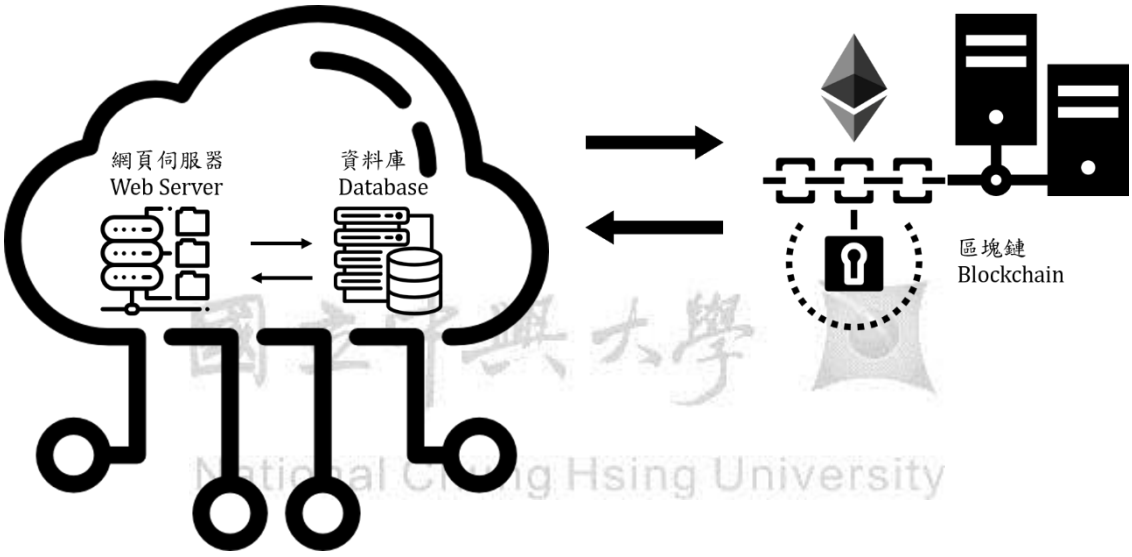


圖 2.6 區塊鏈作為連接器間接存取資料示意圖

3. 研究方法

傳統資料庫使用主從式 (client-server) 網路架構，在這種結構中，使用者可以修改儲存在中央伺服器中的資料。中央機構仍保留資料庫的控制權，在使用者連上資料庫前對其進行身份驗證。中央機構負責資料庫的管理，若授權機構遭受攻擊，安全性受到損害，資料可能面臨被篡改、遺失或是被刪除的風險，而機構內具有資料庫訪問權限的管理人，同樣也存在修改資料的風險 (Kirar, Yadav, & Maheswari, 2016)。近年來雲端運算透過網際網路能跨平台、跨地點來存取伺服器、資料庫和各種應用程式服務，雲端儲存更為熱門，但承如前文所述，當機密資料掌握在第三方雲端服務供應商手中時，洩漏的風險相對提高。區塊鏈資料庫由數個節點組成，所有節點都會參與資料管理，在區塊鏈資料庫要新增任何資料，必須得到區塊鏈網絡中多數節點達成一致才能成功寫入，被記錄到區塊鏈資料庫中的資料，將散布於各個節點之中，無法被單一實體掌控。這樣的機制使資料具備安全、透明及可永久記錄等特性，也讓篡改內容變得非常困難。

本研究針對區塊鏈資料庫提出兩種研究方法：第一種是將資料直接寫入區塊鏈，使用者可自行存取資料；第二種是將資料存放在雲端上，將區塊鏈作為連接器，用來加解密雲端存放資料的位置，詳細內容於下文進行說明。

3.1 資料直接存放區塊鏈

本研究將股票模擬交易程式架設在本機端伺服器上，並將區塊鏈環境架設於測試筆電上。在這支程式中我們增加下載檔案的功能，使用者可以透過 HTTP 協定將網站上的交易記錄下載至自己的電腦，並寫進區塊鏈系統，如圖 3.1。

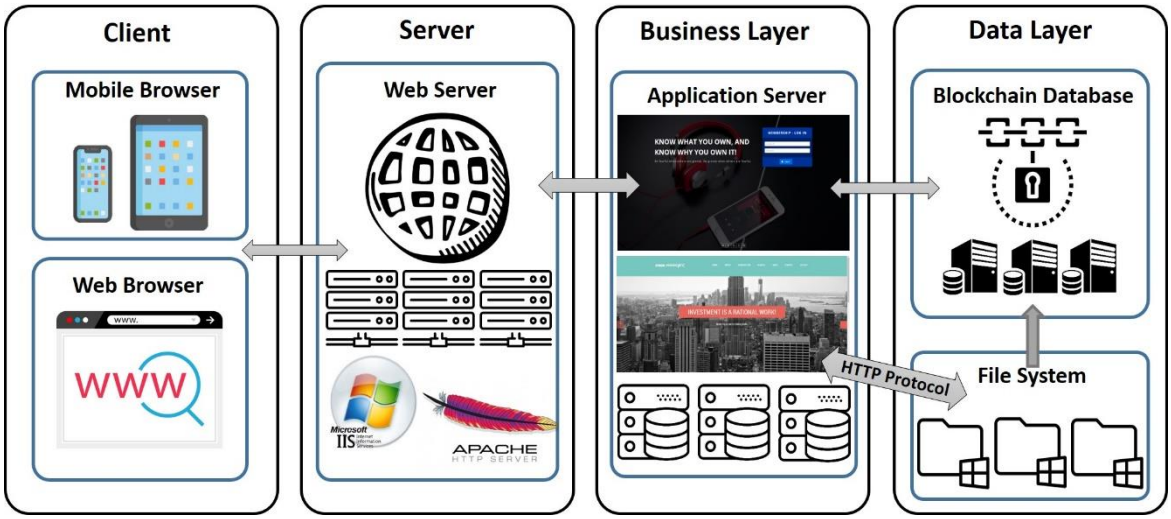


圖 3.1 網頁應用架構圖

我們使用對稱金鑰加密法，將寫進區塊鏈的資料進行加密，進一步保護資料的完整性。我們設計了兩種版本將資料寫入區塊鏈，其一是一鍵完成，將資料寫進區塊鏈，另一版本則是設計了使用者介面，逐步將資料寫入區塊鏈，如圖 3.2，下文將逐一說明本研究所設計的區塊鏈系統。

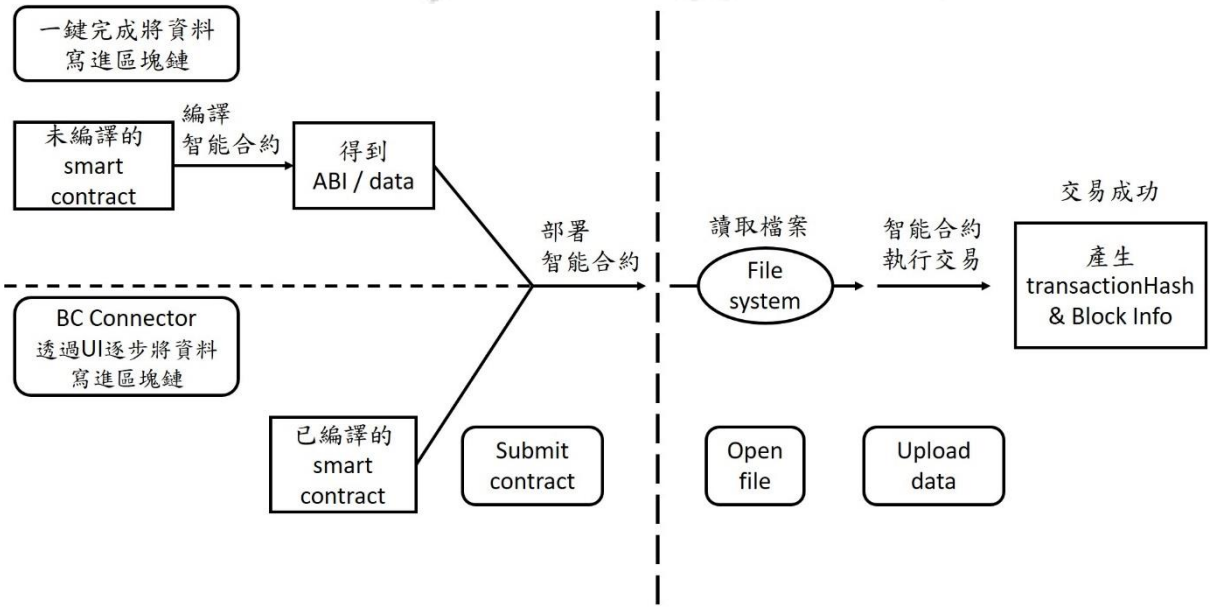


圖 3.2 資料直接寫入本機端區塊鏈之流程圖

3.1.1 區塊鏈環境設定

本論文將區塊鏈架設在 Windows 10 作業系統底下，運行區塊鏈的環境時所使用的工具如下：

- Python 2.7.14
- OpenSSL v1.0.2o Win64
- Node.js v8.9.3
  - File System
  - AES-128-CBC 對稱金鑰加密法
- Git 2.15.1.windows.2
- Microsoft.NET 11.0.9165.1186
- Geth/v1.8.6-stable
- Solidity Compiler
- Microsoft Visual Studio Professional 2015

