**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能大大提升醫療健康的進步及醫生診斷的效率，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，我們將主要以去中心化、無法被更改的區塊鏈做為身分驗證的方式，並使用AES加密演算法及祕密分享作為身分驗證的標準，而秘密分享的加解密過程將透過智能合約於區塊鍊上進行，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

一套具有良好的操作互通性的醫療資訊標準可以廣泛地被用於各醫院及醫材廠商，在資料的使用、傳輸、開發研究上都可以大幅減少人員的負擔。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：缺乏統一的資料規範、不易支援行動裝置及逐漸被國際淘汰 [2]。因此，衛福部正推廣使用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，也減少不同醫材廠商數據標準不一的問題。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，使資料的運用更加彈性。綜合上述優點，國際間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[3]。因此本研究將使用AES加密演算法搭配秘密分享作為身分驗證的標準，透過AES加密演算法在資料庫中儲存使用者密碼的密文，並透過秘密分享分散式儲存加解密金鑰，使金鑰被竊取的難度提升。秘密分享的加解密過程在區塊鏈上透過智能合約運行，利用區塊鏈透明、去中心化、無法被修改的特性來對屬性進行調用及運算。開發過程中使用Hardhat作為智能合約開的框架，幫助智能合約開發、測試及上鏈，並使用Alchemy作為RPC server與鏈上合約互動與調用。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框架構建前端頁面，讓使用者可以清楚簡易地操作FHIR中的API (Application Programming Interface; API)，對資料進行調閱或是修改。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、AES (Advanced Encryption Standardt; AES) 及SSS (Shamir Secret Sharing; SSS)。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。

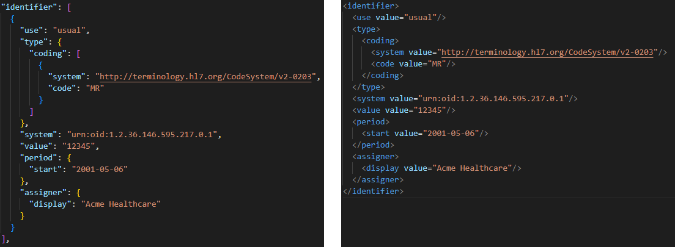
1. **文獻探討**
   1. **電子病歷**

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，免去民眾在不同醫院所中調閱病歷的麻煩及去不同醫院時需要重新建構病例造成的資源浪費，民眾只需要將自己的病歷資料匯出或是許可其他醫院可以調閱需要的資料。FHIR也支援臨床、非臨床資料，並可於不同裝置及平台中互通，可以解決不同醫材廠商資料格式不一產生的應用困難。CDA R2僅可支援XML格式，而FHIR可同時支援XML及JSON。JSON的檔案格式較XML在儲存空間及傳輸上有優勢，但JSON屬於較新的技術，因此使用仍未非常普及。FHIR有龐大的使用者社群，提供其他輔助工具，可將獲取的資料進一步分析並搭配模型進行人體風險預測或其他分析。圖1左為JSON格式、右為XML格式之FHIR官方檔案樣本 [4]。FHIR實作以一系列Resource(資源物件)組成，故使用者可以輕鬆使用、微調這些資源物件，因此能被快速應用及實做於臨床醫療實務，而這些資源物件可由病患的各項資料組成，如身份證字號、姓名、電話等。此外，FHIR能支援行動化應用，能有效解決CDA R2擁有的侷限。在資料互通性上，FHIR進一步瞄準支援行動應用資料交換的互通架構，讓FHIR資料能在各種行動裝置上傳輸資料。在檔案傳輸方式上，台灣EEC以自行開發的專用API，以閘道器來撈取、傳輸CDA R2格式資料，而FHIR採用網頁應用及行動APP常使用的RESTful API做為資料串接的方式，故FHIR能適用於更多網頁應用。

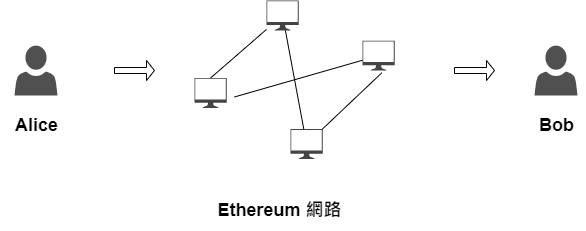


**圖 1. JSON與XML格式之FHIR官方檔案樣本**

* 1. **以太坊**

以太坊[5]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，並給予以太坊的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。



**圖 2. Ethereum網路示意圖**

智能合約則是一種運行在區塊鏈上的程式，並且需要先編譯成EVM可以判別的Bytecode後才能部署上區塊鏈執行，且EVM中的Bytecode有相應對應的價格，執行智能合約時會消耗相應價格的以太幣[6]，用來防止惡意的鏈上資源消耗。智能合約也同時有上鏈後無法更改、公開透明等特點，因此常被用作去中心化需求的開發。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。智能合約被部署上鏈後，需要透過ABI (Application Binary Interface; ABI) 進行調用，ABI為被編譯後的合約原始碼互動的方式，描述了函式的使用方式，包括修飾詞、參數、回傳值等。ABI以JSON格式儲存，但必須先以EVM可以理解的Bytecode格式做轉換後才可以傳送，智能合約同樣以Bytecode的格式回傳[7]。

