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探討：以Ethereum為例

Decentralized Digital Currency Traceability Design Base on Ethereum Platform

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區塊鏈的智能合約並結合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，並透過跨鏈互操作性的方式傳遞訊息，開發實驗系統進行探討與測試，並進一步評估其優缺點，利於後續搜索區塊鏈之實務應用發展的需求之參考。

目 錄

1. 緒論

本章共分兩小節，主要說明本論文之研究背景與動機以及研究目標。

**1.1 研究背景與動機**

「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景。

乙太坊 (Ethereum)[1]在比特幣的基礎上，有了智能合約 (Smart Contract)[2]的功能。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。智能合約有安全性高、交易效率高和可客製化等優點，能在沒有中介的情況下，仍能進行交易。使得區塊鏈在未來能有更多的可能性。

ERC-20 (Ethereum Request for Comment)是一個基於乙太坊智能合約的一種Token標準協議 (EIP，Ethereum Improvement Proposal)，所有的ERC-20代幣都能於乙太坊中進行交易、追蹤或是監測等。乙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。

本研究將利用智能合約建構一「數位貨幣溯源服務」，使用者可對合約中所儲存之歷史交易進行查詢。在區塊鏈上進行資料搜索效益不佳的考量下，意旨設計一區塊鏈智能合約資料庫，使得使用者可以透過範圍搜索的功能，進行相關資料的搜查，例如某時間區段中有哪些交易、發起人是誰、交易儲存於哪一區塊或是交易時間等資訊。最後，我們也將設計一以NodeJS [3][4]打造的UI介面，優化使用者使用該資料庫的方便性。

**1.2 研究目標**

本論文於以太坊 (Ethereum)之智能合約 (Smart Contract)建構「追蹤Token交易之歷史紀錄資料庫」，第一步為建立ㄧ主控之資料庫中心 (tracerCT)，透過tracerCT部署所需之資料庫 (tokenTracer)，同時記錄其合約位址；其中，tokenTracer將會儲存使用者欲追蹤之ERC-20代幣之Transfer交易。

同時，建立具範圍搜索之功能，使得使用者能針對資料庫中所儲存之交易內容進行「時間範圍」或是「區塊區間」等搜索功能。

最後，將完成系統整合並將系統以視覺化介面方式呈現，以期建構出一可追蹤、可客製化與具有範圍搜索功能之區塊鏈智能合約之「數位貨幣溯源服務」。

1. 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。

**2.2 以太坊區塊鏈**

以太坊區塊鏈[1]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，且擁有了智能合約(Smart Contract)[2]的功能，透過其原生貨幣以太幣(Ether)執行去中心化的運行。以太方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。智能合約有安全性高、交易效率高和可客製化等優點，能在沒有中介的情況下，仍能進行交易。使得區塊鏈在未來能有更多的可能性。

ERC-20 (Ethereum Request for Comment)是一個基於乙太坊智能合約的一種Token標準協議 (EIP，Ethereum Improvement Proposal)，所有的ERC-20代幣都能於乙太坊中進行交易、追蹤或是監測等。乙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。