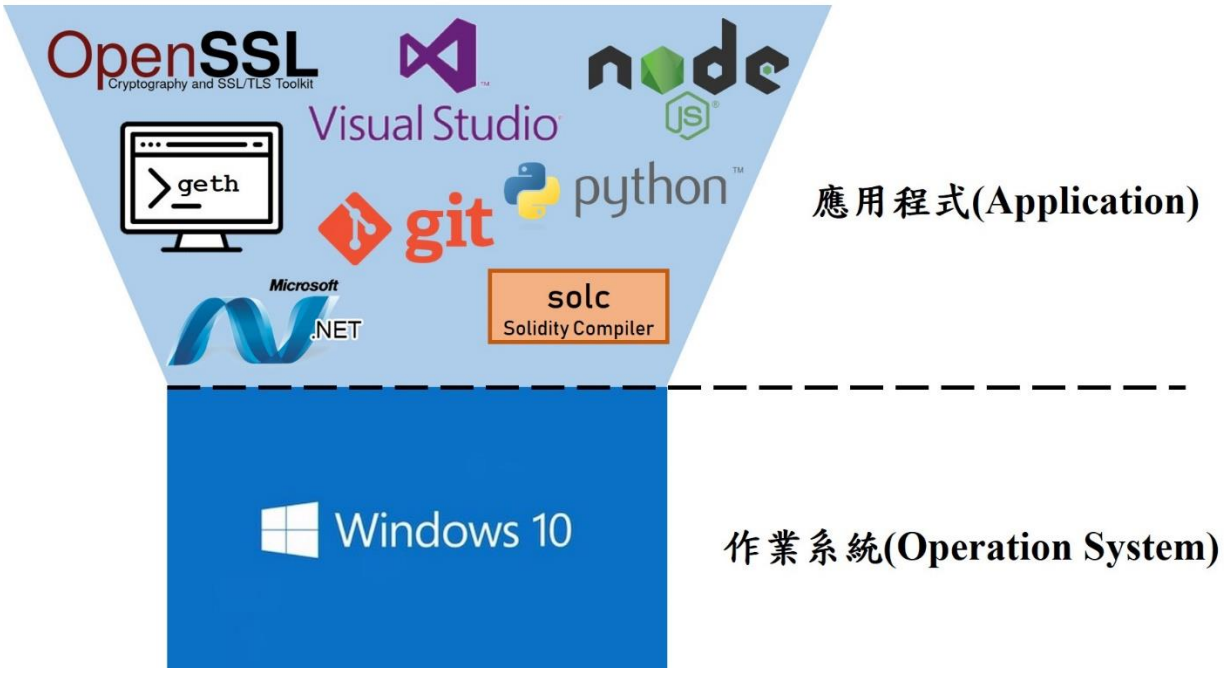


圖 3.3 區塊鏈環境設定示意圖

我們使用以太坊客戶端架設私有鏈網絡，而我們運用編寫智能合約最熱門的程式語言之一—Solidity 來撰寫智能合約，在智能合約提交給區塊鏈網絡之前需要進行編譯及部署，系統架構圖如圖 3.4。

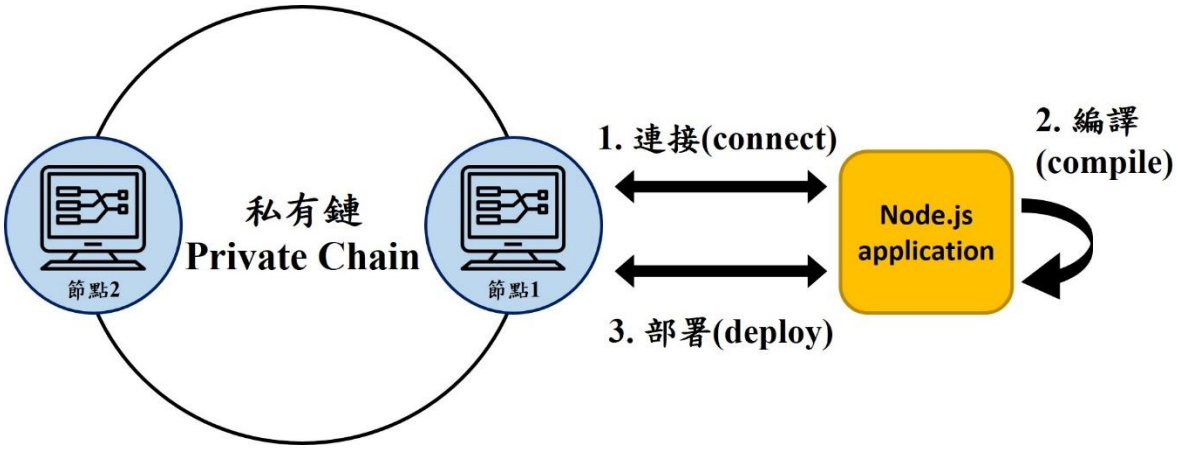


圖 3.4 系統架構圖

在這個私有鏈網絡中包含兩個授權節點 (authority node)，以 node.js 開發應用程式，並使用 web3.js 模組 (module)，透過 JSON RPC API 與節點 1 溝通。

3.1.2 建立創世區塊 (Genesis block) 設定

在網路上同時存在數百條甚至數千條的以太鏈，當以太坊上的兩個節點連接上時，我們可以透過網絡識別碼 (network id) 及創世區塊知道彼此是否連接上同一條鏈。使用者可以創造一條和以太坊公有鏈一樣的創世區塊，但網絡識別碼不一樣的鏈，也可以創造和公有鏈創世區塊不一樣但網絡識別碼一樣的鏈。若這兩項的值皆相同的話，那就必須比較各條鏈的長度，短鏈的資料會被較長的一方覆寫，變成對方的鏈。

通常公有鏈和私有鏈的資料會分別存放在不同的地方，我們利用資料夾參數指定私有鏈資料儲存的資料夾。如下：

```
geth --datadir .ethereum-private --networkid 123456
```

在建立一條新的鏈前，需要先初始化創世區塊，如下方的程式碼，我們先創建出一個 JSON 格式的檔案，裡面記錄創世區塊。只有創世區塊會有 alloc (Brandon Arvanaghi, 2018)，指定地址在初始有多少 wei (以太幣的其中一種單位)。


```
genesis.json
{
  "config": {
    "chainId": 15,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "difficulty": "0x1",
  "gasLimit": "2100000",
  "alloc": {
     "78613609b7744e3beca66cddf45d6762f5a424f3": {
     "balance": "40000000000000000"
  }
}
```

將檔案存成 genesis.json，再用 init 參數進行初始化，如下：

```
geth --datadir .ethereum-private init genesis.json
```

