國立交通大學

網路工程研究所

碩 士 論 文

安 全 資 料 庫 之 完 整 性 檢 查 設 計 與 實 現

Secure Database Design and Implementation for Integrity Checking

研 究 生：劉麗君

指導教授：曾文貴　教授

中 華 民 國　102　年　6　月

安全資料庫之完整性檢查設計與實現

Secure Database Design and Implementation for Integrity Checking

研 究 生：劉麗君　　Student：Li-Chun Liu

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network 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 102 年 6 月

安全資料庫之完整性檢查設計與實現

學生：劉麗君　　指導教授：曾文貴 教授

國立交通大學網路工程研究所 碩士班

摘　要

近年來，科技產業或是個人的龐大資料儲存到雲端資料庫的趨勢。資料交給對方管理時，好奇又惡意的資料庫管理者有可能外洩給非法使用者，例如：資料擁有者非授權的使用者或是外部攻擊者。資料儲存到對方控制的環境下，不被任何人偷竊、修改或刪除是我們考慮的主要課題。為了解決這個問題，我們採用了基礎在 SQL 資料庫的 CryptDB 系統，它可以在加密資料上可以執行最基本的 SQL 查詢運算，且也可以執行關鍵字查詢、order preserving 等功能。但是它沒有資料庫安全原理中大家所關注的另一個議題是資料完整性檢查。因此我們想要在 CryptDB 資料庫系統中加上完整性驗證的功能，讓使用者不須花下載資料的傳輸成本，也能按時地驗證自己資料的完整性。這樣一來，CryptDB 系統就能達到資料庫的基本安全原理，包含資料的隱私性、完整性及可得性。並且使用者的方便性為考量，我們透過網頁伺服器設計並實作使用者與代理伺服器之間的介面，讓使用者更加便利的使用 CryptDB 資料庫系統。

Secure Database Design and Implementation for Integrity Checking

Student：Li-Chun Liu Advisors：Dr. Wen-Guey Tzeng

Institute of Network Engineering

National Chiao Tung University

ABSTRACT

For recent years, it has been a trend that large amount of data in personal information or high technology industry are stored in cloud storage. When we outsource our data, curious and malicious database administrator can possibly leak our privacy to illegal parties such as the parties unauthorized by data owners or external adversaries. In view of this, one important issue we concerned in this study is that how we could prevent someone from stealing, modifying, and deleting the data stored in the cloud. To solve this issue, we adopted a system called CryptDB based by SQL databases which is a system that can execute primitive SQL operators on encrypted data, and which can also run the functions such as keyword search, and order preserving. However, CryptDB has no data integrity check and it is also an important issue concerned in database security principles. Therefore, we want to add integrity check on CryptDB and check our data integrity regularly without downloading data to spare communication cost. In this way, CryptDB can follow the database security principles such as data confidentiality, data integrity and data availability. Besides, for user's convenience, the user interface between the user and the MySQL proxy server via web server are also designed and implemented.

誌　　謝

在我的學習生涯中，要感謝的貴人實在是非常多。首先要感謝我的指導教授，曾文貴老師。沒有他我真的無法想像可以完整一篇碩士論文，他給了我論文的方向能夠決定論文題目、耐心的教導我論文相關的研究方法、思考邏輯不正確的時候協助我導正軌道、每個禮拜抽出他寶貴的時間，叮嚀我的進度讓我能夠最有效率地完成我的碩士論文。並且老師也會教導我報告論文的技巧、做人處事的道理。除此之外，讓我參加實驗室研究計畫的過程中，學習到如何規劃一個完整的研究計劃，按照時程的安排逐步完成研究。除了參加計劃以外舉辦一些論文研討會讓我接觸各種不同的研究領域的相關知識。老師讓我吸收到很多研究知識幫助我完成這篇論文。因此我能夠完成這篇論文，曾文貴老師是我最主要感謝的貴人。感謝口試委員，蔡錫鈞教授以及孫宏民教授的細心指導，讓我的碩士論文更加完美。

接著我要感謝實驗室的學長姐們，謝謝毅睿學長除了學術上的教導以外照顧我生活上的點點滴滴。感謝 Vink 學長帶領我完成研究室的計畫，你總是在我苦惱卡關的時候，就會協助我度過難關。謝謝孝盈學姊雖然已經畢業，總會回來參加實驗室的 meeting，給我一些研究上的建議。學長姐們豐富了我兩年的研究所生涯。你們也是我學習生涯中的貴人之一。我也要感謝我的同學們戴靜瑤、謝維揚、劉正偉以及楊其仁。感謝靜瑤在生活上的陪伴和學習上的教導以及建議。稀少女生的資工系裡，有妳的陪伴讓我解除孤單。妳總是聆聽我的訴苦，讓我消除一些課業以及生活上的壓力。謝謝維揚陪我一起修網工所的課，還有對於口試的種種安排，辛苦你了。感謝所有同學都給予我很多口試有關的意見，謝謝你們。我也要感謝我可愛的室友們之之與韵庭，你們都很了解我的心思，讓我可以即時的跟你們分享我的喜怒哀樂，有了你們的陪伴讓我的生活更加的多采多姿。

我要感謝我的家人，我的媽媽以及兩個妹妹。感謝我母親總是支持我所有決定，讓我無憂無慮的在新竹完成這兩年的學業。也感謝我兩個妹妹填補家裡的生活費，讓我放心地完成我的研究生涯。最後我要感謝我在天之靈的爸爸，您是我心靈上的支柱。我一路能夠完成學業，您是最大的功勞，謝謝您。

目　錄

圖　目　錄

第一章、引言

1.1 前言

過去公家機關或是企業利用紙本的方式儲存所有重要的機密資料。隨著資訊科技技術越來越成熟，不管是許多公家機關或是私人公司甚至是單一使用者都認同管理紙本的資料是耗時又耗人力的一大工程。藉用資訊科技的力量，把紙本資料都改為電子化的趨勢。電子資料比較好管理，且資料擁有者方便存取的同時，任何有心人也能透過竊取資料擁有者的帳號及密碼，更容易獲得重要的機密資料。這些機密資料包含了個人資料、電子病歷、企業之間合作的機密資料、甚至是國家機密資料。為了儲存這麼多龐大的資料，必須借用大量的儲存空間。大多數的公家機關、企業、甚至是個人，借用雲端資料庫，例如：Dropbox、ASUS Web Storage、Amazon 等存放這些機密資料。因此儲存著龐大的這些敏感資料的資料庫成為攻擊者的首要目標，攻擊者中包含了惡意的資料庫管理者、資料擁有者非授權的使用者及外部的攻擊者。所以資料庫安全理當是目前大家所關注的主要議題。

資料庫中要保護的除了資料的隱密性以外不被惡意資料庫管理者修改或刪除資料也是非常重要的。我們能夠驗證資料庫中儲存的資料完整性重要之餘，有授權的合法使用者才能存取資料也是我們要考量的另一個課題。因此資料庫安全原理中包含了資料隱密性、資料完整性及資料可得性。

- **資料隱密性(Data Confidentiality)**

  紙本資料電子化使儲存在資料庫的敏感性資料越來越增加。如果資料庫管理者是惡意的情況之下，資料隱密性是我們大家所關心的議題。敏感性資料分為重要機密資料(例如：公司之間合作案、公司最新開發的產品等)及個人隱私資料。這些敏感性資料被有心人獲取時，像是個人的資料被有心人拿去賣，例如：賣給補習班以便招生等。如果被偷竊的資料是國家機密，後果更是不堪設想的。惡意資料庫管理者擁有最高權限(讀取、插入、更新、刪除)的情況下，敏感資料的隱密性成為我們要首要研究的課題。

- **資料完整性(Data Integrity)**

  資料的隱密性重要之餘，是否能夠驗證資料的完整性也是大家所關注的另一個

安全問題。敏感資料不在資料擁有者自己管理的情況下，可能會被惡意資料庫管理者修改。惡意資料庫管理者為了節省資料庫的空間，有可能只存一些部分資料，像是常被資料擁有者查詢的資料，而有些不常被查詢的資料從資料庫中刪除時，我們希望能夠驗證資料的完整性。因此資料庫安全中不定時的檢查資料完整性也是我們要關心的另一個議題。

- **資料可得性(Data Availability)**

  只有授權的合法使用者才能以明文方式存取資料庫中儲存的資料，是資料庫安全問題中我們要研究的另一個資料可得性問題。被授權的合法使用者包含了資料擁有者及他所授權的合法使用者。我們要防範非授權者不能以明文方式存取使用者資料，以便保護使用者的隱私資料外洩。如果資料存取權最大的惡意資料庫管理者也被列為非授權者時，使用者資料的保護是更加的重要。因此資料可得性成為資料庫的安全原理之一。

為了處理這些安全問題，過去的許多研究者提出了不少技術、協定及方案。使用加密技術是最基本，也是最簡單的方式解決資料隱密性的安全問題。最基本的加密技術分為對稱式與非對稱式加密。對稱式加密的優點是加密速度快，但是使用者之間傳輸加解密金鑰的過程中，不被攻擊者攔截此金鑰是不容易防範的事。非對稱式加密技術的優點是使用者之間不用傳輸解密的金鑰，使用者一方只要把想分享的資訊，利用對方的公開金鑰加密傳送給對方，對方就能用自己的私密金鑰解開此密文。但是非對稱式系統的加解密速度慢，不適合使用在大量資料要加密的應用方面，例如：資料庫中龐大的資料加密。過去的研究中，提出了許多對稱式加密技術，例如：Blowfish、DES、AES 等。其中 AES 加密技術是目前被大家所認可的最先進、最快速且安全性最高的加密技術。因此加解密大量資料的應用方面最適合使用此技術。為了加強資料的安全性，使用者把資料傳送到資料庫之前必須事先加密後才能儲存到資料庫。雖然加密技術可以解決資料隱私性的問題，但加密過程中，基本的計算成本會增加使用者的負擔。並且使用加密技術的同時，使用者也必須管理加密資料過程中參與的金鑰。因此計算成本以及管理金鑰所導致的成本會加諸在使用者身上，且使用者也必須防範傳輸對稱式金鑰的過程中，被攻擊者攔截金鑰的問題。

除了想要保護資料的隱密性之外，使用雲端儲存空間的使用者們關注的另一個議題是能夠不定時地驗證資料是否被修改或是刪除。為了想要節省一點儲存空間成本，惡意資料庫管理者有可能會刪除掉使用者極少存取的部分資料。因此使用者把自己資料給對

方管理的情況下，按時地檢查資料的完整性成為大家所關注的另一個議題。過去的許多研究中，資料的完整性檢查協定都應用在檔案系統上面[4][5][6][7][8][9][10][11]。使用者透過資料傳輸的協定 (FTP：File Transfer Protocol)或是安全殼協定方式(SSH：Secure Shell)與遠端檔案系統溝通，且把資料以檔案方式儲存在檔案系統上。檔案系統的資料之間是沒有關聯性的，比較難在資料之間做運算，例如：資料表之間的相等比對(equality checks)、順序比對(order comparisons)、交集(joins)或是集合(sums)等。相對地，關聯式資料庫系統中可以利用 SQL 查詢運算資料之間的關係。因此我們希望能夠在關聯式資料庫系統上套用完整性檢查協定，可以同時儲存幾個欄位並同時檢查所有欄位中的資料完整性，且同時解決 SQL 查詢語法處理方面的問題。

關聯式資料庫的完整性檢查分為語意上面的完整性[13][14]及使用者儲存的資料本身的完整性。語意上的資料完整性檢查是使用者更新資料後，是否符合資料欄位的定義，例如：員工的薪水欄位中，儲存的資料應該是整數，且大於零、使用者輸入員工名稱的欄位時，應是字串而不是數字型態等，也就是檢查建立資料表時規定的資料型態及使用者儲存到資料庫的資料型態是否是一致的。目前許多研究中，關聯式資料庫的完整性檢查為此類的完整性驗證，這樣的驗證稱為語意上的資料完整性檢查。因此我們希望能夠在以 SQL 語法查詢的關聯式資料庫系統上實作使用者儲存到資料庫的資料本身完整性驗證功能。

最直接能保證資料完整性驗證的方法就是利用一套私密金鑰，事先運算好資料的認證碼(Hash or MACs)後把它們存在使用者端，每次驗證的過程中，洩漏一把私密金鑰給伺服端，讓它能夠驗證資料的完整性，但這方法的驗證次數有限，當私密金鑰用完後使用者必須要重新取回資料，再重新的計算一套 MACs [7]。這樣的方法雖然方便以及簡單，使用者必須要管理一套私密金鑰外，實際狀況上效率會不佳。因此為了能夠有效地檢查資料的完整性，不能利用重新取回使用者的原始資料的方式檢查資料的完整性。我們必須要使用完整的協定才能確保惡意資料庫管理者確實完整的儲存使用者的資料。

過去的研究中，許多學者利用不同的完整性檢查協定的演算法(例如：Provable Data Possession (PDP)、Merkle Hash Tree、Algebraic Signatures 等)提出了完整性檢查的技術。Merkle Hash Tree 的方式驗證資料的完整性是以樹狀的方式維持並儲存使用者的資料結構，並且最後必須驗證資料結構的樹根才能確保資料的完整性。雖然大多數的學者利用此方式提出了不少資料完整性的協定，但像資料庫系統龐大的資料量的狀況之下，要以樹狀方式維持資料結構是比較困難的。並且為了能夠驗證樹根，驗證資料完整性的過程

中，除了要傳送檢查資料的認證碼以外必須一併傳送與要檢查資料認證碼相鄰的其他認證碼。因此除了維持資料結構的困難度以外必須考量到傳輸成本的問題。

Lanxiang Chen[8]提出以代數簽章為基礎的資料完整性檢查協定。代數簽章為基礎的資料完整性檢查協定中，使用者只要儲存兩把私密金鑰及一些亂數值，不需要取回原始資料就能有效的驗證資料的完整性。因此代數簽章的優點是低網路頻寬、合理的計算成本及能抵擋惡意修改使用者資料。並且 challenge 次數越多越能抵擋 collision attacks 而越安全。但是以代數簽章方式驗證資料完整性的缺點在於機率上的安全(probabilistic security)，而非決定性的保證(deterministic guarantee)。雖然近年來不少學者提出了許多完整性檢查的協定，但 2007 年 Giuseppe Ateniese et al.[4]提出的 Provable Data Possession (PDP)協定是最早提出且大家認同較完整的協定，儲存成本、計算成本及通訊成本上較有效。因此我們希望採用 Provable Data Possession 的協定方式驗證資料的完整性。

為了增加儲存資料的空間，使用者把自己的資料儲存在雲端儲存空間上。但由於保護資料的隱密性，使用者要負擔許多計算成本及傳輸成本。為了節省儲存空間，使用者必須負擔其他的成本。因此為了降低使用者所承受的負擔及防範傳輸對稱式金鑰被有心人攔截的問題，最好的解決方法就是加密與完整性檢查的計算交給一個可信任的第三方認證者。第三方認證者可以為使用者分擔許多計算成本，例如：資料的加解密，認證碼(integrity tag)及證明碼(proof)的產生等，且可以減輕使用者金鑰管理的負擔，例如：系統的建造(set up)，金鑰的產生(Key Generation)與金鑰管理(Key Management) 等。除此之外，驗證資料完整性的過程中，也可以降低使用者要傳送給伺服端一些參數的傳輸成本。

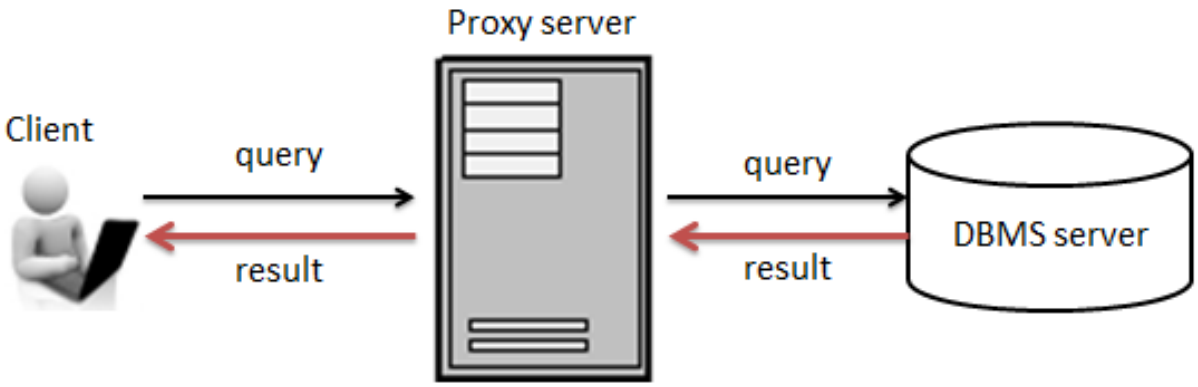


圖 1 Database with proxy server

圖 1 顯示使用者透過代理伺服器與資料庫溝通的最基本架構。透過代理伺服器，使用者想要查詢的資料相關資訊封包傳送給資料庫，而資料庫針對相關資訊，查詢得到的結果也是透過代理伺服器來回傳給使用者。Raluca Ada Popa et al.[1]提出的關聯式資料庫系統(CryptDB)中，MySQL 代理伺服器作為可信任的第三方，利用代理伺服器攔截 SQL 查詢封包、運算加密使用者資料以及改寫封包內容。建立自訂的 C++ library，以便產生加解密運算以及改寫後的 SQL 查詢指令，最後把 SQL 查詢封包中的內容傳送到資料庫。因此我們希望基礎在關聯式資料庫系統上的 CryptDB 系統來實作完整性驗證功能。使用者想要更新或是刪除資料時，只要用一般關聯式資料庫使用的 SQL 語法查詢，代理伺服器會幫使用者插入、更新以及刪除資料。