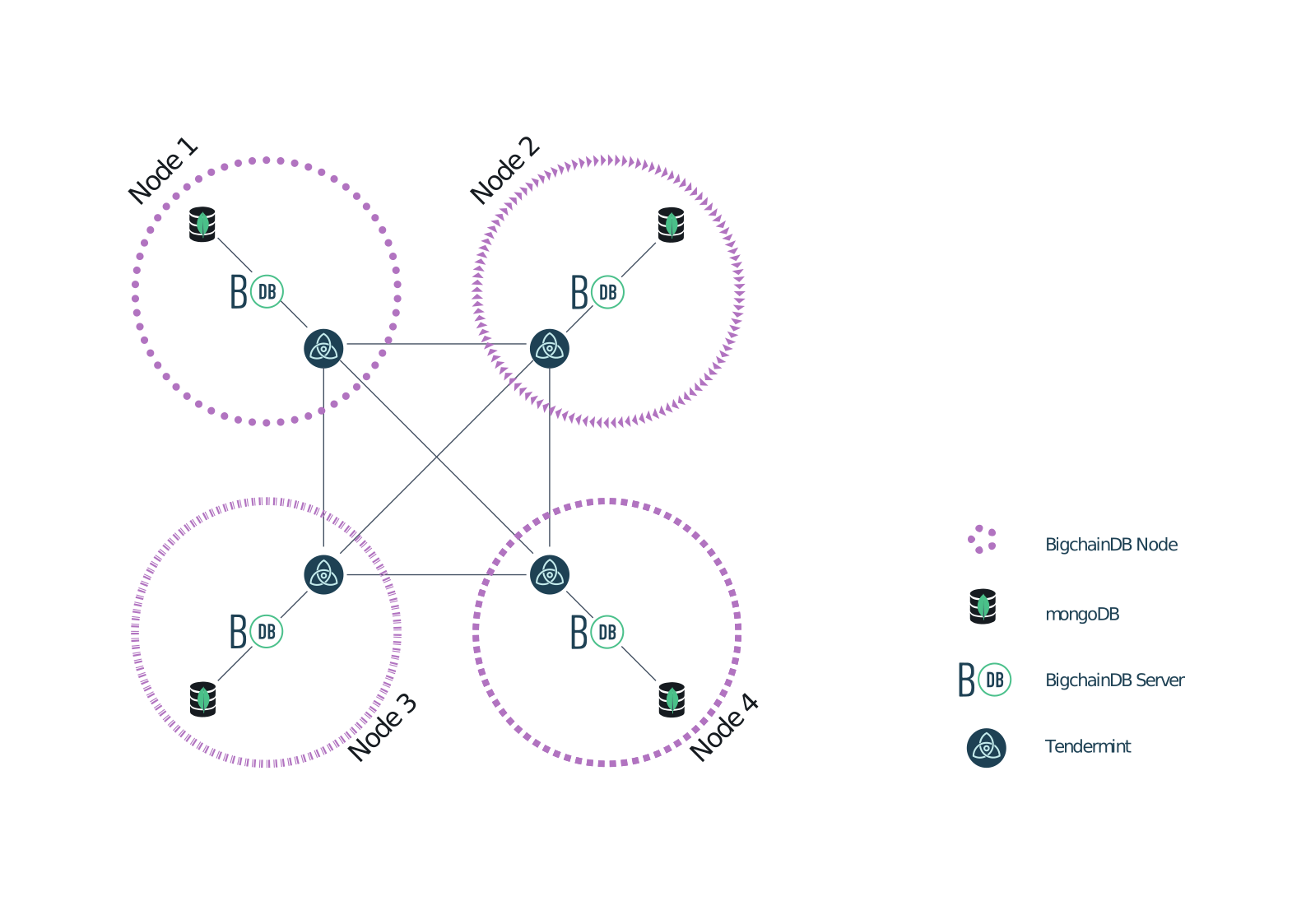
**2.3 Oraclize (Provable)**

1. 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，並針對區塊鏈的搜索需求進行研究。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。

