國立政治大學資訊科學系

碩士學位論文

去中心化數位貨幣溯源服務：設計與以太坊實作

A Decentralized Digital Currency Tracing Service: Design and Implementation on Ethereum

指導教授：郭 桐 惟 教授

研究生：朱 奕 寧 撰

中 華 民 國 一〇九 年 十月

數位貨幣溯源服務設計探討：以 Ethereum 為例

Digital Currency Traceability Design Base on Ethereum Platform

研究生：朱奕寧　　Student：Yi-Ning Chu

指導教授：郭桐惟　　Advisor：Tung-Wei Kuo

國立政治大學

資訊科學系

碩士論文

A Thesis

submitted to Department of Computer Science

National Chengchi University

in partial fulfillment of Requirements

for the degree of

Master

In

Computer Science

中華民國一〇九年十月

Oct. 2020

摘要

「區塊鏈」源自於中本聰 (Satoshi Nakamoto)的比特幣，透過加密技術，利用分散式跳過中介者，使所有參與者一同紀錄、驗證，成為「去中心化的系統」。區塊鏈具有安全性、匿名性、不可篡改以及追溯歷史紀錄等特性，以進行資料的儲存、驗證和傳遞。利用區塊鏈的特性，除了加密貨幣之外，像是合約、病歷追蹤與金融性產品等，逐漸衍生出眾多的應用場景。

區塊鏈的發展日漸成熟，目前已有許多領域和產業與區塊鏈技術整合，利用其資料公開透明、可追溯性以及不可篡改等特性，提高可信度與安全性。隨著時間的推進，區塊鏈上之區塊高度及其交易總量將會不斷成長，因此，如何在龐大的區塊鏈網路中追溯與搜索相關資料，是當前重要的研究課題。

本研究的主要目標是運用以太坊區塊鏈的智能合約與結合 Oraclize(Provable)的服務應用，並透過跨鏈技術以優化目前區塊鏈交易溯源和搜索的方便性。實作方法於以太坊私有鏈之智能合約以儲存公有鏈上之數位貨幣之相關交易，透過跨鏈互操作性的方式傳遞歷史交易，並開發實驗系統進行探討與測試，更進一步評估其優缺點，以利於後續搜索區塊鏈之實務應用發展的需求之參考。

目　錄

第一章　緒論

本章共分二小節，主要說明本論文之研究背景與動機、以及研究問題與研究目標。

1.1 研究背景與動機

「區塊鏈」源自於虛擬貨幣之比特幣(Bitcoin)底層技術。區塊鏈中的所有節點彼此交換訊息並共同維護一共享帳本，被稱為「去中心化的分散式資料庫」。區塊鏈更具有安全性、匿名性、不可篡改以及追溯歷史紀錄等特性，得以進行資料的儲存、驗證和傳遞。過去數年間，區塊鏈技術已有了許多創新應用和產業發展，除了加密貨幣之外，像是商品保證書、醫療應用與金融性產品等，逐漸衍生出眾多的應用場域。

區塊鏈技術蓬勃發展，已然成為火紅的技術焦點，區塊鏈不僅擁有去中心化、資料不可篡改、可追溯、匿名性且公開透明等特性，智能合約也是區塊鏈的另一特色。智能合約是區塊鏈中的一種特殊協議，其合約內容與代碼皆由程式所編寫而成，智能合約就像是個可以被信任的第三方角色，能在沒有中介的情況下，進行交易。智能合約也提供有效管理鏈上資產和使用權限的限制，保護資料不被任意刪除和竄改的風險。

隨著區塊鏈的技術不斷創新突破，市場上也逐漸出現代幣發行(Initial Coin Offering，簡稱 ICO)的風潮。ICO 可以理解為發行代幣，並公開在網上融資，投資者可以透過持有的比特幣或以太幣認購該項目的代幣。大多數的代幣是一基於以太坊智能合約的一種 Token 標準協議(EIP，Ethereum Improvement Proposal) - ERC-20(Ethereum Request for Comment)。所有的 ERC-20 代幣都能於以太坊中進行交易、追蹤或是監測等。以太坊中的代幣交易皆須透過發起交易才能完成動作，而這些交易都會被記錄在區塊鏈上，使交易都具有可追溯性。

面對各式各樣的 ICO 發行，加上區塊鏈系統與鏈上的資料量日漸龐大，而造成歷史交易溯源之困難。然而在實務應用上，很有可能需針對一數位貨幣之歷史交易進行溯源，而交易內容與來源必須不遺漏、交易內容正確且具有可信度。因此，在區塊鏈上進行資料搜索效益不佳的考量下，本研究將利用智能合約建構一「數位貨幣溯源服務」，該服務將擷取與該數位貨幣相關之歷史交易，儲存於智能合約中，使其使用者可對智能合約中所儲存之交易進行查找。