1.2 研究目標

過去的許多研究中，提出了關聯式資料庫的完整性檢查，但只有檢查語意上的完整性，而非使用者儲存的資料本身的完整性。Raluca Ada Popa et al.[1][2][3]利用許多加密技術 (Blowfish、AES in CBC mode、AES in CMC mode、Paillier 等) 設計與實作安全的關聯式資料庫系統。雖然他所提出了“onions of encryption”資料加密技術讓資料庫更加安全，但無保證資料完整性。並且雖然[1][2][3]中提出了 SQL 查詢資料協定，但利用多層的加密方式，加密資料表及資料欄位名稱，導致代理伺服器斷線後重新開啟時，每個使用者必須重新建立資料庫中的所有資料表才能取回想要查詢的資料。並且我們認為資料表及資料欄位名稱的洩漏，不會影響到保護資料內容的隱密性。不加密資料表及欄位名稱的狀況之下，使用者要查詢自己的資料時，更加便利。假設代理伺服器斷線後，使用者想要存取自己的資料時，不需要重新建立所有的資料表才能找出要查詢的資料，只要用資料表名稱直接建立一次就能查詢資料。由於 keyword search 的處理，CryptDB 系統的資料欄位中每 15bytes 必須利用定義符號(delimiter)隔開才能儲存到資料庫。因此我們希望把 CryptDB 系統的“onions of encryption”加密技術修改成簡單的 AES 加密，只要加密使用者要保護隱私的資料內容，不需要多層式的加密及金鑰管理的方式，實作資料庫的安全，並且代理伺服器作為使用者的第三方認證者，加解密使用者的資料及完整性檢查的所有運算，減輕使用者的負擔。除此之外，我們也考量到使用者的方便性，Web server 作為使用者介面，使得使用者更加便利地使用此系統。

1.3 貢獻度

我們在 CryptDB 系統中加上資料完整性檢查功能，系統能夠達到資料庫必備的安全原理包含資料隱密性、資料完整性以及資料可得性，讓 CryptDB 系統更加的完善。

- 由於 CryptDB 系統的資料欄位本身不是儲存整份完整的資料而為了儲存 keyword，處理文字時每 15bytes 長度要用定義符號隔開，因此把 CryptDB 系統的“onions of encryption”加密技術修改成簡單的 AES 加密，並且移除 keyword search、order preserving 以及 add 功能，可以在資料庫中能夠儲存完整的資料。
- 除了 select 查詢指令的 functions 之外，update 以及 delete 等 SQL 查詢指令的 functions 中加上資料的完整性驗證功能，達到更新以及刪除資料。
- 雖然使用者非常關注自己資料的安全性，但是使用上的方便性也是不可或缺的。因此我們把 SQL 查詢指令寫成網頁應用程式，Web server 作為與資料庫溝通的使用者介面，達到使用者的方便性。

1.4 全文架構

本文分為七個章節，分別是引言、相關研究、Provable data Possession 的介紹、系統設計、系統實作、比較與結語。第一章節前言、說明研究動機與目的、貢獻度以及本文架構。第二章節介紹相關研究的優缺點。第三章節會詳細介紹 Provable Data Possession 協定的運作方式。第四章節為系統設計的部分，其中介紹系統架構，資料庫認知的 SQL 語法讀與寫執行。第五章節是系統實作的介紹，終端機上及代理伺服器上發生動作的過程中，記錄下來的畫面擷取，且使用者介面上的操作畫面顯示。第六章節為我們修改 CryptDB 後的系統與 CryptDB 系統之間加密方式的比較。第七個章節是針對我們實作的完整性驗證功能做個簡單的結論，並提出未來系統的延伸及可能性。

第二章、相關研究

過去研究中，許多研究者提出不同的資料完整性檢查協定，例如：PDP、POR 等。Giuseppe Ateniese et al.[4]提出的 PDP 協定是最先提出並較完整的資料完整性檢查協定。PDP 為基礎的完整性檢查協定中通常有五個階段。

第一階段：使用者產生完整性檢查過程中會參與的金鑰，稱為 setup。

第二階段：資料傳送給雲端資料庫之前，先產生認證碼 (integrity tag)，且資料與認證碼一起傳送給雲端資料庫。

第三階段：使用者想要驗證資料完整性時，亂數產生或是計算一些值，傳送要驗證資料的相關資訊，例如：資料區塊總個數、要驗證的資料區塊個數、挑選哪幾個索引等傳送 challenge 給雲端資料庫。

第四階段：雲端資料庫接收到使用者的 challenge 後利用 challenge 中的參數、儲存在本地端的使用者的資料以及認證碼計算證明碼 (proof)，再傳送給使用者。

第五階段：使用者得到證明碼後利用之前傳給雲端資料庫的 challenge 值及證明碼，驗證雲端資料庫上儲存的資料是否完整。

雖然 Giuseppe Ateniese et al.[4]提出的 PDP 協定可以追加動態的完整性驗證，他只提出了靜態資料的完整性檢查。因此 C. Chris Erway et al.[5] 提出了以 skip list 為基礎的動態完整性驗證協定包含資料插入、更新、刪除等的運算。為了計算及傳輸複雜度降低到常數時間，Feifei Liu et al.[6]提出比較有效的動態更新資料完整性檢查協定，改善[5]。

Qian Wang et al.[7]提出了一個 Merkle Hash Tree 為基礎的完整性檢查協定。雖然此協定可以提升資料庫安全性，必須要維護 Merkle Hash Tree 的資料結構及儲存大量的資料區塊的 Hash。圖 2 顯示一個檔案切成 n 個區塊後，資料庫中除了要儲存 n 個資料區塊(Data Block)及它們的 hash 值(Hash of Data Block)以外需要儲存個別 hash 值延升出來的 hash 值(Redundant Storage of Hash Block)，共有(n-1)個 hash。並且完整性檢查的過程中，除了要傳送檢查資料的 hash 以外與它相鄰的 hash 以及所有與此資料有關的路徑上的 hash 給使用者，以便能夠檢查 hash Root 的正確性。這會讓運算成本增加以外傳輸成本也會跟著增加。如果此方法使用在龐大的資料(例如：雲端儲存空間上儲存的資料、資料庫中的資料)完整性檢查時，除了儲存成本增加以外計算及傳輸成本也會越來越可觀。

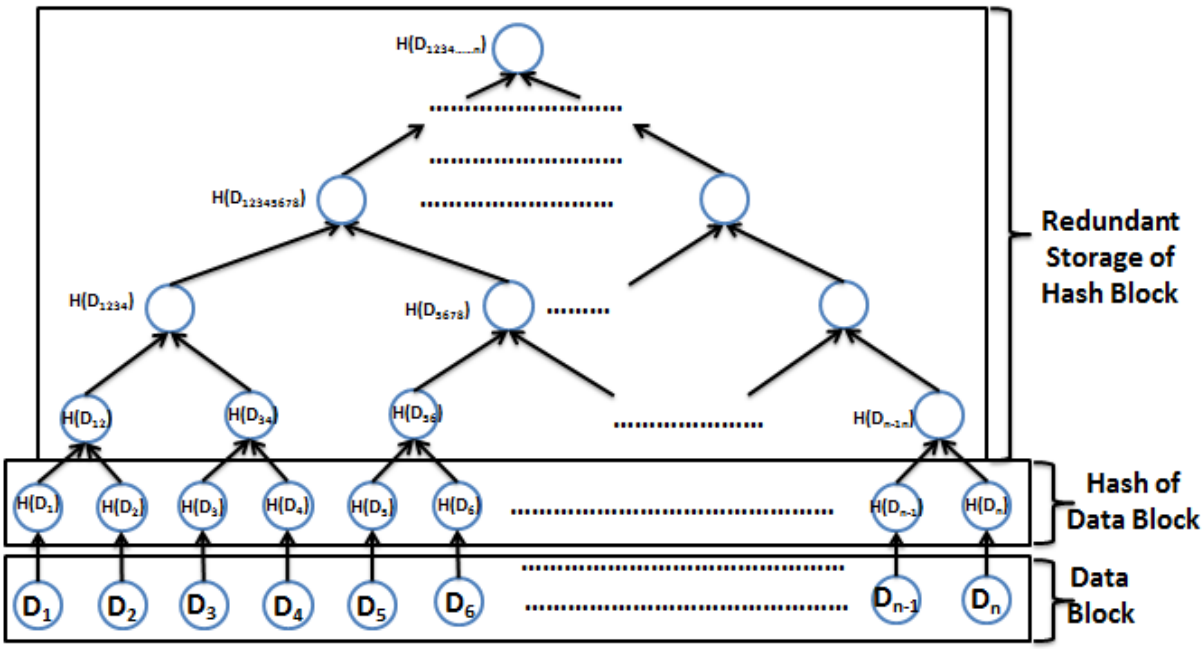
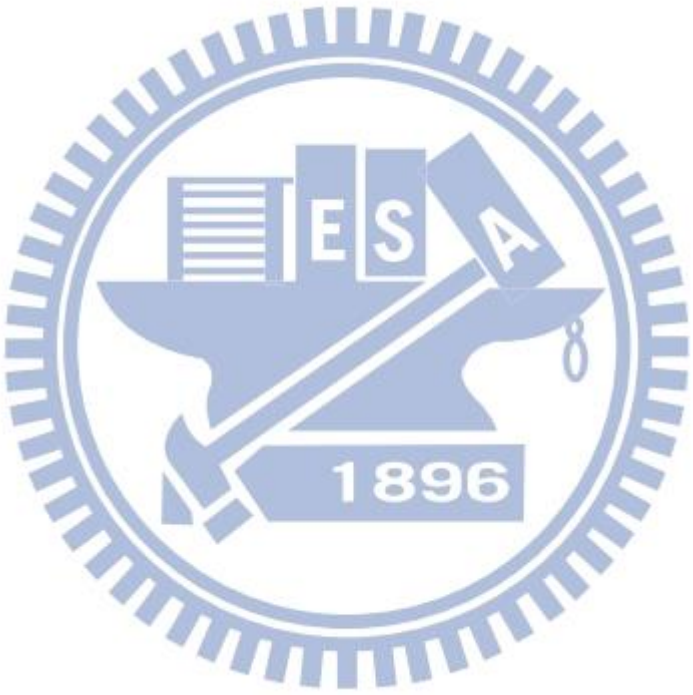


圖 2 Merkle Hash Tree

Zhuo Hao et al.[10]提出的完整性驗證協定也是基礎在 PDP 的協定。計算認證碼的建造方式參考於[4]的方法。S-PDP[4]提出的使用者自己執行的完整性驗證。為了減輕使用者負擔，Zhuo Hao [10]提出交給第三方驗證資料的完整性，欲保證資料的隱私性(privacy preserving)，也就是資料交給第三方驗證完整性時，資料擁有者不需要擔心洩漏資料的隱私。任何第三方驗證者都可以利用公開資訊幫資料擁有者驗證資料的完整性。並且可以動態的更新(Dynamic Data)資料的功能，包含插入、更新及刪除資料等。安全分析也是與[4]的安全性證明一樣基礎在 KEA1- assumption 上面。Syam Kumar P et al.[11]提出的完整性檢查協定的安全性建立在橢圓曲線密碼及 Sobol Sequence(亂數取樣)上面。雖然橢圓曲線密碼只用了較小的金鑰長度就有 RSA 一樣的安全性，且可以應用在計算能力有限的裝置上，而 PDP 的安全程度建立在指數運算的難度上，此安全性等於橢圓曲線密碼上點與點之間的相乘。因此這篇提出的方法與 PDP 協定安全等級一樣，另一種表示方式而已。Hamidah Ibrahim[13]以及 Ali Amer Alwan et al.[14]提出的資料完整性檢查在於關聯式資料庫中，更新資料後檢查是否符合資料欄位規定的型態，而非檢查資料本身的完整性驗證。

Raluca Ada Popa et al.[3]提出安全的關聯式資料庫上 SQL 查詢，包含相等方式比對(DET)、order preserving encryption (SORT, MAX, MIN)、整數資料的加總查詢(SUM)等的查詢，且也提出了關鍵字查詢(Keyword Search)。藉用 SQL 代理伺服器的計算能力，

改寫 SQL 查詢封包內容與資料庫溝通。為了保護使用者資料的安全，提出了新的加密技術“onions of encryption ”。利用多層的加密方式，資料庫中儲存的資料表及資料欄位的名稱與資料一併加密，資料表以及資料名稱重新命名成非資料擁有者猜測不出資料欄位中儲存的資料內容資訊。並且每一種 SQL 查詢層利用不同的加密技術，包含 AES-CBC mode 加密安全性最高的外層、整數資料的加總層利用 paillier 技術加密、相等方式比對(DET)資料層卻利用 AES-CMC mode 的加密技術以及利用 Song et al.[19]提出的加密技術實作關鍵字查詢(Keyword Search)等。雖然此系統利用各種不同的加密技術達到許多功能，卻沒提供資料庫安全原理中最重要的一項，資料完整性檢查功能。



第三章、Provable Data Possession

PDP 是一個不需要從遠端儲存空間取回使用者的原始資料就能檢查使用者資料完整性的模型[4]。由於這個特點，大量資料儲存在遠端伺服器時，可以降低整個檔案從遠端伺服器存取的 I/O 成本，且也可以節省整個檔案的傳輸成本。因此這模型非常適合使用在大量儲存著資料的伺服器(例如：網路檔案系統、資料庫系統等)。儲存成本上面，客戶端只要儲存少量 O(1)的資訊(例如：私密金鑰及資料有關的認證碼(tag))就能驗證資料的完整性。

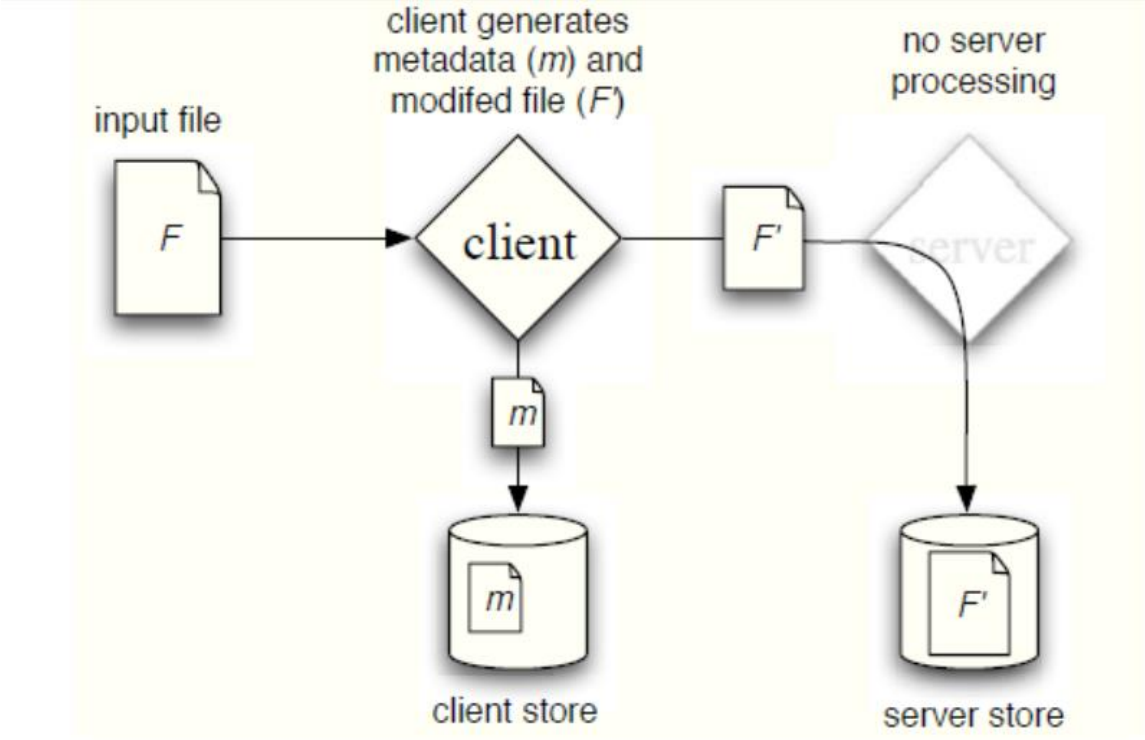
圖 3 顯示最基本的 PDP 協定。首先客戶端要儲存資料到遠端伺服器之前做事先處理，利用使用者的公開金鑰及私密金鑰，計算與資料相關的資訊(metadata 或是 tag)，把 metadata 存在本地端後把資料儲存到遠端伺服器，之後把本地端的原始資料刪除，也可以擴張成 metadata 及資料一起儲存到遠端伺服器[4] (如圖 3(a)所示)。驗證完整性的過程中，客戶端產生並發布 challenge 給伺服端，要求伺服端利用之前所存的資料計算出證明碼(proof)回傳，客戶端利用本地端儲存的 metadata，驗證伺服端的回應[4](圖 3(b) 所示)。過去許多研究者提出了不同的 PDP 協定[4][5][6][7][8][10]，其中 2007 年 Giuseppe Ateniese et al.[4]提出的兩個 scheme，包含 S-PDP(strong data possession guarantee)及 E-PDP(better efficiency PDP)是目前較完整又有效率的 PDP 協定。因此接著介紹 PDP 協定的定義以及建造以[4]提出的兩個 scheme 為主。

