**題目**

**(Title)**

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

摘要

由於病患們會於不同醫療院所接受治療及診斷，故每位病患的醫療資訊將會分布在不同的醫療院所，而這些資訊並沒有統一格式，這對於不同醫院間醫療資訊的交換實屬不易。不同院所間醫療資訊的交換及互通性對於醫療健康的進步及醫生診斷的效率是件非常重要的事，故對於電子病歷擁有統一格式的FHIR (Fast Healthcare Interoperability Resources; FHIR) 及RESTful API便為我們所用。

雖然FHIR提供了電子病歷的統一格式及架構，但並未提供我們安全上的協議，故安全疑慮一直是FHIR的一個問題。在此研究中，為了更安全地使用FHIR協議，我們將以去中心化、無法被更改的區塊鏈做為身分驗證的方式以確保資料的安全，我們將使用Hardhat作為以太坊應用的開發環境。

此外，我們也將使用前端框架開發網頁，並於後端進行身分驗證，再呼叫以太坊中的智能合約依據白名單做二次身分驗證，給予FHIR安全上的保障。

**關鍵詞**：區塊鏈、以太坊、智能合約、FHIR、電子病歷、RESTful API、去中心化、Hardhat、身分驗證。

Abstract

1. 前言

一套完善且可廣泛用於各家醫療院所的醫療資訊標準可以大幅度減少醫療及照護人員工作時的負擔。國際標準組織HL7 International (Health Level Seven International; HL7 International) [1]2005年推出的HL7 CDA R2 (HL7 Clinical Document Architecture, Release 2.0; HL7 CDA R2) 為現今臺灣使用的醫療資訊交換標準。但該標準在資訊量較大的現代面臨以下幾種問題：缺乏統一資料規範規範、難支援行動裝置及逐漸被國際淘汰[2]。因此，衛福部逐步推廣使用HL7組織新設計的FHIR標準來取代舊有的標準。FHIR透過固定的病例格式來解決缺乏統一規範的問題，使病歷可在不同醫療院所中傳遞並使用。FHIR也使用RESTful API使資料可在行動裝置中傳遞，並可與AIoT醫材互動存取及分析資料。綜合上述優點，國際間正廣泛的進行醫療標準的更換以及相關套件的開發。

然而，FHIR並未規定明確的驗證規範，僅在官方文件中建議使用OAuth 作為身分認證標準。因此，本研究將使用以太坊智能合約的白名單作為身分認證的標準，透過區塊鏈去中心化、無法被修改的特性來確保存取者是否擁有存取權限或未授權的資料調閱。開發過程中使用Hardhat框架幫助智能合約開發、測試及上鏈以及Alchemy作為RPC server。合約中使用Merkle樹實作較少gas消耗的白名單，改善常規白名單過度消耗gas成本以及存取速度過慢的問題。在FHIR資料庫中使用JSON( JavaScript Object Notation; JSON)格式取代常規的XML格式，獲取更好的儲存、傳輸效率。最後結合前端Angular框架構建完整前端頁面，讓使用者可以清楚操作FHIR的API(Application Programming Interface; API)。本文結構如下，第二部分為文獻探討，介紹電子病歷、FHIR、以太坊及RESTful API。第三部分介紹研究動機與設計概念，第四部分為實作的過程，最後則為結論。

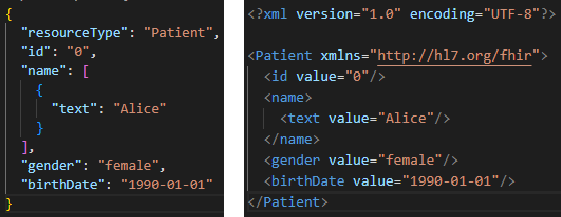
1. **文獻探討**
   1. **電子病歷**

電子病歷是將傳統的紙本病歷電子化並上傳至網路上，有著傳統病歷所沒有的優點，如易查詢、管理及不同醫療院所間的操作互通性，電子病歷內容可包含診療紀錄、藥物過敏史、各項檢查的影像及報告或疫苗施打紀錄等。