初始完成後出現 Successfully wrote genesis state 的訊息，如圖 3.5，就可以進入主控台視窗 (Console) 了。

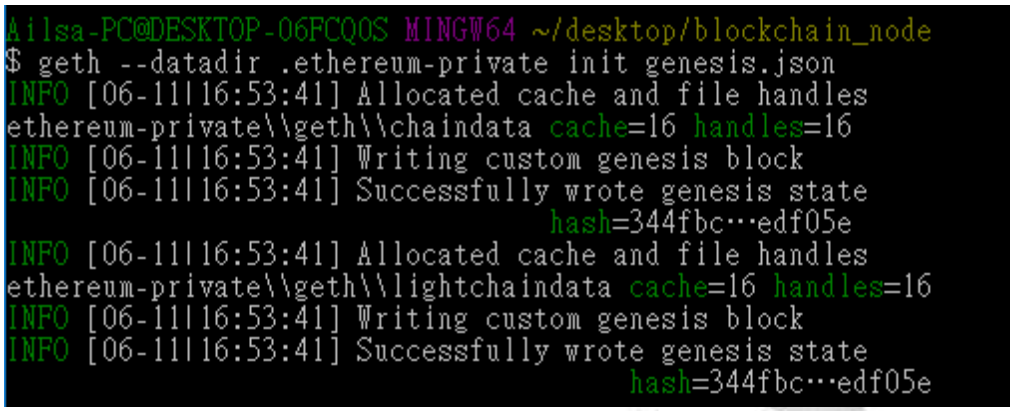


圖 3.5 創世區塊完成初始化

若要進入建立好的私有鏈，每次都要指定--datadir、--networkid 等參數，若沒有指定則會連到公有鏈上，而--maxpeers 可用來限制節點連線總數，--nodiscover 則可取消 P2P 自動節點搜尋的功能，但仍可透過手動加入或被加入。

3.1.3 啟動以太坊節點

首先，下指令開啟 geth 客戶端，完整程式碼如下：

```
geth --datadir .ethereum-private mine –minerthreads 1 –
rpccorsdomain "*" --rpc --rpccapi "eth,web3,personal" –
-networkid 123456789 –maxpeers 0 console 2>> .ethereum-
private.log
```

輸入上述指令後，如成功便會顯示 Welcome to the Geth JavaScript console! 如圖 3.6，成功啟動後礦工便可以開始挖礦。

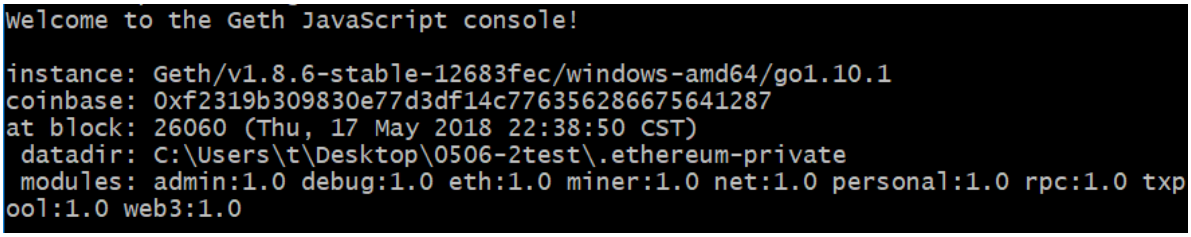


圖 3.6 啟動 geth 客戶端

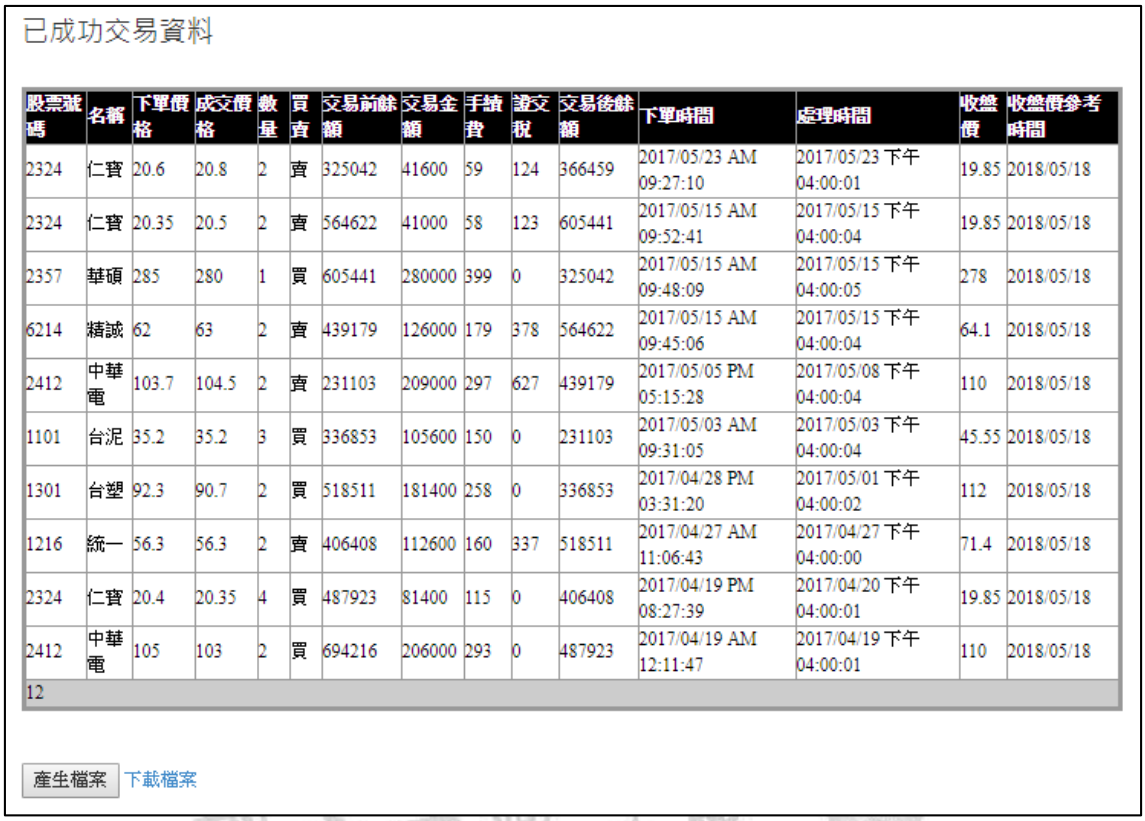
3.1.4 下載檔案

我們在股票模擬程式中新增下載檔案的功能按鈕，如圖 3.7，讓使用者可以將在股票模擬程式中的交易記錄下載至自己的電腦中，再透過BC Connector寫入區塊鏈。

首先，點選「產生檔案」，交易記錄會存到暫存區中，再點選「下載檔案」，資料即會下載一個 CSV 格式的資料到使用者的電腦，如圖 3.7，而下載檔案的主要程式碼如下。

```
using System;
using System.IO;

namespace CreateExcel1
{
    Public partial class download: System.Web.UI.Page
    {
        Protected void Page_Load (object sender, EventArgs e)
        {
            Response.Clear();
            Response.AddHeader ("contect-disposition",
              "attachment;filename=" + Server.UrlEncode ("Data" +
              DateTime.Now.ToString ("yyyyMMddHHmm") + ".csv"));
            //設定標頭
            Response.ContentType = "application/octet-stream";
            Response.BinaryWrite (File.ReadAllbytes
              (AppDomain.CurrentDomain.BaseDirectory + "test.csv"));
        }
    }
}
```



已成功交易資料

股票號碼	名稱	下單價格	成交價格	數量	買賣	交易前餘額	交易金額	手續費	證交稅	交易後餘額	下單時間	處理時間	收盤價	收盤價參考時間
2324	仁寶	20.6	20.8	2	賣	325042	41600	59	124	366459	2017/05/23 AM 09:27:10	2017/05/23 下午 04:00:01	19.85	2018/05/18
2324	仁寶	20.35	20.5	2	賣	564622	41000	58	123	605441	2017/05/15 AM 09:52:41	2017/05/15 下午 04:00:04	19.85	2018/05/18
2357	華碩	285	280	1	買	605441	280000	399	0	325042	2017/05/15 AM 09:48:09	2017/05/15 下午 04:00:05	278	2018/05/18
6214	精誠	62	63	2	賣	439179	126000	179	378	564622	2017/05/15 AM 09:45:06	2017/05/15 下午 04:00:04	64.1	2018/05/18
2412	中華電	103.7	104.5	2	賣	231103	209000	297	627	439179	2017/05/05 PM 05:15:28	2017/05/08 下午 04:00:04	110	2018/05/18
1101	台泥	35.2	35.2	3	買	336853	105600	150	0	231103	2017/05/03 AM 09:31:05	2017/05/03 下午 04:00:04	45.55	2018/05/18
1301	台塑	92.3	90.7	2	買	518511	181400	258	0	336853	2017/04/28 PM 03:31:20	2017/05/01 下午 04:00:02	112	2018/05/18
1216	統一	56.3	56.3	2	賣	406408	112600	160	337	518511	2017/04/27 AM 11:06:43	2017/04/27 下午 04:00:00	71.4	2018/05/18
2324	仁寶	20.4	20.35	4	買	487923	81400	115	0	406408	2017/04/19 PM 08:27:39	2017/04/20 下午 04:00:01	19.85	2018/05/18
2412	中華電	105	103	2	買	694216	206000	293	0	487923	2017/04/19 AM 12:11:47	2017/04/19 下午 04:00:01	110	2018/05/18
12														