3.1 Definition of PDP schemes

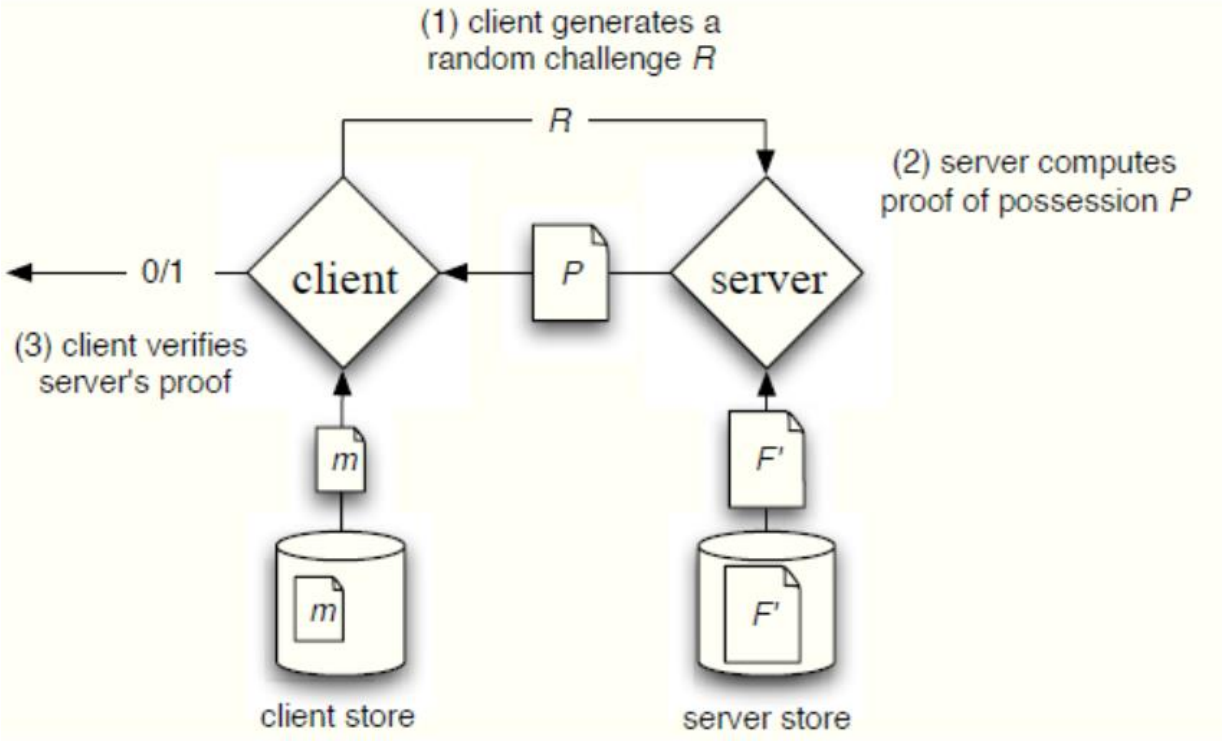
介紹 PDP scheme 的安全定義之前使用者把要儲存到遠端伺服器的檔案切成 n 個區塊：$F = (m_1,\ldots,m_n)$。PDP scheme 以(KeyGen, TagBlock, GenProof, CheckProof)四個演算法組成。

KeyGen(1^k)：使用者為了建立 scheme，利用此演算法(probabilistic key generation)產生一組金鑰。使用者先產生一個安全性參數 k，把 k 輸入到演算法後輸出一組公開金鑰以及私密金鑰(pk, sk)。

TagBlock(pk, sk, m)：輸入公鑰、私鑰及要儲存到遠端伺服器的資料給此演算法後產生



(a) Pre-process and store



(b) Verify server possession

圖 3 Protocol for provable data possession

(資料來源:[4])

驗證時要使用的 metadata(T_m)。T_m 是以($T_{m1},\ldots T_{mn}$)組合而成。每個區塊的 metadata 符合 homomophic 的性質，以便驗證過程中，伺服端根據使用者要求的區塊索引，組成後算出證明碼(proof)。

$$\text{Homomophic tags: } T_{m_i+m_j} = T_{m_i} + T_{m_j} \quad \ldots\ldots [1]$$

GenProof(pk, F, chal, Σ)：遠端伺服器執行 GenProof 演算法後產生證明碼(proof)。此演算法的輸入為公鑰、要驗證的資料、使用者傳來的 challenge 及根據區塊索引組成的 metadata 集合，輸出為證明碼(proof)。

CheckProof(pk, sk, chal, V)：使用者執行 CheckProof 演算法來驗證遠端伺服器回傳回來的 proof 有效性。

3.2 Construction for PDP schemes

PDP 系統以兩個部份來建造，分別為 Setup 及 Challenge。執行 KeyGen 及 TagBlock 演算法來實作 Setup 的部分，而執行 GenProof 及 CheckProof 演算法來完成 Challenge 部分。首先介紹 S-PDP scheme 的建造細節。

- Preliminaries

  建造時會用到的一些符號，假設 p=2p'+1 及 q=2q'+1 都是安全的大質數，N=pq 為 RSA modulus，g 為 QR_N(Quadratic Residue)的 generator，也就是 order 為 p'q'(Z_N^* 的循環子群)，g 的值是 a^2 而得，其中 a 是從 Z_N^* 中隨機選擇而得(gcd(a±1, N))，再選擇安全參數 k, l, λ，其中 λ 是正整數，H 為 cryptographic hash function，(PRF) f 為 pseudorandom function 及(PRP) π 為 pseudorandom permutation[4]。

  $$f: \{0,1\}^k \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^l \quad \ldots\ldots [2]$$

  $$\pi: \{0,1\}^k \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^{\log_2(n)} \quad \ldots\ldots [3]$$

  公式[2]中 k 為輸入值的長度，而 l 是 pseudorandom function 中輸出值的長度。因此 $f_k(x)$函式中，長度為 k 的輸入值 x(0 和 1 組成的值)輸入到 pseudorandom function 後輸出為長度為 l 的值(0 和 1 組成的值)。公式[3]中輸入值為 k 長度的 0 和 1 組成的值，而 $\log_2$(n)是 pseudorandom permutation 中輸出值的長度。因此π_k(x)函式中，

長度為 k 的輸入值 x(0 和 1 組成的值)輸入到 pseudorandom permutation 後輸出為長度為 $\log_2$(n)的值(0 和 1 組成的值)。

- Setup

KeyGen(1^k)：產生一組公鑰 pk = (N, g)以及私鑰 sk = (e, d, v)，其中 e 與 d 要符合($ed \equiv 1 (\mathrm{mod} p'q')$)以外這兩個數的值都要大於 λ，且 e 是很大的秘密質數，g 為 QR_N(Quadratic Residue)的 generator，v 是長度為 k 的 0 和 1 組成的字串，其中 k 為安全參數。最後會輸出公私鑰一組(pk, sk)。

TagBlock(pk, sk, m, i)：使用者利用(N, g)公鑰、(d, v)私鑰、切成 n 個區塊的資料(m)及區塊索引(i)為輸入值來產生資料有關的 metadata。先計算 W_i=v||i，其中(|| 是串聯，$1 \le i \le n$，n 為資料的區塊總個數)，再計算出認證碼。

$$T_{i,m} = (h(W_i).g^m)^d \bmod N \quad \text{.....[4]}$$

最後輸出 (T_m, W)，其中 $T_m = (T_{1,m},...,T_{n,m})$ 、 $W = (W_1,...,W_n)$ 。

- Challenge

GenProof(pk, F, chal, Σ)：使用者為了要求 proof，產生 (c, k_1, k_2, g_s)為 chal，其中 c 為要檢查的區塊個數(隨機選擇、範圍是 1 到 n 之間)，k_1 與 k_2 是長度為 k 的 0 和 1 組成的隨機值、s 為從 Z_N^* 中隨機選擇後計算 g_s 等於 g^s，此 chal 與 (N, g) 公鑰傳送給遠端伺服器。伺服器收到後利用之前儲存的資料區塊及 metadata 計算出 proof。利用 c 值計算出要檢查的區塊索引 $i_j = \pi_{k_1}(j)$，再計算係數值 $a_{i_j} = f_{k_2}(j)$ ，其中 j 的範圍是從 1 到 c 之間。最後計算出 proof。

$$T = T_{i_1,m_{i_1}}^{a_1} .....T_{i_c,m_{i_c}}^{a_c} = (h(W_{i_1})^{a_1} .....h(W_{i_c})^{a_c} .g^{a_1 m_{i_1} + ... + a_c m_{i_c}})^d \bmod N \quad \text{...[5]}$$

$$\rho = H(g_s^{a_1 m_{i_1} + ... + a_c m_{i_c}} \bmod N) \quad \text{.....[6]}$$

最後輸出為 V=(T, ρ)。

CheckProof(pk, sk, chal, V)：使用者收到 proof 後利用(N, g)公鑰、(e, v)私鑰、(c, k_1, k_2, s)為 chal，計算出 $\tau = T^e$、$i_j = \pi_{k_1}(j)$、$W_{ij} = v||i_j$ 及 $a_{i_j} = f_{k_2}(j)$，最後計算以下的值來檢查資料完整性。

$$\tau = \frac{\tau}{h(W_{i_j})^{a_j}} \bmod N \quad \text{.....[7]}$$

如果 $H(\tau^s \bmod N)$ 等於ρ時，表示儲存在資料庫的使用者資料是完整的，否則資料已被修改。[4]提出的第二個 scheme 是 E-PDP，此 scheme 與 S-PDP 的差別在於執行完整性檢查時，此 scheme 效率比較高，但是要犧牲安全性。因為隨機只檢查一個資料區塊，而不檢查 c 個區塊，也就是係數 a_{ij} 為 1，但如果伺服器只要事先計算好 n 個資料區塊中，c 個區塊的各式各樣的組合，伺服器就能通過所有的 challenge，因此使用者以及遠端伺服器的計算都降為只要計算一個次方，但事實上此方法不太安全。[4]提出 PDP scheme 的詳細安全性分析及證明。

第四章、系統設計

4.1 系統架構

圖 4 顯示 CryptDB 系統架構，其中有四個角色，包含客戶端、應用伺服器、代理伺服器及一般關聯式資料庫。比較於一般 client and server 系統，CryptDB 系統多增加了兩個可信任的第三方認證者，分別是 application server 及 MySQL proxy server。利用 application server，使用者與 MySQL proxy server 之間溝通。並且使用者及 SQL 資料庫之間增加了 MySQL proxy server，可以減輕使用者的計算及傳輸負擔。

- Application server

  Application server 是指終端機，使用者只要在終端機下指令的方式，與資料庫的代理伺服器連線，以便可以建立資料庫、資料表以及查詢資料，包含插入、更新、刪除等。輸入要儲存到資料庫的資料時，使用者的資料會被傳送到代理伺服器。代理伺服器會幫使用者處理資料後上傳到資料庫。之後使用者想要從資料庫取回自己的資料時，代理伺服器解密從資料庫取回的資料，且在終端機顯示使用者資料。使用者想要更新或是刪除已經插入到資料庫的資料時，利用 SQL 查詢指令的功能，可以更新及刪除資料。

- MySQL proxy server

  為了能夠避免惡意的資料庫管理者以明文方式看到使用者的隱私資料並外洩給非法使用者，代理伺服器接收透過終端機傳來的 SQL 查詢指令攔截下來後，檢查哪一個欄位的資料需要特殊處理。使用者建立資料表到資料庫的同時，就要決定資料表的哪些欄位資料內容要做特殊處理。建立資料表的 SQL 查詢指令中，檢查到要特殊處理的指令，例如：create table book (id integer, name enc text)，此建立資料表指令的 name 欄位中，找到 enc(encryption) 時，代表 book 資料表中的 name 資料欄位之後要做特殊處理。代理伺服器會 name 欄位分別建立成 keyword search、order preserving 以及 add 三個資料欄位。代理伺服器會把之後使用者插入到 book 資料表的 name 欄位內容分別用不同加密方式加密後分別儲存到三個 keyword search、order preserving 以及 add 欄位。相對地，如果建立以明文方式儲存的資料時，SQL 查詢指令的任何欄位中不會有 enc 指令的出現。

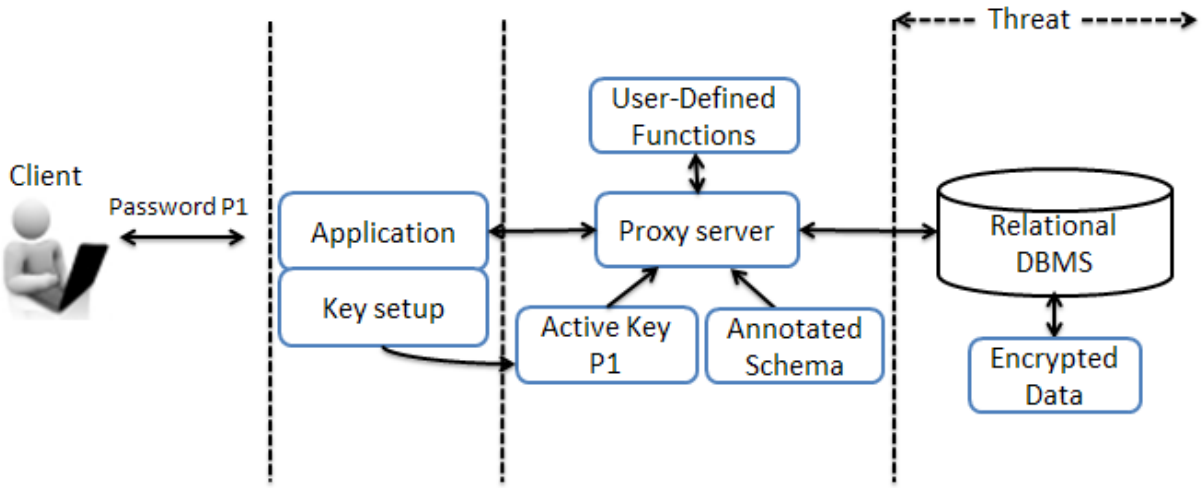


圖 4 系統架構

代理伺服器接收到 SQL 查詢指令後利用 Annotated Schema 中自訂的 C++library 加密資料。並且利用 User-Defined Functions，把 SQL 查詢指令改寫成自訂的欄位格式後傳送資料到資料庫。使用者要查詢資料時，代理伺服器利用 Annotated Schema 中自訂的 C++library 解密資料之後改寫使用者傳送的 SQL 查詢指令，最後把解密後的資料傳送給使用者。由於 CryptDB 系統處理文字資料型態時，因資料欄位中只儲存 keyword 而限制 keyword 長度為 15bytes。每 15bytes 的資料長度利用 delimiter 隔開後才能把文字資料儲存到資料庫。我們想要在 Annotated Schema 中加上資料完整性檢查功能，必須要能夠儲存完整的整份資料而不是文件中的資料被 delimiter 隔開。因此我們從 Annotated Schema 中，移除 keyword search、order preserving 以及 add 功能，達到能夠儲存完整的整份資料，且加上完整性驗證的功能達到資料完整性驗證的功能。為了區分 CryptDB 系統與我們修改後的差別，需要特殊處理的欄位中的指令 enc(encryption)改成 ic(integrity check)。

