**題目**

**(Title)**

**人1**

**1學校**

**E-mail: cnyang@gms.ndhu.edu.tw**

摘要

寫這

Abstract

Write here

1. 前言

現今台灣使用的國際醫療資訊交換標準HL7 CDA R2 (HL7 Clinical Document Architecture, Release 2.0) 為多年前採用的。但該標準在現代面臨以下幾種問題：醫療院所缺乏統一規範、不支援行動裝置、逐漸被國際淘汰。

1. **文獻探討**
   1. **區塊鏈**

區塊鏈的基本概念由中本聰提出[10]，本質為一個去中心化的分散式資料庫，運用密碼學方法產生相關聯的區塊，每一個區塊紀錄一段時間內全網的交易資訊及前一個區塊的編號。從起始區塊串聯至最新區塊，由眾多連接到網絡的節點維護和更新。每一個節點都包含著完整的歷史區塊，即便有一個節點資料遭修改也不會影響到整個區塊鏈的運作。區塊鏈資訊公平、公正、公開，任何人都可從鏈上搜尋任何一個地址的歷史交易資訊。大部分區塊鏈需要有礦工，否則區塊鏈無法正常運作。區塊鏈中的任意節點產生交易時，交易會被廣播到每位礦工手中，此時所有礦工需透過工作量證明機制[11]來決定誰可以驗證這筆交易。礦工將資料打包至區塊中，每個區塊都擁有一定數量的數位貨幣(如:乙太幣)，用來獎勵成功將資料打包至區塊鏈中的礦工。

另外，私有區塊鏈[12]就沒有採礦機制和礦工的角色，因為私有鏈通常作為某個企業、機構內部的的網路，私有鏈上的成員和節點會受限制。驗證新區塊的人在網路搭建時被指定，這樣一來可以規避區塊鏈被篡改的風險。例如IBM 區塊鏈平台就為四百多家企業客戶提供了私有鏈。例如，為沃爾瑪 (Walmart) 提供的方案，能透過區塊鏈提供食品追蹤的可追溯性，從而提升食品供應鏈的透明度，保障食品安全。為快桅 (Maersk) 提供的方案，則在構建全球貿易數位化平臺，提高運輸過程的效率以及流程透明度。事實上，區塊鏈的應用非常廣泛，例如在物聯網[13]，即可不需要中心電腦對所有裝置進行控制，僅使用智能合約來達成如送貨等特定功能，或是用於小額行動支付、雲端計算、數位證書[14-16]等的應用。區塊鏈未來會有更多的發展。其發展大致可以分為三個階段[17]，第一階段為數位貨幣，最著名的即為比特幣。第二階段為協定階段，開始將區塊鏈用於各式資產轉移，如股票、證券等。第三階段則是整合的共同協作平台，因區塊鏈開放API，讓各領域皆有機會利用區塊鏈進行革新。

* 1. **以太坊**

乙太坊的每個區塊鏈都有著不同的功能及目的，我們可以藉由撰寫智能合約將應用程式部署於乙太坊中[18]，所謂的智能合約[19]是「用於計算機執行合約條款的交易協議」，此合約可降低對中間人員的依賴性。將智能合約用於區塊鏈中，可以規範每個區塊以達成其去中心化的需求，在乙太坊的世界裡智能合約的運作得依靠乙太坊虛擬機 (EVM) 來完成。而乙太坊虛擬機是一個運行智能合約的環境，EVM是基於點對點網路協議設計，任何人都可以透過網路參與其中，扮演著驗證者的角色(即礦工)，礦工是支撐整個EVM最重大的支柱，當鏈上的智能合約運行時，必須透過礦工進行打包才能將資料寫入區塊鏈中。

