國立交通大學

資訊科學與工程研究所

碩 士 論 文

雲端儲存系統中個人健康紀錄之安全存取控制–使用屬性加密機制

Secure Access Control of Personal Health Records in Cloud Computing Using Attribute-Based Encryption

研 究 生：戴靜瑤

指導教授：曾文貴　教授

中 華 民 國　一〇二　年　六　月

雲端儲存系統中個人健康紀錄之安全存取控制-使用屬性加密機制

Secure Access Control of Personal Health Records in Cloud Computing Using Attribute-Based Encryption

研 究 生：戴靜瑤　　Student：Ching-Yao Dai

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2013

Hsinchu, Taiwan, Republic of China

中華民國一〇二年六月

雲端儲存系統中個人健康紀錄之安全存取控制-使用屬性加密機制

學生：戴靜瑤　　指導教授：曾文貴

國立交通大學資訊科學與工程研究所碩士班

摘要

Personal Health Record (PHR)是一種醫療資訊紀錄，其來源包含病人紀錄的個人健康資訊如飲食型態、家族病史、醫生處方籤等，由病人所擁有並且管理。近年來雲端技術的蓬勃發展，各式各樣的雲端應用服務應運而生，其中雲端儲存系統提供了使用者便利的資料儲存服務，而病人將 PHR 存放在雲端儲存系統帶來了許多好處，像是有利於他集中管理、節省儲存空間以及方便分享給其他使用者，然而此種做法卻衍生了新的問題：病人要如何對儲存在雲端的 PHR 達到安全的存取控制？一旦病人將儲存 PHR 的任務交由雲端儲存服務供應商幫忙處理，病人便無法即時監控他的 PHR 是否遺失抑或是遭到竊取甚至濫用。換句話說，雲端儲存服務供應商並不保證 PHR 的是否被安全存取以及 PHR 的隱密性，病人的隱私資料未得到妥善的保護。在與其他使用者共享 PHR 的部分，病人對其 PHR 理應擁有絕對的控制權，要如何讓病人能夠針對不同層級的使用者給予相對的權限來存取他的 PHR 亦是一個重要的議題。因此，在病人將 PHR 存放在雲端儲存系統之前，需要透過一套安全且有效率的加密技術用以保護病人的隱私不致洩漏，本篇研究提出一個 PHR 系統的框架，利用 Decentralizing Attribute-Based Encryption (DABE)的技巧來加密病人的 PHR，達成在雲端儲存服務上，對 PHR 安全的存取控制與共享的目標，進而提升對病人隱私資料的保護。而我們所提出的框架也會滿足細粒度存取控制、資料隱私與可拓展性等安全需求。最後我們實作了一個雲端化的 PHR 系統，展示我們的系統是可行的。

Secure Access Control of Personal Health Records in Cloud Computing Using Attribute-Based Encryption

Student：Ching-Yao Dai Advisor：Dr. Wen-Guey Tzeng

Institute of Computer Science and Engineering
College of Computer Science,
National Chiao Tung University

Abstract

Personal Health Record (PHR) is an information related to the care of a patient himself and is maintained and managed by patient. PHR contains variety of health data such as a patient's diet, family history, and prescription record etc. Nowadays with the emergence of cloud computing, many cloud services are provided, including the cloud storage which enables people to store and manage their data in remote storage conveniently. Deploying cloud computing platform in PHR system is not only inexpensive but also provides wide-area access and large storage capability for a patient, who can control and share his PHR with other people. However migrating PHR to cloud storage incurs new security problem: how does patient enforce secure access control of his PHR? Although cloud delivers many resources as a service, unfortunately it also gives attackers new possibilities to launch attacks. On the other hand, cloud service provider does not promise the security and confidentiality of PHR. Besides, patient may share his PHR with many people under different access policy. Therefore, we propose a framework of PHR system, which leverages Decentralizing Attribute-Based Encryption (DABE) to encrypt PHR and enforce access control policy. This framework satisfies several security properties such as fine-grained access control, data privacy and scalable access. And we implement a PHR system to display that this system is effective.

誌　　謝

時光荏苒，研究所兩年的生活轉眼就接近尾聲，在研究所求學的期間，一路上受到許多人的幫助，本篇論文能夠順利完成，首先要感謝我的指導老師曾文貴教授，在研究上對我的指導與提點，從老師的身上學到的無論是在研究上或是待人處事上都應認真以對的態度，而在報告上老師也總是提醒我們要能意簡言賅，清楚地表達自己想法讓聽眾得以理解，這兩年的學習令我獲益匪淺。

感謝口試委員蔡錫鈞教授與孫宏民教授能夠撥冗前來，你們給的建議讓我能夠改進我的論文，並思考其他的可能，使我的論文更加完整。

感謝實驗室的學長姊們，總是不吝解答我的疑問，讓我不管是在課業上或是報告上都能不斷改進，感謝你們這兩年來的照顧。感謝實驗室的同學們，研究所的生涯有你們一起互相幫助互相砥礪使我能夠更堅定地走過這一段日子。

最後，感謝始終在背後默默支持我的父母與家人，因為有你們我才有動力走到現在，非常感謝你們的關懷與鼓勵。也感謝所有在這兩年的時光中幫助我的人，你們的幫助都讓我感念在心，謝謝你們。

目　　錄

表　目　錄

圖　目　錄

第一章、緒論

1.1 Personal Health Record (PHR)

在台灣，電子化病歷(Electronic Health Record, EHR)已經行之有年，使用電子化病歷的優點是能夠節省紙張成本、減少錯誤率、有助於病歷的紀錄與流通，進而改善醫療服務的效率，提升醫療品質。但是這些病歷資料皆由各家醫療機構所保存管理，資料無法互通，民眾的病歷資訊分散在各個醫療機構，使得民眾的病歷資訊都是片段的，缺乏完整性，更造成醫療資源上整合與管理上的困難。有鑒於此，在二〇一二年九月，行政院衛生署與微軟(Microsoft)攜手合作，開始推動 Personal Health Record(PHR)與雲端系統結合的計畫，這個計畫是透過雲端平台，整合民眾在健康、醫療、保險等相關機構所持有的個人健康紀錄，提供民眾自主管理個人健康資訊的功能。

PHR 是一種醫療資訊紀錄，來源包含病人紀錄的個人健康資訊如飲食型態、運動習慣、自行量測的血壓資料等，或是門診用藥紀錄、醫療保險、疫苗施打等資料，PHR 涵蓋了民眾一生中所有與個人健康相關的資訊與紀錄，民眾可經由行動裝置登錄雲端平台管理他們的 PHR，隨時隨地依其需求存取 PHR，並與他人分享個人的健康醫療資訊。也就是說，PHR 是一種由民眾個人自主管理的資料，具有高度隱私性，民眾對於自身的 PHR 擁有最高的管理權限，依其意願與需求決定哪些人可以存取他的 PHR。

總而言之，PHR 的推動與普及除了讓民眾個人的健康醫療紀錄趨於完善，亦讓一般民眾能夠更加便利地管理他們的健康紀錄，有助於他們掌控自身的健康狀況，另一方面，醫護人員能獲得更完整的健康資訊作為診斷的依據，避免重複檢驗，提升醫療品質，同時也減少醫療資源的浪費。

1.2 PHR 在雲端儲存系統的存取控制與安全性議題

近年來雲端技術的蓬勃發展，各式各樣的雲端應用服務應運而生，其中雲端儲存系統提供了使用者便利的資料儲存服務，常見的雲端儲存服務有 Dropbox、Amazon S3、Google Drive 等，使用者毋須負擔建置儲存設備的成本，雲端儲存服務提供了大量的儲存空間與彈性的資料存取，如圖 1 所示，使用者可以將以往習慣存在單機電腦中的個人資料夾存放到遠端的設備中，通過網路能夠即時上傳、下載與管理各種型態的資料，使用者不需要煩惱資料同步與攜帶的問題，並且可以透過雲端平台與其他使用者共享資料。

PHR 與雲端系統的結合帶來了幾項好處，存放在雲端平台的 PHR 有利於集中管理、節省儲存空間，雲端化的 PHR 能提供民眾即時的病歷資訊並且不受時間與地點的限制，民眾可隨時調閱 PHR，並且將這些資訊分享給醫護人員、保險公司或是家人朋友等，民眾亦可以根據其需求授權其他使用者去存取他的 PHR，利用雲端平台便利地與其他人共享資料，然而此種做法卻衍生了新的問題：民眾要如何對儲存在雲端的 PHR 達到安全的存取控制？

過去人們將資料儲存在個人的電腦上時，所有的安全責任都歸屬在自己身上，一旦民眾將儲存 PHR 的任務交由雲端儲存服務供應商幫忙處理，那麼 PHR 若是遺失或是遭到竊取，甚至被有心人士濫用誰又該負起責任呢？換句話說，雲端儲存服務供應商並不保證 PHR 是否被安全存取，也不保證 PHR 的隱密性，就連雲端儲存服務供應商本身也不一定是完全可信任的，他們只提供一個儲存空間，民眾的 PHR 未獲得妥善的保護，這些含有個人隱私的資料面臨洩漏出去或遺失的風險。

此外，在與其他使用者共享 PHR 的部分，民眾對其 PHR 理應擁有絕對的控制權，要如何讓民眾能夠針對不同層級的使用者給予相對的權限來存取他的 PHR 亦是一個重要的議題，由於 PHR 是具有高度隱私性的資料，除了被授權者准許存取之外，其他不正當的存取行為都應該被避免，因此，民眾在將 PHR 存放在雲端儲存系統之前，需要經由一套安全且有效率的加密技術來對 PHR 進行加密，用以保護病人的隱私不致洩漏，同時在 PHR 加密的情況下，如何讓民眾能夠對 PHR 達到安全的存取控制也是一個值得深思的問題。

圖 1　雲端儲存系統架構

1.3 研究目標

1.3.1 傳統上的加密機制與其限制

有鑑於上一段所提到的 PHR 在雲端系統上存取的安全性問題，在將 PHR 傳送出去到雲端伺服器之前，勢必得經過一些加密處理，避免隱私資料在雲端上被洩漏出去或是被不正當的存取。在一般的資料存取控制系統中，傳統上最常使用的技巧是以下兩種：

- 對稱式加密系統(Symmetric Key Cryptography, SKC)

  這種加密技巧，較著名的有 AES、DES 等，對稱式加密系統在加解密的過程中都是使用同一把金鑰，故在加解密時非常有效率，但若是想要將此種技巧應用在對 PHR 的存取控制上的話，還需要引入額外的方法才能達成。此種加密法最大的問題在於加解密都是使用同一把金鑰，假使使用者的金鑰遭攻擊者所攔截，攻擊者即可解密所有的 PHR，對於 PHR 保護也隨之瓦解，使用者的隱私也洩漏出去了。

- 公開金鑰加密系統(Public Key Cryptography, PKC)

  這種加密技巧，較著名的有 RSA、ElGamal 等，系統中每個人都有一把公開金鑰與一把私密金鑰，它是利用公鑰來加密資料，擁有對應私鑰的人則可以成功解密，所以它比對稱式加密還要安全，但是要將此種加密法應用在 PHR 的存取控制上得付出龐大的金鑰管理成本，需要有一個 Public Key Infrastructure(PKI)來發送與管理金鑰，面對大量的 PHR，每一份 PHR 對應到不同的使用者都擁有不同的存取規則，對於使用者來說，金鑰管理是一個讓人很頭痛的問題，這種加密法所需的運算量也會非常可觀，因此，要運用在 PHR 存取系統上不但不切實際，而且管理金鑰的成本所費不貲。

上述的兩種傳統加密技巧都不適合運用在雲端系統上資料存取控制上，目前最適合套用在這種存取系統的加密方法是屬性加密機制(Attribute-Based Encryption, ABE)，此種加密方法可以支援高精度的資料存取控制，使用者可以依個人需求替 PHR 制定一些存取規則，符合規則的人就允許去存取使用者的部分 PHR，換言之，PHR 只能夠被某些符合其存取規則的特定使用者存取，其他無法滿足存取規則的使用者即使竊取到了 PHR 也無法窺視其內容，ABE 提供了更富有彈性且更加精緻的存取控制方法。故在雲端化的 PHR 系統架構中，使用 ABE 來實現對 PHR 安全的存取控制是一套可行的方案。

1.3.2 利用 ABE 於雲端化 PHR 存取控制系統

儘管將 PHR 的存放轉移到雲端儲存系統的同時帶來了更便利的資料存取與共享模式，然而這種將資料傳送到遠端伺服器管理的方式使得雲端服務提供者也同樣掌握了使用者的資訊，因而使用者不再能完全控制 PHR 的存取與流向，要如何維護 PHR 的安全性就顯得格外重要。因此，我們主要的研究目標主要是希望在雲端化 PHR 存取控制系統中，藉由 ABE 來達成對 PHR 的安全存取控制。ABE 提供更具彈性的存取規則及安全的加密機制，使用者能透過這套系統輕鬆管理他的 PHR。最重要的是，在我們所提出的系統框架下，讓使用者能夠安全地將 PHR 分享給其他人，無需憂慮自己的隱私會暴露於危險之中，唯有通過使用者授權的人能夠合法存取他的 PHR。

在我們的日常生活中，每個人根據他的職業、工作場所等都有足以彰顯自己身分的屬性(attribute)，譬如在 A 醫院中工作的外科醫生，就被賦予“A 醫院”、“外科”與“醫生”這三種屬性，透過 ABE，使用者可以加密他的 PHR，並將存取規則訂為“A 醫院”AND “外科”AND “醫生”，如此一來，只要是符合這三個條件的醫生就可以存取該使用者的 PHR。由上例可知，對於使用者而言，最大的益處莫過於是不需要知道每個被授權者的身分，這意味著他不必記錄每份 PHR 的使用者存取清單，畢竟隨著使用者與 PHR 的數量遞增，這份使用者存取清單也會愈來愈複雜與龐大，造成使用者管理上的困難與不便，故利用 ABE 來加密 PHR 的話，使用者除了無須對所有 PHR 都記錄存取清單外，PHR 亦只需要加密一次，不僅減輕使用者管理上的負擔與運算成本，還提升了系統的可拓展性。

在 PHR 的系統中，由於資料的存取是由使用者自主管理，隨著使用者授權的人數增加，該使用者管理金鑰的負擔也會愈來愈重，但是如果將金鑰管理的任務全部託付給一個中央的授權中心(central authority, CA)，萬一授權中心被惡意侵入，所有的金鑰就被破解了。因此，在我們的系統架構中，我們著重在多授權中心(multi-authority)的 ABE。將部分金鑰授權的任務交付給其他的授權中心幫忙處理，多授權中心的 ABE 不但分散了金鑰管理的風險，使用者本身也僅需管理部分的金鑰，比起傳統的 SKC 與 PKC，被授權者所需要的金鑰數量也會降低。

而在我們所提出的系統框架下，要達成的目標有以下幾點：

- 資料隱密性(Data confidentiality)：

  由於 PHR 中含有民眾個人敏感性的資料，假如洩漏出去的話將會對民眾造成莫大的傷害與困擾，因此我們首要達成的目標即是對 PHR 隱密性的保護，避免未

經授權的使用者揭露 PHR 的內容，確保 PHR 在傳遞及儲存時的隱密性，除了惡意的使用者外，雲端服務提供者亦無法得知 PHR 的內容，所以在上傳到雲端伺服器之前，PHR 必須先經過加密再傳送。

- 以病人為中心(Patient-centric)：

近年來，人們的個人意識抬頭，對自身隱私的保護也日益重視，而 PHR 的內容皆是人們私密的醫療資料，基於對個人隱私的保護，人們有權決定自身的隱私資料該如何管理與授權，所謂“以病人為中心”的觀念即根植於此，PHR 的服務彙整所有病人私人的醫療資料集中存放在網路上的儲存空間，允許病人透過網路自己產生與管理個人的 PHR，讓病人擁有完全控制自己醫療資料的權利，並且有權決定要釋出哪一部分的 PHR 給其他人觀看，充分尊重病人自身的意願與保護他的隱私。

- 細粒度存取控制(Fine-grained access control)：

使用者在將 PHR 的資訊分享給其他的使用者之際，也許會根據其他使用者的身分來判斷要授權給他哪一部分的 PHR 存取權限，而 ABE 在加密時能夠讓使用者自行替 PHR 設定彈性化的存取規則，針對不同層級的使用者授予不同的存取權限，每一份 PHR 僅有被授予解密金鑰的使用者方可存取，杜絕其他不懷好意的使用者的惡意存取，方便使用者對其 PHR 的管理並且增進了 PHR 使用上的彈性。

- 抵禦同謀攻擊(Resistant to collusion attack)：

同謀攻擊指的是在使用者授權給其他人 PHR 的解密金鑰時，這些被授權者彼此之間或許會互相串謀起來，提供彼此的解密金鑰組合出一把新的金鑰，意圖存取不在自己權限之內的 PHR，ABE 的加密機制則可以抵禦同謀攻擊，使得被授權者無法將金鑰互相結合，阻絕被授權者間的不法存取，維護 PHR 的隱密性。

- 可拓展性（Scalability)：

在 PHR 的系統中，每個使用者可能會與各式各樣的人分享他的 PHR，這些人也許是他的家人或朋友，也有可能是醫療機構或保險公司等，換句話說，與使用者共享 PHR 的人可能為數眾多，而且使用者的 PHR 以常理來說也會隨著時間增加，傳統的 SKC 以及 PKC 由於對於資料皆是一對一的加密，無法應付龐大的 PHR 與使用者共享系統，因此在我們系統中會利用 ABE 來支援這種無法預測使用者數量的 PHR 共享。此外，我們的系統亦支援新的授權中心加入系統，授權金鑰給使用者。

