安全性雲端儲存快取系統之實作

1 林俊宏　2 蔣朝勛

國立中山大學資訊工程學系

[1]lin@cse.nsysu.edu.tw, [2]adks3489@gmail.com

摘要

近年來雲端儲存空間盛行，我們習慣將大大小小甚至是重要資料存放在雲端儲存空間上，然而我們所上傳的重要資料卻有可能被雲端儲存服務的公司私下查看，因此對個人的隱私性以及企業的機密性造成很大的威脅，但若是要自行建立雲端儲存空間，所需耗費的成本又十分龐大。本論文主要探討如何實作一個具有隱私性的雲端快取伺服器，該系統主要分為兩個部份，第一部份為本地端的快取伺服器，第二部份則與雲端儲存空間連結，使用者可透過系統將檔案加密後上傳到雲端儲存空間，並且在本地端的伺服器上存放常用的檔案以減少從雲端儲存空間存取檔案的時間。當使用者需要使用檔案時，系統會自動從雲端儲存空間上下載使用者所需的檔案，另外也可分享檔案給同系統內其他的使用者使用。透過這樣的系統，我們不再受限於現有雲端儲存空間平台的限制，可以增進重要資料的安全性，同時也可以提昇存取資料的效能，不需要耗費大量的成本，就解決了現有雲端儲存服務的兩大缺點。

關鍵字：雲端儲存空間、雲端儲存閘道器

ABSTRACT

Cloud storage is very popular in recent years. People usually store their important data in cloud storage. However, the cloud storage provider might secretly access our data. It is really a big threats to personal privacy and confidentially of business information. Even if we want to build our own cloud storage system, it will cost a lot of resources.

This paper will discuss on how to implement a secure cloud storage caching system. The system consists of two main parts: local cache server and the connection between local cache server and cloud storage. Users can encrypt and upload their data to the cloud storage through this system. When they need to access their data, they will using this system to download and decrypt data from cloud storage. Besides, the system will preserve some data in local storage as cache, so it will not necessary that to download files from cloud storage every time. Users can also share their files to other users in the system.

Though systems like this, we will not limit by the disadvantages of cloud storage. We could improve the security of data and the efficiency of accessing data at the same time with low-cost.

Keyword: *Cloud storage, Cloud storage gateway*

1. 序論

目前雲端硬碟已廣為普及，幾乎是人人都擁有一個以上的雲端儲存空間，雲端儲存空間讓我們能夠隨時隨地在不同的裝置上存取、分享檔案。如今遇到一種問題，雲端硬碟公司可能因為某些需求而去查看我們的檔案，如研究與分析廣告的目的。我們無法保證雲端硬碟公司不會擅自讀取我們的文件，所以許多人不敢將機密或重要文件上傳至雲端儲存空間。也因此，有許多公司想要建立自己私有的雲端硬碟系統，但若要自行建立雲端硬碟的系統，除了需要軟體上的支援外，還需要大量的硬體設備，

所需要的硬體成本包括作為系統的機器以及做為儲存空間的硬碟，而其中需要的硬碟數量更是會隨著使用需求以及時間而增加，甚至還需考慮硬體故障時的維修成本，更嚴重是資料有可能會遺失，若要避免這些情況也要耗費不少的成本。因此我們希望建立一個雲端硬碟系統能夠讓我們不需自行準備大量硬體設備，直接將現有的雲端硬碟做為我們系統的最終儲存空間，並且將資料加密過後才上傳至雲端硬碟以確保資料的安全性和隱私性。

因此本論文將探討如何建置一個具有隱私性且不需自行架設大量硬體設備的雲端硬碟系統，使用者可以透過此系統將檔案加密後上傳至我們的系統以後，系統會自動將檔案上傳至雲端儲存空間上，使用者也可以透過本系統取出或分享已上傳的檔案。除此之外，我們會探討如何建立配合的快取系統，快取系統會將使用者經常使用的檔案備份一份於本地的雲端硬碟系統上，需要使用檔案時不必皆從遠端的雲端儲存空間抓取，可增進使用者的存取效率。