* 1. **RESTful API**

RESTful API[8] 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)。

一張含有 文字 的圖片

自動產生的描述

**圖 3. FHIR HAPI 範例**

* 1. **AES**

AES加密標準由美國國家標準與技術研究院於

2001年11月26發布，為目前最多人使用的對稱金鑰加密演算法之一。AES的加密過程是透過被稱為狀態矩陣的初值為明文的矩陣進行運算，加密過程中的重複回合數不同會產生不同長度的金鑰。除最後一輪的加密外，每回合皆會重複四個步驟SubBytes 、ShiftRows、MixColumns及 AddRoundKey。

加密過程的第一步SubBytes會先將明文依序替換為狀態矩陣中。第二步驟ShiftRows將矩陣進行運算，使矩陣中的元素左旋特定的位元組。第三步驟MixColumns中每行透過線性變換互相結合，並與特定的多項式作乘法。步驟四AddRoundKey，矩陣會與主秘鑰產生的回合金鑰進行合併。 在最後一回合中，省略MixColumns的運算。

* 1. **Shamir Secret Sharing**

SSS是秘密分享的一種方式，秘密分享就是將

密文以特殊的方式拆分成碎片後保存，並且需要指定份數以上的碎片才可以將密文恢復。

中，為密文拆分的碎片數，為恢復密文所需的碎片數，且。

在加密的過程中建構多項式，其中為需要分享的秘密，為隨機產生的係數。透過產生個不相等的進入多項式計算，獲得個值，並且保密所產生的。

解密則透過拉格朗日差值法，將組的進行運算後可以求得被銷毀的原建構多項式。

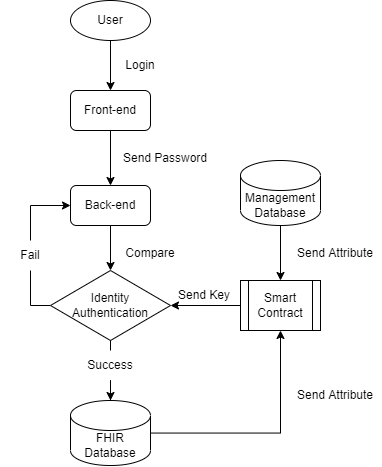
1. **題目**
   1. **動機**

隨著科技的進步，人們的生活越來越仰賴網際網路，生活中有許多東西都朝著數位化的方向發展，如電子支付、網路銀行及線上預約看診等，科技的進步提供了人們生活的便利，而我們便開始思考什麼東西數位化後能為人們帶來更便利及更有效率的生活。由於近年來疫情肆虐，許多醫療行為也漸漸線上化，如遠距診斷及索取處方籤等，因此電子病歷慢慢被廣為使用，但台灣仍使用操作互通性不高及無統一格式的HL7 CDA R2標準，為了降低當前臺灣醫療院所間傳輸資料的成本以及提升操作互通性，故本研究欲引用具有統一格式的FHIR作為電子病歷之標準。在FHIR標準推出後已經被許多國家所採用，並將其定為該國的醫療資訊交換標準，而對於開發者來說也有相當多的開源輔助套件以協助分析資料，此外，還可以搭配人工智慧進行風險預測。由於FHIR標準並未提供電子病歷安全上的規範，本研究擬結合FHIR標準與區塊鏈技術，實作一個具有安全及完整的前後端框架系統。此外，FHIR標準中使用JSON取代常見的XML，以獲取更好的傳輸效能。而智能合約中主要進行密鑰的加解密，將AES演算法獲得的密鑰以秘密分享的形式碎片化後，僅在需要時於鏈上重組並回傳至後台進行解密。本研究以Hardhat做為智能合約之開發環境。前端介面則使用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，期盼能為當前的醫療資訊標準系統提供參考。

* 1. **結合FHIR及智能合約設計概念**

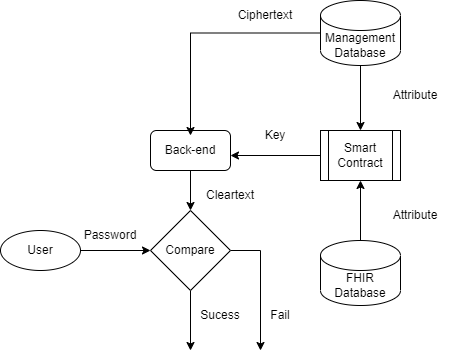
本文所提之FHIR系統是將伺服器儲存

之密碼以AES加密演算法加密後，將金鑰使用SSS的方式，透過調用智能合約於鏈上將金鑰分解成碎片後分散式儲存，並將屬性值回傳至對應的資料庫中儲存。需要進行解密時再調用智能合約，將對應的值傳送至智能合約中並計算金鑰，再將金鑰傳送至後端進行解碼，確認密碼是否一致。透過AES加密演算法搭配秘密分享，攻擊者即便獲得帳號管理伺服器內的密碼密文，仍需破解其餘存有金鑰碎片的地方，獲取足夠數量的碎片後才可以對金鑰進行復原，才能還原使用者的密碼。透過分散式的儲存，可以大幅提升系統遭到破解的難度。FHIR系統示意圖如圖3所示。