1.2 研究問題與目標

實現區塊鏈中數位貨幣的溯源服務，已成為實務應用上必須面對的難題。因此，如何沒有遺漏的取得所有相關歷史交易，以及交易內容是否正確且具有可信度，是目前重要的研究課題。

基於前述之研究背景與動機，本研究將以透過參考現有相關技術與文獻，就針對以太坊數位貨幣之溯源服務進行探討。例如：BigChainDB，其技術內容則是針對分散式資料庫與區塊鏈的結合，以達成兩者之特性；BlockIPFS，是針對分佈式文件系統與區塊鏈整合，但未利用區塊鏈作為儲存交易資料之工具；EQL，可以對區塊鏈進行查詢，並實作出一區塊鏈之查詢語言 EQL(Ethereum Query Language)，依內容所述，該查詢語言能對於區塊鏈區塊(Blocks)與交易(Transactions)進行查詢，但尚未完成智能合約內部數據的查詢；Etherscan，是一個 Ethereum Block Explorer，可以查看區塊鏈上所有發生的交易、交易狀態和交易內容等，但對於歷史交易的查找、或是區塊的搜查皆有限制。上述之相關技術與文獻，於第三章將有更詳細的介紹。本研究之實驗環境於以太方區塊鏈平台，開發實驗系統以進行數位貨幣溯源服務之探討與實驗。

第二章　技術背景

本研究以探討數位貨幣溯源服務為目標，將以太坊區塊鏈作為開發平台，並以其智能合約結合 Oraclize(Provable)實作區塊鏈跨鏈互操作性應用，將存在於主鏈(Mainnet)上之數位貨幣歷史交易儲存至私有鏈中，使其使用者可對儲存之歷史交易進行查詢。本章共分三小節，首先簡介區塊鏈之相關技術背景，接著針對以太坊區塊鏈平台及其智能合約簡述其相關特性，最後則是介紹 Oraclize(Provable)服務是如何可信地與區塊鏈外部溝通。

2.1 區塊鏈

「區塊鏈」源自於中本聰 (Satoshi Nakamoto)的比特幣，透過複雜的密碼學加密資料，再利用數學分散式演算法，成為「去中心化的系統」，也是作為比特幣的底層技術，或稱為「去中心化的分散式資料庫」。區塊鏈將舊有的資料庫重新改造，把資料分成不同的區塊，每個區塊透過特定的協議鏈結到上一區塊，將區塊互相接連呈現一完整的歷史數據。區塊鏈上的每筆交易，都能由區塊鏈的結構設計達到溯源的功能，並能針對每筆交易進行驗證。在區塊鏈的每一個區塊當中，皆會擁有一時間戳進行紀錄，以表示該區塊是由當時的時間所生成的，造就了不可篡改、不可偽造的系統。區塊鏈中所有參與的節點在沒有中心的情況下，節點們透過去中心化的共識機制、遵循著規則驗證訊息內容，並寫入時間戳後生成區塊，再將其廣播至所有節點。透過分散式的方式達成數據儲存、交易驗證、訊息傳遞即為區塊鏈的核心技術。

自從比特幣的出現後，區塊鏈的技術逐漸受到重視，公開透明、公平競爭。隨後世界便開始探索區塊鏈技術在各行各業中發展的可能性，因此，除了常見的公有鏈以外，隨著應用場景的不同，更衍伸出適合企業、產業界使用的私有鏈與聯盟鏈。依照使用者對區練練權限程度不同，種類可略劃分為三種類型：

(1) 公有鏈：任何人都可以訪問，發送、接收、驗證交易，並參與共識過程的區塊鏈。我們最熟悉的區塊鏈，大多屬於公有鏈，像是比特幣、以太幣等。

(2) 私有鏈：區塊鏈的權限被一定程度地進行了限制，須受到授權才能成為節點，並非任何人都能參與。例如摩根大通(JP Morgan)引領的 Quorum 就是私有鏈的代表之一。

(3) 聯盟鏈：聯盟鏈與私有鏈相似，區塊鏈的開放程度與權限也是有所限制的，而授權的節點通常為企業與企業間有合約的關係等。舉例來說，Hyperledger Fabric，以及 R3 Corda 皆是聯盟鏈的一種。

此外，區塊鏈上的交易是在於打包交易並出塊，通過驗證與確認交易是否有效，使交易順利完成，並讓區塊鏈中的所有節點進行更新、以及擁有相同的帳本內容，這樣打包交易並出塊的機制就是「挖礦」。在以太坊區塊鏈中，Ether(ETH)是其中的燃料，在區塊鏈上進行交易時，必須經由礦工運算打包後上鏈，使用者必須支付礦工該交易的手續費，其交易手續費是以 Gas 計算，並以 Ether 支付。於以太坊主網當中，每個區塊都有 800M 的大小限制(Block Gas Limit)，而一般標準交易的 Gas Limit 約為 21,000，一個區塊約可以容納 380 筆交易。交易的 Gas Limit 會依據交易的內容而有所不同，每筆交易的發起方都須設置交易的 Gas Limit 和 Gas Price，不同的操作會產生不同的 Gas 成本，若是交易需花費的 Gas Limit 太高或是超出區塊鏈所設定的大小限制，則可能發生交易無法打包至區塊中，而滯留在交易池(Transaction Pool)的狀況。