產生檔案 下載檔案

圖 3.7 下載檔案

3.1.5 編譯及部署智能合約

我們使用 solidity 撰寫智能合約的原始碼，將智能合約原始碼編譯成可在以太坊虛擬機上執行的 bytecode，並取得智能合約的 ABI，而 ABI 中記錄這個智能合約提供的功能及應該要傳入的參數值。指令碼如下：

```
echo "var greeterCompiled=`solc --optimize --combined-
json abi,bin,interface record.sol`" > record.js
```

下方圖 3.8、圖 3.9、圖 3.10 為第一種版本，一鍵完成將資料寫入區塊鏈。

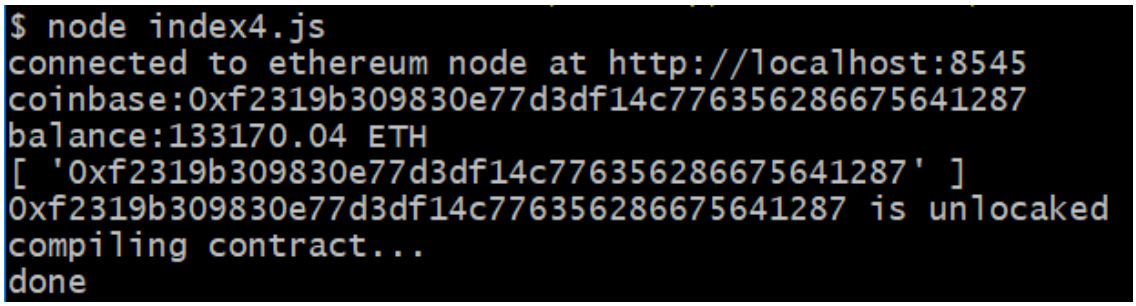


圖 3.8 連接以太坊節點及編譯智能合約

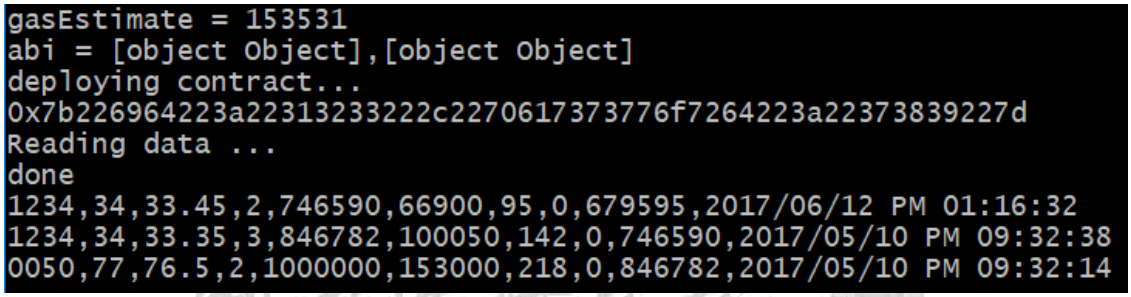


圖 3.9 部署智能合約及叫出欲寫入區塊鏈之數據

圖 3.10 產生 Transaction Hash、Block Info 及印出寫入區塊鏈的資料

下方圖 3.11 提供第二種版本，為了讓使用者方便操作，本研究設計了使用者介面，讓使用者能逐一操作每個步驟。首先，使用者按下上方的 (1) Submit Contracts 按扭後，在 (2) Smart Contract 欄位會產生 address，接著在上方的功能列點選 File 後點選 Open File，並選取欲寫入區塊鏈的檔案。確定檔案後檔案路徑會顯示在 (3) 的地方，檔案內容會顯示於 (4) 的資料欄，再按下 (5) Upload Data 後，資料就會開始被寫入區塊內。當資料寫進區塊鏈後會在 (6) 產生出加密後的 Transaction Hash，而使用者也可以透過下方的 (7) Transaction Record Inquiry 查詢寫進區塊鏈的資料，查詢的資料會顯示在區域 (8)。

為了使資料的安全性更高，我們使用 AES-128-CBC 對稱金鑰加密法，將準備寫入區塊鏈的資料先加密，在下方查詢資料欄所顯示的 Encrypt code 就是加密過後的亂碼，即使被有心人士拿到了 Transaction Hash，在區塊鏈中也看不到檔案內容。

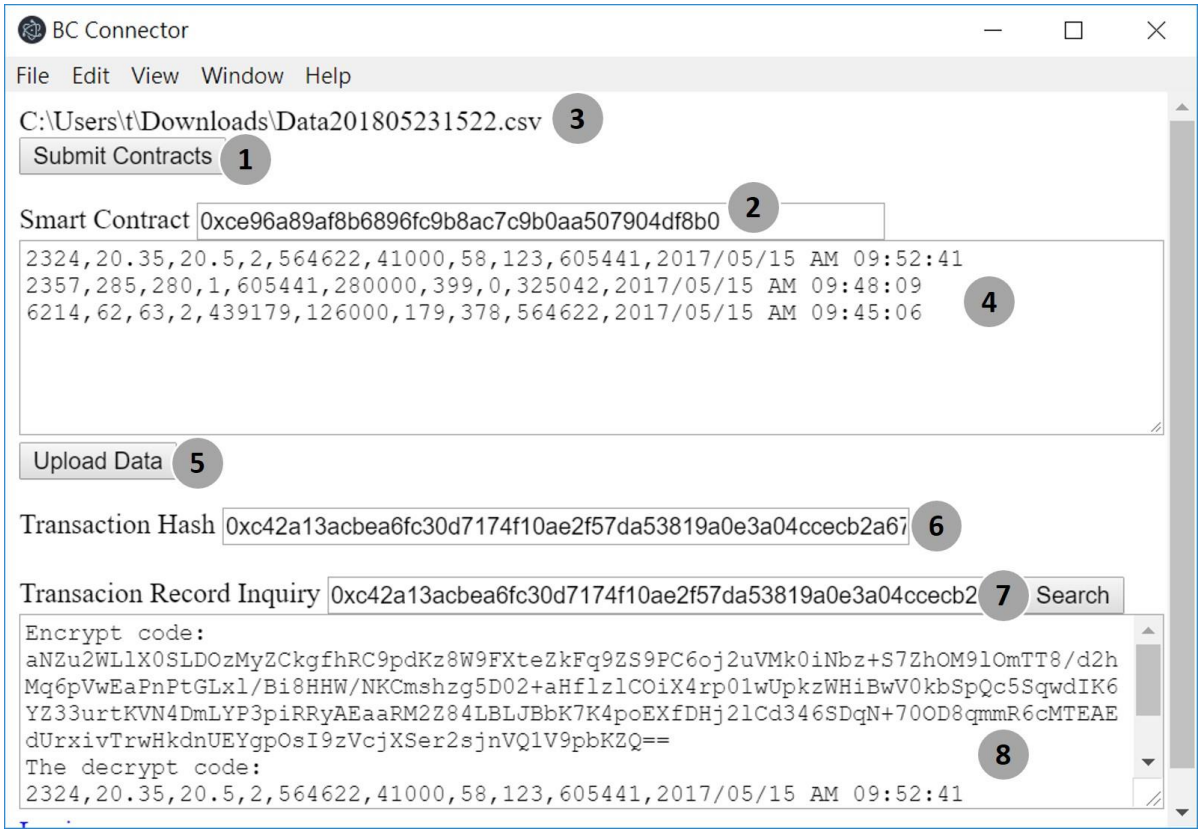


圖 3.11 BC Connector 使用者介面

智能合約會創造一個 instance，而 instance 將會被放上區塊鏈，如圖 3.12，運行在以太坊虛擬機的每一個以太坊節點上面，一旦智能合約部署成功時會得到一