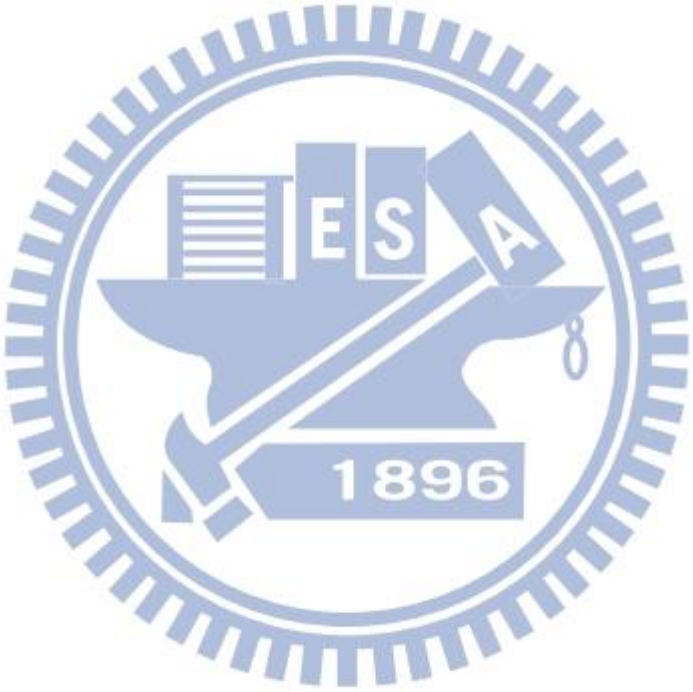
1.4 貢獻

1. 我們提出了一個以 ABE 來做存取控制的 PHR 系統之框架，在此框架之下，是以病人為中心，將 PHR 儲存於雲端系統上，使用者可以自主管理他的 PHR。利用多屬性授權中心的 ABE，讓使用者可以安全地與他人分享 PHR。由於系統中金鑰的管理是分散在各個授權中心，因此我們可以減少其管理金鑰的負擔。此外，透過 ABE 的加密機制保證使用者 PHR 的安全問題，而我們的系統中亦無需一個中央的授權中心，解決金鑰集中的問題。
2. 在我們的系統中，透過多屬性授權中心的 ABE，讓每個授權中心分別控制互斥的屬性集合，這些屬性是代表人們的身分或是職業，而我們也讓使用者成為一個授權中心，控制與 PHR 種類相關的屬性集合，要注意的是，我們將大部分的金鑰管理分散給其他授權中心負責，使用者則負責一小部分的金鑰管理，讓使用者可以根據其需求即時授權給其他使用者。在加密 PHR 時，使用者可決定那些身分的人有資格存取他的 PHR，並選擇 PHR 的種類做加密，因此在我們的系統框架中，提供了更富有彈性的存取規則，讓使用者管理調度上能夠更符合現實上的需求，進而達到對 PHR 細粒度存取控制。
3. 我們也實作了一個 PHR 的系統，在此系統之下，使用者可以經由我們的介面來上傳並加密他的 PHR，提供使用者一個方便及友善的操作介面來管理他的 PHR，而我們在效能分析的部分也會展示我們的系統是可行的。

1.5 全文架構

本論文其餘的章節部分整理如下：在第二章的部分會介紹目前利用 ABE 來做加密的 PHR 系統，並說明在我們的系統框架中欲改善的部分，第三章則是本系統會用到的相關數學背景，像是 Bilinear pairing 與 ABE 的存取結構。第四章由最基本的 ABE 開始，介紹各種不同形式的 ABE，像是 Ciphertext Policy Attribute-Based Encryption（CP-ABE），Key Policy Attribute-Based Encryption（KP-ABE）等以單一授權中心為基礎的 ABE，以及本系統會使用到的 Decentralizing Attribute-Based Encryption（DABE），第五章則進入我們所提出的系統框架，詳細介紹我們的 PHR 系統中的建構方法，各種情境與流程，此外，並分析 PHR 系統的安全性。第六章是系統實作的部分，我們藉由一些實驗來檢測

我們的系統在加解密或是產生金鑰時所需的時間，並且展示實作的系統介面與功能。最後，第七章為全篇總結與未來展望。



第二章、相關研究

電子病歷的推動與普及是目前醫療界的趨勢，以電子病歷取代紙本病歷的好處除了減少錯誤率與節省資源，也提高了病歷的可用性，促進醫療品質。以醫事人員或醫療研究者的角度而言，病歷電子化使得他們可以便利地的取得病人的醫療資訊，進而紀錄與分析，增進看診與研究的效率；對病人來說更是減少了重複檢驗的麻煩，不必每次到新的醫院就得重新填寫病歷，電子病歷亦更便於攜帶與傳遞。

以病人為中心的概念是始於 Szolovits et al. [18]，他們提出一套個人醫療系統，整合病患個人所有醫療相關資訊，其後一個以病人為中心的健康管理系統 Indivo 被提出來 [19]，使用者可以在這套系統上管理與保存他的健康資訊，Indivo 提供一套安全機制讓使用者能夠對其健康資料做加密並制定存取規則，唯有取得使用者私密金鑰的人可以解密。

PHR 的服務允許使用者透過網路集中管理他的 PHR，像是微軟的 Health Vault [21] 及 Google Health [20] 即是將 PHR 系統建置於雲端環境之上，提供使用者自主管理 PHR 的服務，但隨之而來的問題是，使用者該如何安心的將 PHR 儲存在第三方並做到對資料的完全控制與分享？使用者不僅在乎 PHR 使用上是否便利，更關心他們的 PHR 之安全與隱密性。舉例來說，假若使用者患有某些特殊疾病(如：愛滋病、痲瘋病等)，而他的 PHR 中隱含有這項資訊卻被洩漏出去的話，可能會導致使用者遭到社會上的歧視與排擠，因此，維護使用者 PHR 的隱私絕對是首要注意的議題。

為了解決醫療資料存取的安全性問題，近幾年來有許多學者提出了解決的方法，Lee et al. [16] 利用對稱式加密系統與電子簽章的技巧來管理金鑰，他們的方法是運用智慧卡（smart card）來儲存使用者的私密金鑰，當醫生需要病人的醫療資訊，必須徵求病人同意之後，病人再以私密金鑰解密，這種技術雖然很安全，智慧卡也便於攜帶，但是為了滿足緊急狀況，這個方法將所有的私密金鑰一併存放在 TA (Trusted Authority)，以便緊急狀況時醫生能夠去擷取病人的資料，此舉無疑造成了金鑰集中的安全性問題，TA 隨時隨地可以存取病人的資料。而此系統需要智慧卡的技術來達成，每一個病人皆須配有一張智慧卡，在解密的時候病人必須在場提供智慧卡，醫生才能觀看病人的病歷，對於醫生來說是很不方便的，病歷也無法與其他人共享。

與傳統公開加密系統相較之下，使用 ABE 於 PHR 這種多人存取的系統是一套可行的解決方案。Hupperich et al. [17] 提出了一套由病人自我控制的 EHR 系統，使用 ABE

來做加密，這套系統需要智慧卡來儲存 transaction code（TAC），如圖 2 所示，加密 EHR 時，病人利用其智慧卡通過 TAC 服務認證取得 TAC，再將 TAC 及 ID 傳送給醫生 1，醫生 1 產生 EHR 後利用這兩個屬性將 EHR 加密，若醫生 2 想存取病人的 EHR，則需病人給予 TAC，醫生 2 再依此 TAC 與病人 ID 向 PKG 要求私密金鑰，並以私密金鑰解密病人的 EHR。但是此系統依賴一個 TAC 服務來產生金鑰，同樣存在金鑰集中的問題，另一個問題是，EHR 的加密是由醫生執行，病人無法確保醫生在加密時是否會新增存取規則，使得醫生本身能夠自由存取該 EHR，如此一來病人的隱私便無從保證。

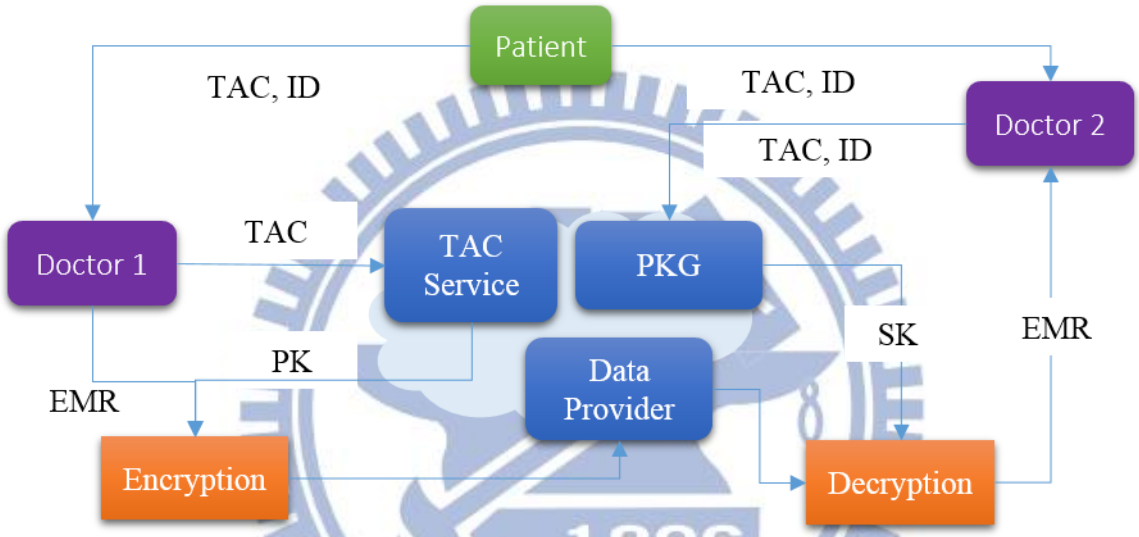


圖 2　Patient-Controlled System

[10][11][12][13][14][15] 皆是以 CP-ABE 為基礎來做資料的存取，儘管在這幾個系統中，不像[17] 是將加密交由醫生來執行，而是病人自己制定存取規則並加密，但同樣系統中只有一個屬性授權中心，如圖 3 所示，此授權中心管理所有的屬性，產生公開金鑰，之後病人再利用這些公開金鑰制定存取規則並上傳到雲端伺服器，而醫生須事先向授權中心要求私密金鑰方可存取病人之 PHR，[10][11][12][13][14][17] 同樣都將金鑰集中在一個授權中心之下，將金鑰管理的責任全部交託給一個授權中心管理，無疑讓授權中心掌握過大的權限，使得他可以任意去存取病人的 PHR，另外一個問題是，負責產生與管理所有的金鑰對於授權中心而言負載太重。

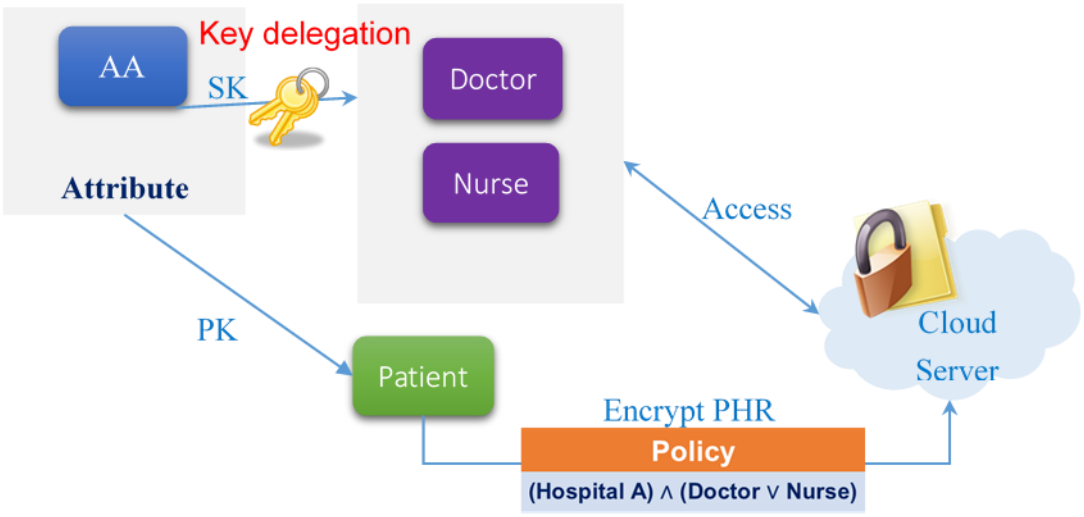


圖 3　使用 CP-ABE 的 PHR 系統架構

[9]的系統則是一個多層的架構，首先利用 KP-ABE，病人將 EHR 以種類作為屬性來加密，利用 IBE 來傳送 KP-ABE 的金鑰給醫院，保證金鑰傳輸的安全，在醫院之下的人如醫生、護士等，依其身分使用 CP-ABE 產生私密金鑰，由醫院決定那些身分的人可以存取 EHR，並定義存取規則使用 CP-ABE 將 KP-ABE 的金鑰加密，簡言之，此系統裡用 KP-ABE 來分類 EHR，再使用 CP-ABE 來篩選可以存取 EHR 的使用者。如圖 4 所示。

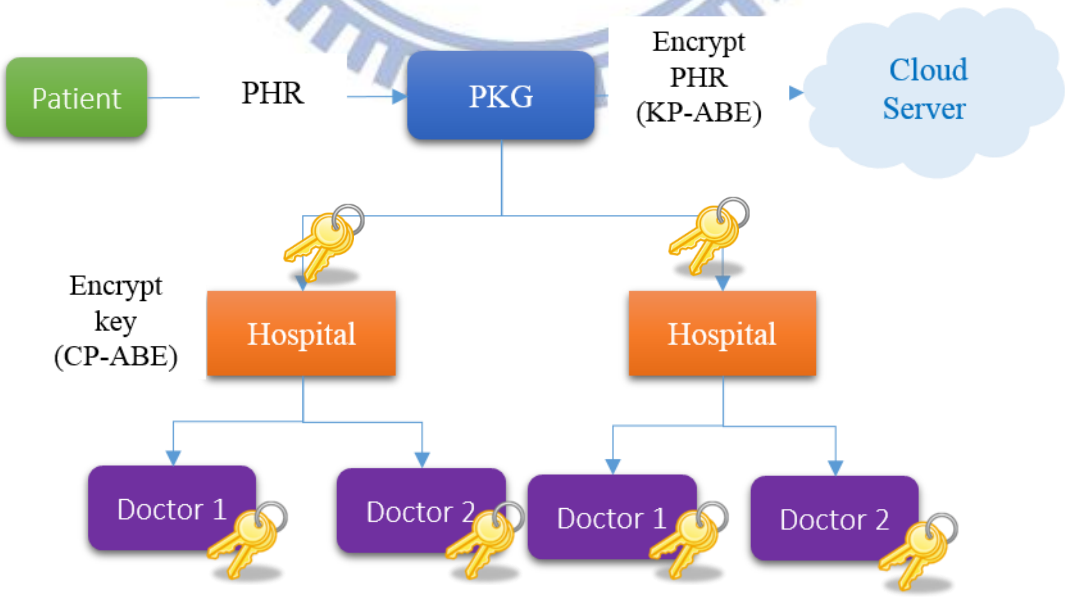


圖 4　Hierarchical 的 PHR 系統

為了解決金鑰集中的問題，[8]首先提出多屬性授權中心的 PHR 存取控制系統，作者將被授權者分成公開領域與私人領域兩部分，對於公開領域的人是以 MA-ABE 來做存取控制；私人領域則是以 KP-ABE 來做存取控制，參考圖 5。換句話說，在公開領域之中，有多個屬性授權中心分別管理一些屬性，這些屬性與身分或職業相關，被授權者透過請求從授權中心取得私密金鑰，病人利用屬性中心的公開金鑰來加密 PHR，制定存取規則決定誰可以解密他的 PHR，然而，在他們的系統中，授權中心的數量是固定的，新的授權中心出現整個系統必須重新設定，使用者所能制定的規則也必須是 CNF，並且加密時一定要從每一個屬性授權中心至少選擇一個屬性來做加密，使得規則沒有彈性，被授權者亦必須從所有的授權中心取得金鑰才能滿足存取規則。另外，私人領域的部分則是運用 KP-ABE，將 PHR 根據其分類的屬性加密，將存取規則定義在被授權者的私密金鑰上。

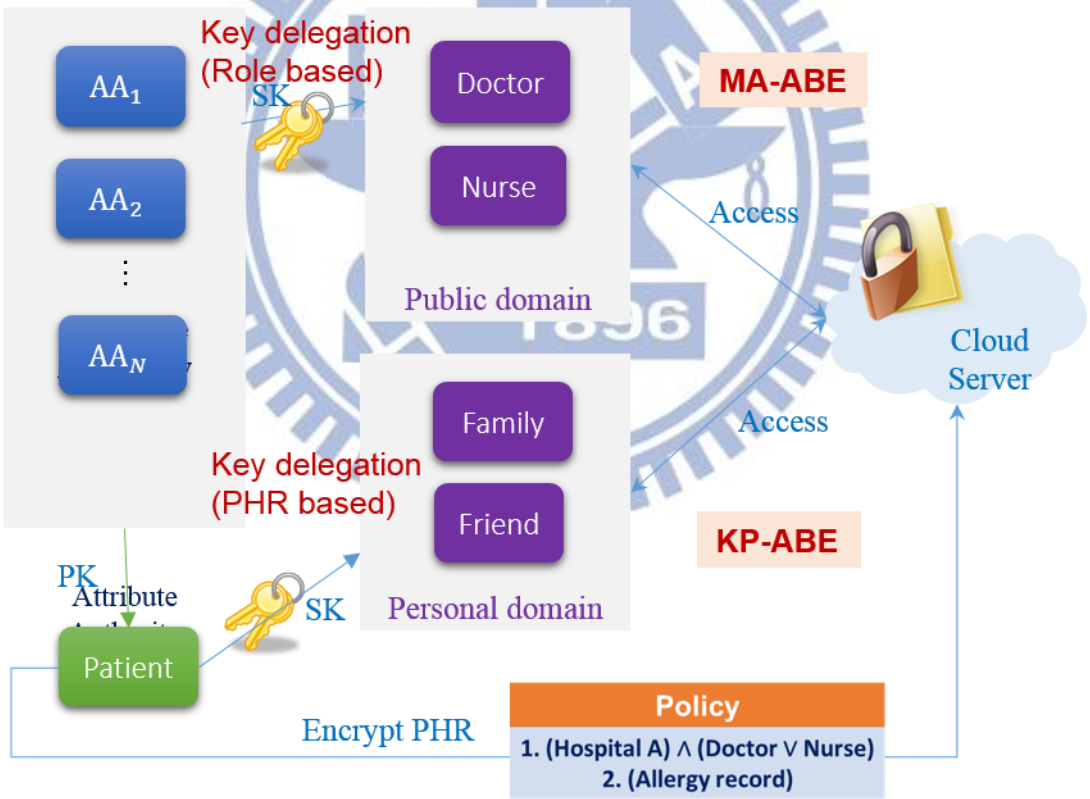


圖 5　使用 MA-ABE 的 PHR 系統

第三章、數學背景

3.1 Bilinear Pairing

令G_1為一個秩(order)為質數 P 的加法循環群，G_2為一個秩同樣為 P 的乘法循環群，g 是在G_1底下的生成元(generator)，則存在一線性映射函數

$$e: G_1 \times G_1 \rightarrow G_2$$

此映射函數符合以下的條件：

- 雙線性(Bilinear)：

  $\forall P, Q \in G_1, \forall a, b \in Z_p^*$， $e(aP, bQ) = e(P, Q)^{ab}$

- 非退化性(Non-Degenerate)：

  $\exists P, Q \in G_1, e(P, Q) \neq 1$

- 可計算性(Computability)：

  $\forall P, Q \in G_1, e(P, Q)$可以有效率地被運算出來。

3.2 存取結構(Access Structure)