整體而言，區塊鏈的應用技術不斷創新，仍有必要持續探索區塊鏈此一新興科技的潛能，在實務應用上，則必須解決區塊鏈的交易擁塞的情形、須支付的 Gas Fee 過高，以及區塊的 Gas Limit 的限制。有鑑於此，為了實現數位貨幣溯源的服務，本研究將實驗環境建立於私有鏈當中，為解決沒有足夠的 Ether 予以支付交易手續費的狀況、打包交易並出塊的速度、以及自定義足夠大的區塊 Gas Limit，且利用智能合約的多元功能，進行數位貨幣溯源的探討與實驗，並針對以太坊區

塊鏈的私有鏈平台與 Oraclize(Provable)的服務應用結合，儲存以太坊公有鏈上之數位貨幣之相關交易進行溯源，透過跨鏈互操作性的方式傳遞歷史交易，並開發實驗系統進行探討與實作，利於後續搜索區塊鏈之實務應用發展的需求之參考。以下將針對以太坊區塊鏈與 Oraclize(Provable)服務分別簡述其相關特性。

2.2 以太坊區塊鏈

以太坊區塊鏈在比特幣區塊鏈的基礎上，衍伸了一個具開源性與開發性高的區塊鏈平台，且支援開發多元化區塊鏈分散式應用程式，稱之為智能合約(Smart Contracts)，並透過其原生貨幣以太幣(Ether)執行去中心化的運行。以太坊區塊鏈也具有圖靈完備(Turing Complete)的特性，使用者能透過 JavaScript 和 Python 等現有語言在以太坊虛擬機(Ethereum Virtual Machine，EVM)上開發分散式應用程式，或稱為 Dapps，並共同維護以太坊區塊鏈之完整性。

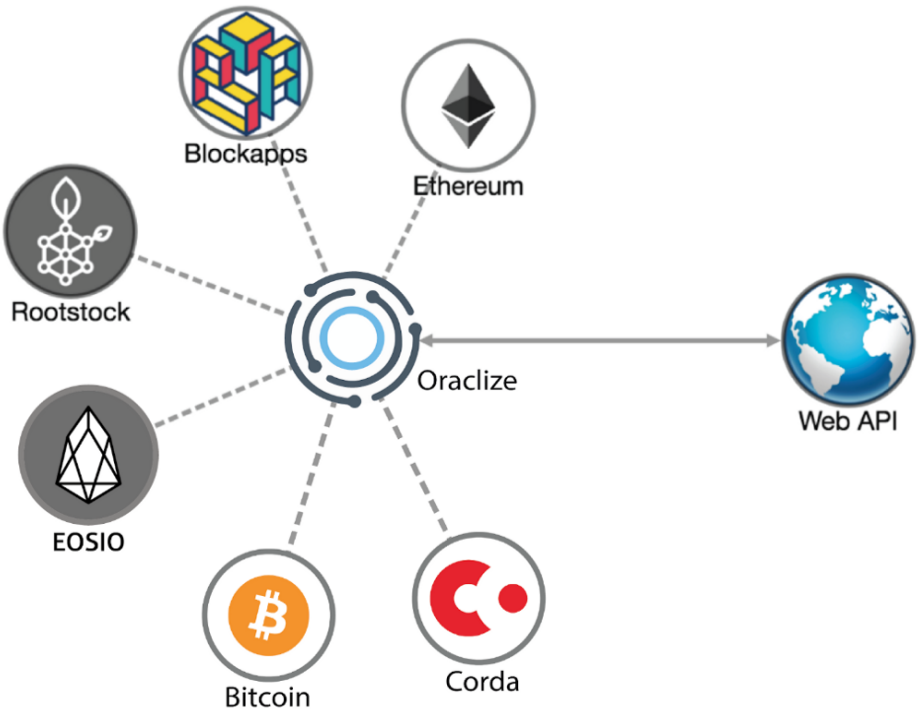
智能合約是區塊鏈中的一種特殊協議，其合約內容與代碼皆由程式所編寫而成，可用於發行客製化代幣、紀錄資料或是金融交易等功能。智能合約具有高安全性、高交易效率與可客製化等，我們也能透過智能合約儲存資料，且本身也能傳遞資料訊息到外界。智能合約就像是個可以被信任的第三方角色，能在沒有中介的情況下，進行交易。智能合約使得區塊鏈在未來能有更多的可能性。