個地址 (address)，它就像記憶體位置一樣，取得這個位置後搭配正確的介面資訊就可以執行這個智能合約，如圖 3.13。

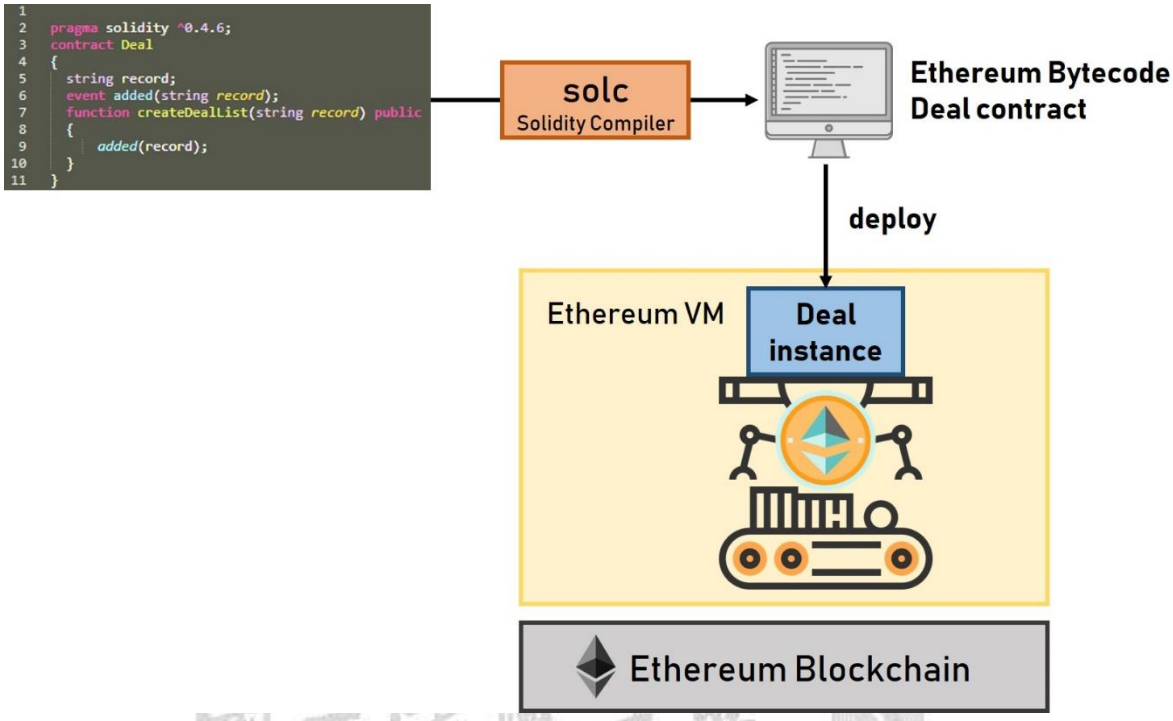


圖 3.12 以太坊虛擬機運行在以太坊節點示意圖

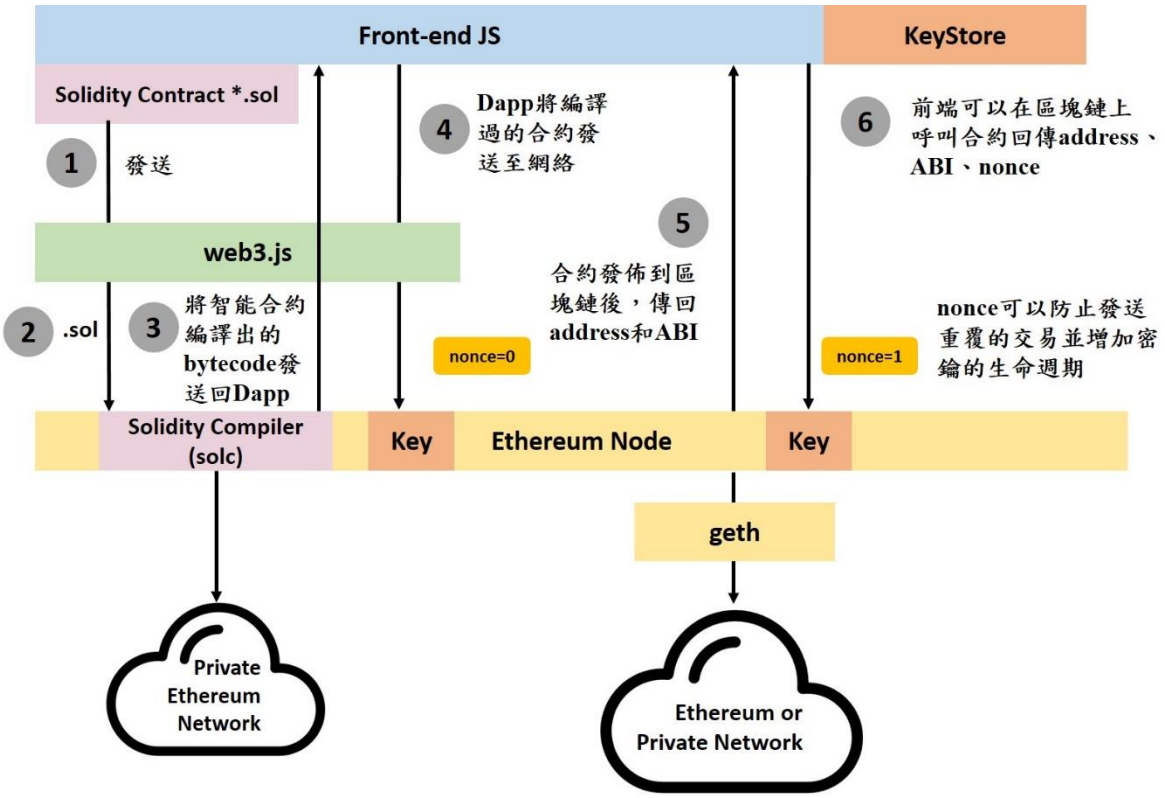


圖 3.13 部署智能合約工作流程圖

3.2 將存放在雲端的資料位址寫入區塊鏈

雲端儲存服務已被廣泛運用在數據傳輸和資料儲存的過程，帶來了許多便利，但同時也存在著一些問題 (Januzaj, Ajdari, & Selimi, 2015)。多數的使用者會利用雲端來儲存資料，而大量的資料存放在集中式的雲端伺服器上，容易受到駭客的攻擊，導致資料的安全性面臨威脅。因此，本研究將網頁伺服器、資料庫、抓股價程式及處理交易的程式放上雲端系統，讓使用者透過瀏覽器買賣股票，藉此收集資料，並利用此資料作為寫入區塊鏈的測試資料。

3.2.1 雲端架設

我們將原先架設在本機端的股票模擬交易程式移轉到 Azure 雲端平台上，收集使用者透過這支模擬程式進行股票買賣所產生的資料作為實驗依據。

圖 3.14 股票模擬交易程式運行在 Azure 雲端平台

3.2.2 以區塊鏈作為連接器

我們使用 JavaScript、HTML 和 CSS 建構跨平台的桌面應用程式 Electron 來設計開發我們的區塊鏈連接器使用者介面。因為 Electron 的後端使用 node.js，前端則使用 JavaScript html，透過 React 函式庫使開發更便利。

在啟動 geth 之後，隨即開啟另一個 Git Bash 視窗，搭配 electron 指令來執行我們的專案。將帳號、密碼及下單時間作為使用者存放交易資料的指標 (pointer)，加密後寫入區塊鏈後，可直接登錄股票模擬交易程式的交易記錄頁面查詢指定下單日期的交易資料，如圖 3.15 及圖 3.16。