Definition 3.2.1 (Access Structure)

令$\{P_1, \ldots, P_n\}$為一個 party 的集合，假如對 $\forall B, C$：

$$if\ B \in \mathbb{A}\ and\ B \subseteq C\ then\ C \in \mathbb{A}\ 成立$$

則一個存取結構 $\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}}$是單調性(monotone)的。而一個單調性的存取結構是在$\{P_1, \ldots, P_n\}$下的一個非空子集合的 collection $\mathbb{A}$，也就是$\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \backslash \{\}$。在 $\mathbb{A}$ 中的集合則稱作被授權的集合(authorized sets)，反之，不在 $\mathbb{A}$ 中的集合被稱作是非授權的集合(unauthorized sets)。

3.3 Linear Secret-Sharing Scheme

Definition 3.3.1 (Linear Secret-Sharing Schemes, LSSS)

在一組 party 的集合 $\mathcal{P}$ 之下的一個 secret sharing scheme Π 如果是線性的，會滿足以下條件：

1. 每個 party 的 share 集合起來會形成一個在 Z_p 底下的向量(vector)。

2. 存在一個矩陣 A，A 稱作是Π的 share-generating matrix。矩陣 A 有 l 個列與 n 個行。對於所有$x = 1, \dots, l$，第 x 個在 A 中的列會被 party $\rho(x)$所標示。($\rho: \{1, \dots, l\} \to \mathcal{P}$)若有一組行向量 $v = (s, r_2, \dots, r_n)$，$s \in Z_p$為 secret，而 $r_2, \dots, r_n \in Z_p$為隨機選擇的變數，則 Av 會產生一組有 l 個 share 的向量對應到 Π，。每一份 share $(Av)_x$ 屬於 party $\rho(x)$。參考圖 6。

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{l1} & \cdots & A_{ln} \end{bmatrix}_{l*n} \begin{matrix} \leftarrow \rho(1) \\ \\ \leftarrow \rho(l) \end{matrix}$$

$$Av = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_l \end{bmatrix}$$

圖 6　　generating matrix

Linear reconstruction property：

假設 Π 是一個對應到存取結構 $\mathbb{A}$ 的 LSSS。令 S 為 authorized set，並且定義 $I \subseteq \{1, \dots, l\}$，$I = \{x | \rho(x) \in S\}$。向量 $(1,0, \dots,0)$ 可以經由 A 的列做線性組合所生成出來，定義所有生成這組向量的列的 index 之集合為 I。由於 Π 是 LSSS，因此存在一組常數集合 $\{\omega_x \in Z_p\}_{x \in I}$，任何對應到 Π 的合法 share 之集合 $\{\lambda\}_x$，我們可以找到一組子集合使得：$\sum_{x \in I} \omega_x \lambda_x = s$，而且這些常數集合 $\{\omega_x\}$ 可以在多項式的時間內被運算出來，找出這些常數 $\{\omega_x\}$ 所需的時間與 sharing-generating matrix A 的大小有關。

Example 3.3.1 [Shamir t-out-of-n threshold scheme]

假設有一組 party $\{P_1, \dots, P_n\}$，$1 \le t \le n$，$t, n \in N$。定義存取結構為 $A = \{S \subseteq \{P_1, \dots, P_n\} | |S| \ge t\}$，dealer 以 $s \in F_p$ 作為 secret，接者 dealer 任意選擇 t-1 個在 F_p 之下的數，$r_1, \dots, r_{t-1}$作為多項式的係數，利用 secret s 以及這些係數定義一個 degree 為 t-1 的多項式：

$$p(x) = r_{t-1}x^{t-1} + r_{t-2}x^{t-2} + \cdots + r_1 x + s$$

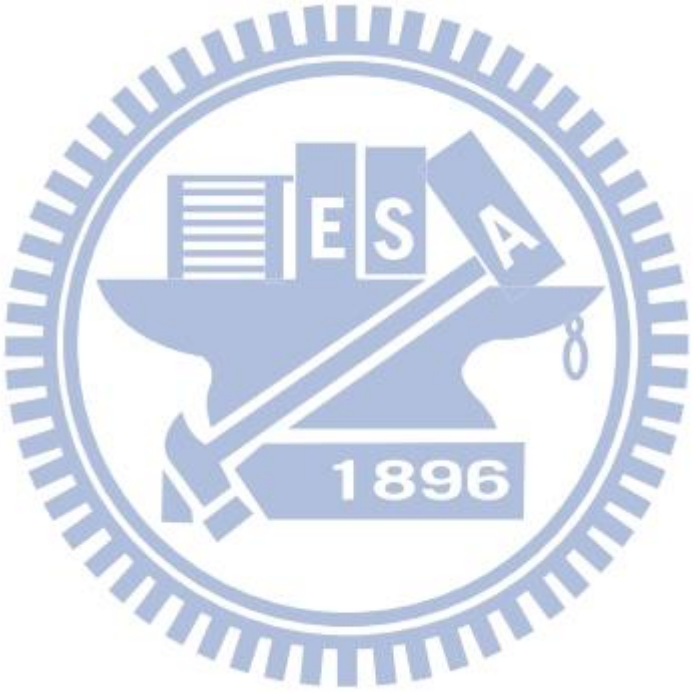
設定 $p(0) = s$。Dealer 將 share p(i) 分給 party P_i（$i \in F_p$），只要有至少 t 個 share 就可以重建回此多項式 p(x)。意即是任意 authorized set S 就可以利用 Lagrange interpolation 重新還原 secret s。

Theorem 3.3.1 [Lagrange interpolation]

給定 t 個相異的點$\{x_i, y_i\}$，$1 \leq i \leq t$，這些點由方程式 $f(x) = y$ 所產生，$f(x)$ 的 degree 不超過 t 。則此 $f(x)$ 可以表示成：

$$f(x) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \leq j \leq t \\ i \neq j}} \frac{x - x_j}{x_i - x_j}$$

令$S = \{x_1, \ldots, x_t\}$，定義 Lagrange coefficient $\Delta_{x_i,S} = \prod_{x_j \in S, x_j \neq x_i} \frac{x - x_j}{x_i - x_j}$ ，我們可以利用 Lagrange interpolation，代入 $x = 0$ ，即可算出 $f(0) = s$ 。



第四章、Attribute-Based Encryption

4.1 Attribute-Based Encryption (ABE)

ABE [2]是 Sahai 與 Waters 在一九九五年所提出，ABE 最早的原型可以追朔到 Shamir 在一九八五年的時候提出的 Identity-Based Encryption（IBE）[1]，在 IBE 中，使用者能夠以他由一個可信的 Private Key Generator（PKG）在驗證使用者身分後產生。然而，無論是 IBE 或是傳統的公開加密系統，它們的加密都是一對一的加密，換言之，每一份加密的資料只對應到一把解密的金鑰，假若想要把資料分享給多個使用者，同一份資料就必須利用各個使用者的公開金鑰進行多次的加密，使得金鑰管理不易且沒有彈性，因此，IBE 跟公開加密系統都不適用於一對多的使用者模式。

而 ABE 的出現解決了上述的問題，Sahai 和 Water 提出了 Fuzzy Identity-Based Encryption，也就是廣為人知的 ABE。使用者在加密資料時所定義的存取規則，並非考慮接收者的身分，而是由接收者手上所握有的屬性（attribute）來決定該接收者是否符合他的存取規則，符合的話接收者則得以解密資料。ABE 可視作 IBE 的延伸，只是將原本用來識別使用者身分轉變成一個屬性的集合。也就是說，ABE 與傳統上的加密系統最大的差別在於它是一種一對多的加密，正是基於這樣的特性，使得 ABE 可以對資料達到細粒度存取控制，與傳統公開加密系統相較之下更富有彈性，同時也降低使用者管理上的複雜度。

在這之後，陸續又有學者提出了 Ciphertext-Policy Attribute Based Encryption（CP-ABE）與 Key-Policy Attribute-Based Encryption（KP-ABE），這是 ABE 最主要的兩種類別，此兩種的差異在於存取規則是嵌在金鑰抑或是密文之中。另外，CP-ABE 與 KP-ABE 皆是單授權中心的屬性加密系統（single authority），在近來的研究中，也有許多人探討多授權中心的屬性加密系統（multi-authority），以下小節會一一詳細介紹這些 ABE 加密演算法。

4.2 Key Policy Attribute-Based Encryption（KP-ABE）

KP-ABE 最早是由 Goyql et al. [3] 所提出，在這個加密系統中，是將存取規則與私密金鑰（Private key）做連結，密文則與一組屬性作連結。例如：假設有一份健康資料 A 以｛“2012”，“心臟病”，“血液檢查”｝這組屬性加密，另一份健康資料 B 以

{“2012”，“高血壓”，“血液檢查”} 這組屬性來做加密，如果 Alice 手中的私密金鑰中所連結的存取規則是“2012”AND “心臟病”，則 Alice 可以存取所有在 2012 年與心臟病相關的健康資料，但無法存取資料 B。而若另一個人 Bob 的私密金鑰所連結的存取結構是 2012”AND “血液檢查”，則 Bob 可以存取資料 A 與 B。

定義 access tree

在 KP-ABE 中，是將一個存取規則以 access tree 表示，每一個非葉的子節點是一個 k-out-of-n 的 threshold gate，n 是該節點的子節點數，k 是 threshold value，若 k=1，n=2 時，此節點是一個 OR gate，k=2，n=2 時，此節點是一個 AND gate。每一個葉節點分別代表一種屬性，以下定義三種函式：

1. $parent(x)$ 回傳節點 x 的父節點。
2. *att(x)* 回傳該葉節點所連結的屬性。
3. *index(x)* c 回傳節點 x 的 index。

圖 7 是一個 access tree 的範例，在此例中的存取規則是（Attr1 And Attr2) OR (2 of (Attr3,Attr4,Attr5))，

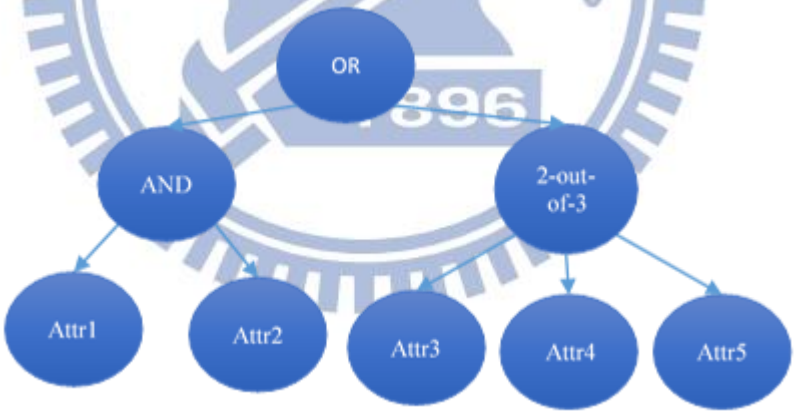


圖 7　　KP-ABE 中的 access tree

KP-ABE 是建立在 symmetric bilinear pairing，e: $G_1 \times G_1 \rightarrow G_T$，$G_1$的 order 是質數 p ，g 為G_1 的一個生成子。

以下是 KP-ABE 的四個演算法：

Setup

定義一個 attribute universe $\mathcal{U} = \{1,2,\dots,\mathrm{n}\}$，替每一個在 $\mathcal{U}$ 中的屬性 i 選擇一個值 t_i，$t_i \in Z_p^*$，並且選擇一個 $\mathrm{y} \in Z_p$，則公開金鑰為：

$$\mathrm{PK} = (T_1 = g^{t_1}, \dots, T_n = g^{t_n}, \mathrm{Y} = e(g,g)^y)$$

主金鑰(Master key)為：

$$\mathrm{MK} = (t_1, \dots, t_n, \mathrm{y})$$

Encryption

選擇一個 secret s，$\mathrm{s} \in Z_p$，將一組屬性的集合 γ 與信息 M 做加密，密文為：

$$\mathrm{E} = (\gamma, E' = MY^s, \{E_i = T_i^s\}_{i\in\gamma})$$

Key generation

此演算法會回傳一個私密金鑰，該金鑰與存取結構 A 做連結。

首先會先將存取規則轉換成 access tree，對每一個在 access tree 中的非葉節點 x，選定一個多項式 p_x，而此多項式的 degree 為 $t_x - 1$，對根節點 R 設定 $p_R(0) = y$，其他的節點 x 設定為 $p_x(0) = p_{parent(x)}(index(x))$，再選擇 $t_x - 1$ 個任意常數來完成這個多項式，最後，令 S 為 access tree 中葉節點之集合，對每一個屬性所連結的葉節點，產生私密金鑰：

$$D_x = g^{\frac{p_x(0)}{t_{att(x)}}} \quad for\ x \in S$$

Decryption

首先定義一個遞迴演算法 DecryptNode(CT, SK, x)，對每一個在 access tree 中的葉節點 x，

$$\mathrm{DecryptNode(CT, SK, x)} = \begin{cases} e(D_x, E_i) = e\left(g^{\frac{p_x(0)}{t_i}}, g^{st_i}\right) = e(g,g)^{sp_x(0)} & for\ i = att(x) \in \gamma \\ \perp \quad otherwise \end{cases}$$

由於 $p_x(0) = p_{parent(x)}(index(x))$，對每個節點 x，利用內插法得到 $\mathrm{e(g,g)}^{\mathrm{sp}_x(0)}$，一直重複次演算法直至遇到根節點，最後可得對根節點 R，

$$\mathrm{e}(D_R, E_i) = \mathrm{e}\left(g^{\frac{p_R(0)}{t_i}}, g^{st_i}\right) = e(g,g)^{sp_R(0)} = e(g,g)^{sy} = Y^s$$

即可得明文：

$$\mathrm{M} = \frac{E'}{Y^s}$$

4.3 Ciphertext Policy Attribute-Based Encryption (CP-ABE)

CP-ABE 是由 Bethencourt et al [4]等人所提出，與 KP-ABE 相反，CP-ABE 是將密文與存取規則連結，私密金鑰則與一組屬性作連結。例如：有一份健康資料加密時，制定存取規則為“A 醫院”AND(“醫生”OR“護士”)，表示 A 醫院的醫生或是護士可以根據他們所持有的屬性來存取這份健康資料，其他像是 B 醫院或是 C 醫院則由於不符合存取規則而無法存取該資料。

CP-ABE 是建立在 symmetric bilinear priring，e: $G_1 \times G_1 \to G_T$，G_1的 order 是質數 p ，g 為G_1 的一個生成子，並定義一個雜湊函數 H：$\{0,1\}^* \to G_1$。

以下是 CP-ABE 的四個演算法：

Setup

選擇兩個變數 $\alpha, \beta \in Z_p$ ，則公開金鑰為：

$$\mathrm{PK} = (G_1, \mathrm{g}, \mathrm{h} = g^{\beta}, \mathrm{f} = g^{\frac{1}{\beta}}, e(g,g)^{\alpha})$$

主金鑰(Master key)為：

$$\mathrm{MK} = (\beta, g^{\alpha})$$

Encryption

此演算法會將信息 M 加密於存取結構 A 之下，首先會先將存取規則轉換成 access tree，由根節點 R 開始，選定一個多項式 p_R ，而此多項式的 degree 為 $t_R - 1$，選擇一個 secret s，$s \in Z_p$，設定 $p_R(0) = s$，再選擇 $t_R - 1$ 個任意常數來完成這個多項式。其他的節點 x 設定為 $p_x(0) = p_{parent(x)}(index(x))$，同樣地，再選擇 $t_x - 1$ 個任意常數來完成這個多項式 p_x。令 X 為 access tree 中所有葉節點的集合，則密文為：

$$\mathrm{CT} = (\mathrm{A}, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s$$

$$\forall \mathrm{x} \in \mathrm{X}: \; C_x = g^{p_x(0)}, C_x' = H(att(x))^{p_x(0)})$$

Key generation

此演算法會輸入一組屬性集合 S ，最後輸出一組金鑰對應到此集合 S 。首先任意選取一亂數 $\mathrm{r} \in Z_p$，針對每個屬性 $\mathrm{j} \in \mathrm{S}$ ，任意選取一個亂數 $r_j \in Z_p$ ，最後運算出私密金鑰為：

$$SK = (D = g^{\frac{\alpha + r}{\beta}}$$

$$\forall \mathrm{j} \in \mathrm{S}: \; D_j = g^r H(j)^{r_j}, D_j' = g^{r_j})$$

Decryption

對於每個非葉節點 x ，令 i = attr(x)，若 i ∈ S ，則：

$$\text{DecryptNode}(\text{CT}, \text{SK}, \text{x}) = \frac{e(D_i, C_x)}{e(D_i', C_x')}$$

$$= \frac{e\left(g^r H(i)^{r_i}, h^{p_x(0)}\right)}{e(g^{r_i}, H(i)^{p_x(0)})}$$

$$= e(g, g)^{rp_x(0)}$$

若 i ∉ S ：

$$\text{DecryptNode}(\text{CT}, \text{SK}, \text{x}) = \perp$$

若 x 為非葉節點，考慮遞迴的情況：

令 z 為 x 的子節點，呼叫DecryptNode(CT, SK, z)，輸出 F_z ，令 S_x 為大小為 t_x 的任意集合，收集這些子節點 z，使得 $F_z \neq \perp$ ，如果 S_x 不存在，則 DecryptNode(CT, SK, x) = ⊥。