ERC-20(Ethereum Request for Comment)是一個基於以太坊智能合約的一種 Token 標準協議(EIP，Ethereum Improvement Proposal)，所有的 ERC-20 代幣都能於以太坊中進行交易、追蹤或是監測等。以太坊中的代幣交易皆須透過發起交易(Transaction)才能完成動作，而這些交易都會被記錄在區塊鏈上，使交易都具有可追溯性。

2.3 Oraclize (Provable)

Oraclize(Provable)作為區塊鏈外部世界之資料提供者，能夠在區塊鏈上提供可信任的數據傳送服務，其目的是在區塊鏈建立一條可與外部溝通的橋樑，為了解決智能合約取得數據的限制，在保證可信的情況下，使得智能合約具有取得外部數據的能力。此服務也不僅局限於以太坊，目前也可以使用在其他區塊鏈平台上，例如 Bitcoin、EOS、Rootstock、Fabric。

Oraclize(Provable)在以太坊上部署了名為 usingOraclize 的智能合約，只需要在自己的智能合約當中引用，並依據 Web APIs 的使用方法進行調用即可。Oraclize(Provable)主要是透過監聽特定 Event 來接收合約所發出的 Query 請求，處理完成後，再由 Oraclize(Probable)主動呼叫合約的 callback function 將資料回傳。此外，Oraclize(Provable)也提供了五種資料來源，使智能合約取得外部資料，包括 URL 資料來源、WolframAlpha 資料來源、IPFS 資料來源、Random 資料來源、以及 Computation 資料來源。

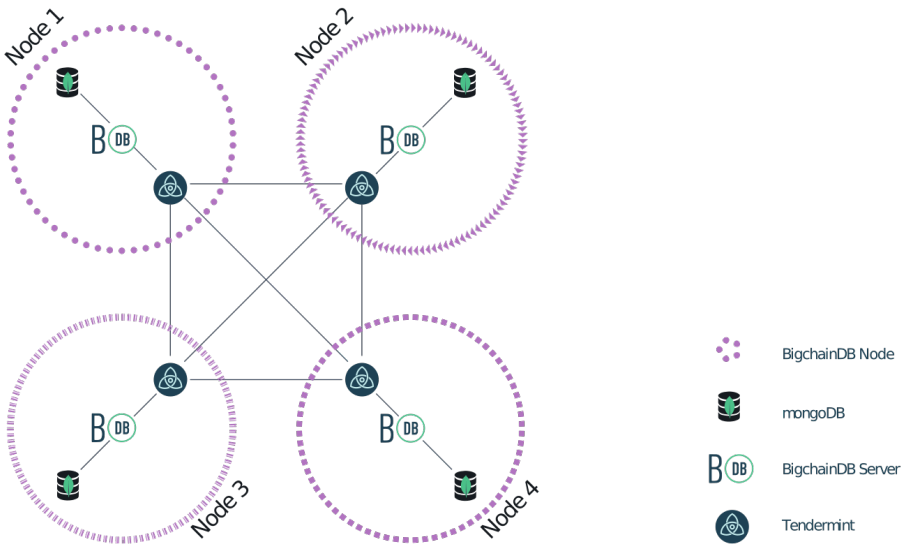


第三章　相關研究

本研究將提出建構於區塊鏈智能合約的數位貨幣溯源服務，以參考現有相關技術，並針對數位貨幣溯源服務之需求進行研究。本研究將以此作為出發點，進行該題目之探討與延伸，並開發該服務系統作為依據。本章共分為四小節，第一小節為 BigChainDB，該論文則是針對分散式資料庫與區塊鏈的結合，以達成兩者之特性；第二小節為 BlockIPFS，該論文針對分佈式文件系統與區塊鏈整合，並未利用利用區塊鏈作為儲存資料之工具；第三小節為 EQL，該論文針對區塊鏈查詢的功能，實作出一查詢語言 EQL(Ethereum Query Language)，此工具可針對區塊(Blocks)與交易(Transactions)進行查詢，但尚未完成智能合約內部數據的查詢；第四小節為 Etherscan，是一個 Ethereum Block Explorer，可以查看區塊鏈上所有發生的交易及其交易狀態等。