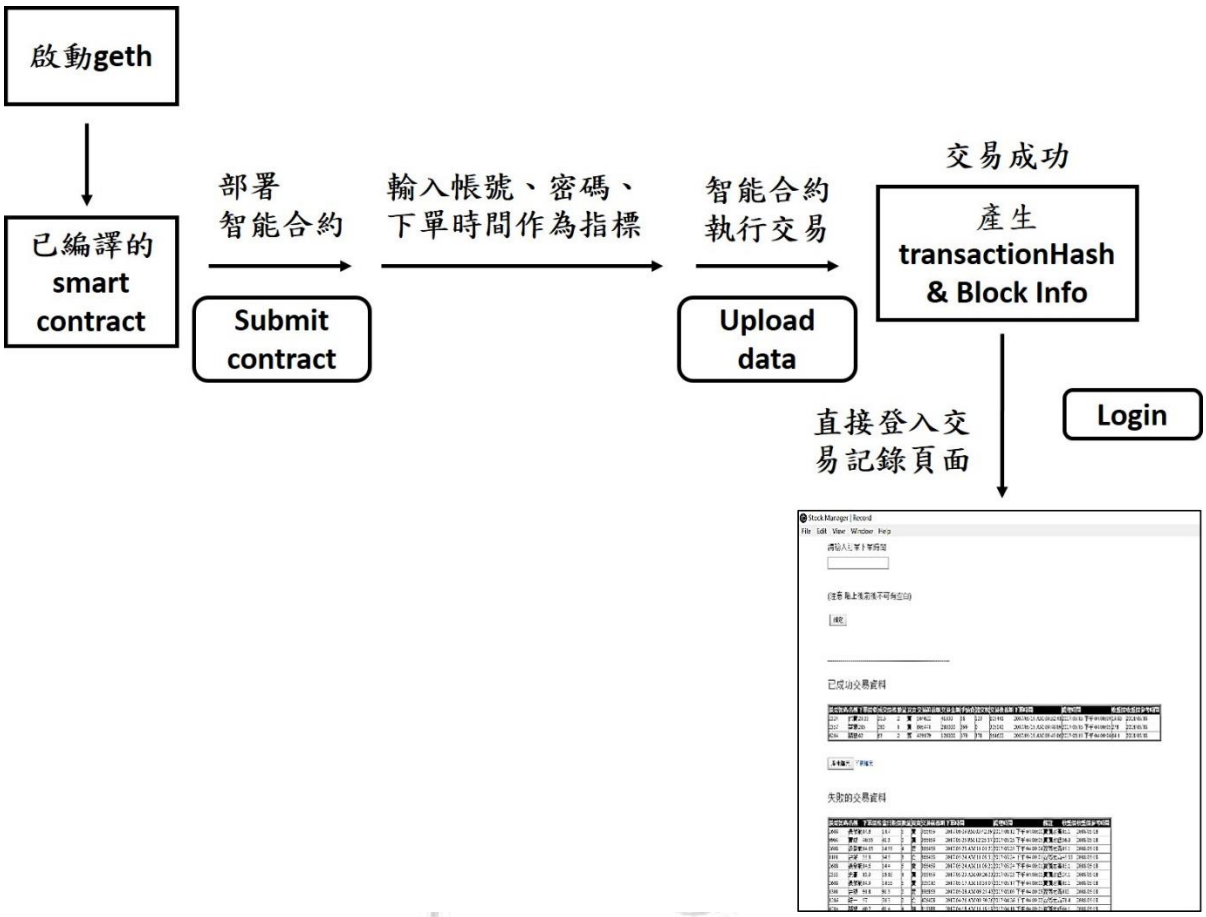


圖 3.15 透過 BC Connector 登入股票模擬程式流程圖

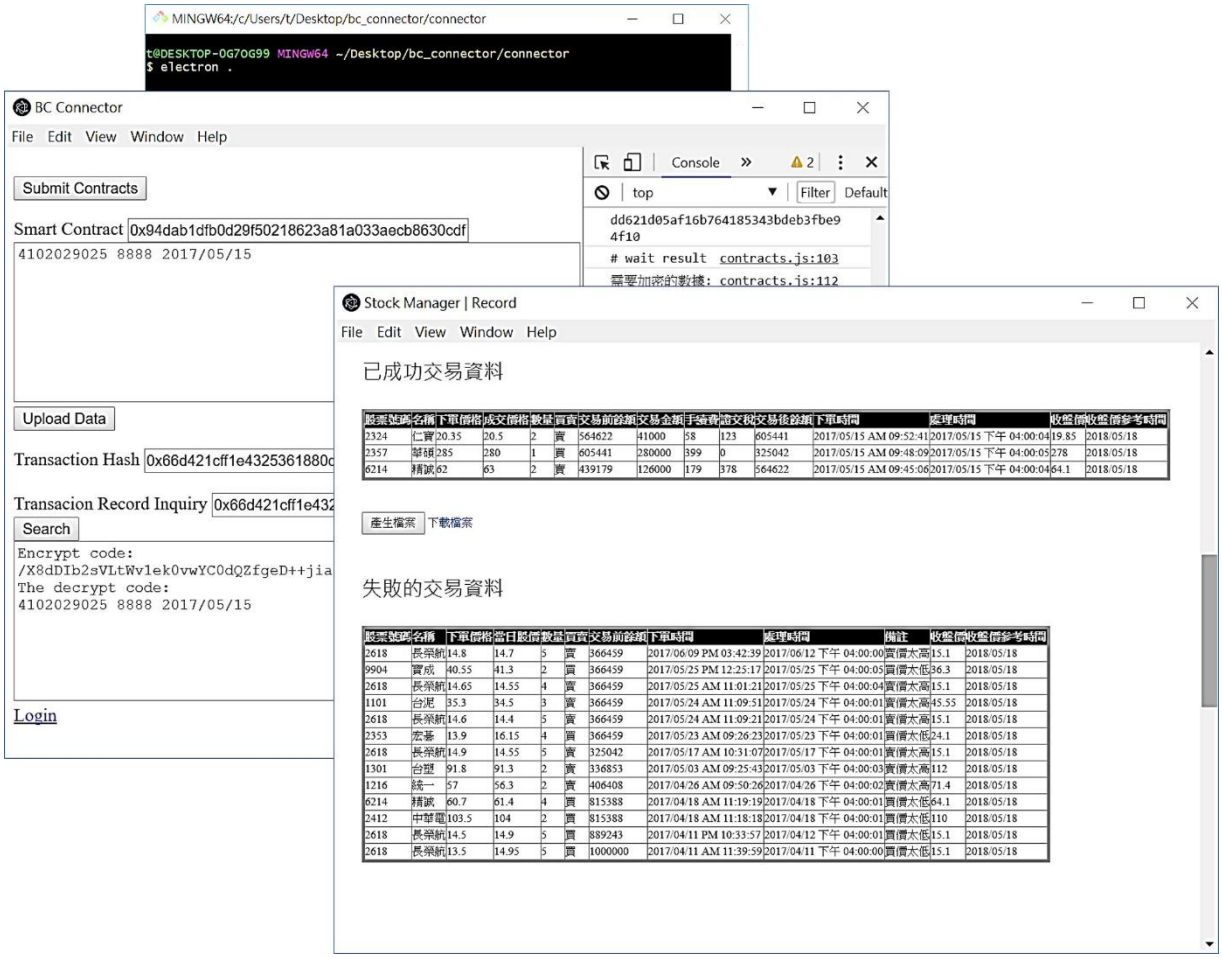


圖 3.16 利用 BC Connector 進入股票模擬交易程式查詢資料

4. 研究結果

本研究根據第三章所提出的研究方法進行一連串的測試及分析。

4.1 股票模擬交易應用程式測試結果

為了測試性能，我們使用檢測網站速度、效能的免費網站工具 Pingdom Website Speed Test 來了解和評估所提出的服務性能，並且通過實驗來衡量網站的加載時間。我們將美國聖荷西定為測試區域，在測試中使用的伺服器規格如下：

- 本機端伺服器
  - 處理器(CPU)：Intel® Core(TM) Quad CPU Q9400 @ 2.67GHz.
  - 記憶體(RAM)：4GB
  - 作業系統(OS)：Windows Server 2008 R2 Enterprise 64bit
  - 機器所在位置：台灣
- 雲端伺服器
  - 雲端平台：Microsoft Azure Server
  - 機器所在位置：美國中部

網頁加載時間及寫入區塊鏈的時間測試結果如表 4.1。從下表可以看出雲端伺服器比本機端伺服器的速度更快，我們推測這是因為雲端服務供應商專注於雲端儲存業務所造成的優勢，這種差異還受到雲端服務供應商與本研究電腦設備之間的網絡速度、測試位置和設備功能等影響。

表 4.1 應用程式架設於本機端與雲端的加載時間測試結果

加載時間	股票模擬應用程式架設於本機端伺服器(sec)	股票模擬應用程式架設於 Azure 雲端伺服器(sec)
1	2.07s	1.03s
2	2.54s	1.02s
3	2.05s	1.23s
4	2.01s	1.11s
5	2.52s	1.02s
6	2.40s	1.08s
7	2.31s	1.15s
8	2.23s	1.24s
9	1.99s	1.09s
10	2.36s	1.15s
平均時間	2.48s	1.12s