系統架構中，介紹惡意資料庫管理者外洩資料給其他非授權者的威脅，代理伺服器從應用伺服器接收到的 SQL 查詢指令改寫，也就是加密使用者想要保護隱密資料的內容。CryptDB 系統保證，資料庫管理者不會得到加密資料的金鑰。因此 CryptDB 系統可以防範好奇又資料庫存取權限最高的資料庫管理者或是外部攻擊者不會以明文方式獲得敏感性資料，也確保使用者的隱私資料不會外洩給任何人。我們的目標要達到資料的隱密性以外資料的完整性以及資料可得性。代理伺服器接收到插入、或是更新等的 SQL 查詢指令時，利用自己的私密金鑰加密使用者資料，並且改寫成加密後自訂的資料欄位格式寫入資料庫。此方法允許像明文資料庫系統上做 SQL 查詢一樣，可以做加密資料

的 SQL 查詢。由於 CryptDB 系統想要處理多種不同功能，因此作者提出了新加密資料方式(onions of encryption)。為了達到更高的安全性，此加密方式除了加密要特殊處理的資料內容以外也一併加密了資料表以及資料欄位名稱。因此會出現了幾個問題，第七章節特別介紹會出現的兩個問題。因此 onions of encryption 修改成簡單的 AES 加密方式。雖然我們的加密資料方式不會隱藏資料表名稱、資料欄位名稱、資料表的結構、資料筆數、資料欄位的型態或是資料的大小，但可以保護到使用者要隱藏的資料內容的隱密性，也就是使用者的敏感性資料從不以明文方式出現在資料庫。

4.1.1 前置作業

使用 CryptDB 系統時，與一般系統一樣使用者事先註冊到資料庫成為合法使用者，之後使用此帳號與密碼登入到資料庫。由於我們為使用者設計了使用者介面，因此登入系統時，可以選擇兩種方式登入，包含終端機下指令或利用使用者介面的方式。如果使用終端機下指令時，CryptDB 系統預設 port 為 3307 來導向到代理伺服器，例如：

mysql -u lichun -ppassword -h 140.113.207.138 -P 3307 cryptdbtest

其中-u 代表 user account，後面輸入的 lichun 為使用者帳號、-p 代表 user password，後面輸入的 password 為使用者密碼、-h 代表 host(要連線的資料庫 IP)，後面輸入的 140.113.207.138 為要連線的資料庫 IP、-P 代表 Port，3307 為代理伺服器的 port(資料庫的 port 預設為 3306)、最後 cryptdbtest 為要連線的資料庫名稱)，透過這樣的方式，使用者透過代理伺服器與資料庫溝通。如果透過使用者介面登入時，我們幫使用者設計好介面，透過代理伺服器與資料庫連線。因此使用者介面可以讓一般不熟悉使用終端機下指令的使用者也能夠方便地管理自己的資料。

4.1.2 資料完整性驗證的系統流程

圖 5 顯示我們在 CryptDB 系統上提出的資料庫完整性驗證的系統流程，以使用者介面作為介紹。使用者登入之後可以安全的上傳，下載、檢查資料的完整性、更新以及刪除資料。架構中共有六個階段。

第一階段：使用者透過 Web server 上傳資料到代理伺服器。

第二階段：代理伺服器從 Web server 接收到要上傳資料時，當初建立要上傳資料的資料

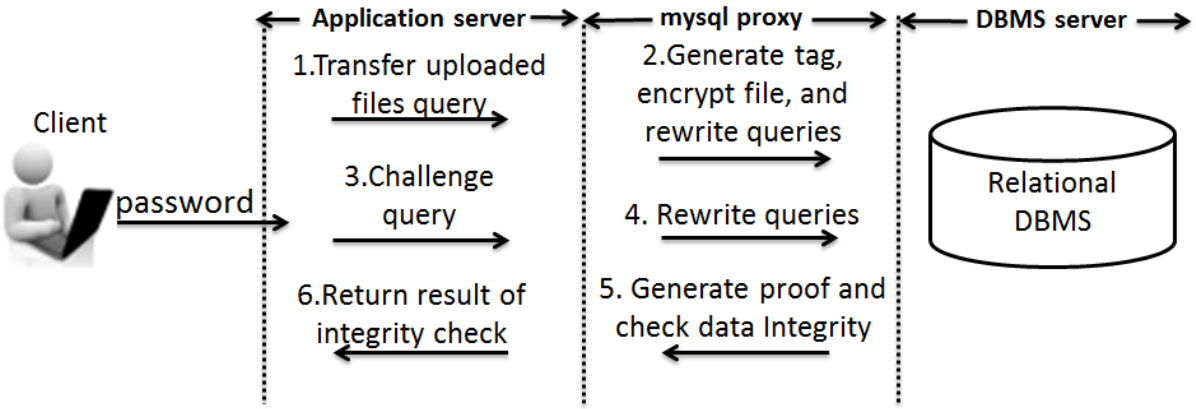


圖 5 資料完整性驗證的系統流程

表時有著 ic 指令欄位中的資料內容產生認證碼及加密資料，並且把原來的 SQL 查詢封包格式改寫成新增認證碼的欄位後密文資料與認證碼一併儲存到資料庫。

第三階段：我們設計的使用者介面上顯示出資料表中儲存的所有 primary key 的欄位，而使用者可以選取 primary key 欄位後面的下載、驗證完整性、更新以及刪除資料的功能，透過 Web server 送出使用者想要查詢的訊號。

第四階段：代理伺服器收到 SQL 查詢指令後改寫成當初儲存到資料庫時的資料結構格式，再把 SQL 查詢指令傳送到資料庫。

第五階段：代理伺服器從資料庫接收到加密資料及認證碼後先把資料解密，且產生證明碼(proof)。為了避免重送攻擊，產生認證碼使用的參數，每次以亂數方式產生。產生完證明碼後驗證此證明碼，並檢查結果及解碼後的明文一併傳送給 Web server。

第六階段：Web server 收到檢查結果後利用網頁訊息視窗，顯示驗證結果。如果使用者要下載資料時，解密後的檔案傳送給使用者。並且上傳、更新以及刪除資料成功時，透過網頁訊息視窗顯示完成這些查詢的結果。

4.2 加密資料上執行讀與寫

加密資料上執行 SQL 查詢時，一樣可以使用大家所熟悉的 SQL 查詢指令。差別在於使用者一開始建立資料表時，要決定之後此資料表中的哪一個欄位資料要做完整性檢查驗證，而 SQL 查詢指令中，想要完整性驗證的欄位名稱及資料型態之間增加一個 ic

指令。之後使用者插入資料到此資料表時，對應到有著 ic 指令欄位的資料會被加密及產生完整性檢查所需要的相關資訊。系統利用相等的比對方式，搜尋資料。假設 SQL 查詢為“select * from book;”時，代理伺服器會把 book 資料表中的所有資料回傳回來。其中把資料加密的部分，解密成明文回傳，並且驗證加密資料的完整性。如果 SQL 查詢為“select * from book where filename='file.txt';”時，代理伺服器會回傳 filename 等於 file.txt 的那一筆資料，此時也會一樣回傳資料的完整性驗證結果。

4.2.1 執行寫入查詢

系統除了能建立可以檢查完整性的資料表，也就是保護隱私性的資料以外也可以建立明文資料表。欲達到隱私性及完整性檢查時，資料表建立的過程中就需要改寫 SQL 查詢指令。圖 6 對照以明文方式儲存的資料表內容(圖 6(a))及有著完整性檢查資訊(也就是認證碼)的資料表內容(圖 6(b))。使用者想要檢查 filedata 欄位內容的完整性時，建立資料表的 SQL 查詢指令為

create table book(filename text, filedata ic text);

代理伺服器收到 SQL 查詢指令後檢查到 filedata 欄位中有著 ic(integrity check)指令時，增加兩個儲存 IV(initialization vector)及認證碼(integrity_tag)的欄位，如圖 6(b) 所示。IV 的欄位名稱命名為要檢查完整性的欄位名稱 (filedata) 加上 salt，認證碼欄位則 integrity_tag。SQL 查詢指令中也可以有多個欄位同時檢查資料的完整性，也就是可以在多個欄位中有 ic 指令的出現。

資料插入到資料表時，插入資料的 SQL 查詢指令與一般 SQL 查詢指令一樣為

insert into book values ('file.txt', 'test');

代理伺服器會利用 filedata 欄位的內容及相關金鑰產生認證碼，且利用自己的 master

book

filename	filedata
file.txt	test

(a)

book

filename	filedata	filedatasalt	integrity_tag
file.txt	XXXXX	12345678	OOOOO

(b)

圖 6 明文及加密資料表對照

key，加密 filedata 欄位的內容。最後密文資料及認證碼分別存回 SQL 查詢封包的 filedata 欄位及新增加的認證碼欄位。加密資料時，系統使用 AES-CBC mode，而加密使用到的 IV(initialization vector)以亂數產生並寫到 filedatasalt 欄位、最後認證碼存入 integrity_tag 的欄位，filedata 及 integrity_tag 的內容儲存成 BLOB 型態的檔案，圖 6(b) 中的 XXXXX 以 BLOB 型態儲存的 filedata 檔案以及 OOOOO 以 BLOB 型態儲存的 integrity_tag 檔案。

4.2.2 執行讀取查詢

使用者想要檢查資料完整性時，傳送 SQL 查詢指令到代理伺服器。檢查完整性資料的 SQL 查詢指令為

select * from book where filename='file.txt';

資料庫接收到 SQL 查詢指令後用相等的方式比對，回傳 filename 等於 file.txt 的那筆資料。代理伺服器接收到符合 SQL 查詢封包的資料後先解密 filedata 欄位的內容，利用解密的結果及認證碼產生證明碼，為了防止重送攻擊，參與計算證明碼的 challenge 每次以亂數方式產生。最後檢查證明瑪，驗證資料的完整性，檢查結果傳到 Web server。Web server 以網頁訊息視窗方式顯示驗證結果。如果使用者想要下載資料時，把解密後的結果，以檔案方式傳送給 Web server。

使用者想要動態的更新或是修改資料時，SQL 查詢指令為

update book set filedata='testtest' where filename='file.txt';

代理伺服器收到 SQL 更新指令時，與插入或是上傳指令一樣利用 filedata 欄位的新資料('testtest')產生認證碼。接著利用自己的 master key 加密新資料後新資料以及認證碼上傳並覆蓋 file.txt 檔案的內容。

如果使用者想要刪除資料時，SQL 查詢指令為

delete from book where filename='file.txt';

代理伺服器接收到刪除資料指令時，把 SQL 查詢封包改寫成自訂的資料欄位格式後把查詢送給資料庫。最後資料庫會刪除 filename 等於 file.txt 的那筆資料。所有 SQL 查詢中，filename 的欄位可以作為 primary key 的角色，以便讓使用者能夠查詢資料。

第五章、系統實作

使用自訂的C++ library及Lua模型設計並實作代理伺服器。C++ library包含了query parser、query encryptor、tag generator、proof generator、check proof generator 及query decryptor module。利用 query encryptor 加密資料並改寫 SQL 查詢封包內容、tag generator function 產生認證碼、proof generator function 產生證明碼、check proof generator function 驗證證明碼以及利用 query decryptor 解密使用者要下載的密文資料。C++ library 中包含除了加解密、改寫 SQL 查詢內容及檢查完整性的功能以外建立這些功能時，會使用到的 User-Defined Functions。運算過程中，利用這些 User-Defined functions 定義執行 SQL 查詢時會使用到的新資料結構。

使用者透過兩種方式，包含開啟終端機下指令的方式以及使用者介面的方式使用此資料庫系統。圖 7 顯示以終端機下指令為主的系統實作內部架構，架構中包含終端機、自訂的 SQL 代理伺服器，其中包含加解密器、認證碼產生器、證明碼產生器、改寫 SQL 查詢封包的模型，與關聯式 MySQL 資料庫。系統透過 Lua 模型接收使用者透過終端機傳送的 SQL 查詢指令，利用自訂的 C++library 改寫封包內容後儲存到資料庫。使用者想要讀取資料時，代理伺服器也是透過 Lua 模型，傳送符合使用者查詢的資料，把資料解密後的明文資料及完整性驗證的結果給使用者。MySQL 資料庫的部分是安裝沒有修改過原始碼的一般資料庫。為了提升使用者的方便性，所有在終端機上操作的 SQL 查詢指令分別寫成網頁應用程式，實作使用者介面。使用者可以透過 Web server 為介面，操作資料的查詢動作，包括上傳、下載、驗證完整性、更新及刪除資料。

5.1 代理伺服器及終端機上的結果資訊

使用者在終端機上操作 SQL 查詢資料時，代理伺服器上可以顯示所有查詢資料的細部資訊，例如：使用者要建立資料表時，傳送的 SQL 查詢指令、使用者要儲存到資料庫的所有資料、為了儲存加密 AES-CBC mode 時參與的 IV 及完整性驗證時參與的認證碼欄位，代理伺服器改寫的 SQL 查詢封包內容、資料庫回傳符合使用者查詢資料條件的所有資訊包括加密資料、資料的認證碼、資料的證明碼及解密後的明文資料等。

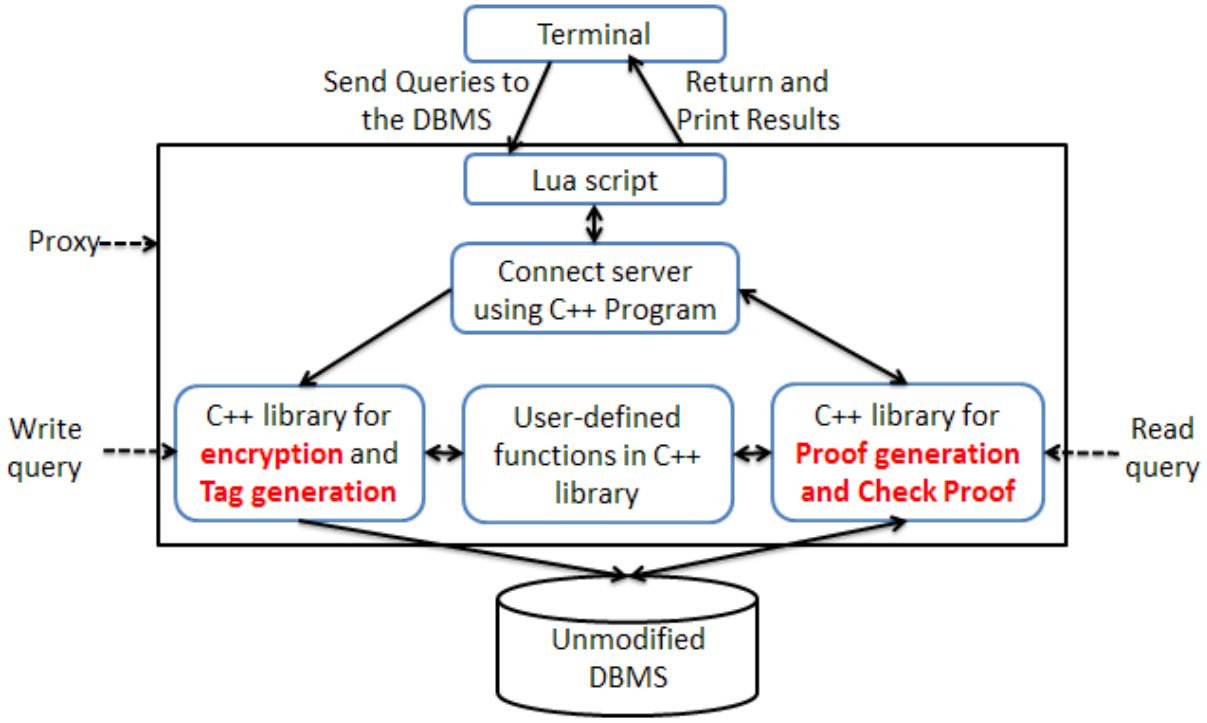


圖 7 系統實作內部架構

5.1.1 建立資料表、插入及查詢資料

在關聯式資料庫系統上建立資料表時，使用的 SQL 查詢指令為

create table book(filename text, filedata blob);