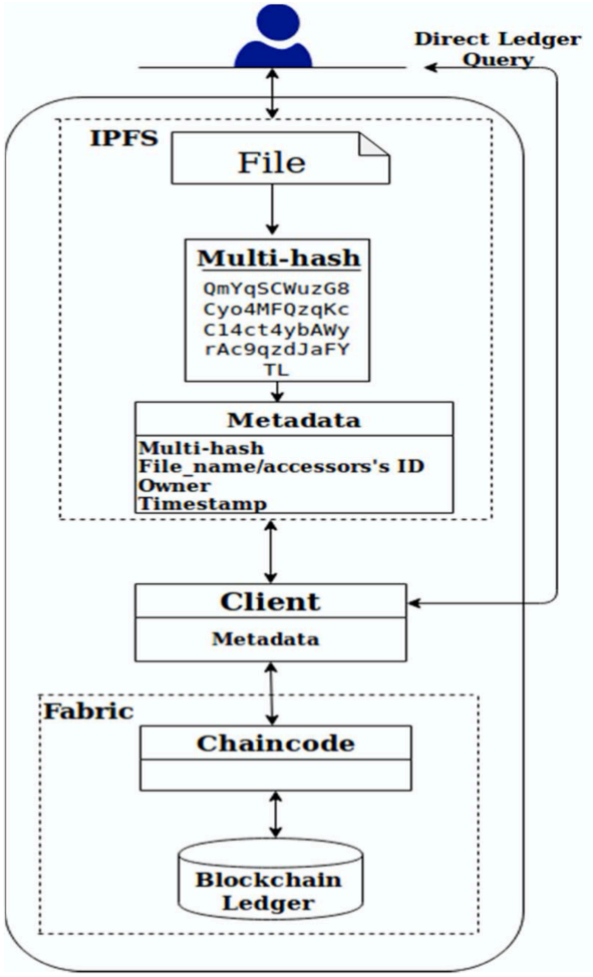
3.1 BigChainDB

BigChainDB [5]是一個可用的去中心資料庫。它的設計起源於分散式資料庫，加入了很多區塊鏈的特性，像是中心控制、不可改變性、數字資產的建立和移動。BigChainDB 繼承了現代分散式資料庫的特性：吞吐量和容量都是與節點數量線性相關、功能齊全的 NoSQL 查詢語言，以及高效的查詢和許可權管理。因建構在已有的分散式資料庫上，它在程式碼層面也繼承了企業級的健狀性。下圖為 BigChainDB 之系統架構圖，根據現版本之設計，每個節點都有一 mongoDB 資料庫用於儲存鏈上資料之用，以此使用者皆能透過常規之 SQL 語言進行鏈上資訊(Block、Transaction 等)之搜索，即不包含智能合約內部數據查詢。



3.2 BlockIPFS - Blockchain-enabled Interplanetary File System for Forensic and Trusted Data Traceability

在此論文中，探討了將區塊鏈與分佈式文件系統 (IPFS，Interplanetary File System)整合，該系統利用區塊鏈 (該論文實作於須身份驗證之區塊鏈 - Hyperladger Fabric)既有的安全性和可追溯性，其溯源性可用於審核和保護文件的作者身份，並可作為調查的證據和爭議解決。開創一個更安全的文件共享平台，該平台具有清晰的審核記錄，以提高數據的可信度以及作者身份保護，方法提供了一條明確的途徑來追溯與給定文件相關的所有活動 (如添加文件、訪問文件等)。下圖為論文中 BlockIPFS [6]之架構，用於說明文件上傳之步驟，其文件本身由 IPFS 加密和管理，而解密文件的密鑰存儲在區塊鏈上的特定通道中。



3.3 Ethereum Query Language

此論文提出了以太坊查詢語言 (EQL，Ethereum Query Language) [7]，一種允許用戶通過編寫類似 SQL 的查詢從區塊鏈中檢索信息的查詢語言。查詢提供了豐富的語法，可以指定數據元素來搜索分散在多個記錄中的信息。EQL 使從區塊鏈中搜索、獲取、格式化和呈現信息變得更加容易。雖然提供了使用者易於查詢區塊資訊的方法，但還缺乏取得智能合約內部資訊的功能。下圖是 EQL 對 Block 查詢的語法範例。

```
SELECT block.parent.number, block.hash,
    block.timestamp, block.number,
    block.amountOfTransactions
FROM ethereum.blocks AS block
WHERE block.timestamp BETWEEN date('2016-01-01')
    AND now() AND block.transactions.size >10
ORDER BY block.transactions.size
LIMIT 100;
```

3.4 Etherscan APIs

Etherscan，是一個 Ethereum Block Explorer，可以查看區塊鏈上所有發生的交易、交易狀態或是查詢 ETH 錢包餘額等功能。其中，Etherscan 也能瀏覽相關 ERC20、ERC721 合約之代幣價格、相關交易以及代幣持有者等。該網站也提供了統計圖表和數據，進而分析供應量的增長、代幣價格的漲幅或是交易頻率等服務。

第四章　系統設計實作

本研究依前章所述之相關研究作為延伸，於以太坊區塊鏈平台之智能合約實作數位貨幣溯源與搜索服務，實踐合約導向程式語言 Solidity 建構資料儲存、資料搜索與資料追溯的機制，使用者能直接對於該智能合約進行搜索。透過智能合約結合跨鏈技術作為提供區塊鏈交易溯源與搜索之功能，利於後續搜索區塊鏈之實務應用發展的需求之參考。