2. 背景知識

2.1 雲端儲存空間

雲端儲存，透過網路把資料存放到遠端第三方的伺服器上而非存到本地端的硬碟上。目前有許多業者都推出了雲端檔案的儲存服務，包括：Dropbox、Google Drive、OneDrive、iCloud、ASUS WebStorage、Amazon S3 等等。這些服務各有優缺點，但都提供了基本額度的免費空間，當使用者有更大空間需求時，可再透過付費的方式增加可用的空間。利用第三方的雲端儲存空間除了便利等好處之外，也有他的缺點，首先就是企業最關心的安全性問題，畢竟將重要的機密文件存放在第三方的伺服器總是會讓人擔心是否有外洩的問題，也因此阻擋了許多想要使用第三方雲端儲存空間的企業。其次是速度問題，因為檔案儲存於遠端的第三方伺服器內，我們若想要存取檔案就必須透過廣域網路 WAN(Wide Area Network)，因此網路速度不會比在區域網路 LAN(Local Area Network)存取還要快速。

接下來會介紹一些現行的雲端儲存平台，首先是本系統後端所使用的 Dropbox，Dropbox 這間公司提供的平台服務有雲端儲存、檔案同步、個人雲端空間、客戶端應用軟體等等。個人雲端空間即為每個使用者都有自己個人的雲端空間，我們可以在上面進行儲存、讀取、建立、刪除、修改等動作。而檔案同步的部份，Dropbox 本身提供一個網站可以讓使用者在上面操作自己的雲端儲存空間，除此之外 Dropbox 還提供客戶端應用軟體可以讓我們在電腦上設立特定的資料夾，當我們對該資料夾的內容進行更動時，Dropbox 會自動同步檔案至個人的雲端空間上。

在應用程式介面的部份，Dropbox 將所有的功能主要分為三個部份：Datastore API、Sync API 與 Core API，其中最重要的是 Core API，其提供了最重要的操作，如：存取檔案、複製、移動、修改名稱等等。Core API 提供的 SDK 有：Python、Ruby、PHP、Java、Android、iOS、OS X 等。但其實這些 API 都是以 RESTful Web API，所以也可以採取 HTTP 的方法去對 Dropbox 進行操作。

2.2 加密技術

對稱式加密為加密演算法中的一種類別，這種演算法於加密以及解密時，所使用的金鑰是同一把，也因此稱為對稱式加密。

非對稱加密為加密演算法中的一種類別，這種演算法於加密以及解密時，所使用的金鑰是非對稱式的金鑰，非對稱式金鑰是指一對加密金鑰與解密金鑰，這兩個金鑰是數學相關，用某使用者金鑰加密後所得的訊息，只能用該使用者的解密金鑰才能解密。

本系統在加密上會使用到兩種加密方法 AES 與 RSA，首先在檔案加密是採用對稱式金鑰加密法，然後再利用非對稱式金鑰加密法將 AES 金鑰加密，這樣做的原因是非對稱式金鑰加密法在加解密速

度上比較快速，所以拿來加密長度較小的 AES 金鑰，但是安全性上 RSA 比較好，因此利用 RSA 加密來兼顧速度與安全。

3. 系統架構

本節會介紹系統的架構，在操作上使用者會透過瀏覽器連上本地端安全快取伺服器上的網頁伺服器，透過網頁操作上傳、下載、刪除等功能以後，伺服器再將所做的修改傳至後端的雲端儲存空間。如圖 3-1，主要會將運作過程分為兩大部份，一部份在本地端的區域網路內，另一部份都是透過廣域網路連上雲端儲存空間。在區域網路內，使用者可透過個人電腦、筆記型電腦、智慧型手機、平板電腦等裝置透過瀏覽器進行操作，操作完成後，本地端的伺服器會依照情況決定是否修改雲端儲存空間的內容，例如上傳時，則將上傳至本地端伺服器的檔案也上傳至雲端儲存空間。