此時，資料庫中建立了 book 名稱的資料表，其中包含 text 型態的名稱為 filename 的欄位以及 blob 型態的名稱為 filedata 的欄位。之後插入資料到此資料表時，資料內容都以明文方式儲存。為了區分建立明文及密文資料表的差別，之後想要檢查完整性的欄位中加上(ic)指令。filedata 欄位內容想要檢查完整性時，欄位名稱及檔案型態之間要加上一個 ic，如圖 8 所示。此時，代理伺服器會把 SQL 查詢封包攔截下來後改寫成增加兩個加密資料時需要的 IV (initialization vector) 欄位(也就是要檢查完整性的欄位名稱接著加上 salt (filedatasalt))及認證碼 (integrity_tag)欄位後在資料庫建立資料表。因此資料庫建立完成的資料表欄位中，明文資料表建立時的兩個欄位 filename、filedata 變成四個欄位 filename、filedata、filedatasalt 以及 integrity_tag。其中，不需要加密的欄位(filename) 資料型態為依照使用者傳送的 SQL 查詢指令中的型態、需要加密資料的欄位型態及認證碼(integrity_tag)欄位型態都預設為 blob 的型態、IV (initialization vector)的欄位為 unsigned big integer 的型態。

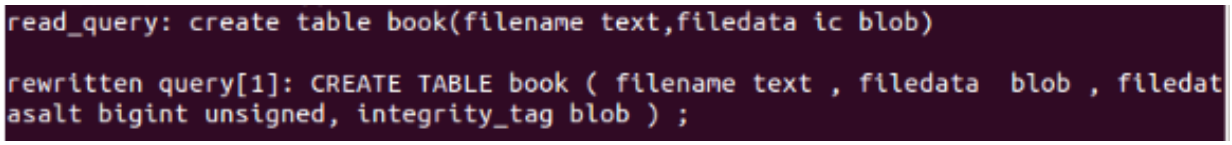


圖 8 建立資料表及改寫欄位內容

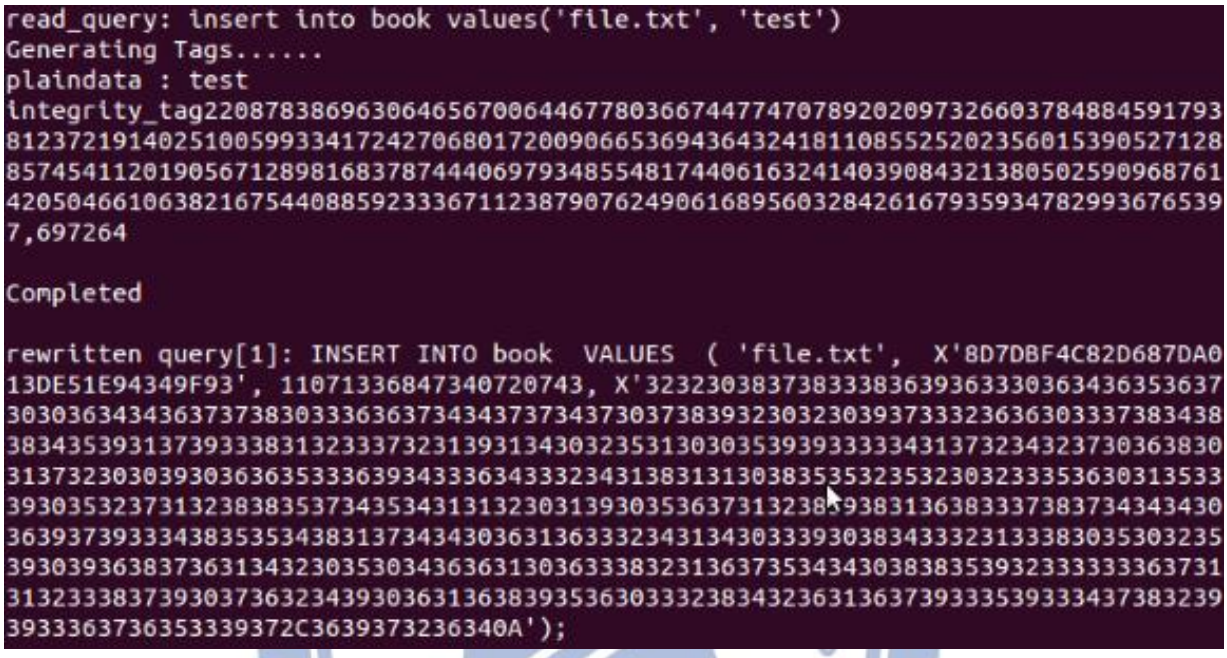


圖 9 插入資料

使用者插入資料到資料庫時，使用的 SQL 查詢指令為

insert into book values (‘file.txt’, ‘test’);

代理伺服器攔截 SQL 查詢封包後取出對應到 filedata 欄位的資料內容產生認證碼及加密 filedata 欄位的資料。圖 9 顯示 filedata 欄位的資料(test) 產生認證碼(integrity_tag)的結果。認證碼的內容中逗點前面的值為 metadata(T_m)的值、逗點後面的值為 W 的值。接著代理伺服器把攔截下來的原本資料內容(‘file.txt’, ‘test’)改寫成(‘file.txt’, X’8D7DB ….40A’, 11071……0743, X’323230…….340A’)，其中第一個 filename 欄位在建立資料表時已經決定以明文方式儲存，因此對應到 filename 欄位的資料是 file.txt，不需要加密或是改寫而放回原來的值、第二個欄位為插入 filedata 欄位的資料 (test)加密後編碼成 16 進位方式的結果、第三個欄位是利用 AES-CBC mode 的技術，加密資料時參與的 IV (initialization vector)值，這值以亂數方式產生並寫到代理伺服器新增的 filedatasalt

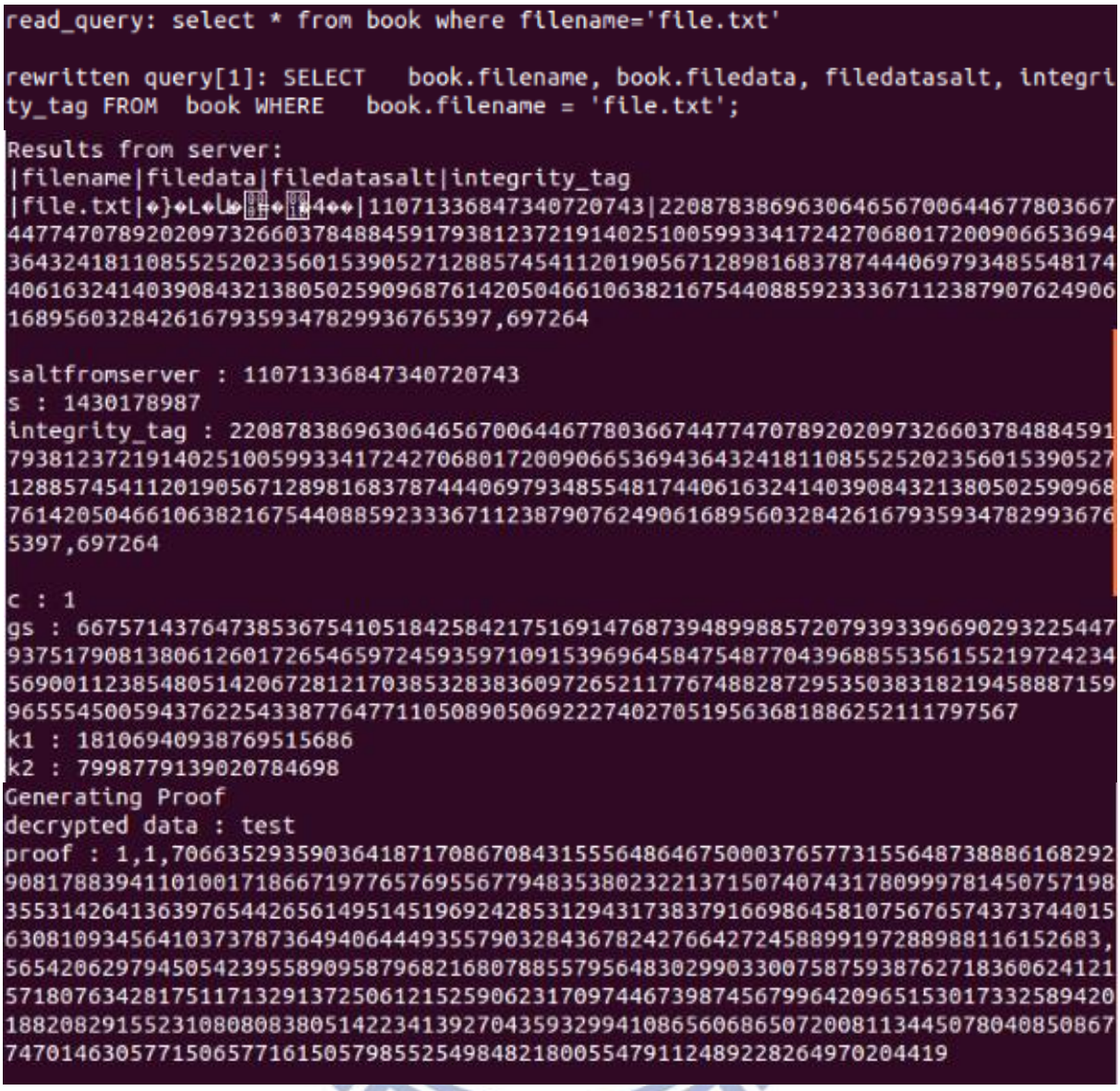


圖 10 產生證明碼過程

欄位、第四個欄位是產生好的認證碼的欄位以 16 進位方式編碼，最後把改寫後的 SQL 查詢指令傳送到資料庫。

當使用者想要驗證資料表 book 中儲存的 filename 等於 file.txt 的資料完整性時，使用的 SQL 查詢指令為

select * from book where filename='file.txt';

代理伺服器攔截此SQL查詢封包後把*改寫成book資料表中的所有欄位，包含filename、filedata、filedatasalt 以及 integrity_tag，之後把此 SQL 查詢指令傳送到資料庫查詢符合的資料。資料庫回傳book資料表中filename等於file.txt的資料，圖 10 中顯示Results from

server 底下的資料為資料庫回傳的資料。filename 欄位以明文儲存而回傳資料 file.txt 也是以明文方式回傳。filedata 欄位是以密文方式儲存，而回傳資料時以亂碼方式呈現。filedatasalt 欄位中儲存著加解密資料時，AES-CBC mode 中會參與的 IV，而最後一個欄位是運算證明碼時需要參與的認證碼。saltfromserver 是 filedatasalt 欄位中的值。代理伺服器利用 IV(filedatasalt) 及 master key 解密資料，圖 10 顯示 decrypted data 是資料庫回傳的 filedata 欄位資料解密後的結果。完整性驗證過程中，要產生一些亂數來計算證明碼，其中 s 為從 Z_N^*中隨機選擇、c 為 n 個區塊資料中選擇一個區塊資料、利用 s 計算 gs 等於 g^s、使用 k1 做 pseudorandom permutation 與 k2 為計算係數值使用。最後產生證明碼(proof)，其中包含四個數值，第一個數值為資料個數、第二個為資料的總共資料區塊個數、第三個則是 PDP 中產生 proof 值的 T、最後一個數值為ρ的值。圖 11 顯示取出 proof 的所有參數產生驗證結果，check result 等於 1 是驗證結果為正確， 否則表示非授權使用者包含惡意資料庫管理者或是外部攻擊者有修改使用者的資料或是認證

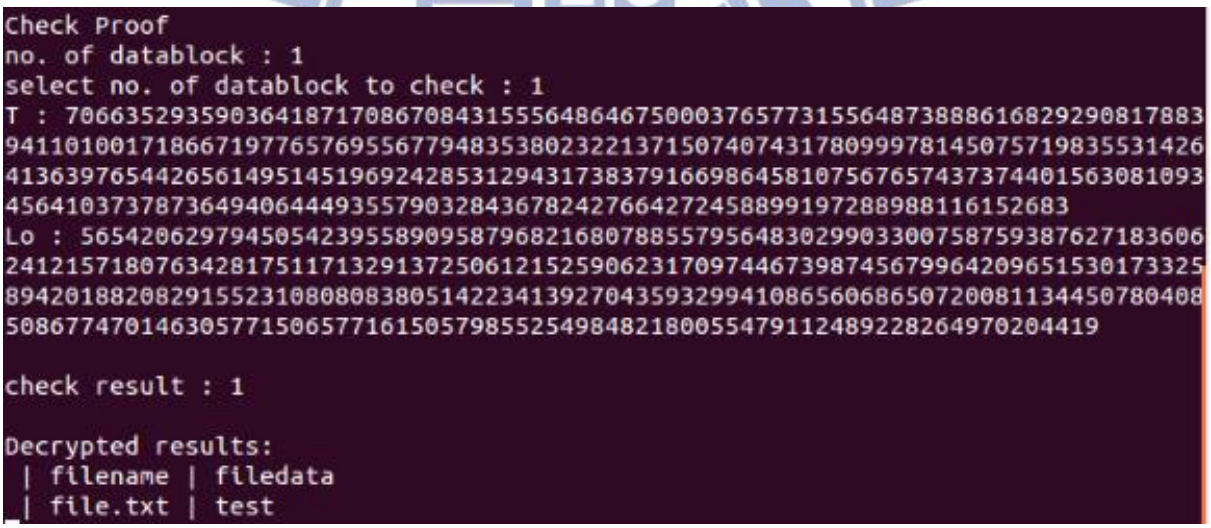


圖 11 驗證證明碼過程

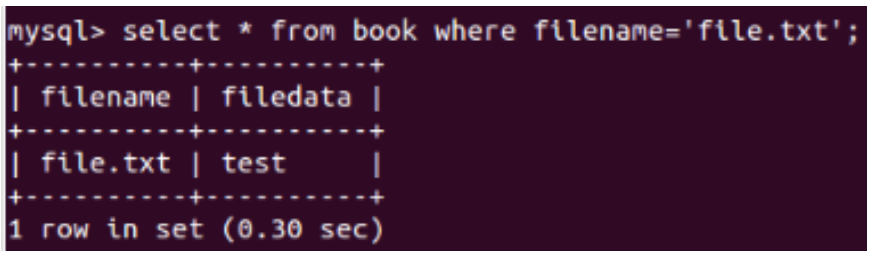


圖 12 終端機上查詢資料的結果

圖 13 資料已被修改後驗證完整性

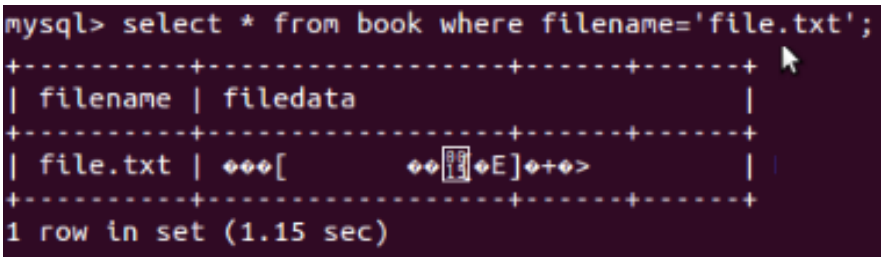


圖 14 資料已被修改後終端機上查詢的結果

碼。最後可以看到符合使用者查詢資料解密後的結果(Decrypted results)，並且代理伺服器把結果回傳給使用者。圖 12 是終端機上顯示符合使用者要查詢的資料結果。

5.1.2 使用者資料被刪除後完整性檢查結果

為了節省儲存空間，惡意資料庫管理者刪除使用者的部分資料的情況之下，使用者必須不定時地驗證資料的完整性。圖 13 顯示使用者查詢 book 資料表中的檔案名稱為 file.txt 的資料時，資料庫回傳符合的資料。接著代理伺服器會解密 filedata 欄位的資料，filedata 欄位內容已經被修改的情況之下，解密出來的資料(decrypted data 為亂碼)的結果是無意義的資料。因此完整性驗證的結果(check result 為 0)是不正確的。最後可以看到代理伺服器回傳解密後的資料(Decrypted results)中資料則是無意義的資料。圖 14 顯示代理伺服器回傳解密後的明文資料給使用者時，終端機上顯示的結果，此結果符合於代理伺服器上顯示的 Decrypted results 結果。