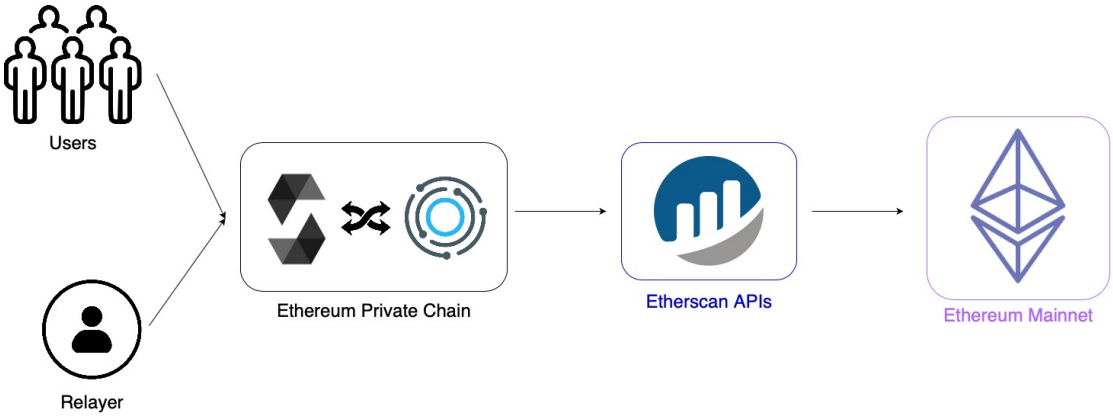
本章共分為五小節，第一小節闡述數位貨幣溯源服務之系統架構；第二小節介紹 Oraclize(Provable)服務與 Ethereum-Bridge；第三小節說明擷取之交易資料處理方式；第四小節描述 Query 之功能；第五小節展示系統介面與操作流程。

4.1 實作系統架構

本實驗架構主要由一個使用者、一個中繼者(Relayer)、以太坊區塊鏈之 Dapp，以及 Oraclize(Provable)服務組成，圖 x 為本研究實作之系統架構圖。

為實現數位貨幣溯源之服務於以太坊區塊鏈，使用 go-ethereum (geth)工具建立數個節點(enode)，其中一個節點架設 NodeJS 伺服器負責區塊鏈與使用者之溝通。利用以太坊智能合約設計其系統，其智能合約再透過 Oraclize(Provable)服務以利用 Etherscan 提供之 APIs 取得相關歷史交易資訊，並將其資訊儲存至合約當中，以利使用者之後續查詢。

中繼者(Relayer)為獨立於以太坊區塊鏈外的 NodeJS 伺服器，負責觸發合約中之 Oraclize(Provable)服務，並同時監控服務是否正常運作，以確保數位貨幣追溯交易之功能不會中斷。



4.2 Oraclize 與 Ethereum-Bridge 介紹

4.2.1 Oraclize

Oraclize(Provable)是一個兼容以太坊智能合約的數據傳送服務，能夠自定義智能合約，在區塊鏈與真實世界之間建立一可信的通道，在保證可信的情況下，使其可透過 APIs 取得以太坊外部真實世界的資訊。

```
1.  contract tokenTracer is usingProvable, Parser {
2.      // oraclize results
3.      function __callback(bytes32 myid, string memory _result) public {
4.          if (msg.sender != provable_cbAddress()) revert();
5.          // 更新合約餘額
6.          tracerBalance = address(this).balance;
7.
8.          oraclizeIsRunning = false;
9.          // 檢查是否有回傳值
10.         if (bytes(_result).length != 0) {
11.             savingTx(_result);
12.         }
13.     }
14.
15.     // call oraclize
16.     function traceTx() payable public {
17.         uint gasLimit = 50000000;
18.         oraclizeIsRunning = true;
```

```
19.
20.         // 設定為每次 Oraclize 取得的交易筆數[:Count]
21.         string memory apiStr1 = "json(https://api.etherscan.io/api?module=logs&action=getLogs&fromBlock=";
22.         string memory apiStr2 = "&toBlock=latest";
23.         string memory apiStr3 = "&address=0x";
24.         string memory apiStr4 = "&topic0=0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef";
25.         string memory apiStr5 = "&apikey=HTI3IX924Z1IBXIIN4992VRAPKHJI149AX).result[";
26.         string memory apiStr6 = "][transactionHash, blockNumber, timeStamp, topics, data]";
27.         string memory apiUrl = string(abi.encodePacked(apiStr1, uint2str(syncBlockHeight), apiStr2, apiStr3, parseAddrressToString(tokenContract), apiStr4, apiStr5, uint2str(syncIndex), ":", uint2str(syncIndex + 50), apiStr6));

28.         provable_query("URL", apiUrl, gasLimit);
29.     }
30. }
```