* 1. **多付款鏈小額付款及具找錢能力之小額付款**

Rivest 等人的研究 [1] 中首次提出了使用雜湊鏈的小額付款機制。客戶隨機選擇最後一個payword，然後以相反的順序計算payword鏈*ai*=*h*(*ai*+1)，其中 *i = n*−1, *n*−2, …, 0、而 *h*(·) 是雜湊函數，例如類似 SHA 的函數。這個小額付款鏈*A*=(*a*0, *a*1, ..., *an*)中的每個payword是相同的面額，且*a*0值是用來作驗證的起始值。這種使用單一付款鏈的小額付款機制稱之為SCM (single-chain micropayment; SCM)。所謂的多付款鏈小額付款機制 (multi-chain micropayment; MCM)則是使用多條付款鏈，且每一付款鏈代表了不同的面額值。實際上，論文[1] 中已經提到了 MCM 的概念。顯然地，同樣的支付金額，使用MCM 的雜湊鏈長度和和驗證的雜湊函數運算會比SCM少。例如: MCM可使用兩個面額*d*A 和 *dB* (其中*d*A < *dB*) 的付款鏈*A*=(*a*0, *a*1, …, *an*)、和*B*=(*b*0, *b*1, …, *bn*)來付款，*A*鏈和*B* 鏈可混合使用。實際上，MCM是多個不同面額SCM的組合。

MCM 雖然減少了雜湊函數的數量，但有兩個問題：如何在付款中找到最小的雜湊鏈而減少運算，另外要如何平均使用每一付款鏈，使它們在付款後的長度還能一樣。論文 [2]介紹了在考慮計算和存儲成本時有效地使用 MCM的方法。為了解決上述兩個問題，論文 [2]提出了激進模式和平衡模式。激進模式具有較少的雜湊函數運算，而平衡模式可以避免特定付款鏈的耗盡 (註: 當付款鏈用完時，MCM 會變成 SCM。實際上，解決上述兩個問題的徹底解決方案是在MCM中加入找零的能力。具找錢能力之小額付款機制[3] 也是一種使用多付款鏈的小額付款機制。論文[3]提出了三種具找錢能力之小額付款機制: 分別使用了計數器模式的區塊加密、雜湊函數、具金鑰的雜湊函數等三種方法。

1. **植基於以太坊的線上小額付款機制**
   1. **動機**

這些以太坊的小額付款通道要完成智能合约運算時，需付出以太坊gas成本[7-9]。我們撰寫了智能合约將小額付款機制部署於以太坊虛擬機，完成更高效的線上小額付機制。智能合约主要有兩部分，一個是初始化交通道來訂定對應的金額、另一個是關閉 (結算交易) 通道，達成線下供應商所得的金額可以轉至線上 區塊鏈的金額。交易在初始化和結算交易時，都需要花費gas來運算。這造成了小額付款通道從線下至線上小額付款的額外消耗。本論文擬結合Merkle 樹及多面額的付款鏈，實作一個需要較少以太坊gas成本的線上小額付款機制。我們的方案不僅適用於MCM付款機制，也適用於傳統SCM付款機制。

* 1. **結合Merkle樹及多面額付款鏈設計概念**

本文所提減少以太坊gas成本的線上小額付款機制是利用不同面額的付款鏈，以不同高度的Merkle 樹結合完成。我們方法使用Merkle樹結合MCM付款鏈的示意圖如圖1所示。我們以兩個面額*dA 和 dB* (其中*dA < dB*)的MCM舉例說明。令*A*鏈和*B* 鏈的產生方式如下: 付款鏈*A*=(*a*0, *a*1, …, *an*)、和*B*=(*b*0, *b*1, …, *bn*′)，其中*ai*=*h*(*ai*+1)， *i = n*−1, …, 0、*bi*=*h*(*bi*+1)， *i = n*′−1, …, 0。為了實用，小額面值的 *A* 鏈比大額面值的 *B* 鏈會有較大的鏈長(也就是 *n* > *n*′)。所以，Merkle樹的高度在*A* 鏈會比*B*鏈高(也就是 *hA* > *hB*，如圖1所示)。且供應商執行智能合約做結算交易時，原本MCM做雜湊運算確認時。一般而言，大額面值的 *B* 鏈比小額面值的 *A* 鏈會有較少的運算。這是因為金額不大的小額付款，大面額的*B*鏈請款會有較少的運算。以圖1而言，若是*A* 鏈與*B*鏈剛好分成4個子鏈與2個子鏈。