5.1.3 修改認證碼後完整性檢查結果

惡意資料庫管理者除了可能會修改使用者的資料以外也有可能修改認證碼。圖 15 顯示使用者查詢檔案名稱為 file.txt 資料時，代理伺服器回傳符合的資料結果。回傳資料認證碼的第一個字(圖 15 中框起來的部分)被修改的情況之下，代理伺服器解密資料時，不會影響解密的結果，而解出使用者原本儲存的資料(decrypted data 為 test)。但是因為認證碼已經被修改，最後驗證完整性檢查的結果 (check result 為 0)是不正確的。因此系統可以判斷惡意資料庫管理者或是外部攻擊者修改的資料部分為使用者儲存的資料或是認證碼。因為解密資料結果都是正確的，最後代理伺服器上顯示的的結果

圖 15 認證碼已被修改後驗證完整性

(Decrypted results)是使用者一開始儲存的 file.txt 檔案的內容 test。

5.1.4 動態資料的更新及刪除結果

當使用者想要動態的更新資料表中的 filename 等於 file.txt 檔案的內容時，使用的 SQL 查詢指令為

update book set filedata='testtest' where filename='file.txt';

代理伺服器接收到 SQL 查詢指令時，插入資料到資料庫時一樣，利用新資料以及自己的 master key 產生認證碼，接著加密新資料('testtest') 後以 16 進位方式編碼。最後把 SQL 查詢指令的 filedata 內容換成加密並編碼好的新資料(X'9A608….8546')、filedatasalt 的欄位中放入加密時亂數產生的 IV(12278……0135)、integrity_tag 的欄位則放入產生好的認證碼以 16 進位方式編碼後的內容(X'31333…….340A')，如圖 16 所示。改寫完

```
read_query: update book set filedata='testtest' where filename='file.txt'
Generating Tags......
plaindata : testtest
integrity_tag1323041226687066223765399145893006760300066125690407830815367518733
97817683152151660599019934247636993735670748310944134103077601268508670137309160
23430918016800756931932494398904564583021883224693939407033472186357915901442773
31871684710147810880397349436855948627753119486194661376277171941360351485028760
24,697264

Completed

rewritten query[1]: UPDATE book SET  filedata = X'9A608CD970805ECA61A9E3FAA23885
46', filedatasalt = 12278788873662250135, integrity_tag = X'3133323330343132323636
36383730363632323337363533393931343538393330303637363033303030363631323536393034
30373833303831353336373531383733333937383137363833313532313531363630353939303139
39333432343736333639393337333536373037343833313039343431333431303330373736303132
36383530383637303133373330393136303233344330393138303136383030373536393331393332
34393433393839303435363435383330323138383332323436393339339339343037303333343732313
38363335373931353930313434323737333331383731363834373130313437383130383830333937
33343934333638355359343836323737353331313934383631393436363133373632373731373139
34313336303335313438353032383736303234 2C3639373236340A' WHERE   book.filename =
'file.txt';
```

圖 16 動態更新資料的結果

```
read_query: delete from book where filename='file.txt'

rewritten query[1]: DELETE FROM  book WHERE   book.filename = 'file.txt';
```

圖 17 動態刪除資料的結果

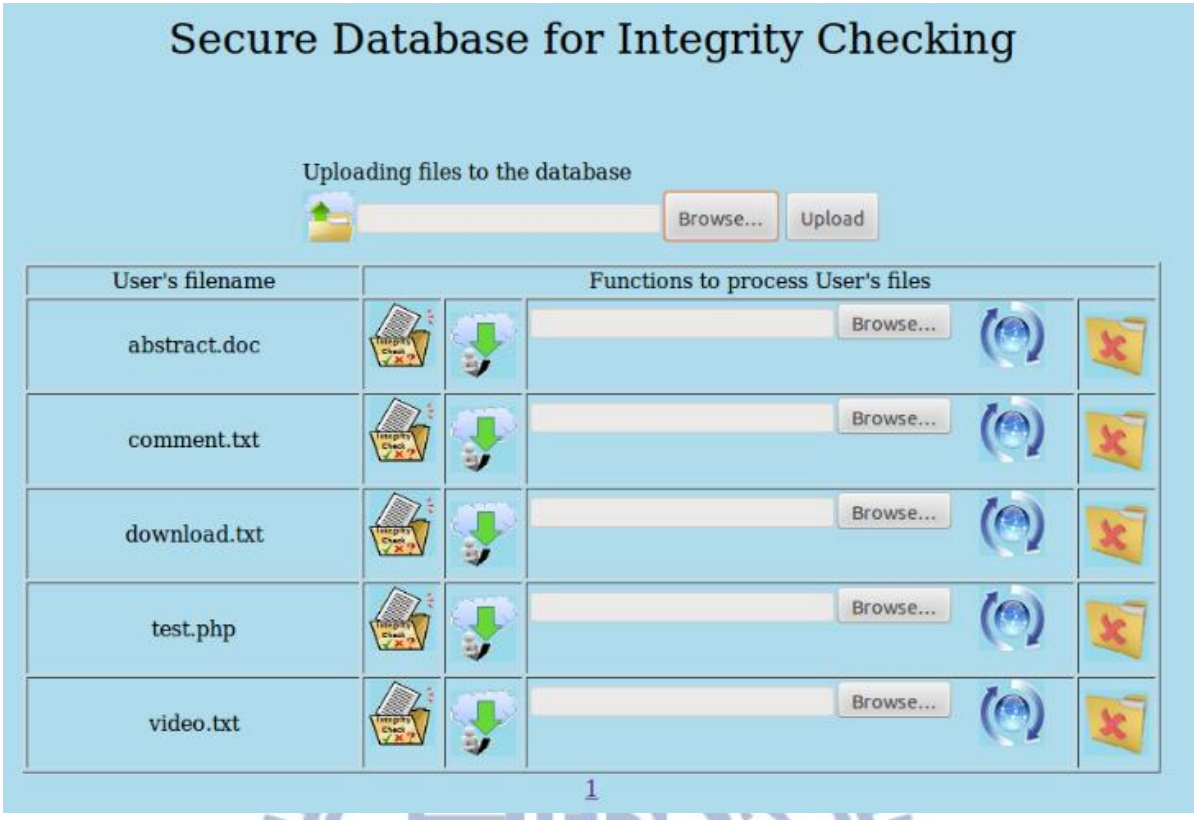


圖 18 使用者介面

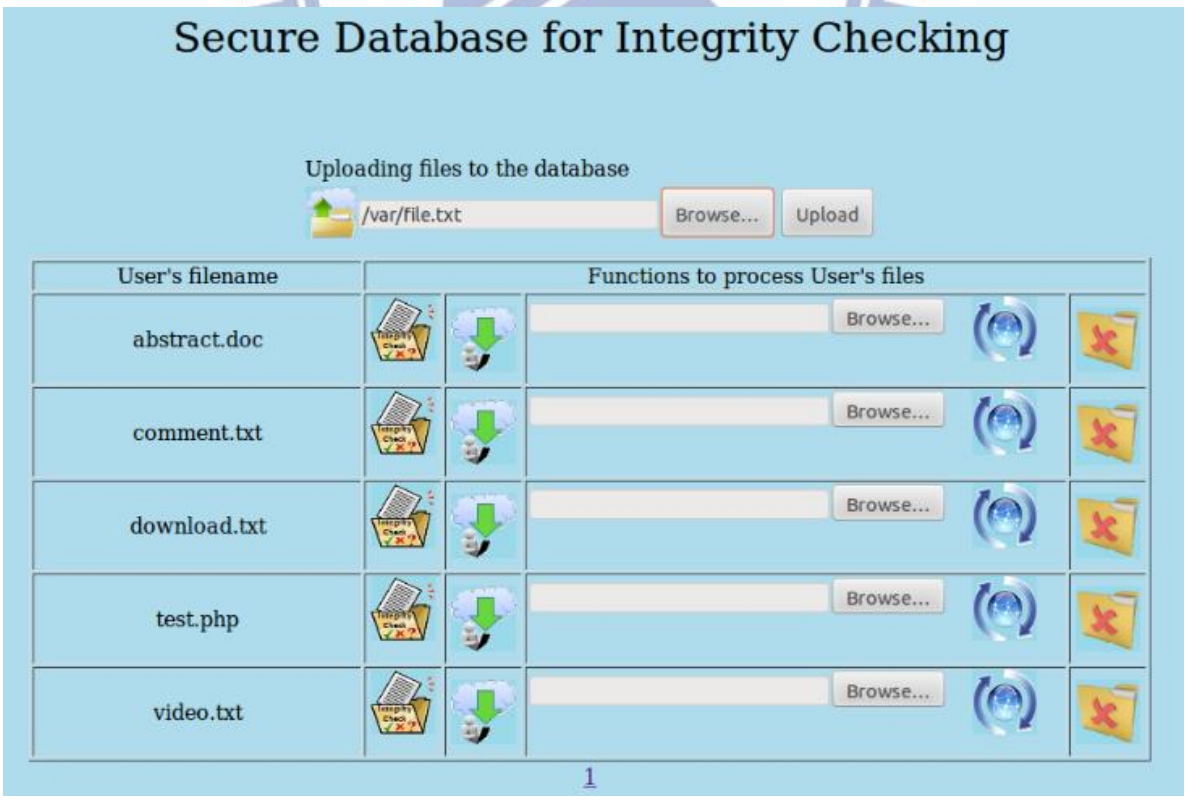


圖 19 要上傳的檔案瀏覽並選擇後結果

SQL 查詢指令內容後傳送到資料庫更新資料的內容。

當使用者想要動態的刪除資料表中的 filename 等於 file.txt 檔案的內容時，使用的 SQL 查詢指令為

delete from book where filename='file.txt';

代理伺服器改寫 SQL 查詢指令為自訂代理伺服器的資料結構格式後傳送此指令給資料庫刪除資料，如圖 17 所示。

5.2 使用者介面上顯示的結果

為了提升使用者的方便性，利用 Web server 作為使用者介面，使用者可以操作資料上傳、下載、完整性的驗證、更新以及刪除，如圖 18 所示。使用者介面會顯示資料表中儲存的所有資料，也就是作為 primary key 的欄位資料，且每頁顯示五筆資料。資料表的上面顯示可以上傳使用者想要儲存到資料庫的資料。並且使用者可以下載、資料完整性驗證、更新以及刪除資料庫中已經存在的資料。使用者想要上傳一筆新的資料時，利用 Browser 瀏覽並選擇一個要上傳到資料庫的檔案。

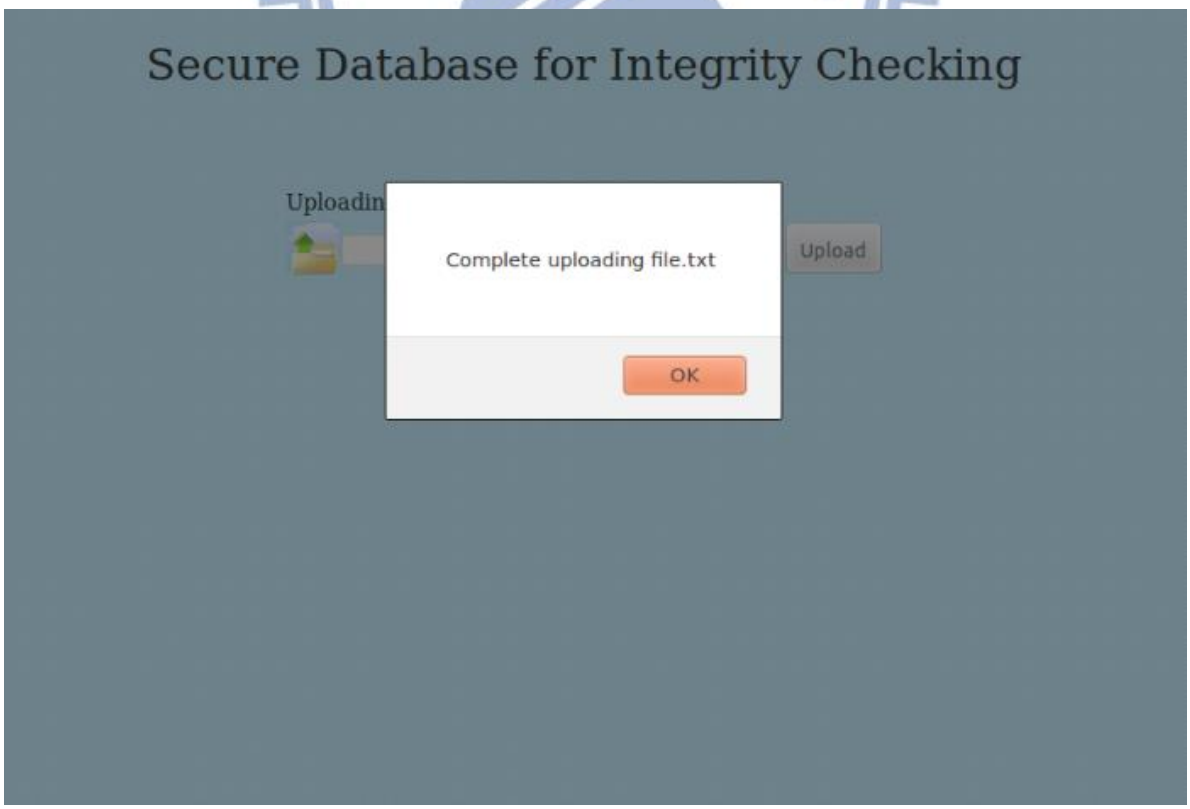


圖 20 使用者介面顯示上傳成功的訊息

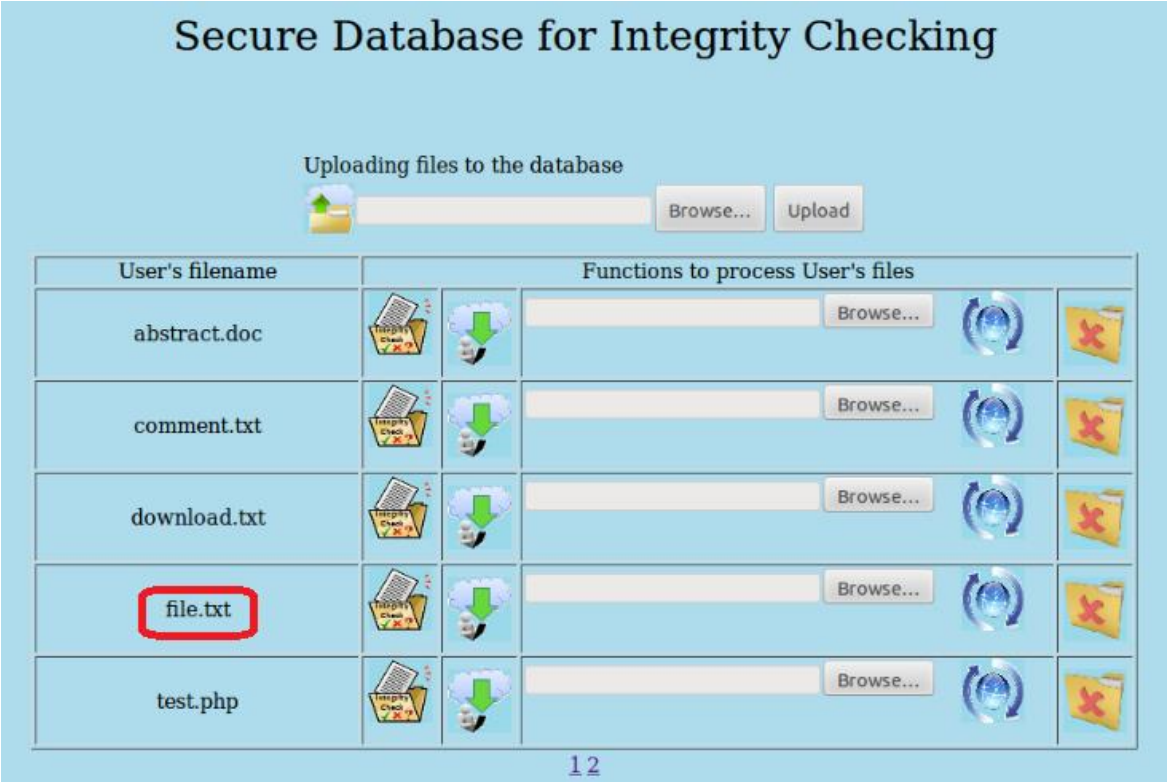


圖 21 使用者介面顯示上傳成功的結果

圖 19 顯示使用者選擇了要上傳到資料庫的檔案名稱為 file.txt 。選擇完後 Upload 按鈕上傳檔案。上傳成功後跳出網頁訊息來顯示檔案已經上傳成功的訊息，如圖 20 所示。圖 21 顯示 file.txt 檔案已經上傳並儲存到資料庫的結果。上傳資料功能中，終端機上操作的建立資料表及插入資料表的 SQL 查詢指令寫成網頁應用程式，透過 Web server 傳送 SQL 查詢指令到代理伺服器。代理伺服器幫使用者加密資料及產生認證碼，且把 SQL 查詢指令的內容改寫成增加加密資料時參與的 IV 及認證碼儲存的欄位。最後把加密資料及認證碼一併儲存到資料庫。