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S_x'}(0)}, where\ i = index(z), S_x' = \{index(z) : z \in S_x\}$$

$$= \prod_{z \in S_x} \left(e(g, g)^{rp_z(0)}\right)^{\Delta_{i,S_x'}(0)}$$

$$= \prod_{z \in S_x} \left(e(g, g)^{rp_{parent(z)}(index(z))}\right)^{\Delta_{i,S_x'}(0)}$$

$$= \prod_{z \in S_x} (e(g, g)^{rp_x(i)})^{\Delta_{i,S_x'}(0)}$$

$$= e(g, g)^{rp_x(0)}$$

將根節點 R 入此遞迴演算法，可得 $\text{DecryptNode}(\text{CT}, \text{SK}, \text{R}) = e(g, g)^{rp_R(0)} = e(g, g)^{rs}$，最後可算出明文：

$$\text{M} = \frac{\tilde{C}}{e(C, D) \Big/ e(g, g)^{rs}}$$

$$= \frac{\tilde{C}}{e(h^s, g^{\frac{\alpha+\beta}{\gamma}}) \Big/ e(g, g)^{rs}}$$

4.4 Multi-Authority Attribute-Based Encryption (MA-ABE)

[2][3][4] 的 ABE 皆是建立在單屬性授權中心(attribute authority)之下，意即是系統中只有一個授權中心負責金鑰的產生與授權。然而，在現實生活中，存在許多的授權中心，這些授權中心各別掌管一些屬性，譬如由圖 8 所示，在一個醫療的系統中，一個在 A 醫院服務的內科醫生，經醫療機構認證後取得“physician”這個屬性，而“internal medicine”這個屬性也許是從醫療學會取得，“Hospital A”則是服務的醫院所給予，這些授權中心各自獨立運作，每個人可以個別到這些授權中心經由驗證取得所需的屬性。

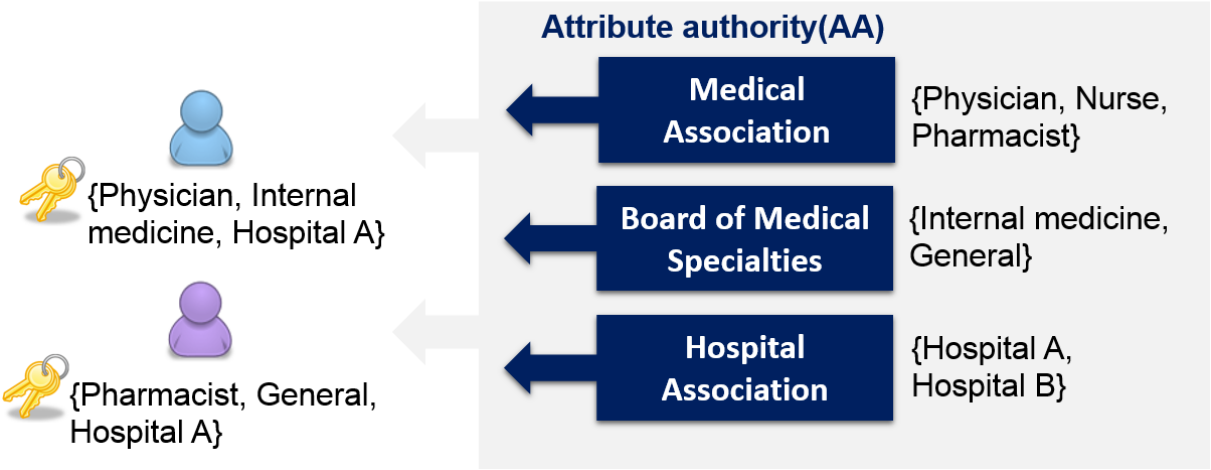


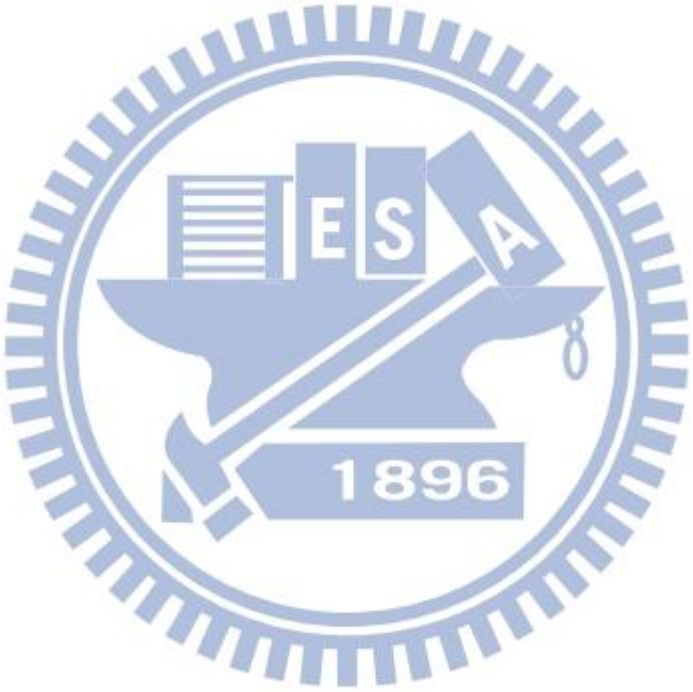
圖 8　多屬性授權中心

Chase [5] 於二〇〇七年發表了一篇 KP-ABE 的多授權中心加密系統，其中，每一個使用者擁有一個獨特的識別碼(GID)，利用此識別碼，使用者可將從不同屬性授權中心取得的金鑰連結在一起，這個系統需要一個中央的授權中心(central authority)來控制主金鑰，並且透過他來將從其他授權中心取得的金鑰整合在一起之後，產生一把新的金鑰再授權給使用者，但是，這樣導致中央授權中心控制了所有金鑰，使得他有權去解密所有的資料，反倒變成了安全上容易被攻擊的目標。而 Chase 提出的機制有一項限制是在存取規則上只支援 Conjunctive normal form(CNF)，系統中的授權中心數量也是在一開始系統設定時就必須決定好，無法彈性增減。

之後，Chase et al. [6] 提出了改進的方法，利用分散式虛擬亂數函數(distributed pseudo-random function)的技巧，移除了中央授權中心，儘管如此，這個方法除了在存取

規則上同樣只支援 CNF 之外，授權中心的數量同樣是在系統設定時就被固定，無法讓新的授權中心動態加入系統。

二〇〇九年，Lewko et al. [7] 提出了 DABE，此為 CP-ABE 的多屬性授權中心加密機制，在此機制之下，每個使用者都有一個識別碼，同樣是用來連結來自不同授權中心的金鑰。不過這個系統不需要一個中央的授權中心，故不會遭遇因金鑰集中所造成的安全性問題。除此之外，在系統初始化之後，任何的使用者依然可以隨時加入系統，成為新的授權中心，在存取規則上更是支援了所有的 Boolean formula，使用者可以選擇某些授權中心所管理的屬性任意組合來制定存取規則，在制定存取規則上極富彈性。因此，我們選擇 Lewko 所提出的加密機制來實現我們的 PHR 系統，而詳細的演算法會在第五章介紹。



第五章、利用 ABE 達成存取控制之 PHR 系統

在這個章節，介紹我們所提出的，以病人為中心的 PHR 系統框架，此 PHR 系統是建立於雲端環境，我們將闡述如何利用 ABE 來實現對 PHR 於雲端架構之下的安全共享。

5.1 問題闡述

在 PHR 系統中，存在數以千計的 PHR 擁有者與使用者，在此我們稱系統中 PHR 的擁有者為病人，而存取 PHR 的人稱為被授權者或使用者，亦即是需經由病人授權許可，被授權者始能存取該病人之 PHR，每一個病人對其 PHR 有絕對的控制權，他可以產生、管理與刪除他的 PHR。所有病人的 PHR 皆儲存在雲端伺服器上，被授權者可能來自各行各業，也許是醫事人員、醫療研究者或是病人的家人、朋友等。這些被授權者透過網路連結雲端伺服器取得病人的 PHR，被授權者可以依其權限存取來自不同病人之 PHR，同樣一份 PHR，每位被授權者的存取權限有所差別，與病人所定義的存取規則息息相關。

5.2 系統架構

在我們的系統架構中，主要有四個角色：屬性授權中心(Attribute Authority, AA)、病人、被授權者(也就是 PHR 系統中的使用者)與雲端伺服器。

如圖 9 所示，屬性授權中心各別掌管一個屬性集合，這些屬性可以用來代表使用者的身分與職業，授權中心並替其所控制的屬性產生公開金鑰與私密金鑰，公開金鑰的部分，授權中心會將其公告在網頁上；私密金鑰的部分是使用者提出要求，經授權中心驗證使用者身分之後才授權給該使用者。而在我們的系統中，病人本身也是一個授權中心，其所掌管的屬性是以 PHR 的分類為主，假若病人想分享某部分的 PHR 給某一使用者，可以即時授權一把私密金鑰給該使用者。被授權者則是通過屬性中心取得符合其身分的私密金鑰，再從雲端伺服器下載他想存取之 PHR，依其取得之私密金鑰解密該份 PHR。病人從授權中心所產生的公開金鑰中選擇一部分金鑰來加密他的 PHR，之後再將 PHR 上傳到雲端伺服器儲存。雲端伺服器則是負責接受使用者上傳與下載的請求，並提供空間儲存 PHR。

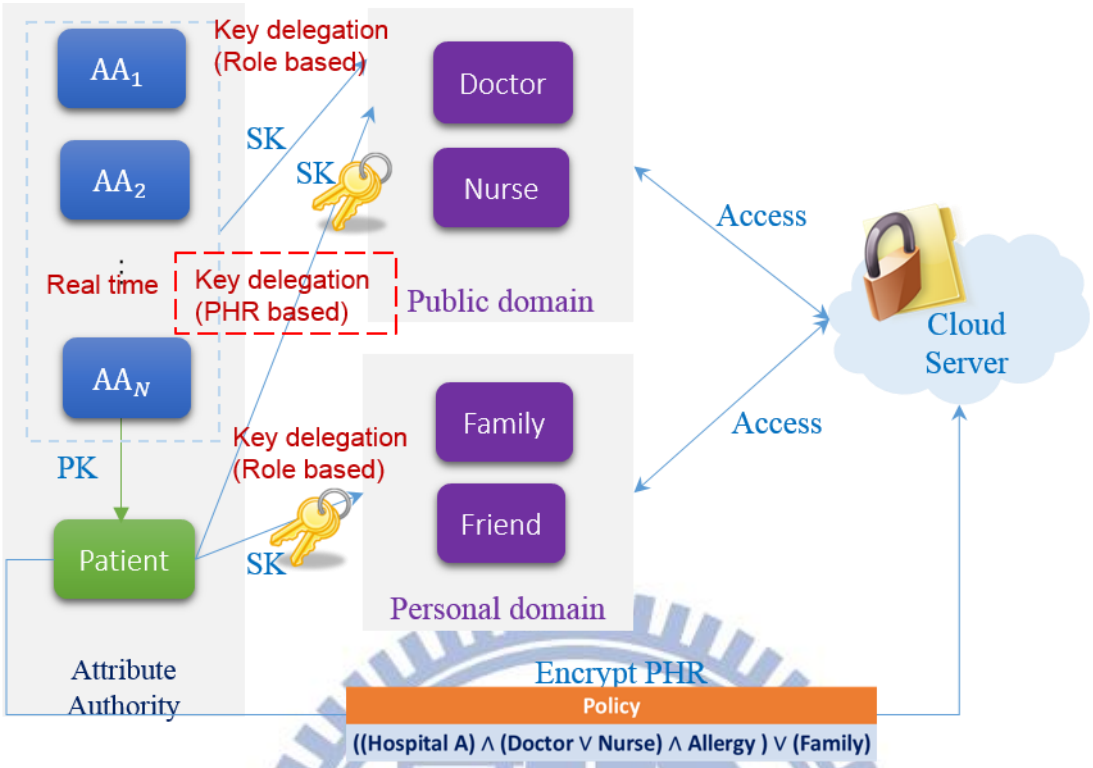


圖 9　系統架構

5.3　安全模型

在我們的系統中，假設雲端伺服器是誠實但好奇的(honest but curious)，他們提供使用者儲存的服務，讓使用者能夠正常上傳與下載檔案，遵循我們的演算法運作，但是他們會窺探使用者的檔案內容，意圖從中獲取使用者的私密資訊。而各個屬性加密中心我們則假設都是完全可信任的，這些中心各自管理一些屬性並產生公開金鑰供使用者加密時使用，使用者們可以向這些授權中心提出要求，通過驗證之後取得含有屬性的私密金鑰。最後，我們假定使用者會試圖去存取不在其權限內的檔案，換言之，若是其私密金鑰無法滿足解密檔案所需的存取規則的話，使用者們或許會串謀起來，將各自擁有的私密金鑰結合起來用以滿足更多檔案的存取規則，藉此存取權限外的檔案。

5.4　系統框架

我們提出此框架的目的在於提供使用者一個安全存取 PHR 的結構與減輕其管理金要的負擔，主要的想法是將系統中大部分的金鑰管理交由屬性授權中心負責，被授權者被劃分成兩部分：公開領域(public domain)與私人領域(personal domain)，以病人視角來

說，公開領域使用者的身分為醫生、護士、醫療人員等具有專業身分之人物；而在私人領域的使用者則是病人熟識或是信任的人，譬如家人或朋友等。在此兩種領域中，我們利用 ABE 來定義 PHR 的存取規則，以使用者的身分與 PHR 的種類作為屬性，藉此分配每一個使用者的存取權限。病人毋須記錄使用者存取清單，通過 ABE 加密 PHR，允許病人對他的 PHR 做更細緻化的存取控制，依據不同的角色與需求制訂更有彈性的存取規則。同時，病人本身僅需管理私人領域之使用者的私密金鑰與 PHR 種類相關的金鑰。就使用者數量而言，公開領域之使用者所涵蓋的範圍遠遠大於私人領域，故我們將與公開領域使用者身分相關之屬性金鑰交由屬性授權中心管理，大大減輕病人管理金鑰的負擔。

5.4.1 金鑰設定

在我們的系統中，屬性的種類包含被授權者的身分與 PHR 的種類，前者像是醫生、護士、藥師等，後者則像是病人的過敏記錄、各科處方簽、手術紀錄等。而在公開領域中，由屬性授權中心來管理與發送金鑰，這些屬性授權中心分別控制與使用者身分職業相關的屬性，並產生公開金鑰與私密金鑰，公開金鑰的部分可以公布在該授權中心的網頁上供病人下載；私密金鑰的部分則由使用者向授權中心提出請求，經由授權中心驗證之後取得。譬如說一個具有醫生身分的使用者可向醫生公會請求私密金鑰，待醫生公會審核通過之後方可取得含有醫生屬性的金鑰，使用者若是在 A 醫院任職的話也可以向醫院請求私密金鑰，也就是使用者可以向多個授權中心請求與其身分相符的私密金鑰，最後這個使用者手中可能握有“醫生”、“內科”、“A 醫院”這些屬性的私密金鑰。

另一方面，病人主要負責管理與其 PHR 種類相關的屬性與少部分私人領域中的使用者屬性。系統中事先定義一些屬性來分類 PHR，像是手術紀錄、血液測試等，此外，還定義朋友、家人屬性，讓病人能夠管理在私人領域中的使用者。當病人註冊進入系統後，會替這些屬性分別產生公開金鑰與主金鑰，病人可依其家人、朋友身分給予相對應的私密金鑰。此外，在公開領域的部分，我們首先利用身分的屬性來過濾那些人可以存取病人的 PHR，接著透過病人即時授權一把與 PHR 種類相關的私密金鑰來決定該使用者可以存取病人哪一類的資料。舉例來說，當病人到醫院看病時，可以即時授權一把私密金鑰給醫生，此私密金鑰含有手術紀錄這個屬性，若病人將其 PHR 的存取規則訂為“A 醫院”AND “醫生” AND “手術紀錄”的話，則 A 醫院中的醫生，且有被病人

即時授權“手術紀錄”這個屬性的醫生可以存取病人的 PHR，但 A 醫院中的其他醫生則無法滿足該 PHR 的存取規則，因而無權去對該病人之 PHR 做存取之動作。

5.4.2 PHR 的存取規則

存取規則能夠以 Boolean formula 來表示，其中的每一項代表一種屬性，例如：$(a_1\ AND\ a_2)\ OR\ a_3\ OR\ a_4$，$a_1, a_2, \ldots$為屬性，而一個 Boolean formula 亦可以一棵 access tree 來表示，所有葉節點分別代表一個屬性，非葉節點與根節點則是代表 AND 跟 OR。而在加密時我們會將 Boolean formula 轉換成一個 LSSS matrix，詳細過程在 6.2 節會介紹。

為了同時讓公開領域與私人領域的使用者可以存取病人的 PHR，在制定存取規則時，病人可以將存取規則定義為如圖 10 所示：

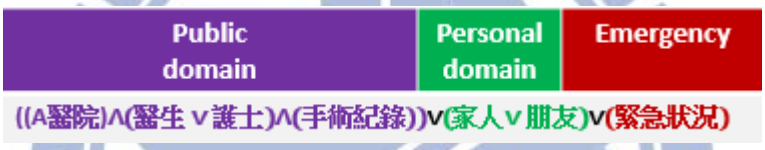


圖 10 存取規則

也就是將公開領域與私人領域中使用者的身分屬性以 OR 連結在一起，如此一來，無論是 A 醫院的醫生護士，或是病人的家人朋友，滿足存取規則的人都可以存取該份 PHR，病人只要在加密時定義好 PHR 的存取規則，即可讓公開領域與私人領域中符合權限的使用者存取 PHR，換言之，病人不僅無需針對不同領域的使用者設定不同的存取規則，更不需要重複加密 PHR。另外，若是要支援病人在緊急狀況時也能提供 PHR 給醫療人員使用的話，可將“緊急狀況”這個屬性利用 OR 加入存取規則中。

5.4.3 系統設定與金鑰發送

系統最初會定義一組與 PHR 種類相關的屬性，像是診所處方簽、手術紀錄等，當一個病人加入系統之際，病人根據這組屬性產生公開金鑰與私密金鑰，公開金鑰可公布在病人個人的網頁上，私密金鑰則由病人自己保存。而各個屬性授權中心加入系統之際會定義一組與使用者身分相關的屬性，同樣對每一個其所管理的屬性產生公開金鑰與私密金鑰，在我們的系統中，由於病人本身亦管理部分的金鑰，故可將病人視為一個屬性授權中心。