如程式碼所示，智能合約調用了 provable_query 為 oraclize 提供之方法，並於區塊鏈外調用這個 query，得到結果之後，則會調用智能合約中的 callback 功能將結果作為參數傳入。有了 Oraclize(Provable)服務，當以太坊內部需要外部資訊時，只需使用 APIs 呼叫，並寫好相對應之 callback 方法即可。

4.2.2 Ethereum-Bridge

本實驗環境為以太坊私有鏈，我們必須使用 Ethereum-Bridge 作為 Oraclize(Provable)對外部世界溝通的橋樑。啟動 Ethereum-Bridge 後，它則會開始監聽鏈上 Oraclize(Provable)發出的 events。作為橋樑的角色，每當它接收到 events 時，於區塊鏈外部執行該 APIs 請求，並將最終結果傳回該智能合約中。

4.3 資料處理方式

透過 oraclize(Provable)服務所取得之區塊鏈外部資訊後，我們必須將資料分析後再將其儲存至智能合約中。於我們所開發之智能合約中繼承之「jsmnSol」功能(此工具為 GitHub 上之開源專案，並以此進行資料處理，且為 Solidity 語言所開發之 JSON parser)，以利處理從區塊鏈外部傳回之資訊內容。

```
contract tokenTracer is usingProvable, Parser {
    bytes32[] transactionHash;
    address[] sender;
    address[] receiver;
    uint[] value;
    uint[] blockNumber;
    uint[] timeStamp;

    // 取得已儲存之交易筆數
    uint public transactionCount;

    mapping(bytes32 => bool) private isExist;

    function savingTx(string memory json) internal {
        uint returnValue;
        JsmnSolLib.Token[] memory tokens;
        uint actualNum;

        // (returnValue, tokens, actualNum) = JsmnSolLib.parse(json, 9);
        (returnValue, tokens, actualNum) = JsmnSolLib.parse(json, 450);

        // 每次 Oraclize 取回之交易筆數
        for (uint i = 0; i < 50; i++) {
            JsmnSolLib.Token memory a = tokens[1 + 8*i];
            bytes32 _transactionHash = parseStringTo32Bytes(JsmnSolLib.getBytes(json, a.start, a.end));
            // 避免重複紀錄同一筆交易
            if (!isExist[_transactionHash] && _transactionHash != "") {
                isExist[_transactionHash] = true;
                transactionHash.push(_transactionHash);
                JsmnSolLib.Token memory b = tokens[2 + 8*i];
                uint _blockNumber = parseHexToUint256(JsmnSolLib.getBytes(json, b.start, b.end));
                blockNumber.push(_blockNumber);
```

```
33.                 JsmnSolLib.Token memory c = tokens[3 + 8*i];
34.                 timeStamp.push(parseHexToUint256(JsmnSolLib.getBytes(json, c
    .start, c.end)));
35.                 JsmnSolLib.Token memory d = tokens[6 + 8*i];
36.                 sender.push(parseAddr(subString(JsmnSolLib.getBytes(json, d.
    start, d.end), 26, 66)));
37.                 JsmnSolLib.Token memory e = tokens[7 + 8*i];
38.                 receiver.push(parseAddr(subString(JsmnSolLib.getBytes(json,
    e.start, e.end), 26, 66)));
39.                 JsmnSolLib.Token memory f = tokens[8 + 8*i];
40.                 value.push(parseHexToUint256(JsmnSolLib.getBytes(json, f.sta
    rt, f.end)));
41.
42.                 // 更新下回開始搜尋之 blockNumber, 需避免同一 block 有多筆交易
43.                 syncBlockHeight = _blockNumber;
44.                 syncIndex = 0;
45.             } else if (_transactionHash == "") {
46.                 syncBlockHeight = realBlockHeight;
47.                 syncIndex = 0;
48.                 break;
49.             }
50.         }
51.         if (transactionCount == transactionHash.length) {
52.             syncIndex += 50;
53.         }
54.         transactionCount = transactionHash.length;
55.     }
56. }
```

4.4 Query 功能介紹

本實驗為實現數位貨幣溯源之服務，為此我們也提供了 Query 功能，以利使用者可對儲存於智能合約中之數位貨幣歷史交易進行搜索。於該智能合約中擁有「區塊範圍搜索」與「時間範圍搜索」兩大功能，以及針對交易發送方或接收方進行查找。