使用者介面上顯示的資料表中，所有作為 primary key 的資料名稱後面提供四個功能，包含完整性檢查、下載、更新以及刪除資料的功能。使用者想要驗證資料完整性時，滑鼠移到想要檢查資料完整性檔案名稱後面的第一個按鈕，使用者介面會顯示 IntegrityCheck 的文字，如圖 22 所示。按下此按鈕後，代理伺服器從資料庫回傳符合的資料及隨機產生一些 challenge 值計算證明碼。最後檢查證明碼是否正確後回傳驗證結果給 Web server。Web server 會彈出一個網頁訊息視窗，顯示完整性檢查的結果。圖 23 顯示使用者資料完整的被儲存在資料庫時，網頁視窗上顯示正確的驗證結果。如果使用

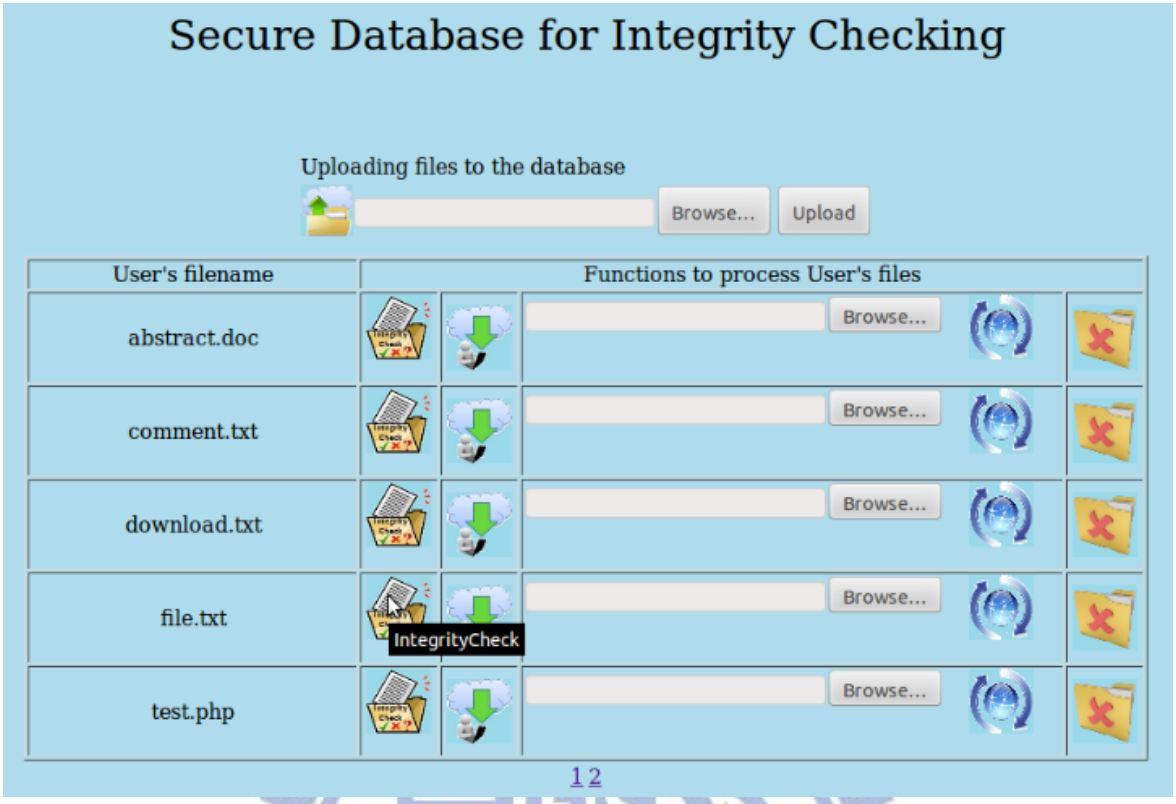


圖 22 顯示完整性驗證功能

圖 23 使用者介面上的完整性驗證結果

者的資料或是認證碼被非授權者修改時，網頁視窗上會顯示不正確的驗證結果，如圖 24 所示。使用者要下載資料時，使用者選擇想要下載資料的檔案名稱後面的第二個按鈕，使用者介面會顯示 download 的文字，如圖 25 所示。Web server 傳送給代理伺服器的 SQL 查詢指令與資料完整性檢查時一樣

select * from book where filename='file.txt';

代理伺服器收到指令後改寫成

select book.filename, book.filedata, book.filedatasalt, book.integrity_tag from book where filename='file.txt';

傳送到資料庫查詢。代理伺服器解密資料庫回傳回來的資料後傳送給 Web server。Web server 會先彈出一個網頁訊息視窗，詢問使用者是否下載此檔案，如圖 26 所示。使用者想要下載此檔案時，按 OK 的按鈕，否則按 Cancel 取消下載。使用者想要更新資料表中的檔案時，利用更新功能中的 Browser 瀏覽要更新的檔案，如圖 27 所示。圖 28 顯示成功地更新資料完成的結果。使用者想要刪除已經存在資料庫的某個資料時，使用者介面也提供刪除資料的功能，如圖 29 以及圖 30 所示。