**圖 1. Merkle樹結合MCM付款鏈的示意圖**

實際使用時，我們可以調整 *j* 和 *j*′值來調節線下小額付款金額轉換成線上區塊鏈加密貨幣的結算交易時程，也可以控制智能合約雜湊運算的數目。註: 這直接影響到智能合約消耗以太坊gas的費用。圖1是只有兩種面額的情況，我們可以很容易將這個概念擴展到更多面額的MCM。

圖1是基於二元 Merkle 樹。首先是將付款鏈均勻的找出其葉節點 (·) 。例如，付款鏈*A*=(*a*0, *a*1, …, *an*)，我們選了*a*0、*aj*、*a*2*j*、*an*−*j*+1當葉節點，而付款鏈*B*=(*b*0, *b*1, …, *bn*′)，我們選了*b*0、*bj*′當葉節點。然後使用雜湊函數*h*(·) 計算得內部節點(·): Ha= *h*(*b*0, *bj*′), Hb= *h*(*an*−*j*+1, *a*2*j*), Hc= *h*(*aj*, *a*0), Hbc= *h*(Hb, Hc), and Habc= *h*(Ha, Hbc)。這些內部節點儲存於結算交易的智能合約，用於線下小額付款金額轉成區塊鏈貨幣驗證用。例如，若*A*鏈用至*a*2*j*則供應商會有*an*−*j*+1和 *a*2*j*，智能合約可以使用一個雜湊運算Hb= *h*(*an*−*j*+1, *a*2*j*)驗證以得到(*n*−2*j*)×*dA*金額的加密貨幣。若是，傳統以太坊的小額付款通道要完成智能合约運算就需要 (*n*−2*j*) 個雜湊運算來驗證*an*−*j*+1和*a*2*j*是否相等以得到(*n*−2*j*)×*dA*金額。因為MCM，我們一般會選擇*j*′ *<j* (因為*dB*面額比*dA*面額)。圖1中的雙面額MCM例子，我們甚至可以一次執行5個雜湊運算用Habc驗證得到(*n*×*dA*+ *n*′×*dB*)區塊鏈貨幣，而傳統方法則需要(*n*+ *n*′)個雜湊運算。

我們使用Remix跑SHA256雜湊函数。一次雜湊函数運算所需的gas包含了運算雜湊函數的智能合約，還有precompiled contract等。假設*n*=1000、*n*′=150，若是一次性執行線下線上轉換，原來的小額付款機制需要1150次的雜湊函數運算。所需的gas成本，與我們僅需5次的雜湊運算方法相比(註:我們智能合約中使用的雜湊函數運算是先將請款者輸入之驗證碼進行XOR運算後，再計算其雜湊值，所需gas比傳統雜湊函數運算略高)。最後，我們的智能合約減少了整體以太坊gas所需的成本。

圖1的Merkle 樹是二位元樹，事實上我們的方法也可以使用多位元樹。調整MCM不同面額付款鏈的子鏈長度和多位元Merkle 樹件，我們可以在以太坊gas的消耗及轉換區塊鏈加密或秘的金額做適當的取捨。如果只用Merkle樹來計算單一付款鏈SCM的內部節點，圖1的MCM小額付款機制就變成SCM小額付款機制。若是使用MCM小額付款機制有找錢功能，就可以完成植基於以太坊的MCM線上小額付款機制。