假如使用者想要取得私密金鑰的話，他必須向屬性授權中心提出請求，並證明自己的身分，屬性授權中心驗證後會回傳一把與其身分相符之私密金鑰，如圖 11 所示，我們假設這些請求與金鑰的傳遞都是在一個安全的通道上。爾後使用者即可以此把金鑰作為身分的證明，只要身分符合病人 PHR 中存取規則所規範，使用者就能存取病人儲存在雲端上的 PHR。

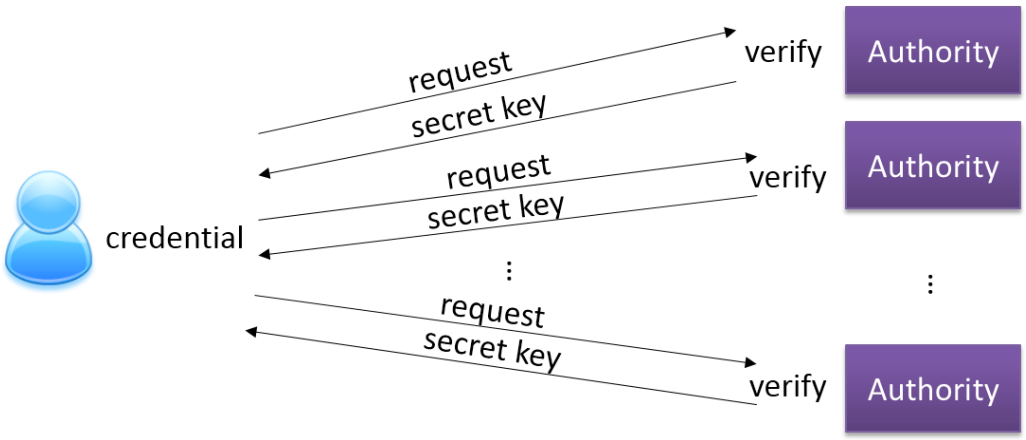


圖 11　金鑰授權

5.4.4　PHR 的加密與儲存

PHR 產生的途徑可能是來自於病人的個人資料、血壓紀錄或是診所處方簽等，當病人取得 PHR 後，可將 PHR 傳送到雲端伺服器儲存，與其他使用者共享，而在上傳前，必須先加密 PHR，以保障 PHR 資料的隱密性。但是在這裡並非直接以 ABE 對 PHR 加密，因為 PHR 的資料可能很龐大，直接用 ABE 加密所花費的時間會非常可觀，因此在進行加密時，會先產生一把 AES 的金鑰，先以此金鑰加密 PHR，接著再以 ABE 加密此把 AES 金鑰，並定義存取規則。加密結束後，使用者才將已加密的 PHR 連同經 ABE 加密的 AES 金鑰上傳至雲端伺服器存放，其存放格式如下圖 12：

FID	ABE(k)	E_k(PHR)

圖 12　加密之 PHR 格式

FID 為該 PHR 之識別碼，k 為 AES 金鑰，ABE(k)是以 ABE 加密之 AES 金鑰，$E_k(PHR)$ 則是以 AES 加密之 PHR。

5.4.5 PHR 的解密與存取

如圖 13 所示，使用者透過網路，向雲端伺服器提出請求，傳送他想要下載的 PHR 之 FID 給雲端伺服器，雲端伺服器再根據 FID 找出對應的 PHR 及其 AES 金鑰並回傳給使用者。使用者獲得 PHR 與 AES 金鑰後，假若他手中所持有的私密金鑰與存取規則相符合的話，則使用者可以解出 AES 金鑰，並用此把金鑰解密 PHR，取得明文的資訊，反之，若其私密金鑰無法滿足存取規則時，將無法正確解回 AES 金鑰，換句話說，使用者並無存取該 PHR 之權限，無法將 PHR 正確解密回來。

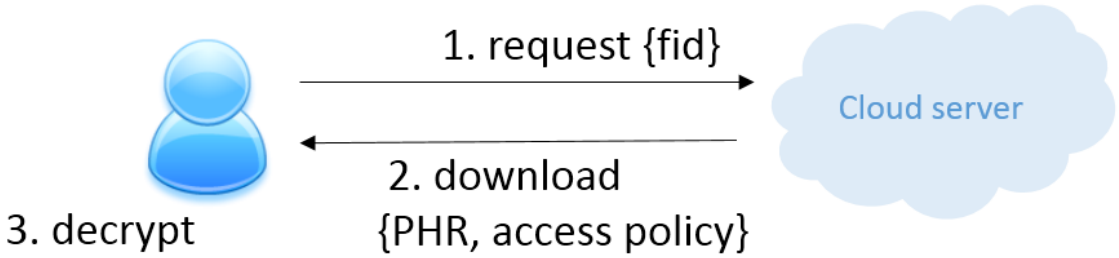


圖 13 PHR 解密與下載

5.5 PHR 系統-使用 DABE 做存取控制

5.5.1 DABE

本研究所提出的框架主要是運用 DABE 來達成對資料的存取控制，DABE 中每一個使用者都會被賦予一個全域識別碼 GID，GID 為該使用者所獨有，用來連結使用者從不同授權中心所取得之私密金鑰，防止使用者之間互相串謀，將彼此之私密金鑰連結起來。而 DABE 主要分成五個演算法：

Global setup

在這個演算法中定義系統中全域的初始設定。選定一個 bilinear pairing $e: G_1 \times G_1 \rightarrow G_T$，$G_1$、$G_T$的序為質數 N ，而 g_1 為 G_1 底下的一個生成子。此外，定義一個雜湊函數 $H: \{0,1\}^* \rightarrow G_1$，g、N、H 為公開的參數，在初始設定後公布給全系統的使用者。

Authority Setup

授權中心對於每一個他所掌管的屬性 i，為其選擇兩個任意的常數 α_i、$y_i \in Z_N$，公開金鑰為 $\mathrm{PK} = \{e(g_1, g_1)^{\alpha_i},\ g_1^{y_i}\ \forall \mathrm{i}\}$，而主金鑰為 $\mathrm{MK} = \{\alpha_i, y_i\ \forall \mathrm{i}\}$。

Encryption

加密演算法讀入欲加密的信息 M 與存取規則，首先會將存取規則轉換成一個 n × l 的 LSSS matrix A 與一個函數 ρ，ρ 將 LSSS matrix 中的列對應到屬性。接著隨機選擇一個常數 $\mathrm{s} \in Z_N$，以及一個隨機的向量 $\mathrm{v} \in Z_N^l$，$\mathrm{v} = \{\mathrm{s}, v_1, \ldots, v_{l-1}\}, v_i \in Z_N$，運算 s 的 share，$\mathrm{A} \cdot \mathrm{v} = \{\lambda_1, \ldots, \lambda_n\}$，再隨機選擇一個向量 $\mathrm{w} \in Z_N^l$，$\mathrm{w} = \{0, u_1, \ldots, u_{l-1}\}, w_i \in Z_N$，得 $\mathrm{A} \cdot \mathrm{w} = \{w_1, \ldots, w_n\}$。對每一個在存取矩陣中的列 A_x，選定 $r_x \in_R Z_N$，則可算出密文為：

$$C_0 = Me(g_1, g_1)^s,$$
$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x},$$
$$C_{2,x} = g_1^{r_x},$$
$$C_{3,x} = g_1^{y_{\rho(x)} r_x} g_1^{w_x}\ \forall x$$

KeyGen

授權中心針對某個使用者，其全域識別碼為 GID，對每一個欲授權的屬性 i，授權中心產生私密金鑰為：

$$K_{i,GID} = g_1^{\alpha_i} H(GID)^{y_i}$$

Decryption

假設密文是被加密在 LSSS matrix (A, ρ) 之下。解密者首先利用他的全域識別碼算出H(GID)，倘若解密者所持有的私密金鑰 $\{K_{\rho(x),GID}\}$ 對應到存取矩陣的列 A_x，使得 $(1,0,\ldots,0)$ 可以由這些列的子集合所生成，則對於每一個屬性 x，解密者可算出：

$$C_{1,x} \cdot \frac{\mathrm{e}(\mathrm{H}(\mathrm{GID}), C_{3,x})}{\mathrm{e}(K_{\rho(x),GID}, C_{2,x})} = e(g_1, g_1)^{\lambda_x} e(H(GID), g_1)^{w_x}$$

然後，解密者選擇一組常數 $c_x \in Z_N$，使得 $\sum_x c_x A_x = (1,0,\ldots,0)$，接著再算出：

$$\prod_x (e(g_1, g_1)^{\lambda_x} e(H(GID), g_1)^{w_x})^{c_x} = e(g_1, g_1)^s$$

最後可以得到明文為:

$$\mathrm{M} = {C_0}/{e(g_1, g_1)^s}$$

5.5.2 系統流程

以下我們利用 DABE 來建構我們的系統，主要會用到前一節所介紹到的五個演算法，Global Setup、Authority Setup、KeyGen、Encrypt、Decrypt：

系統初始

1. 系統開始運作之初，執行 **Global Setup** 定義 bilinear pairing e 以及雜湊函數 H，公開參數為 N、g、H。
2. 屬性授權中心最初先定義其所管理之屬性，並執行 **Authority Setup**，產生公開金鑰與主金鑰，並將公開金鑰公布給系統中所有使用者，主金鑰則妥善保存用以授權。
3. 每一個加入系統的使用者皆會賦予一個獨特的全域識別碼 GID，由使用者所保管，於要求授權金鑰時使用。

金鑰產生與授權(屬性授權中心)

1. 使用者傳送請求給授權中心，告知授權中心欲被授權之屬性與其 GID。
2. 經屬性授權中心審核通過之後，執行 **KeyGen**，產生與使用者所要求之屬性相對應的私密金鑰，並回傳給使用者。
3. 使用者儲存從屬性授權中心所取得之金鑰，待解密 PHR 時使用。

金鑰產生與授權(病人)

1. 使用者傳送請求給病人，告知授權中心欲被授權之屬性與其 GID。
2. 病人根據想要授權給使用者之 PHR 種類，執行 **KeyGen** 產生與該種類相對應的屬性之私密金鑰，並即時授權給使用者。
3. 此後，使用者可運用此私密金鑰去存取該病人之 PHR，假若使用者所持有之私密金鑰符合 PHR 之存取規則即可成功解密。

PHR 加密

1. PHR 之產生：病人由醫院、醫療機構取得健康資料或是個人紀錄量測之血壓與飲食紀錄等。
2. 假使病人想將 PHR 的資料與其他使用者共享，首先他先選擇一把 AES 金鑰，將 PHR 以 AES 作加密。

3. 經由 AES 加密後，再訂定存取規則，根據存取規則中會使用到的屬性，選取屬性授權中心的公開金鑰與自己所管理之公開金鑰，執行 **Encrypt** 將 AES 金鑰加密。
4. **Encrypt** 會先將存取規則由 Boolean formula 轉換成一個 n × l 的 LSSS matrix，n 為屬性的數量，並定義映射函數 ρ，$\rho(x)$ 代表存取矩陣中每一列 x 所映照到的屬性。加密後的 PHR 包含密文、LSSS matrix 與 ρ。
5. 接著再為 PHR 選擇一個 FID，此 FID 不可與系統中其他 PHR 重複。
6. 最後，病人將加密的 PHR 及 AES 金鑰上傳到雲端伺服器。

PHR 解密

1. 使用者提供要下載之 PHR FID 給雲端伺服器，雲端伺服器依照 FID 傳回 PHR 與 AES 金鑰。
2. 首先找出使用者私密金鑰中與 PHR 存取規則相對應之屬性，運算在存取矩陣中是否可找出一子集可以線性組合出 $(1,0,\dots,0)$，若存在這樣的子集的話，使用者再執行 **Decrypt**。
3. 若使用者提供的金鑰正確的話即可解回 AES 金鑰，否則 **Decrypt** 會回傳 $\perp$。
4. 解回 AES 金鑰後即可將 PHR 解密，若使用者提供的私密金鑰中之 GID 一致，則可正確解密 PHR，否則將解密失敗。

5.6 安全分析

在[7]中有 DABE 詳細的安全證明。在我們的 PHR 系統中，所要達到的安全性包含 PHR 的隱密性與抵禦使用者間的同謀攻擊。不論是未經授權的使用者或是雲端供應商，即使他們獲得 PHR 密文也無法解密。

一個經授權的使用者唯有在其所擁有的屬性滿足存取規則的情況下方可解密，換句話說，由存取結構轉成的 LSSS matrix 中，我們可以找出一個屬性的子集合 X，選擇一組常數 $c_x \in Z_N$，使得 $\sum_{x\in X} c_x A_x = (1,0,\dots,0)$，如此才能算出：

$$\prod_x \left(e(g_1,g_1)^{\lambda_x} e(H(GID),g_1)^{w_x}\right)^{c_x} = e(g_1,g_1)^s \qquad (1)$$

而未經授權的使用者所擁有的屬性無法找出一組列的集合使得其作線性組合之後為 $(1,0,\dots,0)$，故不能算出 $e(g_1,g_1)^s$。

接著我們討論使用者間的同謀攻擊，也就是使用者並非單獨使用其所被授權的金鑰來作解密，而是與其他使用者聯合起來，意圖存取不在其權限之內的 PHR。假設這些同謀者之中可以找出一組屬性的子集合 X，使得 $\sum_{x\in X} c_x A_x = (1,0,\dots,0)$，儘管看起來這些使用者的私密金鑰滿足了存取規則，但是在解密時，由(1)的式子可看到必須算出 $e(H(GID), g_1)^{w_x}$，然而，每一個使用者的私密金鑰中的 GID 都是相異的，因此最後仍然無法正確算出 $e(g_1, g_1)^s$，這些同謀者無法解密成功。而雲端伺服提供商亦無法解密，這是因為他並未被授權任何私密金鑰，即使與其他使用者串通起來，由於上述的理由，雲端伺服器提供商還是無法解密 PHR 獲取任何資訊。

所以利用 DABE，讓使用者可以對 PHR 達到細粒度的存取控制，並且保證了資料的隱密性與抵禦同謀攻擊，而在屬性授權中心的部分，假設有 N 個屬性授權中心，DABE 最多可達到 N-1 個屬性授權中心同謀攻擊的安全性，換言之，除非所有的屬性授權中心都串謀起來或是全被攻擊者擊潰，否則無法產生出系統中所有使用者的私密金鑰。

第六章、實作

系統實作的部分，我們會介紹建構 PHR 系統所需的工具與環境，實作 DABE 的演算法，並提供使用者一個操作介面用以管理他的 PHR。

6.1 DABE 實作

6.1.1 開發環境設定

DABE 主要包含五個演算法，在實現這些演算法上，我們使用的開發語言是 Java，開發軟體為 Ellipse 4.2.0，程式運行於 2.4 GHz Intel core i5 的 CPU 之上，作業系統為 Windows 7，此外，為了實現 DABE 中 bilinear pairing 的運算，我們引入 Java Pairing-Based Cryptography(JPBC)套件。

6.1.2 JPBC 函式庫[22]

JPBC 函式庫係由 Ben Lynn 所開發，他將原本由 C 語言撰寫的 Pairing-Based Cryptography(PBC)改成 java 的 porting。JPBC 支援 symmetric 與 asymmetric pairing 的運算，主要分為六個種類：Type A、Type A1、Type D、Type E、Type F、Type G，我們選擇 Type A 來做 paring 的運算。Type A 建立在橢圓曲線 E: $y^2 = x^3 + x$ 之上，E 是一個 supersingular curve，因此 Type A 為 symmetric pairing，field 的大小為 512 bit，embedding degree 為 2，而群的序是一個 160 bit 的質數。根據表 1，我們所選擇的型態其安全強度與 RSA 1024 bit 的安全強度相等。

Symmetric key size	RSA	Elliptic curve key size
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

表 1　　NIST 建議的金鑰長度

6.1.3 實作

本節描述我們如何實作 DABE，程式主要的架構如圖 14：

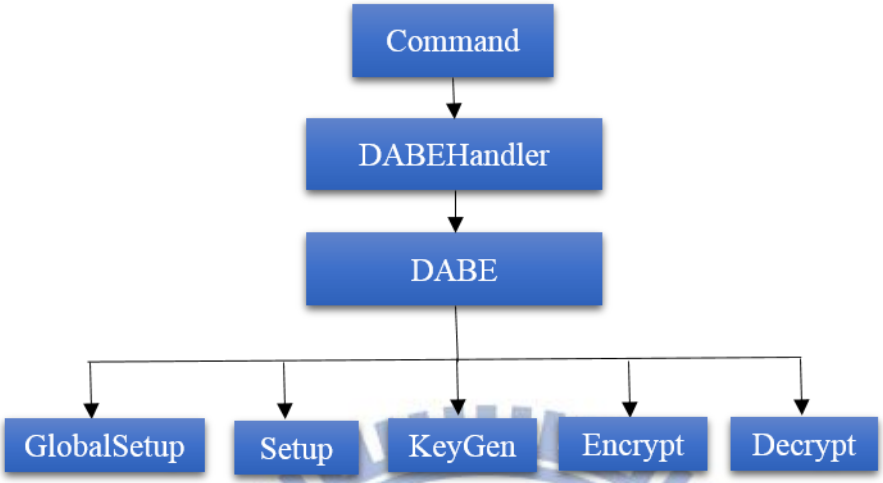


圖 14 程式架構

由 command 讀入指令，DABEHandler 判斷目前要執行哪一種演算法，而在 DABE 中包含以下五個程式：

GlobalSetup

產生 pairing 以及生成子 g。

Setup

指令格式：set <pkFile> <mkFile> <attribute1> <attribute2>.....

輸入一組屬性，此程式會對每一個屬性產生公開金鑰與主金鑰，並分別寫入到公開金鑰與主金鑰檔案之中。

KeyGen

指令格式：key <gid> <skFile> <mkFile> <attribute1> <attribute2>.....

首先讀入使用者的 GID，接著再讀取授權中心主金鑰檔案，以及要授權給使用者屬性清單，最後將產生給使用者的私密金鑰寫入檔案。

Encrypt

指令格式：enc <file> <encFile> <accessPolicy> <authority1> <authority2>.....

程式讀入定義好的存取規則，此存取規則以 Boolean formula 表示，再讀入存取規則中所需要使用到的授權中心之公開金鑰。接著將 Boolean formula 的運算式轉為前序的運算

式，再建立 access tree，根據 access tree，產生 LSSS matrix，再以 LSSS matrix 做加密，最後將結果輸出至檔案。

Decrypt

指令格式：dec <gid> <file> <aesKeyFile> <decFile> <skFile1> <skFile>.....