4.5 系統介面與操作流程

為了讓使用者容易操作其去中心化數位貨幣溯源服務，本研究也為此系統提供Dapp，以提升使用時之便利性。

4.5.1 註冊欲溯源之數位貨幣合約位址

4.5.2 數位貨幣歷史交易之查找

第五章　討論與結論

本章節統整本研究之研究成果，共分為二小節，先是針對本研究提出整體的結論說明，以及本研究的研究貢獻；接著針對本研究之議題提出未來研究方向之建議。

5.1 討論

本研究目標在於實作以太坊區塊鏈之數位貨幣溯源服務。透過現有技術方案與相關文獻，根據溯源服務在實務上的可能作法進行研究與探討，再藉由實作與實驗，針對實務應用層面進行探討與評估，甚至找出問題並建議解決方案。

下面我們探討針對區塊鏈數位貨幣之溯源溯源服務，其影響溯源之效率的可能因素，以及實作中遭遇的難題：

- 區塊大小影響溯源效率：透過 Etherscan 所提供之 APIs 取得數位貨幣之歷史交易，而每筆歷史交易都有一定的容量大小，但區塊鏈中有區塊大小(Block Gas Limit)的限制，故無法一次性地將所有歷史交易同步完成。因此，本實驗架設一以太坊區塊鏈之私有鏈，並自定義其區塊大小(Block Gas Limit)，使得我們可以大量取得歷史交易，並利用批次的方式，使我們能更快速地同步所有歷史交易。
- Etherscan APIs 之使用次數限制影響溯源效率：我們能透過 Etherscan APIs 取得以太坊公鏈上之數位貨幣歷史交易，但對於一般開發者而言，使用 API 有對於每個 IP 位址每秒只能使用五次 API 的限制。
- Ethereum Bridge 處理 Oraclize(Provable)之 Query 速度影響溯源效率：由於本實驗將數位貨幣溯源服務建立於私有鏈之上，當使用 Oraclize(Provable)之 Query 時，需搭配 Ethereum Bridge 工具監聽其智能合約所發出之特定事件(Events)，並作為中介者負責傳遞鏈外訊息。Ethereum Bridge 並無多執行緒

的多工處理能力，因此當它接收到事件時，只能將其放入佇列中，採先進先出的方式逐一處理。

- Solidity 無提供 Json 資料型態之處理方法：數位貨幣溯源服務乃是利用以太坊區塊鏈智能合約所建立而成，將取得之歷史交易儲存於智能合約當中，以利後續之查找和追溯。不過，於智能合約獲取之歷史交易內容皆為 Json 格式之資料，而 Solidity 本身並無提供該資料的處理方式，故透過「jsmnSol」此 GitHub 上之開源工具，同時也是為 Solidity 語言所編成之 Json Parser 進行資料處理。

5.2 結論

本研究之「去中心化數位貨幣溯源服務」是一建立於以太坊區塊鏈之私有鏈平台之服務，透過私有鏈上之節點共同維護數位貨幣溯源之交易證明，以保證其公正性和安全性。該服務也利用了 Oraclize(Provable)作為可信任之交易資料提供者，為了解決智能合約取得外部數據之限制，並結合由 Etherscan 所提供之 APIs，以確保數據之正確性。在實務應用上，隨著區塊鏈的演進與數位貨幣的發展趨勢，可能須面臨更透明化、更嚴謹之溯源服務，甚至是政府機關之區塊鏈監管機制。因此，未來在區塊鏈數位貨幣之溯源議題時，希望能持續改進，例如：(1)數位貨幣流向追蹤，透過資料分析與追查，評估各個錢包地址是否存有洗錢之動機等。(2)納入監理機制：於每一種數位貨幣之智能合約中加入一監理者角色，並監聽該數位貨幣之所有交易內容；或是所有交易皆需經由監理者認可才能發送，藉此達到監管與治理的效果。

參考文獻

[1] Ethereum Foundation. 2014. Ethereum's white paper. (2014). https://en.Bragagnolo et al.wikibooks.org/wiki/LaTeX/Bibliography_Management

[2] Ethereum Foundation. 2018. Solidity Documentation Release 0.4.20. (2018). https://media.readthedocs.org/pdf/solidity/develop/solidity.pdf

[3] Ethereum Foundation. 2018. JSON RPC. (2018). https://github.com/ethereum/wiki/wiki/JSON-RPC

[4] Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. https://nodejs.org/en/

[5] BigChainDB is a software that has blockchain properties and database properties. https://www.bigchaindb.com

[6] E. Nyaletey, R. M. Parizi, Q. Xhang, K. R. Choo, "BlockIPFS - Blockchain-Enabled Interplanetary File System for Forensic and Trusted Data Traceability", *IEEE International Conference on Blockchain (Blockchain)*, Atlanta, GA, pp. xxxx-xxxx, 2019.

[7] S. Bragagnolo, H. Rocha, M. Denker, S. Ducasse, "Ethereum Query Language", IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), Gothenburg, Sweden, pp. xxxx-xxxx, 2018.

[8] The Provable™ blockchain oracle for modern DApps. https://provable.xyz

[9] Etherscan.io https://etherscan.io