4.2 區塊鏈應用程式測試結果

我們首先比較將資料存放至雲端和將資料存放在雲端後產生的位址寫入區塊鏈之間的差異，由表 4.2 可看出直接將股票模擬應用程式產生的交易資料存放至雲端花費的時間較短且較平均，而將雲端資料位址存放至區塊鏈，增加寫入區塊鏈的步驟，因此需花費較長的時間，相較於只將資料存放在雲端，寫入區塊鏈雖較為耗時，但安全性更高。

表 4.2 將資料存放至雲端及雲端資料寫入區塊鏈之比較

加載時間	股票模擬應用程式資料直接存放至 Azure 雲端平台(sec)	股票模擬應用程式在 Azure 產生出資料位址後寫入區塊鏈(sec)
1	21.14s	51.42s
2	20.31s	64.36s
3	15.92s	40.55s
4	14.54s	52.31s
5	24.64s	36.96s
6	18.51s	88.49s
7	16.61s	37.12s
8	19.92s	35.71s
9	23.16s	21.33s
10	17.18s	46.40s
平均時間	19.19s	47.45s

再者，我們比較直接將資料存放至區塊鏈及透過區塊鏈作為連接器將雲端資料寫入區塊鏈中兩者的差異。將礦工的數量設為變數，來測試寫入區塊鏈的速度，本研究測試中使用的伺服器規格如下：

- 區塊鏈伺服器
  - 處理器 (CPU)：Intel® Core(TM) i5-2400 CPU @ 3.10GHz　3.30GHz
  - 記憶體 (RAM)：8GB
  - 作業系統 (OS)：Windows 10 Professional 64bits
  - 機器所在位置：台灣

我們分別將礦工的人數設定為單人 (如表 4.3)、雙人 (如表 4.4)、五人 (如表 4.5)、十人 (如表 4.6)，就以本實驗的環境設定看來，當礦工的人數越多時，寫入區塊鏈的速度相對越快，但就單張表來看，寫入區塊鏈的時間差異仍然很大。我們認為是因為區塊鏈使用雜湊函數 (hash function) 設計複雜的數學題，要部署上區塊鏈的智能合約，即為候選區塊要猜到正確答案才能成為鏈上的最新區塊，而猜的時間目前尚未有平均值，因此會造成每一次寫入區塊鏈的時間差距甚大。答案會由節點計算後一同被打包進區塊中，猜到答案的節點就擁有將候選區塊放到鏈上的權利，並獲得一定的回饋金，如圖 4.1。

表 4.3 單人礦工寫入區塊鏈之時間測試結果

寫入時間 (sec)	以本機端網頁資料 直接寫入區塊鏈	以雲端資料透過 區塊鏈作為連接器
1	33.55s	53.77s
2	35.62s	63.25s
3	61.98s	45.39s
4	63.24s	87.38s
5	28.46s	36.01s
6	20.68s	34.60s
7	37.68s	20.22s
8	91.42s	49.44s
9	56.24s	51.20s
10	31.66s	35.85s
平均時間	46.03s	47.71s

表 4.4 雙人礦工寫入區塊鏈之時間測試結果

寫入時間 (sec)	以本機端網頁資料 直接寫入區塊鏈	以雲端資料透過 區塊鏈作為連接器
1	49.95s	30.23s
2	19.96s	33.13s
3	25.23s	16.74s
4	37.22s	22.15s
5	23.54s	23.68s
6	21.37s	19.19s
7	20.52s	23.56s
8	12.62s	26.73s
9	28.17s	16.25s
10	22.69s	20.29s
平均時間	26.13s	23.20s

表 4.5 五人礦工寫入區塊鏈之時間測試結果

寫入時間 (sec)	以本機端網頁資料 直接寫入區塊鏈	以雲端資料透過 區塊鏈作為連接器
1	11.65s	8.48s
2	20.63s	9.04s
3	22.09s	14.81s
4	13.91s	21.32s
5	19.36s	17.04s
6	15.75s	15.82s
7	11.51s	14.82s
8	16.99s	21.31s
9	17.05s	14.29s
10	10.74s	11.17s
平均時間	15.97s	14.81s

表 4.6 十人礦工寫入區塊鏈之時間測試結果

寫入時間 (sec)	以本機端網頁資料 直接寫入區塊鏈	以雲端資料透過 區塊鏈作為連接器
1	14.65s	13.02s
2	12.39s	16.32s
3	7.38s	28.38s
4	24.92s	14.27s
5	9.87s	19.32s
6	16.79s	14.31s
7	13.70s	22.91s
8	25.85s	29.28s
9	15.67s	15.79s
10	8.08s	7.74s
平均時間	14.94s	18.13s

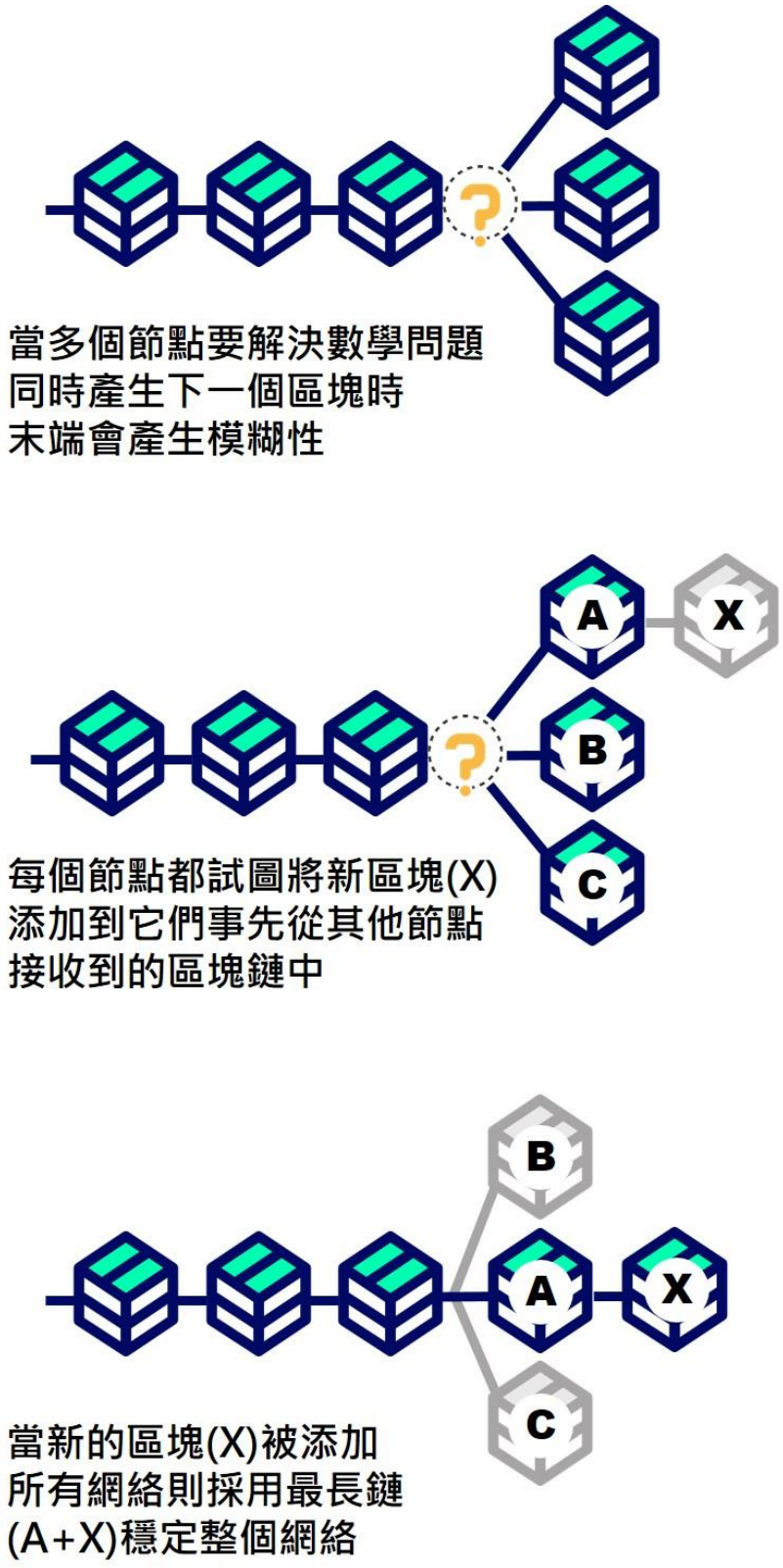


圖 4.1 解決區塊鏈模糊邏輯

4.3 優劣勢分析

用區塊鏈技術來加密資料達成分散式儲存，是輔助雲端儲存的服務核心。透過本研究提出的方案，結合區塊鏈點對點 (P2P) 的網路特性，並採用分散式共識演算法，將資料記錄在區塊鏈中，而區塊鏈會持續延長，且節點之間可相互連結 (Drescher, 2017)，資料一旦寫入區塊中就無法再篡改，有利於解決雲端供應商無法保證使用者數據的安全性，且將資料分散存放在網絡中的各節點能使網絡交易、認證與協作等成本降低，同時有效解決傳統分散式資料庫的同步問題。相對於寫入鏈中需要網絡中多數的節點驗證，導致寫入資料庫的時間較長，因此需要高頻交易、資料量傳輸過於龐大者，目前仍不適合運用區塊鏈解決方案 (Wüst, Gervais, 2017)。