解密時，讀入要解密的檔案及其 AES 金鑰，使用者必須輸入其 GID 與其所持有之私密金鑰，解密成功的話，程式會成功將檔案解密並寫入新的檔案，反之，解密失敗的話程式會出現錯誤訊息並終止。

6.2 Boolean formula 到 LSSS matrices 的轉換

存取規則可以用 monotonic Boolean formula 來描述，而 LSSS 的存取結構亦可由 Boolean formula 轉換而成。有許多標準的技巧可以執行此一轉換。其中一種像是可以將 Boolean formula 表示成 access tree，內部的節點為 AND 與 OR 這兩種邏輯閘，而葉節點則對應到各個屬性。LSSS 中列的個數與 access tree 中葉節點的個數相同。

假設有一個 Boolean formula：“(A AND B) OR (C AND D) OR E”，在實作上我們會先將此 Boolean formula 轉成前序式表示：“OR OR AND A B AND C D E”，接著再建造 access tree，如圖 15 所示：

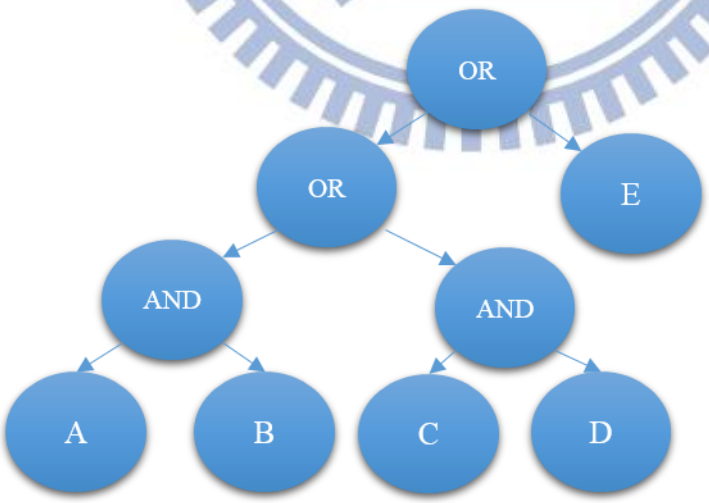


圖 15 Access tree

Access tree 建造完成之後，利用 Lewko [7]中所提的方法進行轉換。在[7]中提到 LSSS matrix 等同於 Monotone span program (MSP)，有關此演算法的正確性可參考[23]。首先對 access tree 的每一個節點產生一個長度為 1 的 vector，並初始一全域變數 c 為 1，假使父節點為 OR，其 vector 為 v，則其子節點所對應的 vector 亦為 v，若父節點為 AND，vector 為 v 的話，則先將 v 以 0 做 padding 使得 v 的長度等於 c，設定一子節點的 vector 為 $v|1$，另一子節點的 vector 為$(0,\dots,0)|-1$，$(0,\dots,0)$的長度為 c，使得這兩個節點的 vector 總和為 $v|0$。最後再將 c 加 1。對 access tree 遞迴執行這些動作，直到每個節點的 vector 都產生出來。以上述的例子來說，首先根節點為 OR，產生 vector 為 (1)，其左節點 OR 的 vector 為 (1)，右節點 E 的 vector 亦為 (1)，接著由於又是 OR，故其左右節點 AND 的 vector 也都是紀錄為 (1)，對左邊 AND 節點而言，其左節點標示為 (1,1)，其右節點則標示為 (0,-1)，並將變數 c 加 1，此時 c = 2，對右邊的 AND 節點來說，先將 v 做 padding 為 (1,0)，其左節點標示為 (1,0,1)，右節點則為 (0,0,-1)，並將 c 加 1，此時 c = 3，所有的節點都標示完後，將全部的 vector 都用 0 padding 到 c 的長度，最後可以得到 LSSS matrix 為：

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix}$$

此 LSSS matrix 每一列分別代表屬性 A、B、C、D、E 這些屬性，定義一個 ρ 函數：

x	**1**	**2**	**3**	**4**	**5**
$\boldsymbol{\rho}$**(x)**	A	B	C	D	E

表 2　　ρ 函數

x 代表 LSSS matrix 中的列，ρ(x)則是每一列對應到的屬性，使用者在加密後會保存 LSSS matrix 及 ρ 函數以供解密時查詢。如果使用者想要加密一個 secret s，則將 LSSS matrix 與 (s, r_1, r_2) 相乘，可得 $(s + r_1, -r_1, s + r_2, -r_2, s)$，只要使用者提供的屬性符合存取規則即可解回 secret s，換句話說，在 LSSS matrix 中，符合存取規則的屬性所對應到的列，將這些列做線性組合可以得到 (1,0,0)，譬如 A、B 屬性分別對應到 (1,1,0)與 (0,-1,0)，(1,1,0) + (0,-1,0) = (1,0,0)。

6.3 PHR 系統實作

在這節中，介紹我們所實作的 PHR 系統，此系統採用 Apache+PHP+MySql 組合，在 windows 7 系統之下，架設了一個網頁伺服器，並利用資料庫來儲存使用者的個人資料與其 PHR 檔案資訊，除此之外，提供使用著一個友善的操作介面，使得他們可以透過網頁輕鬆地上傳、下載、管理 PHR。在雲端伺服器方面，我們使用 Dropbox 提供的雲端儲存服務，將使用者的 PHR 存放在 Dropbox 上，讓其他被授權的使用者可以透過 Dropbox 來存取 PHR，共享資料。圖 16 是這個系統主要的架構圖。

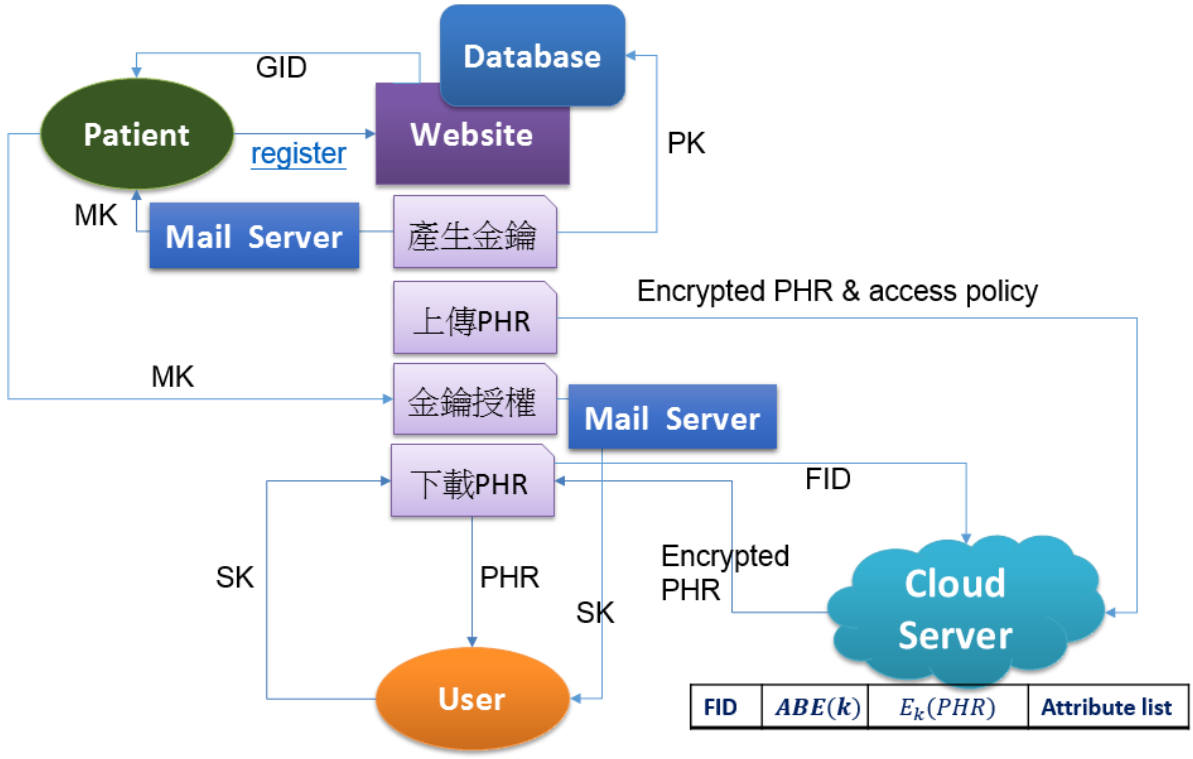
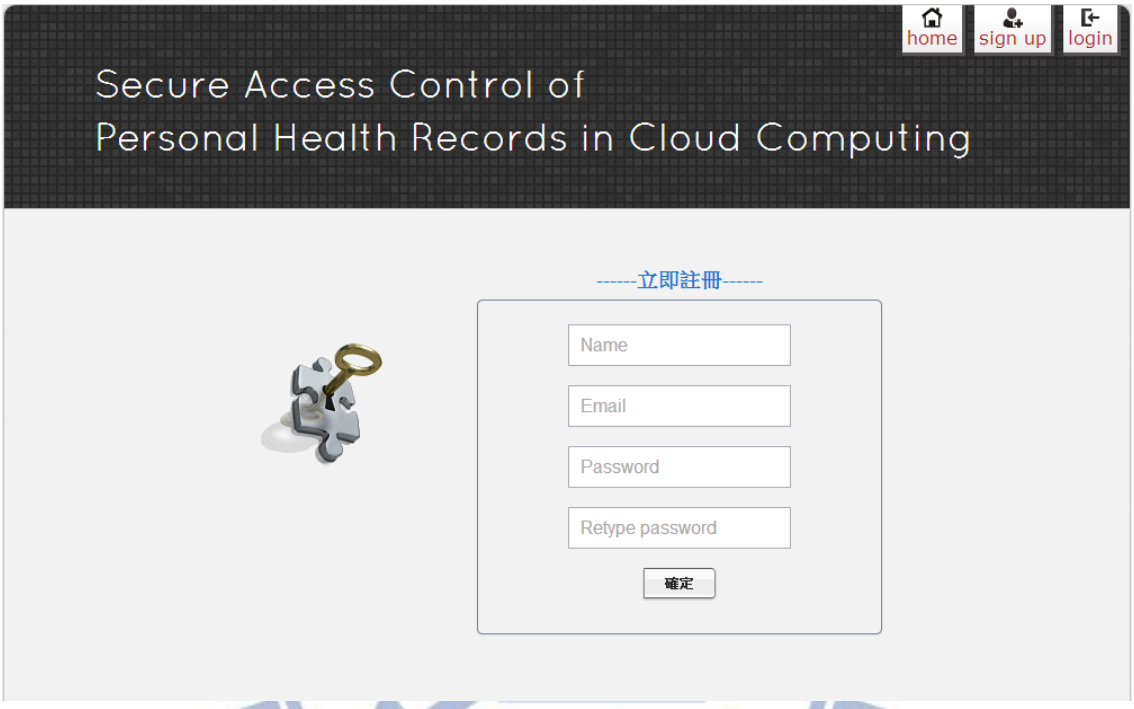


圖 16 PHR 系統架構圖

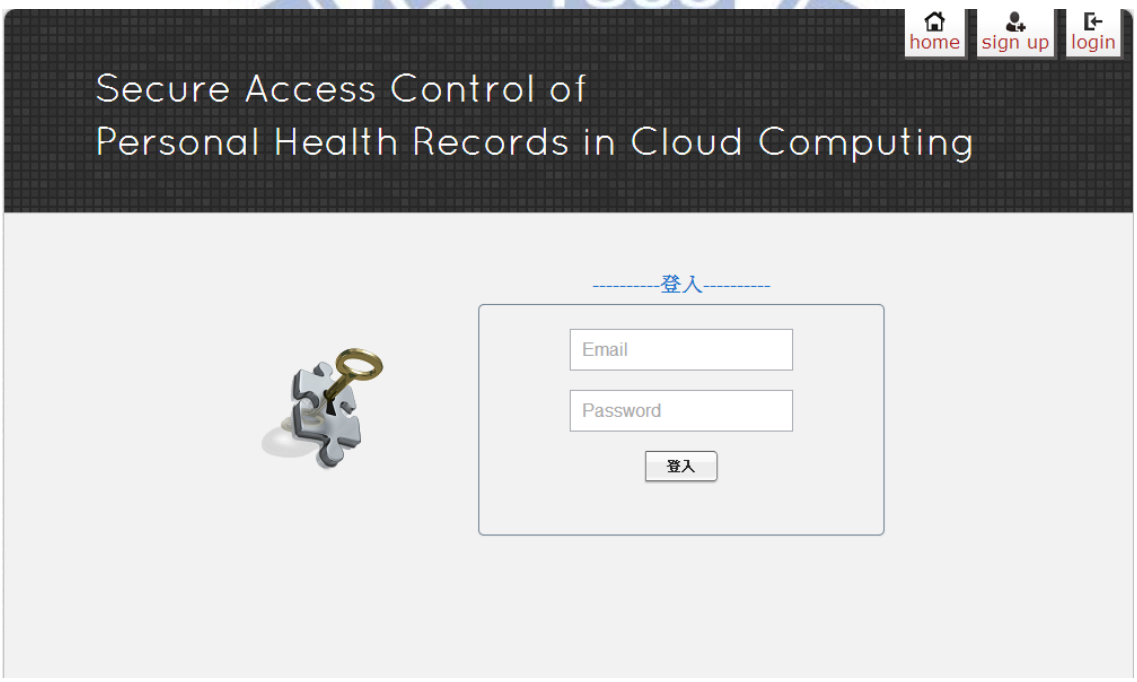
在這個網頁中，主要提供五種功能：註冊、產生金鑰、上傳 PHR、金鑰授權與下載 PHR，使用者只要連結到我們的網頁就可以使用 PHR 管理的服務。而在這個系統中，會與 6.1 節中 DABE 做結合，以 DABE 中的 GlobalSetup、Setup、KeyGen、Encrypt、Decrypt 五個程式來實現系統所提供的五種功能。系統運轉之初首先執行 GlobalSetup 設定基本參數，其他的系統流程如下：

註冊

參考圖 17，病人首次進入這個系統必須先註冊一個帳號，帳號為病人的電子郵件信箱，而病人的名稱與電子郵件信箱在之後的金鑰授權中會使用到。註冊完成後系統會為病人產生一個獨有的 GID，並將病人的個人資料與 GID 存入資料庫中，系統將以 GID 或使用者名稱來識別病人身分。此後，利用此 GID，其他使用者可以產生私密金鑰並授權給病人。病人登入之後，可以點選左方列表中的功能，選擇要執行的動作。



(a)



(b)

圖 17　(a)註冊，(b)登入

產生金鑰

首先，病人可以產生自己的公開金鑰與主金鑰，如圖 18 所示，在此我們將 PHR 約略分為十個種類：皮膚科類資料、牙科類資料、骨科類資料、眼科類資料、健康紀錄、過敏記錄、急救紀錄、精神科類資料、手術紀錄與復健紀錄等，前七種類別為系統預設選取，最後三種類別由病人根據其需求自由選取。使用者選定好類別後，系統將這些類別作為屬性，執行 Setup 產生公開金鑰與主金鑰，接著系統會將公開金鑰儲存在網頁伺服器之中，主金鑰則利用郵件傳送給病人，如圖 19 所示，病人必須自己妥善保管此主金鑰，以待日後授權金鑰時使用。

圖 18　金鑰授權

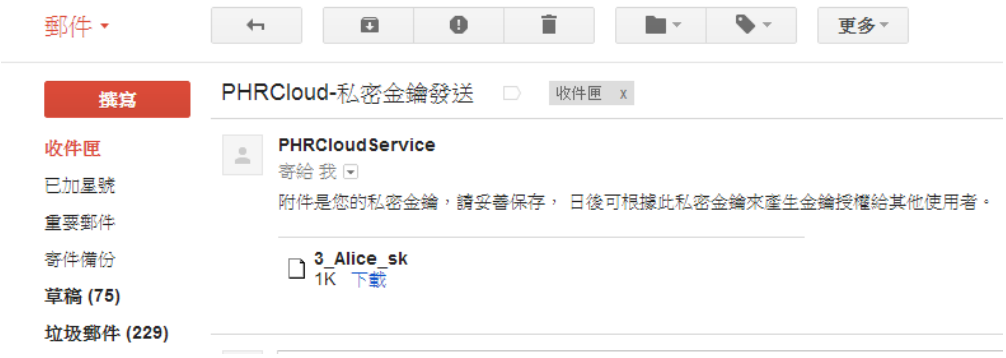


圖 19　使用者透過信箱收到主金鑰圖

上傳 PHR

病人可以上傳他的 PHR，參考圖 20，病人先選擇想要上傳的 PHR，並為此 PHR 選擇合適的種類，接著選擇可以存取此 PHR 之使用者身分，而被授權者的身分主要劃分為公開領域與私人領域兩種，公開領域中的屬性有醫院與醫事人員兩種類別，這兩種類的屬性分別由其他兩個屬性授權中心所管理，在系統初始之際這兩個授權中心就設定完成，為其所管理的屬性產生公開金鑰，並接受使用者的請求授權私密金鑰。而在這個系統中，病人可以使用這兩個屬性授權中心的公開金鑰來對其 PHR 進行加密。病人依據被授權者的身分，從公開領域與私人領域中至少選取一種身分的被授權者(例如：病人可選取“A 醫院”AND“醫生”或者只選取“家人”等)，決定被授權者身分後即可上傳 PHR，系統會根據使用者所選取的 PHR 種類與使用者身分作為屬性執行 Encrypt 將 PHR 做加密，加密完畢後將 PHR 與 AES 金鑰一併上傳至雲端伺服器。如圖 21 所示，PHR 在加密後會上傳到 Dropbox，每一份 PHR 都賦予一個獨特的 FID，加密的 PHR 包含以 AES 加密的 PHR 密文檔案與以 ABE 加密的 AES 金鑰檔案，由圖 21 可看到，譬如 8756 與 8756_key 這兩個檔案就分別代表加密的 PHR 以及 AES 金鑰，使用者要存取 PHR 時即是透過 FID 來尋找想要的 PHR。