而電子病歷的發展對於醫療健康的進步及不同院所間病歷的交換十分重要，電子病歷能有效地降低醫療成本及加速診斷流程使得醫院及病患能有更完善、有效率的醫療體驗。

* 1. **FHIR**

以臺灣電子病歷交換中心的電子病歷標準與其參考的 CDA R2 標準對比。首先，FHIR透過統一資料格式，使醫療資料可於不同醫療院所中相互傳輸，免去民眾去不同醫院調閱病歷的麻煩及去不同醫院時需要重新填寫病例的資源浪費。FHIR也支援臨床、非臨床資料，也可於不同裝置及平台中互通，讓手機、電腦、醫療儀器等都可XML格式，FHIR則同時支援XML及JSON。JSON的檔案格式較XML簡單，因此在儲存空間及傳輸效率上有相當大的優勢，目前也有套件支援將XML轉為JSON格式的資料。FHIR有龐大的使用者社群，提供大量輔助工具，可將獲取的資料進一步分析並搭配人工智慧進行人體風險預測。圖1左為JSON格式、右為XML格式之資料。

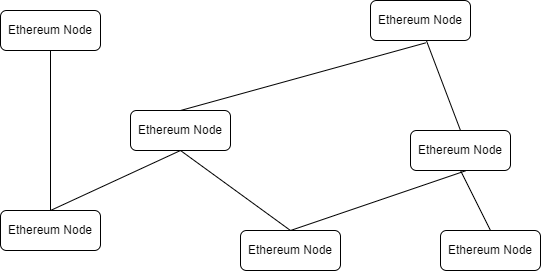


**圖 1. JSON與XML格式之FHIR資料**

* 1. **以太坊**

以太坊[3]是一種底層區塊鏈技術，作為分佈式

計算機網路運行可以驗證區塊及交易的程式。以太坊目前採用工作量證明作為共識機制，透過大量算力找出隨機調整數來決定下個區塊的添加者，並給予原生代幣以太幣作為獎勵，該代幣可作為鏈上Gas消耗或是其他用途，這個過程通常被稱為挖礦。以太坊的運作節點中存在以太坊虛擬機(Ethereum Virtual Machine; EVM)，用來在固定環境中運行智能合約，圖2為Ethereum網路示意圖。智能合約需要先編譯成EVM可以判別的指令碼後才能部署上區塊鏈執行，並且EVM中的指令碼有相應對應的價格，因此執行智能合約時會消耗相應價格的以太幣[4]，用來防止惡意的鏈上資源消耗。智能合約則是一種運行在區塊鏈上的程式，使用者可以在支付Gas後執行該合約的內容，智能合約也同時有上鏈後無法更改、公開透明等特點。另外，一般的網路服務在部署後仍需要維持伺服器的運作才能確保服務的運行，但區塊鏈上的智能合約僅需於部署時支付費用即可儲存於以太坊的各個節點中。



**圖 2. Ethereum網路示意圖**

* 1. **RESTful API**

RESTful API[5] 是一種能讓兩個電腦系統安全地透過網際網路交換資訊的介面，REST全名Representational State Transfer (表現層狀態轉移)為一種軟體架構，對於API的運作方式施加了條件，可以輕鬆實作和修改，替任何API系統提供可視性及跨平台可移植性，而RESTful API只是REST架構的Web服務，而Web API可被視為用戶端及Web上資源間的通道。

由於RESTful API具有可擴展性、靈活性及獨立性的優勢，故被許多工程師廣為使用。RESTful API的基本功能和網際網路相同，用戶端使用API聯絡伺服器，而RESTful API請求須包含唯一資源識別符 (伺服器通常使用URL進行資源識別)及方法 (通常使用HTTP實作RESTful API)，四種常見的HTTP方法以告知伺服器需要對資源做什麼：GET (用於存取伺服器上指定URL的資源)、POST (用於向伺服器傳送資料)、PUT (用於更新伺服器上的現有資源)、DELETE (用於請求移除資源)。