1. **結合Merkle 樹及多面額付款鏈智能合約實作**
   1. **智能合約的佈署**

Solidity 是一種合約導向語言，可用於部屬合約於各種不同的區塊鏈平台上，本研究透過線上編譯器Remix IDE 作為編譯的環境，完成編譯後，將程式碼編譯成二進制(Contract Bytecode)後，才佈署屬於EVM。圖2、和圖3是結合Merkle樹及多面額MCM線上小額付款機制的OPEN智能合約、和CLAIM智能合約。在OPEN智能合約的open function中先輸入請款者的位址用來作日後以太坊將線下小額付款轉至線上區塊鏈加密貨幣的認證，並且設定申請時間以避免請款者遲遲未請款導致資金凍結於智能合約中。MCM與SCM不同處在於付款者需輸入*A*鏈與*B*鏈所對應的面額價值，以及各自的內部節點來作確認交易使用。另外，CLAIM智能合約的claim function中執行交易請款流程，請款者必須輸入請款金額以及驗證碼，如果請款者與創建合約所輸入之請款者不符，則智能合約會拒絕驗證。將請款者輸入之驗證碼進行XOR運算後，再計算其雜湊值來驗證是否與付款者於創建時所建立之內部節點相同。若一樣，系統將請款金額傳送至請款者錢包。

function open(address \_receiver, uint \_validityTime,uint \_wordValueb, uint \_wordValue, bytes32 \_wordRoot,bytes32 \_wordnodeb,bytes32 \_wordnodec,bytes32 \_wordnodea,bytes32 \_wordnode\_bc) public

payable

checkOwner

checkState(States.Init)

{

receiver = \_receiver;

require(now < now + \_validityTime \* 1 minutes);

expirationTime = now + \_validityTime \* 1 minutes;

wordValue = \_wordValue;

wordValueb = \_wordValueb;

root = \_wordRoot;

state = States.Open;

nodea=\_wordnodea;

nodeb=\_wordnodeb;

nodec=\_wordnodec;

node\_bc=\_wordnode\_bc;}

**圖 2. MCM線上小額付款OPEN智能合約**

function claim(bytes32 \_word,bytes32 \_word2,bytes32 \_word3,bytes32 \_word4, uint \_wordCount, uint \_wordCountb) public

checkReceiver

checkState(States.Open)

{

state = States.Locked;

wordScratch = \_word; wordScratcha1 = \_word2;

wordScratchb = \_word3; wordScratchb1 = \_word4;

bytes32 checkword = keccak256(wordScratcha1 ^ wordScratch);

bytes32 checkworda = keccak256(wordScratchb ^ wordScratchb1);

bytes32 checkwordr = keccak256(checkworda ^ checkword);

if(checkword != nodec || checkword !=nodeb || checkworda !=nodea) {

state = States.Open;

revert();

}

if(checkwordr != root) {

state = States.Open;

revert();

}

if (msg.sender.send(\_wordCount \* wordValue+\_wordCountb \* wordValueb)) {

selfdestruct(owner);

}}

**圖 3. MCM線上小額付款CLAIM智能合約**

如前面敘述，若只用Merkle樹來計算單一付款鏈SCM的內部節點， MCM小額付款機制就變成SCM小額付款機制。圖4、和圖5是結合Merkle樹及單一面額SCM線上小額付款機制的OPEN智能合約、和CLAIM智能合約。主要部分與MCM一致，只需依照不同的Merkle樹結構，適當修改MCM的智能合約即可得SCM的OPEEN和CLAIM智能合約。但是特約商店的請款者如果超過智能合約設定的時間沒有進行請款，則將資金將返還使用者(如圖6所示)。

function open(address \_receiver, uint \_validityTime, uint \_wordValue, bytes32 \_wordRoot,bytes32 \_wordnodeb,bytes32 \_wordnodec) public

payable

checkOwner

checkState(States.Init)

{

receiver = \_receiver;

require(now < now + \_validityTime \* 1 minutes);

expirationTime = now + \_validityTime \* 1 minutes;

wordValue = \_wordValue;

root = \_wordRoot;

state = States.Open;

nodeb=\_wordnodeb;

nodec=\_wordnodec;