**圖 4. FHIR系統示意圖**

使用者在Angular框架之前端頁面發出登入請求後，由後台接收並調用身分驗證的功能。該功能會將此帳號恢復AES金鑰所需之屬性分別由身分控管、FHIR病歷資料庫傳送至鏈上合約，由合約透過恢復函式重新計算出金鑰後回送至後台。後台接受到金鑰後將儲存之密文解密，並將得到的明文與使用者輸入之密碼比對，相同後才可以調用相關的FHIR API，不同則拒絕此次登入請求。身分驗證系統示意圖如圖4所示。



**圖 5. 身分驗證系統示意圖**

通過身分驗證後才可以開始使用該系統的功能，合法調用使用者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。本研究使用開源的智能合約框架Hardhat進行測試及部署，Hardhat提供許多插件供開發者使用，在撰寫、測試及部署上皆對開發者十分友善。本研究同時使用開源的智能合約倉庫OpenZeppelin來協助開發，OpenZeppelin提供不同的智能合約模板，透過這些已進行測試過並被社群大量使用及維護的模板，確保合約的安全、完整性並簡化開發中的繁瑣流程。圖6為SSS金鑰拆解程式碼。當使用者註冊、更改密碼時後台必須從鏈上調用此程式碼，將傳入之金鑰作為亂數生成多項式的常數，搭配傳入的屬性進行運算，求得所有屬性的值並回傳至伺服器儲存，並且不對該函式進行保留。圖7為其中的函式程式碼，該程式碼為多項式的實作，使程式碼較為整潔也方便做維護及修改。

function encode(int256 \_att0, int256 \_att1, int256 \_att2,

int256 \_key) public payable returns (int256, int256, int256){

    int256 coef0 = \_key;

    int256 coef1 = random();

    int256 coef2 = random();

    int256 attVal0 = encodePolynomial(coef0, coef1, coef2, \_att0);

    int256 attVal1 = encodePolynomial(coef0, coef1, coef2, \_att1);

    int256 attVal2 = encodePolynomial(coef0, coef1, coef2, \_att2);

    return (attVal0, attVal1, attVal2);}

**圖 6. SSS金鑰拆解程式碼**

function encodePolynomial(

int256 \_coef0, int256 \_coef1, int256 \_coef2,

int256 \_variable)

private pure returns (int256) {

  return \_coef0 + \_coef1 \* \_variable +

\_coef2 \* \_variable \* \_variable;}

**圖 7. 程式碼**

圖8為SSS金鑰重組程式碼 。當使用者發出登入請求後，後端由鏈上調用該段程式碼，並依據傳入的參數，透過拉格朗日差值法回復原加密函式，函式求得之常數項及為AES加密法之密鑰。圖9為其中函式之程式碼，該段程式碼為拉格朗日插值法針對求常數項展開的方程式，減少完整展開造成的資源消耗。運算完後將求得的密鑰回傳至後端，與使用者輸入之密碼進行比對。

function decode(int256 \_att0, int256 \_att1, int256 \_att2,

int256 \_attVal0, int256 \_attVal1, int256 \_attVal2)

public payable returns(int256) {

int256 key = decodePolynomial(\_attVal0, \_att0, \_att1, \_att2) +

               decodePolynomial(\_attVal1, \_att1, \_att0, \_att2) +

               decodePolynomial(\_attVal2, \_att2, \_att0, \_att1);

  return key;}

**圖 8. SSS金鑰重組程式碼**

function decodePolynomial

(int256 \_y, int256 \_x1, int256 \_x2, int256 \_x3)

private pure returns (int256) {

  return \_y\*\_x2\*\_x3/((\_x1-\_x2)\*(\_x1-\_x3));}

**圖 9. 程式碼**

* 1. **前後端及智能合約串接**
  2. **FHIR**

圖7及圖8為圖1之兩種格式的官方完整檔案大小，從圖片可以得知，使用JSON格式的檔案大小為3.82KB，而XML的檔案大小則式4.44KB。在簡單的病例下即可觀察出資料格式差別造成的檔案大小差異，因此當病例的內容及數量增加時必定會需要消耗更大的資源去儲存。

一張含有 文字 的圖片

自動產生的描述

**圖 8. 圖1之範例JSON格式檔案大小**

一張含有 文字 的圖片

自動產生的描述

**圖 9. 圖1之範例XML格式檔案大小**

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. FHIR Security:

https://hl7.org/FHIR/security.html#authentication

1. FHIR Resource Patient – Examples:

https://hl7.org/fhir/patient-example.html

1. Ethereum : https://ethereum.org/zh-tw/developers/docs/
2. 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>
3. Alchemy-What is an ABI of a Smart Contract?: https://www.alchemy.com/overviews/what-is-an-abi-of-a-smart-contract-examples-and-usage
4. RESTful API: https://aws.amazon.com/tw/what-is/restful-api/