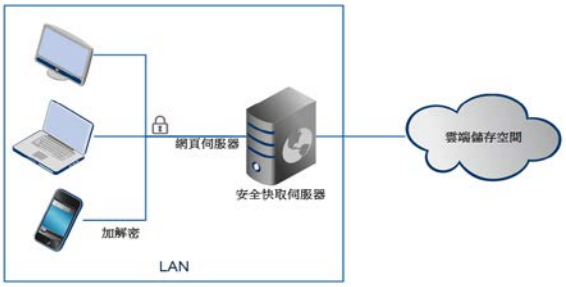

圖 3-1　　系統運作圖

4. 實作技術探討

4.1 上傳及加密

本節主要介紹客戶端在上傳檔案時，系統會進行的操作流程，如圖 4-1 所示為上傳的大致流程。在上傳檔案進行後，首先會透過 HTML5 File API 去讀取檔案內容，讀取檔案內容後會送進加密系統將檔案進行加密，在加密完成以後再把檔案內容送進傳輸的地方，傳輸的地方為了方便使用者看到目前的進度，所以是透過 XMLHttpRequest 和伺服器上的 PHP 進行溝通並傳輸檔案。在檔案上傳時，會先將檔案切成小區塊(chunk)，不管是讀取檔案內容、加密、傳輸，每個流程所收到的資料都是檔案的其中一個小區塊，每個區塊的大小為 65535 bytes，這樣做可以避免在大檔案傳送時，網頁伺服器端因為沒有開啟大檔案設定，而無法進行傳輸。所以從圖 4-1 我們可以看到讀取、加密、傳輸都是一次對一個 chunk 進行處理，等到所有的 chunk 都讀取完以後，才會完成上傳的步驟。

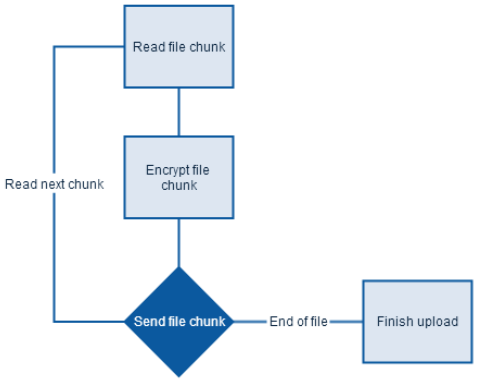

圖 4-1 上傳流程

4.2 下載及解密

本節主要介紹客戶端在下傳檔案時，系統會進行的操作流程，下載檔案的流程幾乎跟上傳的流程相似，如圖 4-2 所示。跟上傳時一樣，在下載檔案時，為了方便使用者看到目前的進度，會先透過 XMLHttpRequest 和伺服器上的 PHP 進行溝通並取得加密過的檔案內容，然後再透過 HTML5 File API 去讀取檔案內容，讀取檔案內容後會送進解密系統將檔案進行解密，每一個區塊解密完成後會透過 HTML5 FileSystem API 將內容附加到一個暫存檔案，在所有區塊解密完成以後，會模擬使用者點擊下載的動作，跳出一個視窗讓使用者選擇要將檔案存放的位置，之後便會將暫存檔移動至使用者要求的位置。

總共流程大致有六項：從 Dropbox 下載、檔案金鑰解密、讀取檔案內容、解密檔案內容、附加至暫存檔案、完成下載檔案。進行檔案下載時，會先

由伺服器上的快取系統檢查檔案是否存在本地端的伺服器，若是不存在的話，才會將檔案從 Dropbox 上抓取至本地端的伺服器。在進行檔案解密前，客戶端會需要先從伺服器上取得一組 AES Key 和 Initialization vector 才能夠解密檔案，從伺服器取得這一組金鑰以後，因為金鑰已先使用 RSA 加密，所以我們在客戶端上需要先利用 JSEncrypt 將金鑰以 RSA 解密過後才能使用。讀取部份與上傳檔案的讀取部份類似，將內容讀取之後再送去解密。解密檔案時，則是於客戶端上利用 AES 解密演算法來解密，模式也是選擇 CBC 模式，因此除了 AES 的 Key 以外，還會需要一個 Initialization vector，所以在下載流程的一開始會先跟伺服器索取，索取完以後再透過使用者的 RSA Private Key 將二者解密供這邊使用。附加檔案的流程，我們必須把所有的檔案區塊都附加到一個暫存檔案，等所有區塊都解密完，才會將此暫存檔案送給使用者。完成下載檔案的部份，則是先取得 Blob 的 URL 以後，再將網頁上事先埋好的超連結元件做修改，首先修改超連結下載的檔名、再來是將超連結的 URL 修改為 Blob 的 URL，都修改完成以後會去模擬點擊超連結的動作，如此就會在使用者的視窗上出現與平常下載檔案一樣的畫面。