圖 20　上傳 PHR

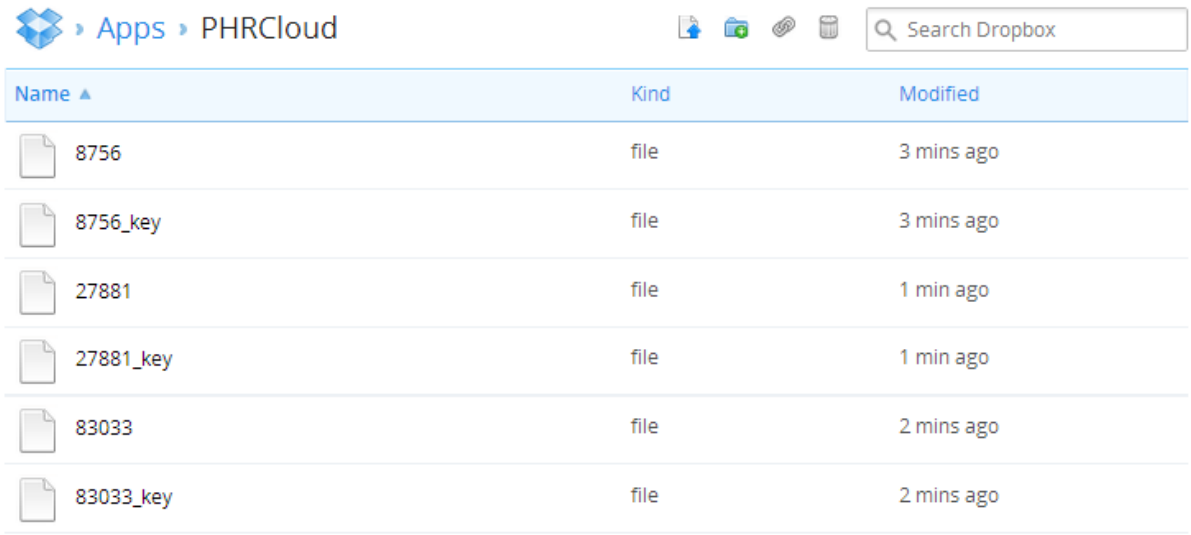


圖 21　雲端伺服器上的 PHR

另外，我們也提供一個頁面顯示已上傳的 PHR 資訊，如圖 22 所示，此頁面顯示病人 Alice 已上傳到雲端伺服器的 PHR 資訊，像是 PHR 的 FID、上傳時間與存取規則，病人可通過這個頁面瞭解自己上傳了那些 PHR 及加密規則，並且可以點取 delete 將儲存在雲端伺服器上的 PHR 刪除，進行簡單的 PHR 管理。

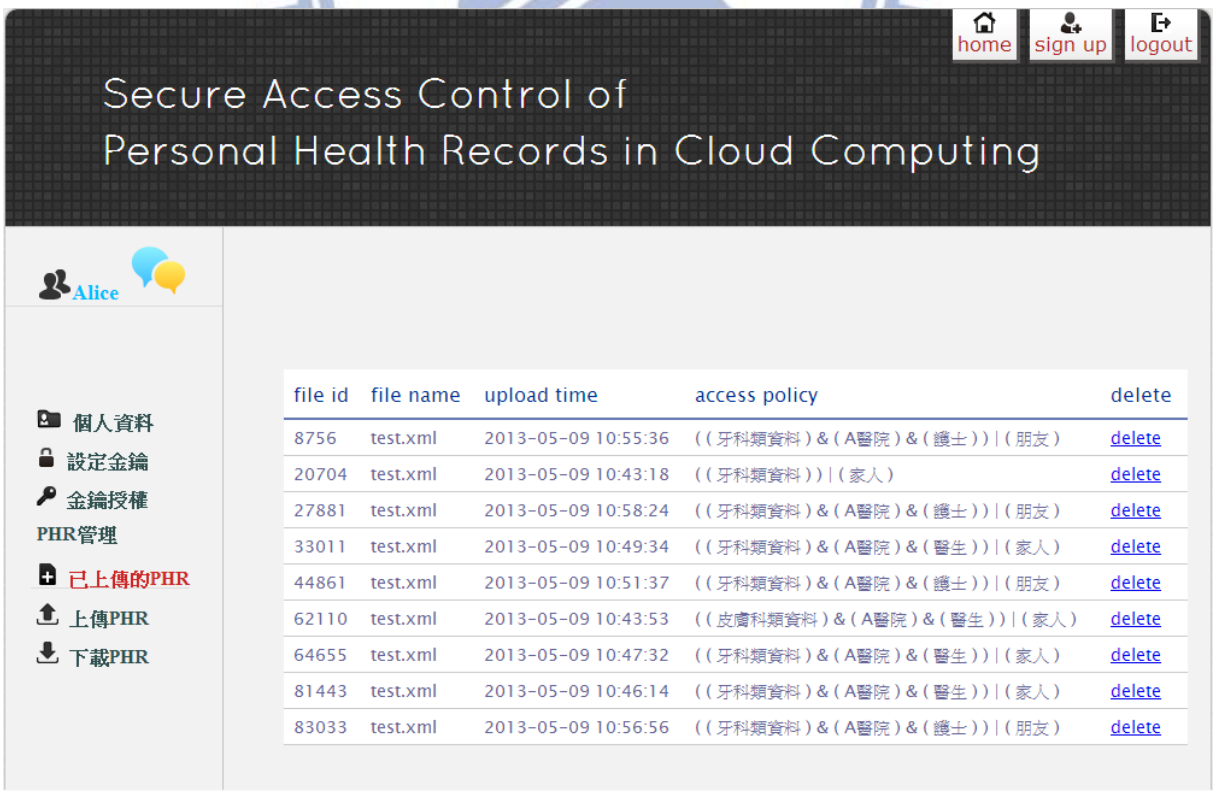


圖 22　已上傳的 PHR 頁面

金鑰授權

參考圖 23，假設 Alice 想將她的 PHR 分享給其他的使用者，Alice 可以授權私密金鑰給該使用者。首先 Alice 先選取要授權的 PHR 種類，接著選擇被授權者的身分，被授權者的身分可能是家人、朋友或其他使用者，若是其他使用者，則與其身分相關的屬性則是由其他屬性授權中心授權，像是醫生是從醫療機構取得含有“醫生”屬性的私密金鑰。選擇完 PHR 種類及被授權者身分後，Alice 必須上傳他的主金鑰，並且輸入被授權者的電子郵件信箱或是名稱，系統會根據這兩種資料取找出被授權者的 GID，執行 KeyGen 產生該被授權者的私密金鑰，最後將私密金鑰寄到被授權者的信箱。

圖 23　金鑰授權頁面

圖 24　被授權者收到私密金鑰圖

假設 Alice 授權給醫生 Bob 觀看他有關牙科類資料與過敏紀錄的 PHR，則 Bob 會收到系統發送的私密金鑰，如圖 24 所示，此私密金鑰為 Alice_Bob_牙科類資料_過敏記錄_key，表示由 Alice 授權給 Bob，此金鑰含有“牙科類資料”與“過敏紀錄”這兩種屬性，Bob 需自行妥善保存此金鑰。

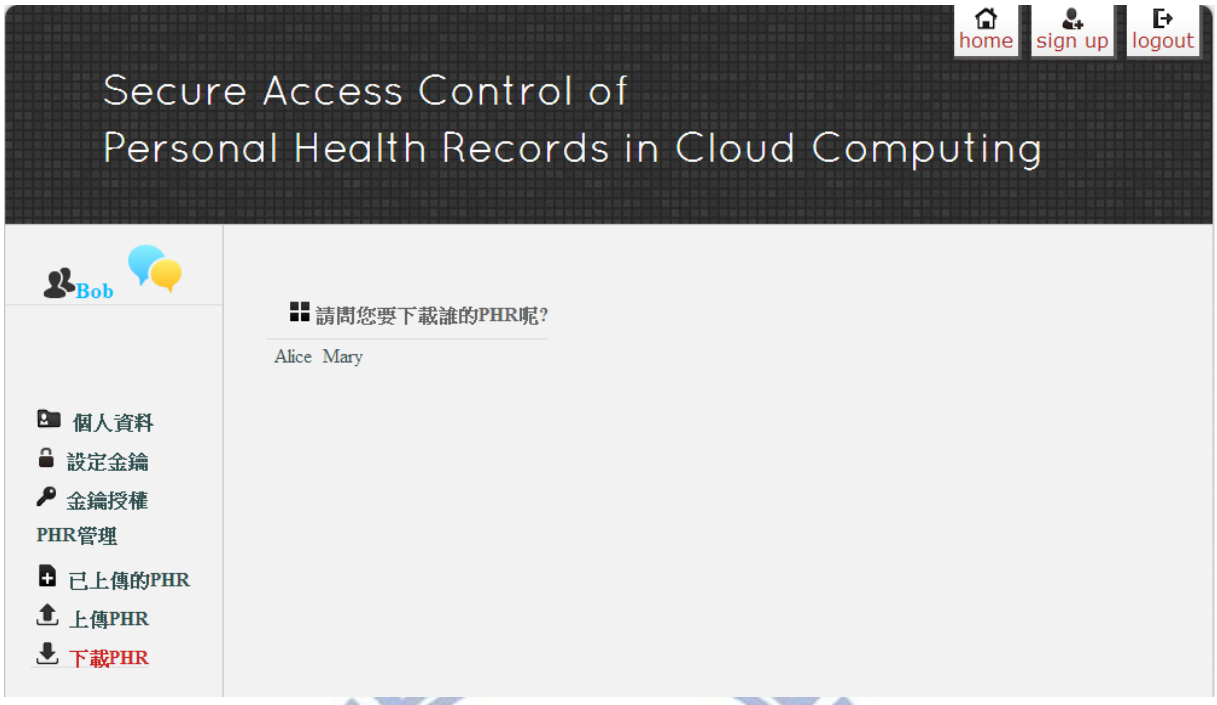
下載 PHR

假使醫生 Bob 已經從醫療機構與他工作的醫院取得“醫生”與“A 醫院”這兩把私密金鑰，而病人 Alice 透過此 PHR 系統授權給 Bob 含有“牙科類資料”與“過敏紀錄”屬性之私密金鑰，則 Bob 登入系統後，可以從下載 PHR 中看到授權給他私密金鑰的病人名稱，參考圖 25(a)，Bob 可以選擇要觀看 Alice 或 Mary 的 PHR。如果 Bob 點選要觀看 Alice 的 PHR，如圖 25(b)所示，網頁會列出所有 Bob 可觀看的 PHR，Bob 可參考每一份 PHR 的存取規則，將他的私密金鑰上傳。像是 Bob 想要存取 FID 為 33011 的這份 PHR 的話，此 PHR 的存取規則為

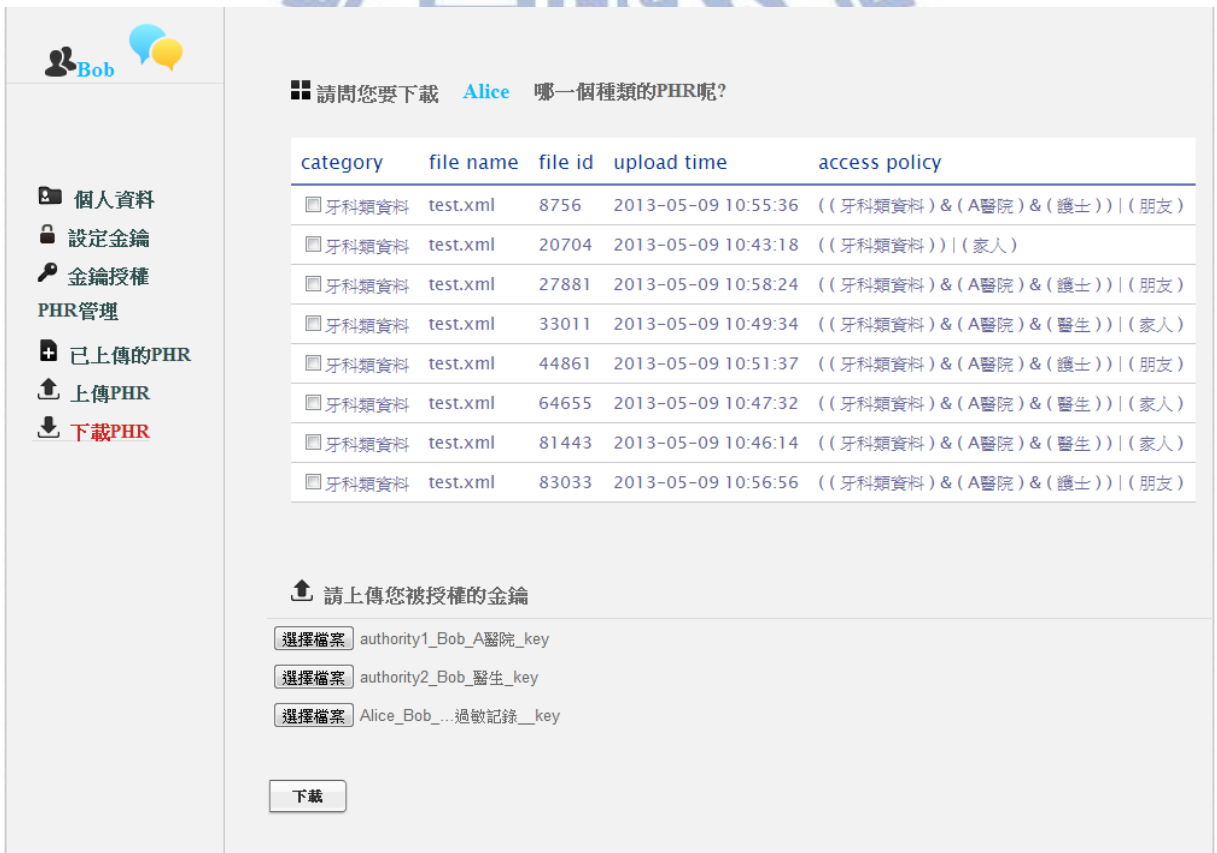
((牙科類資料) & (A 醫院) & (醫生)) | (家人)

由於 Bob 不是 Alice 的家人，因此若要滿足存取規則的話，Bob 必須上傳含有“牙科類資料”、“A 醫院”、“醫生”的私密金鑰。Bob 可以勾選想下載的 PHR，點擊下載後，系統會根據 FID 從雲端伺服器將該份 PHR 下載回來，並且利用 Bob 的私密金鑰與 PHR 執行 Decrypt 將 PHR 解密。

解密時若私密金鑰確實符合存取規則，Decrypt 程式可以解回 AES 金鑰，並且成功將 PHR 解密，如圖 26(a) 所示，此範例是加密一份病人的牙科處方簽，若醫生 Bob 成功解密則可以觀看正確的 PHR 內容。反之，假使 Bob 的私密金鑰不符合存取規則系統會告知 Bob 解密失敗。另外，Bob 如果想將自己的私密金鑰與其他使用者的結合起來以便獲取更大的權限去存取原本無法存取的 PHR 的話，在這邊也會解密失敗，這是由於每一把金鑰中的 GID 不同所致，因此即使被授權者們將金鑰結合起來用以符合存取規則，依然無法正確解密 PHR，如圖 26(b) 所示，解密出來的 AES 金鑰是錯誤的，故無法將 PHR 正確解密回來。

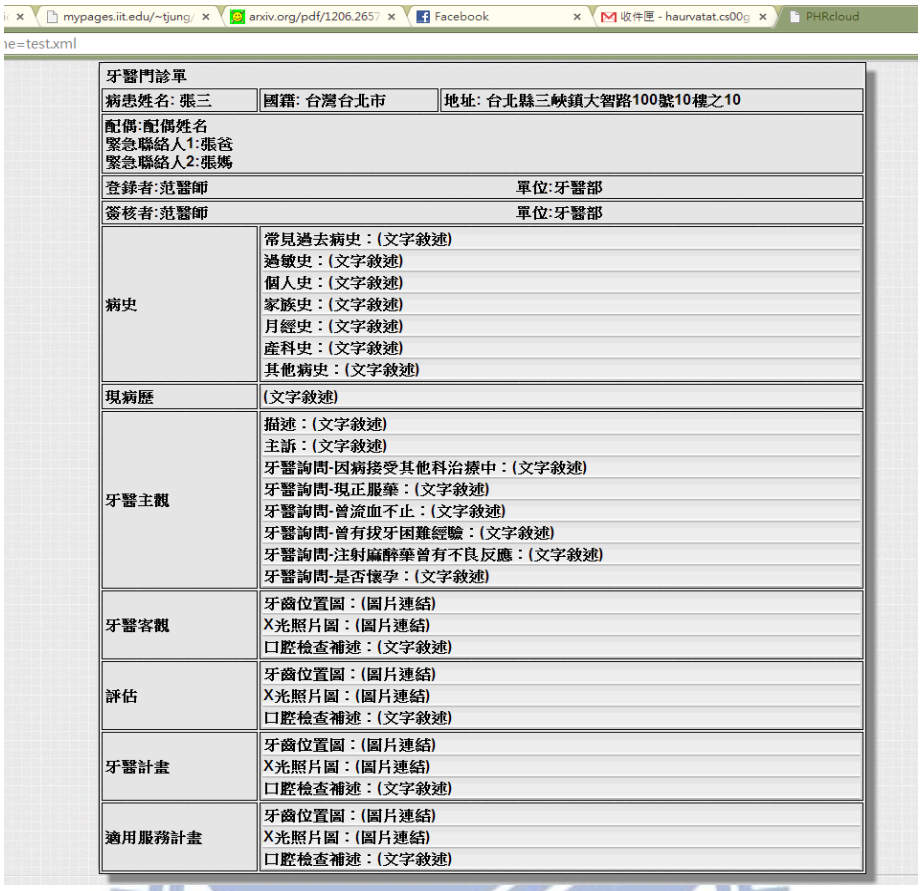

home
sign up
logout
Secure Access Control of
Personal Health Records in Cloud Computing
Bob
請問您要下載誰的PHR呢?
Alice Mary
個人資料
設定金鑰
金鑰授權
PHR管理
已上傳的PHR
上傳PHR
下載PHR

(a)