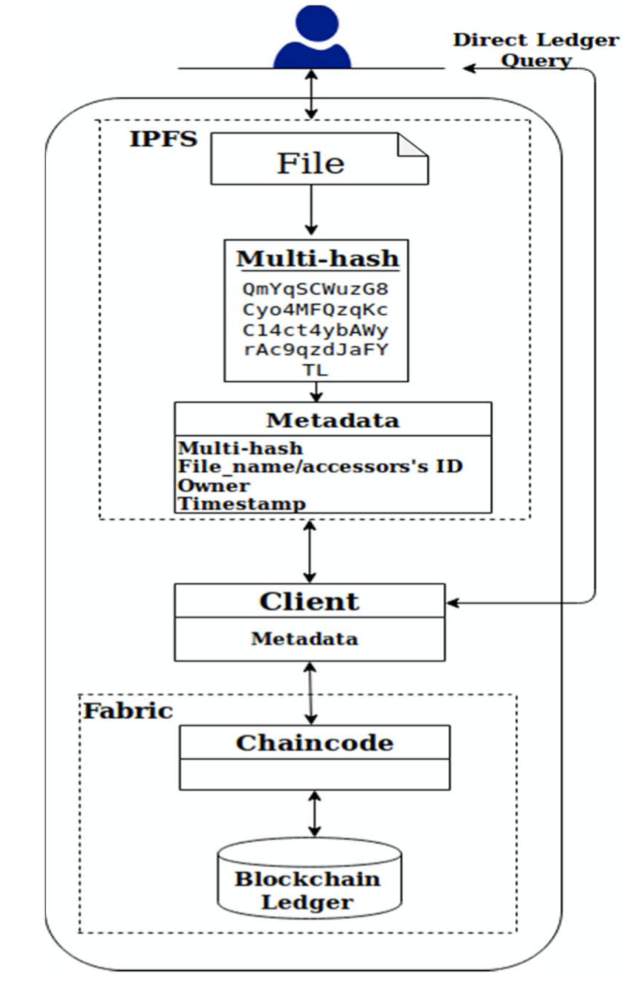
**3.1 BigChainDB**

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。



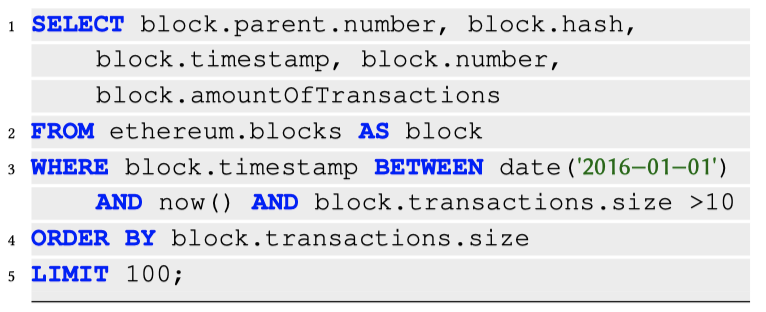
**3.2 BlockIPFS - Blockchain-enabled Interplanetary File System for Forensic and Trusted Data Traceability**