}

**圖 4. SCM線上小額付款OPEN智能合約**

function claim(bytes32 \_word,bytes32 \_word2, uint \_wordCount) public

checkReceiver

checkState(States.Open)

{

state = States.Locked;

wordScratch = \_word;wordScratch2 = \_word2;

bytes32 checkword = keccak256(wordScratch2 ^ wordScratch);

if(checkword != nodec || checkword !=nodeb) {

state = States.Open;

revert();

}

if (msg.sender.send(\_wordCount \* wordValue)) {

selfdestruct(owner);

}

}

**圖 5. SCM線上小額付款CLAIM智能合約**

function refund() public

checkOwner

checkTime

checkState(States.Open)

{

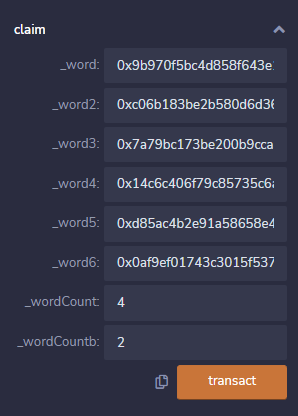
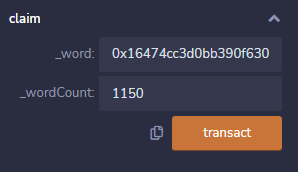
selfdestruct(owner);

}

**圖 6. 線上資金返還智能合約**

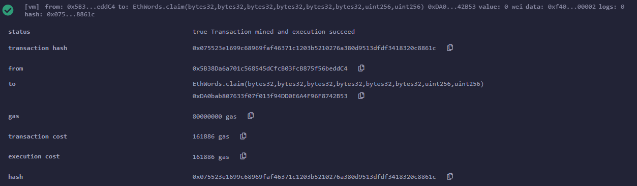
* 1. **Remix執行智能合約所需Gas的實驗**

我們就以3.2節所敘述的使用情境，來比較傳統以太坊線上小額付款機制、與我們MCM線上小額付款機制。假設*n*=1000、*n*′=150，且一次性執行線下線上轉換，原來的小額付款機制需要1150次的雜湊函數運算，而我們需5次的雜湊運算(註:這兩種運算約略不同)。圖7、(a)是MCM以太坊線上小額付款機制，使用者輸入所要請款的驗證碼以及不同面額鏈所要驗證的個數，\_word、\_word1、\_word2及\_word3為申請*A*鏈之驗證碼，\_word4、\_word5為申請*B*鏈之驗證碼。圖7、(b)是傳統以太坊線上小額付款機制，使用者輸入所要請款的驗證碼以及驗證時所需要進行的雜湊次數。圖8為這兩種機制完成claim時所產生的明細，MCM以太坊線上小額付款機制的請款所花費的gas為161886 (如圖8、(a))，而傳統以太坊線上小額付款的請花費的gas為294682 (如圖8、(b))。

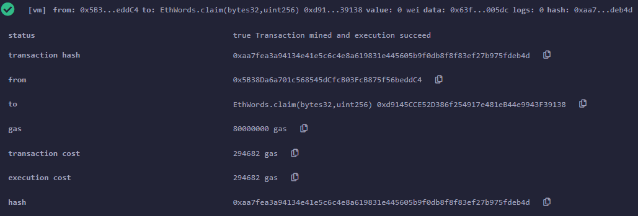
** **

1. (b)

**圖 7. 請款者所輸入參數: (a) MCM以太坊線上小額付款 (b) 傳統以太坊線上小額付款**

****

(a)

****

(b)

**圖 8. CLAIM所花費之gas 明細: (a) MCM以太坊線上小額付款 (b) 傳統以太坊線上小額付款**

1. **結論**

本文結合了Merkle 樹 以及 雜湊鏈的小額付款，以智能合约實作較少以太坊Gas成本的線上小額付款機制。將線下小額付款擴展至線上小額付款。主要設計理念是把付款鏈區分成子付款鏈。把子付款鏈的端點，當作Merkle樹的葉節點，計算出Merkle樹的內部節點儲存於結算交易的智能合約作驗證用。由於我們的方法使用了Merkle樹及多面額的付款鏈。結算交易需要的雜湊運算較少，可有效節省智能合约所需的gas成本。