圖 24 資料被修改後完整性驗證結果

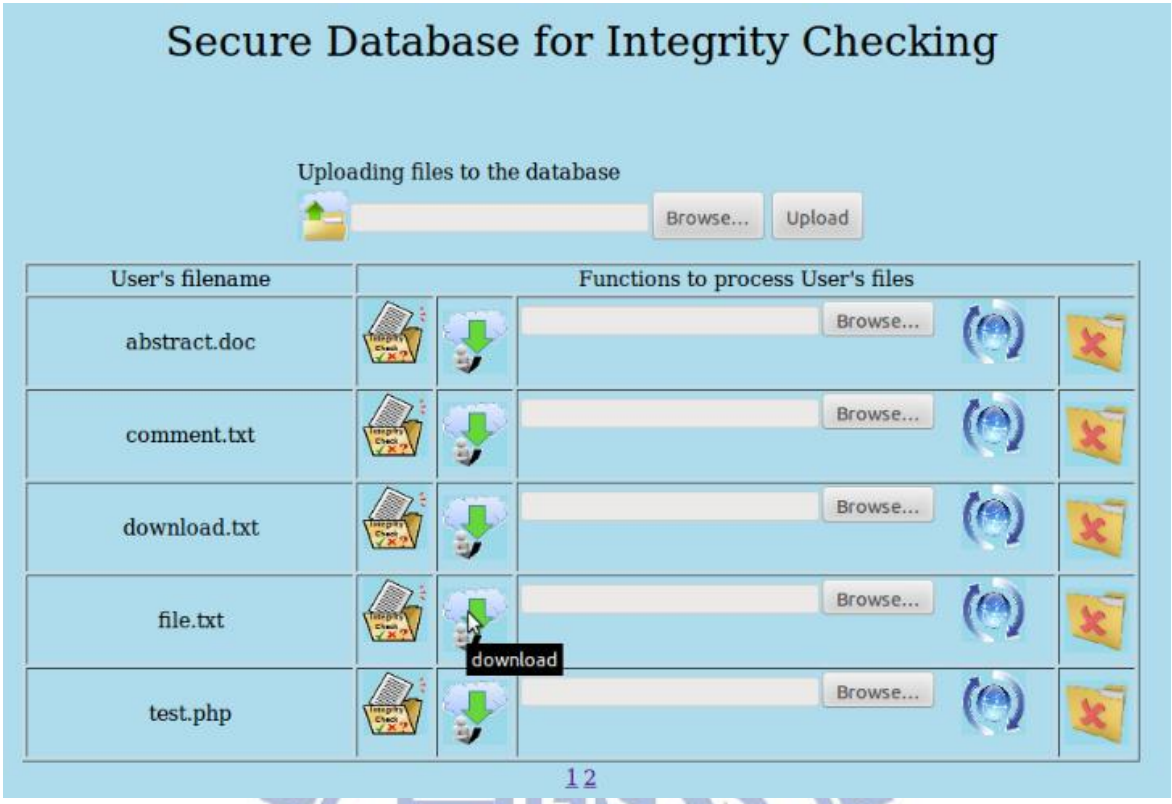


圖 25 使用者介面上選擇下載功能

圖 26 使用者介面上下載資料的結果

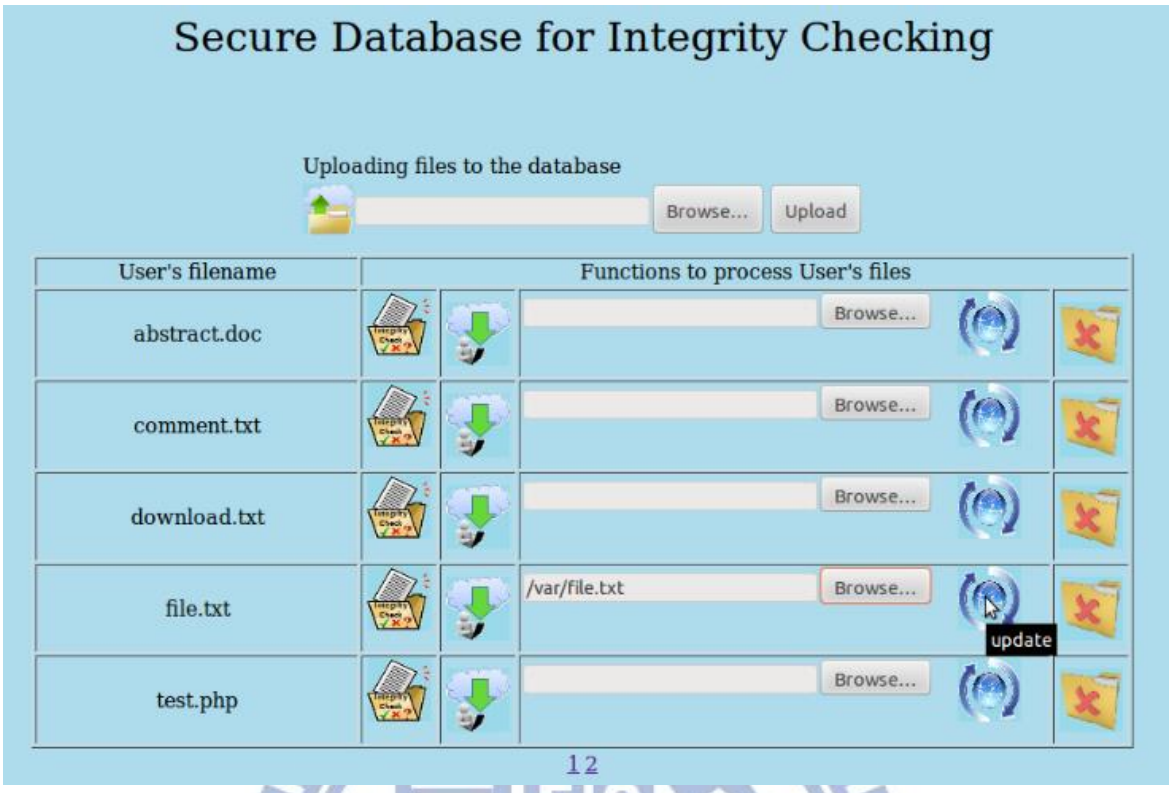


圖 27 使用者介面上選擇要更新的檔案

圖 28 使用者介面上成功地更新資料的結果

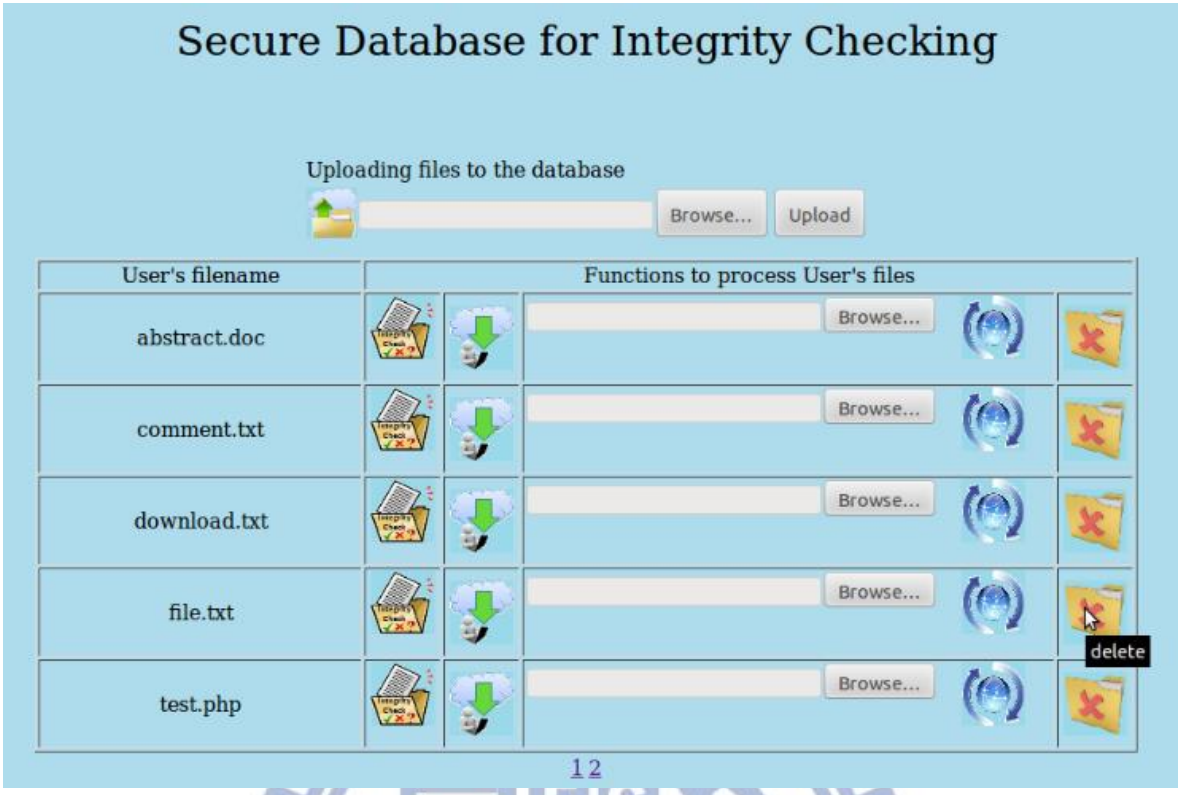


圖 29 使用者介面上選擇刪除資料的功能

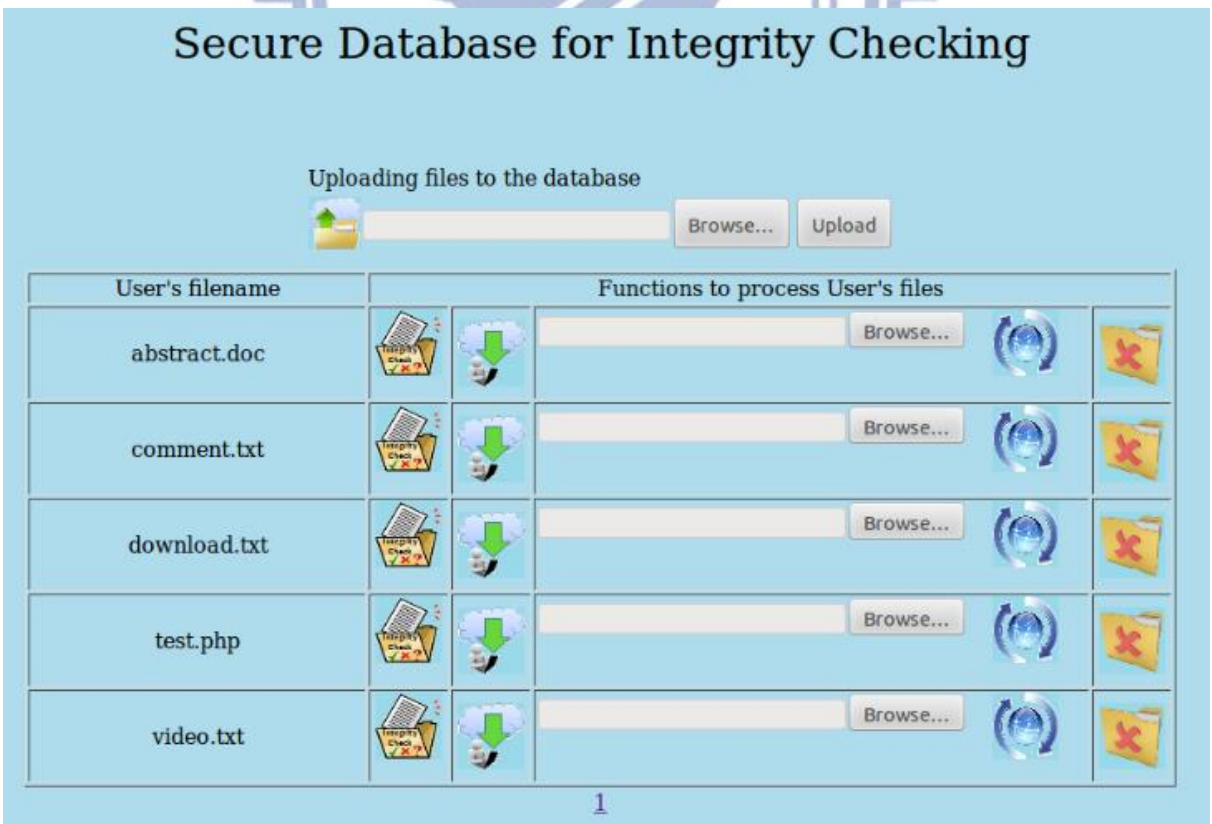


圖 30 使用者介面上刪除資料的結果

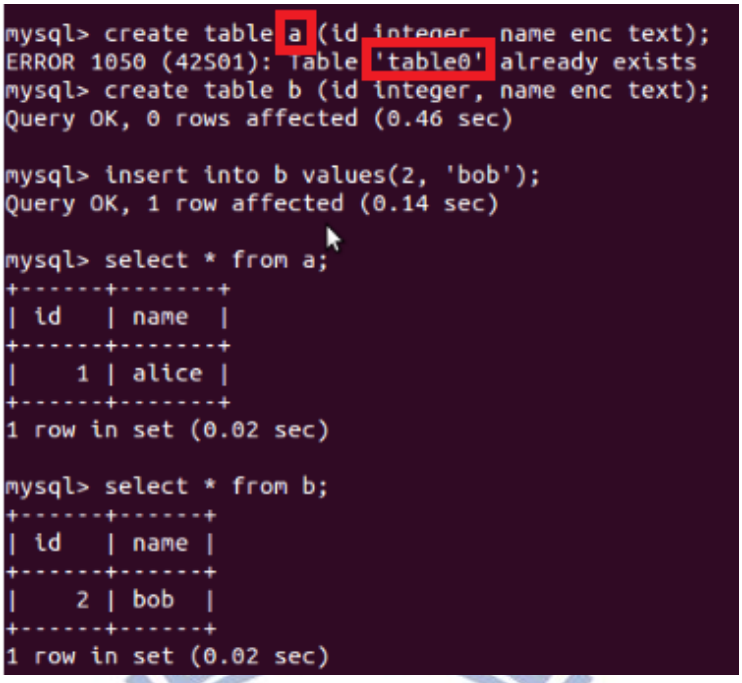
第六章 比較

為了保護資料庫中儲存的使用者資料安全性，Raluca Ada Popa et al. [1][2][3]提出了新的加密方式“onions of encryption”。此加密方式是資料庫中的資料表及資料欄位都與資料一起加密。不管使用者儲存時命名的資料表名稱為何，資料表的命名方式是依照已經儲存的資料表個數順序命名，例如：第一個儲存到資料庫的資料表被命名為 table0、第二個儲存進去的資料表為 table1、依此類推。資料表名稱為匿名的情況之下，惡意資料庫管理者無法猜出此資料表中儲存的資料內容為何。雖然這樣的保護資料方式讓使用者的資料更加安全，但引起了兩個問題。

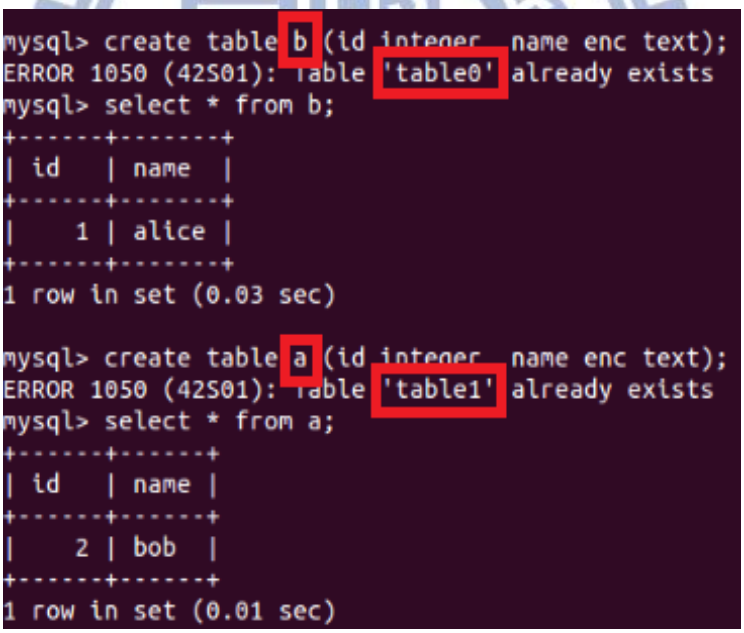
第一個問題為使用者要查詢資料時，必須記得當初建立資料表的明文名稱順序，對照與它的匿名後才能以明文方式取出資料表中的資訊。除此之外，資料欄位名稱也是以匿名方式命名。因此使用者除了要管理資料表名稱以外必須得儲存資料欄位的資訊，包含欄位名稱及型態。但儲存著龐大資料的資料庫而言，資料表個數自然是非常多的，而資料欄位名稱也是如此。使用者必須管理與對照匿名資料表及資料欄位的明文名稱，才能查詢資料，而這樣會造成使用者的負擔。並且也會失去使用者為了節省儲存空間以及減輕管理負擔而交給對方管理資料的意義。

第二個問題是因為資料表的匿名為建立資料表的順序命名，代理伺服器因某些因素(如斷網或是斷電)的情況之下，被關機後再次被開啟時，使用者必須重新建立資料庫中的所有資料表，才能存取想要查詢的資料。假設資料庫中目前有十個資料表，使用者想要查詢的第九個資料表中的某一筆資料時，必須重新依照順序建立完前八個資料表後，才能建立第九個資料表及存取要查詢的某一筆資料。這樣的設計對使用者而言，實際操作上有非常的不方便。並且重新開啟後的代理伺服器，不認得當初建立的明文資料表名稱。因此使用者重新建立資料表時，不管使用者利用任何資料表及資料欄位名稱建立資料表，資料庫都會從第一個開始幫使用者建立資料表。

圖 31(a)顯示代理伺服器斷線前查詢資料的狀況。使用者從資料庫登出後再一次重新連線資料庫時，為了明文以及匿名資料表名稱的對照，必須建立一次資料表。圖 31(a) 的資料表 a 是上次已經建立好在資料庫中的資料表，資料表 a 必須再一次建立才能查詢此表中的資料，而資料表 b 是新建立的資料表，建立完插入新資料後馬上就能查詢資料表中的資料。圖 31(b) 顯示代理伺服器斷線後查詢資料的狀況，我們利用明文資料表



(a) 代理伺服器斷線前查詢資料



(b) 代理伺服器斷線後查詢資料

圖 31 代理伺服器斷線前與後查詢資料狀況圖

名稱 b 來建立第一筆資料，而第二筆資料則利用明文資料表名稱 a 來建立第二筆資料。此時，CryptDB 系統的代理伺服器依照資料庫中已經存在的順序重新建立資料表，而並非對照原來建立的明文資料表名稱。並且重新建立資料表時，必須要符合當初建立資料

表時的欄位個數及欄位的型態。因此為了防範代理伺服器斷線發生的狀況，使用者每次建立好資料表時，必須依序儲存一份已經儲存到資料庫的資料表名稱、資料欄位名稱及型態的查詢，例如：create table a(id integer, name enc text)、create table b(id integer, filename text, filedata enc blob)等。隨著資料庫中儲存的資料越來越龐大，使用者必須管理越來越多的建立資料表查詢而會造成使用者的負擔。

為了解決以上兩個問題，我們利用最基本的 AES 加密技術，修改 CryptDB 系統的加密方式。我們認為以明文方式儲存資料表及資料欄位名稱，不足以威脅到資料內容的安全。只要加密敏感的資料內容，就能解決資料的隱私性問題，也就是雖然讓惡意資料庫管理者以明文方式看到資料表及欄位名稱，無法以明文方式取得資料欄位的內容。這樣的設計，不但可以解決了使用者必須依序儲存資料表的問題，要查詢任何一筆資料表時，利用明文資料表名稱直接建立要存取的資料表，可以立即存取資料。

第七章、結論與未來發展

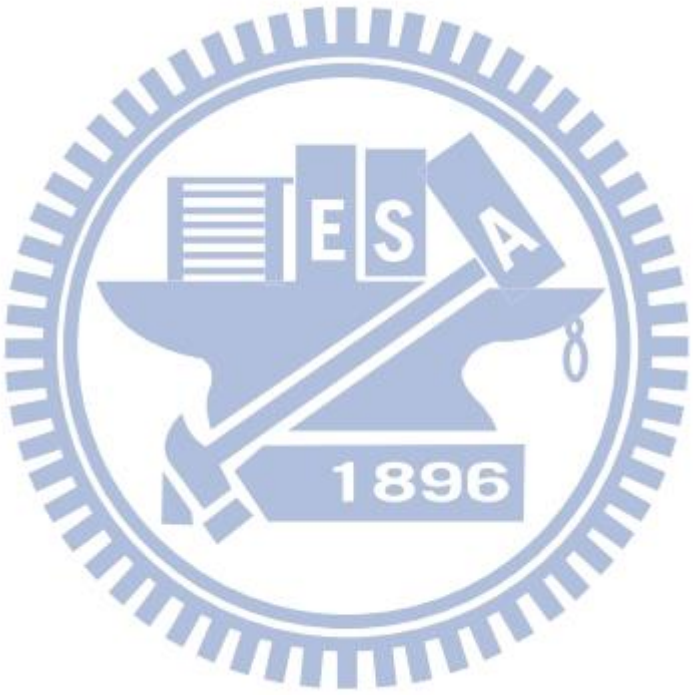
7.1 結論

我們從 CryptDB 系統中移除 keyword search、order preserving 以及 add 功能後，把 CryptDB 系統提出的新加密方式“onions of encryption” 修改成簡單的 AES 加密，不須用 delimiter 隔開就能夠處理完整的資料。代理伺服器及 Web Sever 作為使用者可信任的第三方，自訂的 C++ library 中加上完整性驗證的功能，讓 CryptDB 系統更加完善。代理伺服器加密使用者資料、驗證資料完整性以及運算 SQL 查詢封包的改寫。因為由代理伺服器產生參與計算證明碼的一些參數值，使用者不需要傳送這些參數值給代理伺服器。因此也可以降低使用者的傳輸成本。因為代理伺服器的私密金鑰處理資料的加密，減輕了使用者管理金鑰的負擔，也改善了傳輸對稱式金鑰時，被攻擊者攔截金鑰的缺點。加密技術讓資料庫系統達到資料隱密性、完整性檢查功能可以達到資料的完整性、只有授權的合法使用者才能以明文方式存取資料庫達到資料的可得性。因此 CryptDB 系統達到了資料庫系統安全原理。除此之外，為了使用者的方便性，Web server 作為使用者介面可以查詢資料。使用者介面上也提供了上傳、下載、完整性檢查、更新以及刪除的功能。一般使用者不需要下指令的方式，透過使用者介面就可以查詢資料。因為提出了完整性檢查功能，使用者可以節省下載資料所要花費的傳輸成本，就可以不定時地驗證資料的完整性。因此我們在 CryptDB 系統上增加了一個安全又能降低使用者負擔並能驗證資料完整性的功能。

7.2 未來發展

我們透過 CryptDB 系統設計了一個代理伺服器實作加解密、資料完整性檢查及改寫封包的運算。如果多個使用者要同時存取資料庫的狀況之下，會加重單一代理伺服器的計算負擔。並且也會下降計算及傳輸效率而造成使用者的不便。因此系統可以延伸到利用分散多台伺服器運算。這樣的情況之下，多個使用者的讀與寫執行時間可以同時分散的進行，而執行時間會變快。除此之外，系統可以彈性的擴展 Raluca Ada Popa et al. [1][2][3]提出的 keyword search、order preserving encryption、add 等功能。另外，為了能夠知道資料庫是否誠實的執行並回傳所有使用者要查詢的資料，例如：符合使用者要查

詢的資料共有十筆時，資料庫卻只回傳五筆資料的情況，我們的系統可以利用 Qingji Zheng et al. [20]提出的執行查詢封包回傳資料筆數的完整性檢查功能。利用分散多台伺服器以及執行查詢的完整性驗證功能，加強 CryptDB 系統更加的完善。



參考資料

1. Raluca Ada Popa, Catherine M.S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan, “CryptDB: Processing Queries on an Encrypted Database,” Communications of the ACM, Pages 103-111, September 2012.

2. Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan, “CryptDB: Protecting Confidentiality with Encrypted Query Processing,” in Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011), Cascais, Portugal, October 2011.

3. Raluca Ada Popa, Nickolai Zeldovich, and Hari Balakrishnan,“CryptDB: A Practical Encrypted Relational DBMS,” Technical Report MIT-CSAIL-TR-2011-005, Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, January 2011.

4. Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, Dawn Song, “Provable Data Possession at Untrusted Stores,” CCS’07, 2007.

5. C. Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, Roberto Tamassia, “Dynamic Provable Data Possession,” CCS '09 Proceedings of the 16th ACM conference on Computer and communications security Pages 213-222, 2009.

6. Feifei Liu, Dawu Gu, Haining Lu, “An Improved Dynamic Provable Data Possession Model,” IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), Page(s): 290 – 29, 2011.

7. Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li, “Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,” IEEE Transactions on Parallel and Distributed Systems, 2011.

8. Lanxiang Chen,“Using Algebraic Signatures for Remote Data Possession Checking,”

2011 International Conference on Cyber- Enabled Distributed Computing and Knowledge Discovery (CyberC), Page(s): 289 – 294, 2011.

9. Kevin D. Bowers, Ari Juels, and Alina Oprea, "Proofs of Retrievability : Theory and Implementation," CCSW'09, 2009.

10. Zhuo Hao, Sheng Zhong, Nenghai Yu, "A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability," IEEE Transactions on Knowledge and Data Engineering, VOL. 23, NO. 9, September 2011.

11. Syam Kumar P, Subramanian R, "An Efficient and Secure Protocol for Ensuring Data Storage Security in Cloud Computing,"International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, 2011.

12. Y. Deswarte, J.-J. Quisquater, and A. Saidane, "Remote integrity checking," In Proc. of Conference on Integrity and Internal Control in Information Systems, 2003.

13. Hamidah Ibrahim, "A strategy for semantic integrity checking in distributed databases," Ninth International Conference on Parallel and Distributed Systems, Page(s): 139 – 144, 2002.

14. Ali Amer Alwan, Hamidah Ibrahim, Nur Izura Udzir, "A Framework for Checking Integrity Constraints in a Distributed Database," Third International Conference on Convergence and Hybrid Information Technology, Page(s): 644 – 650, 2008.

15. Iqra Basharat, Farooque Azam, Abdul Wahab Muzaffar,"Database Security and Encryption: A Survey Study,"International Journal of Computer Applications, 2012.

16. Mariana Raykova, Binh Vo and Steven M. Bellovin, "Secure Anonymous Database Search," CCSW '09 Proceedings of the 2009 ACM workshop on Cloud computing security, Pages 115-126, 2009.

17. Hasan Kadhem, Toshiyuki Amagasa, Hiroyuki Kitagawa,"A Novel Framework for Database Security based on Mixed Cryptography," 2009 Fourth International Conference

on Internet and Web Applications and Services, Page(s): 163 – 170, 2009.

18. D\'ecio Luiz Gazzoni Filho and Paulo S\'ergio Licciardi Messeder Barreto, “Demonstrating data possession and uncheatable data transfer,” IACR ePrint archive, Report 2006/150, http://eprint.iacr.org/2006/150.

19. Dawn Xiaodong Song, David Wagner and Adrian Perrig, “Practical techniques for searches on encrypted data,” Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, Page(s): 44 - 55, 2000.

20. Qingji Zheng, Shouhuai Xu, Giuseppe Ateniese, “Efficient Query Integrity for Outsourced Dynamic Databases,” CCSW’12, October 19, 2012.