Bob
請問您要下載 Alice 哪一個種類的PHR呢?
category file name file id upload time access policy
牙科類資料 test.xml 8756 2013-05-09 10:55:36 ((牙科類資料)&(A醫院)&(護士))|(朋友)
牙科類資料 test.xml 20704 2013-05-09 10:43:18 ((牙科類資料))|(家人)
牙科類資料 test.xml 27881 2013-05-09 10:58:24 ((牙科類資料)&(A醫院)&(護士))|(朋友)
牙科類資料 test.xml 33011 2013-05-09 10:49:34 ((牙科類資料)&(A醫院)&(醫生))|(家人)
牙科類資料 test.xml 44861 2013-05-09 10:51:37 ((牙科類資料)&(A醫院)&(護士))|(朋友)
牙科類資料 test.xml 64655 2013-05-09 10:47:32 ((牙科類資料)&(A醫院)&(醫生))|(家人)
牙科類資料 test.xml 81443 2013-05-09 10:46:14 ((牙科類資料)&(A醫院)&(醫生))|(家人)
牙科類資料 test.xml 83033 2013-05-09 10:56:56 ((牙科類資料)&(A醫院)&(護士))|(朋友)
請上傳您被授權的金鑰
選擇檔案 authority1_Bob_A醫院_key
選擇檔案 authority2_Bob_醫生_key
選擇檔案 Alice_Bob_...過敏記錄__key
下載

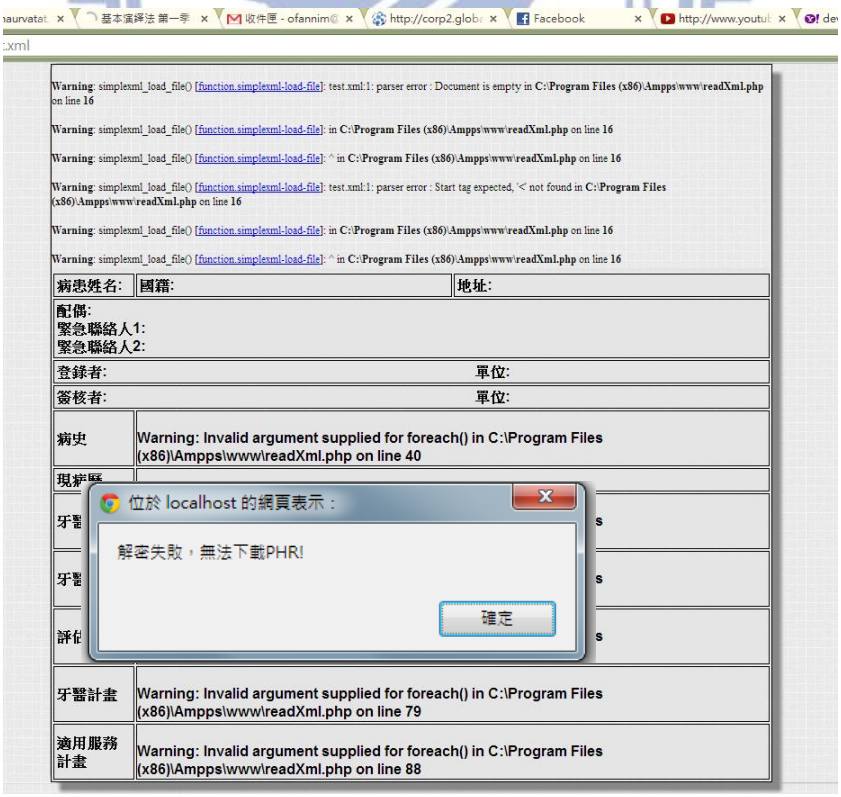
(b)

圖 25　下載 PHR



牙醫門診單		
病患姓名: 張三	國籍: 台灣台北市	地址: 台北縣三峽鎮大智路100號10樓之10
配偶:配偶姓名 緊急聯絡人1:張爸 緊急聯絡人2:張媽		
登錄者:范醫師		單位:牙醫部
簽核者:范醫師		單位:牙醫部
病史	常見過去病史：(文字敘述)	
	過敏史：(文字敘述)	
	個人史：(文字敘述)	
	家族史：(文字敘述)	
	月經史：(文字敘述)	
	產科史：(文字敘述)	
	其他病史：(文字敘述)	
現病歷	(文字敘述)	
牙醫主觀	描述：(文字敘述)	
	主訴：(文字敘述)	
	牙醫詢問-因病接受其他科治療中：(文字敘述)	
	牙醫詢問-現正服藥：(文字敘述)	
	牙醫詢問-曾流血不止：(文字敘述)	
	牙醫詢問-曾有拔牙困難經驗：(文字敘述)	
	牙醫詢問-注射麻醉藥曾有不良反應：(文字敘述)	
	牙醫詢問-是否懷孕：(文字敘述)	
牙醫客觀	牙齒位置圖：(圖片連結)	
	X光照片圖：(圖片連結)	
	口腔檢查補述：(文字敘述)	
評估	牙齒位置圖：(圖片連結)	
	X光照片圖：(圖片連結)	
	口腔檢查補述：(文字敘述)	
牙醫計畫	牙齒位置圖：(圖片連結)	
	X光照片圖：(圖片連結)	
	口腔檢查補述：(文字敘述)	
適用服務計畫	牙齒位置圖：(圖片連結)	
	X光照片圖：(圖片連結)	
	口腔檢查補述：(文字敘述)	

(a) PHR 解密成功



(b) PHR 解密失敗

圖 26 (a) PHR 解密成功(b) PHR 解密失敗

6.4 效能評估

本節對 6.1.3 節中所實作的程式進行效能評估，主要針對 Setup、KeyGen、Encrypt、Decrypt 這四個部分來量測其效能。對每一個演算法，我們量測其在屬性數量從 1 個遞增到 50 個時所需花費的時間，一共進行 100 次的測試最後取其時間平均值。

這四個演算法所量測出來的時間隨著屬性數量的遞增皆是呈線性成長，而圖 27 到圖 30 所量測出來的時間顯示出這四個演算法實際上是可行的。

在 Setup 以及 KeyGen 的部分，參考圖 27 與圖 28，在我們實作的系統中，每個病人實際上所管理的屬性不會超過 50 個，大約是 20 到 30 個之間，而 Setup 與 KeyGen 在 30 個屬性時的運算時間皆不會超過 2 秒，Setup 只需要在病人首次登入時運算一次，病人在每次做金鑰授權也不需要花太多的時間。

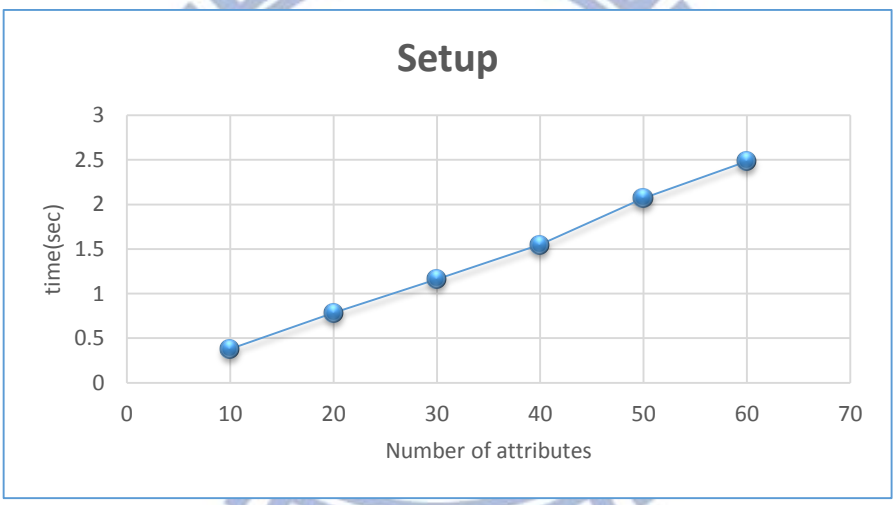


圖 27　初始設定時間

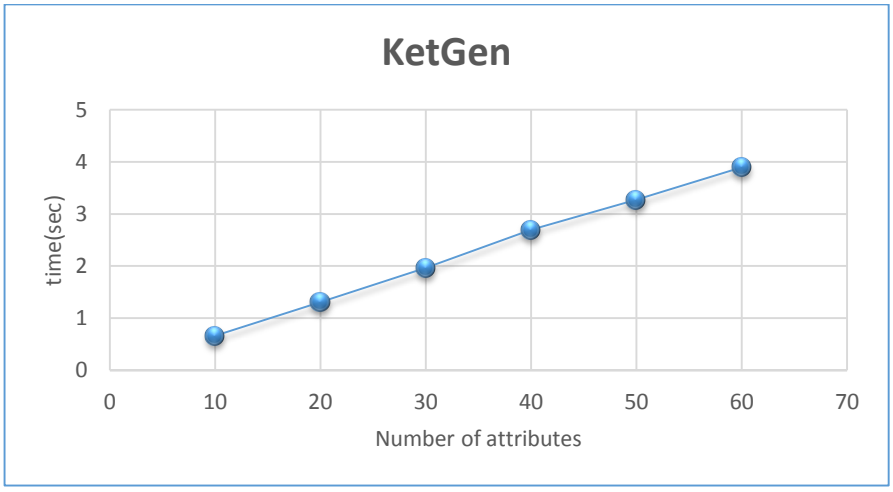


圖 28　金鑰產生時間

另一方面，Encrypt 的部分我們將存取規則設定成只有 OR 出現，參考圖 29，存取規則中的屬性數量到達 50 個的時候，所需的時間約為 4 秒，與[8]中使用 MA-ABE 及 KP-ABE 的加密時間相較之下需要較多的運算時間，這是由於 MA-ABE 和 KP-ABE 是將存取規則與金鑰做連結，而 DABE 的 Encrypt 需先將存取規則從 Boolean formula 轉換成 LSSS matrix，並且做多次的矩陣運算，此外，計算密文時，假設屬性的數量為 N，DABE 需要 5N+1 個 exponential 計算加上一次 Bilinear pairing 的運算，MA-ABE 則需要 N+2 個 exponential 計算，但是 MA-ABE 僅支援 CNF 形式的存取規則，DABE 則可以支援任何 Boolean formula 的存取規則。解密的時間如圖 30 所示，這裡的 x 軸為解密時用來滿足存取規則所需要的屬性數量，解密時的運算必須花費 3N 個 Bilinear pairing 以及 3N 個 exponential 運算，而 MA-ABE 需要 N 個 exponential 運算與 N+1 個 Bilinear pairing 運算，因此相較之下 DABE 需要較多的運算時間。

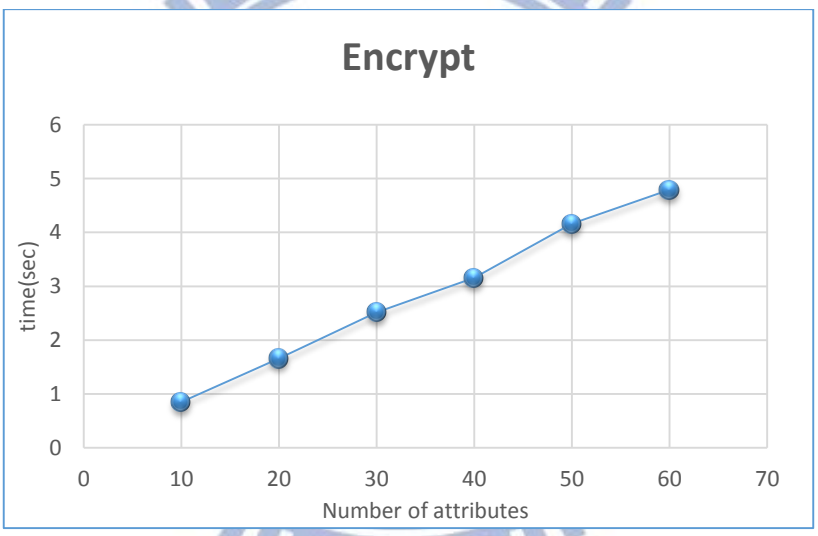


圖 29　加密時間

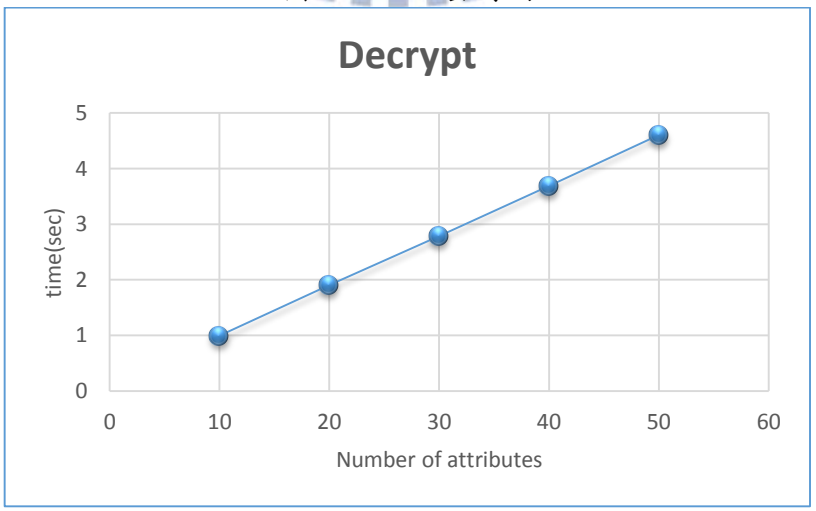


圖 30　解密時間

第七章、結論與未來展望

7.1 結論

本篇論文主要的研究是提出了一個 PHR 系統的框架，在此框架下，我們建構一個以病人為中心的 PHR 系統，病人藉由雲端儲存系統管理與分享其資料，而我們亦利用屬性加密機制-DABE，達到對 PHR 安全的存取控制，保障資料的隱密性、防止被授權者之間的同謀攻擊，讓病人能無後顧之憂地分享他的 PHR，並且讓病人對資料達到細粒度存取控制，也就是讓病人能夠有彈性的根據其需求制訂存取規則以應付不同層級被授權者存取要求。同時我們的系統也是具可拓展性的，允許其他使用者動態加入我們的系統，成為新的授權中心，授權私密金鑰並加密 PHR。最後我們實作出一套 PHR 系統，提供使用者便利的操作介面對其 PHR 進行管理，展示我們所提出的架構是可行的。

7.2 未來展望

本系統雖然支援使用者定義任意 Boolean formula 的存取規則，但是我們所使用的加密機制尚無法支援使用者撤銷(revoke)其金鑰之授權，換言之，假使想要撤銷對於某一被授權者之金鑰授權，目前只能重新產生使用者的公開金鑰與私密金鑰來達成此目的，因此，未來我們可以針對撤銷授權這一部分再做改進，使整個 PHR 系統更趨於完善。

參考文獻

[1] Adi Shamir. "Identity-based cryptosystems and signature schemes." Advances in cryptology. Springer Berlin Heidelberg, 1985.

[2] Amit Sahai and Brent Waters. "Fuzzy identity-based encryption." Advances in Cryptology–EUROCRYPT 2005. Springer Berlin Heidelberg, 2005. 457-473.

[3] Goyal, Vipul, Omkant Pandey, Amit Sahai and Brent Waters. "Attribute-based encryption for fine-grained access control of encrypted data." Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006.

[4] John Bethencourt, Amit Sahai, and Brent Waters. "Ciphertext-policy attribute-based encryption." Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007.

[5] Melissa Chase. "Multi-authority attribute based encryption." Theory of Cryptography. Springer Berlin Heidelberg, 2007. 515-534.

[6] Melissa Chase and Sherman SM Chow. "Improving privacy and security in multi-authority attribute-based encryption." Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009.

[7] Allison Lewko and Brent Waters. "Decentralizing attribute-based encryption." Advances in Cryptology–EUROCRYPT 2011. Springer Berlin Heidelberg, 2011. 568-588.

[8] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, Wenjing Lou. "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption." (2013): 1-1.

[9] Jie Huang, Mohamed Sharaf, and Chin-Tser Huang. "A Hierarchical Framework for Secure and Scalable EHR Sharing and Access Control in Multi-cloud." Parallel Processing Workshops (ICPPW), 2012 41st International Conference on. IEEE, 2012.

[10] Patil, Pooja K., and P. M. Pawar. "PHR Model using Cloud Computing and Attribute based Encryption." International Journal of Computer Applications 65.18 (2013).

[11] Suhair Alshehri, Stanislaw P. Radziszowski, and Rajendra K. Raj. "Secure Access for Healthcare Data in the Cloud Using Ciphertext-Policy Attribute-Based Encryption." Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on. IEEE, 2012.

[12] Changji Wang, Xuan Liu, and Wentao Li. "Implementing a Personal Health Record Cloud Platform Using Ciphertext-Policy Attribute-Based Encryption." Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on. IEEE, 2012.

[13] Mrinmoy Barua, Xiaohui Liang, Rongxing Lu and Xuemin Shen. "Peace: An efficient and secure patient-centric access control scheme for ehealth care system." Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. IEEE, 2011.

[14] Shivaramakrishnan Narayan, Martin Gagné, and Reihaneh Safavi-Naini. "Privacy preserving EHR system using attribute-based infrastructure." Proceedings of the 2010 ACM workshop on Cloud computing security workshop. ACM, 2010.

[15] Luan Ibraimi, Muhammad Asim, and Milan Petkovic. "Secure management of personal health records by applying attribute-based encryption." Wearable Micro and Nano Technologies for Personalized Health (pHealth), 2009 6th International Workshop on. IEEE, 2009.

[16] Wei-Bin Lee and Chien-Ding Lee. "A cryptographic key management solution for HIPAA privacy/security regulations." Information Technology in Biomedicine, IEEE Transactions on 12.1 (2008): 34-41.

[17] Thomas Hupperich, Hans Löhr, Ahmad-Reza Sadeghi and Marcel Winandy. "Flexible patient-controlled security for electronic health records." Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium. ACM, 2012.

[18] Peter Szolovits, Jon Doyle, William J. Long, Isaac Kohane, and Stephen G. Pauker. Guardian angel: patient-centered health information systems. Massachusetts Institute of Technology, Laboratory for Computer Science, 1994.

[19] Kenneth D Mandl, William W Simons, William CR Crawford1, and Jonathan M Abbett. "Indivo: a personally controlled health record for health information exchange and communication." BMC medical informatics and decision making 7.1 (2007): 25.

[20] Google Inc. Google health. https://www.google.com/health/, 2009

[21] Microsoft. Microsoft healthvault. http://www.healthvalut.com/personal/websites-overview.html, 2009

[22] JPBC 函式庫 http://gas.dia.unisa.it/projects/jpbc/index.html

[23] Amos Beimel. Secure schemes for secret sharing and key distribution. Diss. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.