## 6. 致謝

本研究部分成果由科技部計畫，編號 MOST 110-2221-E-259-005-MY2 補助，特此致謝。

**參考文獻**

1. R. Rivest and A. Shamir, “PayWord and MicroMint: two simple micropayment schemes,” Proceedings of the International Workshop on Security Protocols, LNCS vol. 1189, pp. 69-87, 1996.
2. C.N. Yang and H.T. Teng, “An Efficient Method for Finding Minimum Hash Chain of Multi-Payword Chains in Micropayment,” IEEE Conference on E-Commerce, pp.45-48, California, USA, June, 2003.
3. K. Chaudhary, X. Dai X, and JC.Grundy, “Experiences in Developing a Micro-payment system for Peer-to-peer Networks,” International Journal of Information Technology and Web Engineering, vol. 5, pp. 23-42, 2010.
4. C.N. Yang and C.C. Wu, “MSRC: (M)icropayment (S)cheme with Ability to (R)eturn (C)hanges,” Mathematical and Computer Modelling, vol. 58, pp. 96-107, July, 2013.
5. AT. Swe, KKK. Kyaw, “Improved E-cash Protocol,” International of Science & Technology Research, vol. 2, pp. 28-31, 2013.
6. C. Chen, C. Wu, W. Lin, “Improving an On-line Electronic Check System with Mutual Authentication,” International Conference on Advanced Information Technologies, 2010.
7. M. Di Ferrante, “Ethereum payment channel in 50 lines of code,” 2017. Available at https://medium.com/@matthewdif/ethereum -payment-channel-in-50-lines-of-code-a94fad2704bc.
8. M. Elsheikh, J. Clark, and A. M. Youssef, “Deploying Payword on Ethereum,” International Conference on Financial Cryptography and Data Security Workshops, vol. LNCS 11599, pp. 82–90, 2020.
9. H.S. Galal, M. ElSheikh, and A.M. Youssef, “An Efficient Micropayment Channel on Ethereum,” Data Privacy Management, Cryptocurrencies and Blockchain Technology. DPM 2019, CBT, vol. LNCS 11737, 2019.
10. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. Available at https://bitcoin.org/bitcoin.pdf.
11. M. Vukolić, “The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication,” Open Problems in Network Security, pp. 112-125, 2015.
12. Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,” 2017 IEEE International Congress on Big Data (BigData Congress), pp. 557-564, 2017.
13. K. Christidis and M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things”. IEEE Access, Vol.4, pp. 2292-2303, 2016.
14. A. Xu, M. Li, X. Huang, N. Xue, J. Zhang, and Q. Sheng, “A Blockchain Based Micro Payment System for Smart Devices,” International Journal of Design, Analysis and Tools for Integrated Circuits and Systems (IJDATICS), 2016.
15. Jin Ho Park, Jong Hyuk Park, “Blockchain Security in Cloud Computing: Use Cases, Challenges, and Solutions”, Symmetry, 9, 164; doi:10.3390/sym9080164, 2017.
16. 鄭錦楸,李南逸,陳宜樺,冀謙,陳昭宇, 「區塊鏈數位證書」,離島會議 (ITAOI), 2018
17. 楊惟雯，全球區塊鏈發展現況與趨勢，available at http://nmart.pixnet.net/blog/post/65851006-全球區塊鏈發展現況與趨勢。
18. V. Buterin. “Ethereum white paper: a next generation smart contract & decentralized application platform” white paper, 2018.
19. N. Szabo, “Smart contracts, 1994,” Virtual School. Available at http://szabo.best.vwh.net/smart.contracts.html, 1994.