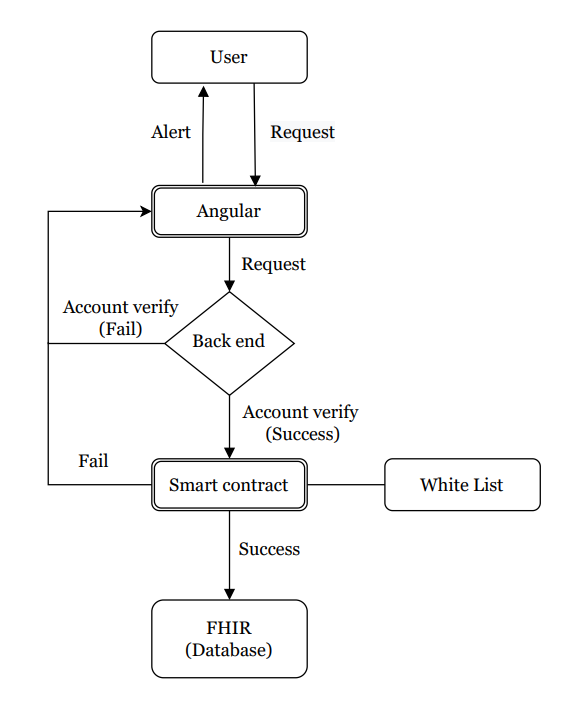
1. **題目**
   1. **動機**

在FHIR標準推出後已經有許多國家採用，並將其定為該國的醫療資訊標準。對於開發者來說也有許多開源的輔助套件可以協助分析資料，可以搭配人工智慧進行風險預測。為了降低當前臺灣醫療院所跨傳輸資料的成本以及趕上國際當前的趨勢，本研究擬結合FHIR標準與區塊鏈技術，實作一個具有完整前端框架之系統。FHIR標準中使用JSON取代常見的XML，獲取更好傳輸的效能。智能合約中主要使用白名單方法決定訪問者的權限，並使用Hardhat做為開發環境。前端介面使用Angular框架搭配後台進行資料的存取及合約的調用。透過完整的系統架構，期盼能夠提供當前的醫療資訊標準系統作為參考。

* 1. **結合FHIR及智能合約設計概念**

本文所提之FHIR系統是利用智能合約在使用

者登入時附加第二層的身分驗證，透過區塊鏈不可竄改的驗證系統，進一步提升帳號的安全。FHIR系統示意圖如圖3所示。



**圖 3. FHIR系統示意圖**

使用者在Angular框架之前端頁面登入後，先對使用者的帳號密碼做識別。通過常規與後台互動的登入方式，先對登入的使用者進行初步的識別，過濾掉大部分的非法登入。接下來進行第二步驟的驗證，與鏈上合約互動。將使用者的錢包地址、帳號序號及部分資料運算後與Merkle樹作結合，並將最後的結果與Merkle Root做比對。透過與帳號地址以外的資料進行運算，可以確保攻擊者不能透過任意存在於白名單中的地址存取資料。

兩步驟的驗證皆通過後才可以合法調用FHIR資料庫內的資料並將資料呈現於前端頁面或是進行其他操作。

1. **結合FHIR及智能合約的完整系統實作**
   1. **智能合約與部署**

作為合約導向語言，Solidity在被部署上鏈後 即可持續運行。本研究使用開源框架Hardhat進行測試及部署，Hardhat提供許多插件供開發者使用，因此在撰寫、測試及部署上都對開發者十分友善。本研究同時使用開源的智能合約倉庫OpenZeppelin來協助開發，該倉庫提供不同的智能合約模板，透過這些已經進行測試過並被社群大量使用及維護的模板，確保開發的安全以及簡化繁瑣的流程。圖4為更改Merkle Root之程式碼，透過該程式碼可以對鏈上儲存的Merkle Root進行更動。當有新的使用者註冊時，必須呼叫該段程式重新計算後Merkle Root後傳送到鏈上進行更動，確認鏈上的Merkle Root代表當前所有用戶之雜湊值。圖5為驗證Merkle值正確性之程式碼，該段程式為主要驗證該次登入合法。該程式會從後台傳入作為驗證使用之Merkle Proof以及該帳號的ID。執行該程式時會先計算葉節點，計算的方式為該用戶的地址以及嘗試登入的帳號的ID雜湊值。獲取葉節點的雜湊值後，使用OpenZeppelin中的驗證功能比對最終結果是否與鏈上儲存的Merkle Root相同。

function setMerkleRoot(bytes32 \_merkleRoot)

public onlyOwner {

    merkleRoot = \_merkleRoot; }

**圖4. 設置Merkle Root之程式碼**

function isWhitelist(

bytes32[] calldata \_merkleProof, bytes32 \_accountID)

public view returns(bool) {

    bytes32 leaf =

keccak256(abi.encodePacked(msg.sender, \_accountID));

    if(MerkleProof.verify(\_merkleProof, merkleRoot, leaf)){

        return true;

    } else{

        return false;

    }

}

**圖 5. 驗證Merkle值正確性之程式碼**

* 1. **前後端及智能合約串接**
  2. **FHIR系統**

1. **結論**

**參考文獻**

1. HL7 : <https://www.hl7.org/>
2. 國家生技醫療產業策進會，”15年前電子病歷交換舊標準4大痛點成了臺灣醫界數位創新困境”。
3. Ethereum : https://ethereum.org/zh-tw/developers/docs/
4. Contract)與分散式網頁應用(dApp)入門。Available at <https://gasolin.gitbooks.io/learn-ethereum-dapp/content/what-is-ethereum.html>
5. RESTful API: https://aws.amazon.com/tw/what-is/restful-api/