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。



**4.4 Etherscan API**

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。

1. 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。

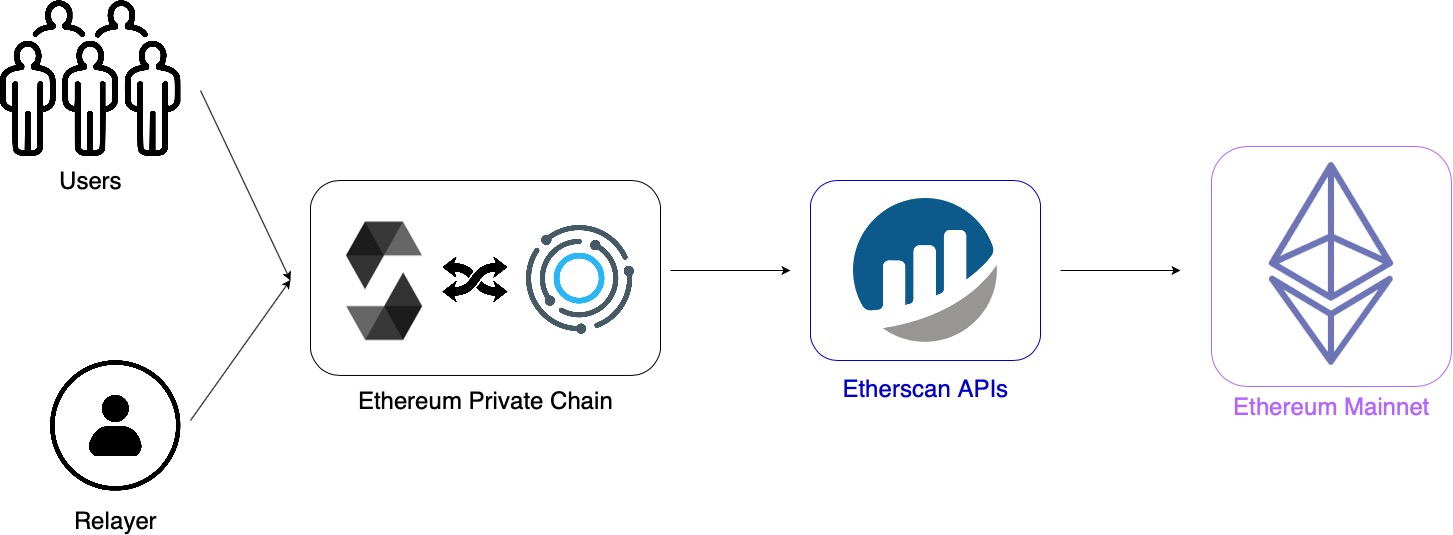
本章共分為五小節，第一小節闡述數位貨幣溯源服務之系統架構；第二小節介紹Oraclize(Provable)服務與Ethereum-Bridge；第三小節說明擷取之交易資料處理方式；第四小節描述Query之功能；第五小節展示系統介面與操作流程。

**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;
8. oraclizeIsRunning = **false**;
9. // 檢查是否有回傳值
10. **if** (bytes(\_result).length != 0) {
11. savingTx(\_result);
12. }
13. }
15. // call oraclize
16. **function** traceTx() payable **public** {
17. uint gasLimit = 50000000;
18. oraclizeIsRunning = **true**;
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。

1. contract tokenTracer is usingProvable, Parser {
2. bytes32[] transactionHash;
3. address[] sender;
4. address[] receiver;
5. uint[] value;
6. uint[] blockNumber;
7. uint[] timeStamp;
9. // 取得已儲存之交易筆數
10. uint **public** transactionCount;
12. mapping(bytes32 => bool) **private** isExist;
14. **function** savingTx(string memory json) internal {
15. uint returnValue;
16. JsmnSolLib.Token[] memory tokens;
17. uint actualNum;
19. // (returnValue, tokens, actualNum) = JsmnSolLib.parse(json, 9);
20. (returnValue, tokens, actualNum) = JsmnSolLib.parse(json, 450);
22. // 每次Oraclize取回之交易筆數
23. **for** (uint i = 0; i < 50; i++) {
24. JsmnSolLib.Token memory a = tokens[1 + 8\*i];
25. bytes32 \_transactionHash = parseStringTo32Bytes(JsmnSolLib.getBytes(json, a.start, a.end));
26. // 避免重複紀錄同一筆交易
27. **if** (!isExist[\_transactionHash] && \_transactionHash != "") {
28. isExist[\_transactionHash] = **true**;
29. transactionHash.push(\_transactionHash);
30. JsmnSolLib.Token memory b = tokens[2 + 8\*i];
31. uint \_blockNumber = parseHexToUint256(JsmnSolLib.getBytes(json, b.start, b.end));
32. 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.start, f.end)));
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

1. 結論與建議

**5.1 結論**

**5.2 未來研究方向**

參考文獻

1. Ethereum Foundation. 2014. Ethereum’s white paper. (2014).  [https://en.Bragagnolo et al.wikibooks.org/wiki/LaTeX/Bibliography\_Management](https://en.bragagnolo/)
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](https://github.com/ethereum/)
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](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 ProvableTM blockchain oracle for modern DApps. <https://provable.xyz>
9. Etherscan.io <https://etherscan.io>