5. 結論與未來展望

5.1 結論

網路上提供各式各樣免費的API服務使開發人員能夠創造更多元的應用程式，這項研究也使用了許多 API 來完成，並結合了現實世界的股票市場。我們將資料放上雲端利用區塊鏈改善及解決，可能發生的問題，並且在股票模擬程式中實現本研究提出的解決辦法，證明此方案的可行性。

在這個研究中分為兩個部分，第一，我們將股市模擬程式的網頁伺服器和資料庫系統架設於實驗室的機房，從中搜集交易資訊，直接存放至我們架設好的私有區塊鏈中，透過分散式節點進行網路數據的儲存、驗證、傳遞和交流。它可以為每筆交易進行記錄、排序及加密。參與者則透過驗證碼，連結交易記錄，再利用區塊鏈的特性，保存執行所有事務的記錄，並保證數據的完整性，從而無法篡改交易歷史。

第二，由於越來越多的中小企業轉向採用雲端運算服務，而區塊鏈創建了安全、有效防止篡改和民主的計算網路。本研究透過區塊鏈結合雲端，上傳的每一個檔案都會被分割成好幾個小塊，每一個小塊將分散儲存在區塊鏈網絡中不同的區塊裡，此外，區塊鏈上也記錄每個資料塊儲存的位置。當需要調閱該檔案時，系統會根據使用者手上的私鑰驗證身份後將其檔案組裝，與集中式系統的儲存相比，將數據儲存在區塊鏈結合分散式雲端中會更佳安全，分散式網絡也讓使用者的數據存放在更多設備上，一台設備不包含完整的文件，這使駭客幾乎不能獲得數據，從而提高可靠性。

5.2 未來展望

隨著資訊與網路科技的快速發展，區塊鏈將會驅動全球經濟成長，目前大多數的數位資訊與交易依賴集中式的資訊架構，如政府機關、金融機構等，然而集中化管理在網路發達的時代問題也陸續浮出檯面，中心化的機構容易成為有心人士攻擊的目標。

基於區塊鏈技術能有效降低認證、網絡交易與協作等成本 (Herbaut, & Negru, 2017)，並透過公私鑰來控制資料讀取與修改權限，兼顧透明度與隱私安全性，且具可信賴的優點，因此在金融保險服務、醫療健康、共享經濟、供應鏈、物聯網與所有權與版權、政府與公共服務等，均有極大的應用效益。

參考文獻

Antonio Madeira. (2017, Sep 28). How to mine Ethereum on a Windows PC - Mining Tutorial | CryptoCompare.com. [Online] Retrieved from https://www.cryptocompare.com/mining/guides/how-to-mine-ethereum

Brandon Arvanaghi. (2018, Jan. 26). Explaining the Genesis Block in Ethereum. Retrieved from https://arvanaghi.com/blog/explaining-the-genesis-block-in-ethereum

Buterin, V. (2015, Aug. 7). On Public and Private Blockchains - Ethereum Blog. [Online] Retrieved from https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

Buterin, V. (2017, Feb 6). The Meaning of Decentralization. [Online] Retrieved from https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274

Christidis, K., & Devetsikiotis, M. (2016). *Blockchains and smart contracts for the internet of things.* IEEE Access, 4, 2292-2303. DOI: 10.1109/ACCESS.2016.2566339

Dai, M., Zhang, S., Wang, H., & Jin, S. (2018). *A Low Storage Room Requirement Framework for Distributed Ledger in Blockchain.* IEEE ACCESS, 6, 22970-22975.

Drescher, D., (2017). *Blockchain Basics: A Non-Technical Introduction in 25 Steps*. New York, NY: Apress Media LLC. DOI: 10.1007/978-1-4842-2604-9_14

Gibson, J., Rondeau, R., Eveleigh, D., & Tan, Q. (2012). *Benefits and challenges of three cloud computing service models.* Proceedins of IEEE International Conference on Computational Aspects of Social Networks (CASoN), pp. 198-205. DOI: 10.1109/CASoN.2012.6412402

Herbaut, N., & Negru, N. (2017). *A model for collaborative blockchain-based video delivery relying on advanced network services chains.* IEEE Communications Magazine, 55(9), 70–76. DOI: 10.1109/MCOM.2017.1700117

Jansen, W. A. (2011). *Cloud hooks: Security and privacy issues in cloud computing.* Proceedings of System Sciences (HICSS), 1–10. DOI: 10.1109/HICSS.2011.103

Januzaj, Y., Ajdari, J., & Selimi, B. (2015). *DBMS as a cloud service: Advantages and*

disadvantages. Proceedings of Procedia - Social and Behavioral Sciences, 195, 1851–1859. DOI: 10.1016/j.sbspro.2015.06.412

Kirar, A., Yadav, A. K., & Maheswari, S. (2016, November). *An efficient architecture and algorithm to prevent data leakage in Cloud Computing using multi-tier security approach.* Proceedings of IEEE International Conference on System Modeling & Advancement in Research Trends (SMART), pp. 271-279. DOI: 10.1109/SYSMART.2016.7894534

Komhar. (2017). Introduction to Solidity: Creating a data contract [Part 1] - Blockgeeks. [Online] Retrieved from https://blockgeeks.com/introduction-to-solidity-part-1/

Lemieux, V. L. (2016). *Trusting records: is Blockchain technology the answer?* Proceedings of Records Management Journal, Vol. 26.110-139. DOI: 10.1108/RMJ-12-2015-0042

Nakamoto, S. (2008, Oct. 31). Bitcoin: A Peer-to-Peer Electronic Cash System. [Online] Retrieved from https://bitcoin.org/bitcoin.pdf

Rouhani, S. & Deters, R. (2017, November). *Performance Analysis of Ethereum Transactions in Private Blockchain.* Proceeding of IEEE International Conference on Software Engineering and Service Science (ICSESS), DOI: 10.1109/ICSESS.2017.8342866

Saxena, V. K., & Pushkar, S. (2016, March). *Cloud computing challenges and implementations*. In Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on(pp. 2583-2588). IEEE.Swan, M. (2015). *Blueprint for a new economy*. Sebastopol, CA: O'Reilly Media, Inc.

Szabo, N. (1994) Smart contract. Unpublished manuscript. [Online] Retrieved from http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

Tianfield, H. (2012, October). *Security issues in cloud computing.* IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1082-1089. DOI: 10.1109/I-SMAC.2017.8058286

Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3), 2084-2123.

Vogelsteller, F., Kotewicz, M., Wilcke, J., &Oance, M. (2018). web3.js Documentation.

Wang, C. W., &Chang, S. E. (2016). *Cloud service in stock trading game: Service virtualization, integration and financial application.* International Conference on Ubiquitous and Future Networks, ICUFN, 2016–Augus, 857–862. DOI: 10.1109/ICUFN.2016.7537158

Wüst, K., & Gervais, A. (2017). *Do you need a Blockchain?*. IACR Cryptology ePrint Archive, 375.

Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., &Chen, S. (2016). *The blockchain as a software connector.* Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016, 182–191. DOI: 10.1109/WICSA.2016.21

Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., ... & Rimba, P. (2017, April). *A taxonomy of blockchain-based systems for architecture design.* Proceedings of IEEE International Conference on Software Architecture (ICSA), pp. 243-252.

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018, Jan. 24). Blockchain Technology Overview. [Online] Retrieved from https://csrc.nist.gov/CSRC/media/Publications/nistir/8202/draft/documents/nistir8202-draft.pdf