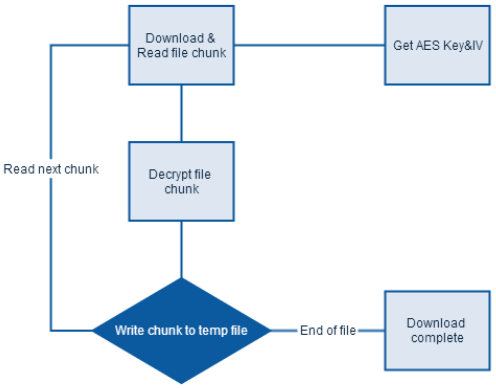

圖 4-2　　下載流程

4.3 分享檔案

在本系統內使用者可以選擇將檔案分享給其他使用者，而在分享檔案時，會先遇到第一個問題是所有上傳至伺服器的檔案都是經過加密的，被分享者即使可以把分享者的檔案下載回去，也沒有辦法使用。所幸每個使用者在上傳檔案時，系統會自動將 AES 的金鑰一起上傳至伺服器，但是第二個問題是為了確保系統沒有解密使用者檔案的權限，我們必須在上傳 AES 金鑰前，先將 AES 金鑰以 RSA 加密過，如此一來，被分享者就算拿到檔案的 AES 金鑰也無法解開，更謬論拿去解密檔案了。

因此，本系統有一個特定機制來處理以上遇到的兩個問題，使用者在分享檔案時，會先跟伺服器索取欲分享之檔案的 AES 金鑰，以及被分享者的 RSA Public Key，取得這兩者之後，系統會用分享者的 RSA Private Key 將檔案的 AES 金鑰解開，解開之後再利用被分享者的 RSA Public Key 將該把解密後的 AES 金鑰加密，最後再將新加密過後的 AES 金鑰上傳回伺服器存於資料庫的 Share Table。

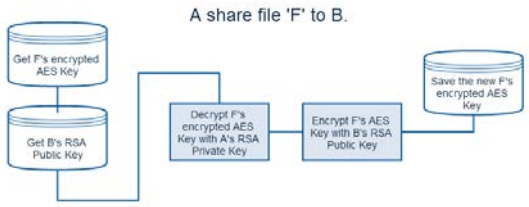

圖 4-3　　使用者 A 分享檔案 F 給使用者 B 流程

4.4 快取系統

由於使用者與本系統伺服器通常位於同一個區域網路底下，所以傳輸的速度不會受到外部連線的影響；若只將檔案存放在遠端的 Dropbox，那整個流程的速度還是會受到外部連線的影響。為了提昇存取的效能，本系統除了在 Dropbox 上存有使用者的檔案以外，也在本地伺服器上做了一個快取系統，該快取系統會在本地端的伺服器內存放使用者常用的部份檔案，傳輸的速度不易受到對外連線至 Dropbox 的速度所限制，可以大幅提昇存取速度。

快取系統預設讓每個使用者可以在本地伺服器上保留 10 個檔案，當使用者超過 10 個檔案時，快取系統會透過 Least Recently Used 的演算法來將

最不常被使用的檔案移出伺服器，並且將新的檔案取代進來。所有的快取檔案都會紀錄於資料庫內的 Cache 資料表，Cache 資料表內有兩個欄位：fileid 與 counter，fileid 是代表哪個檔案，而 counter 的內容則為時間戳記(Timestamp)代表上次使用快取的時間。

快取的替換通常發生在使用者上傳檔案或下載檔案並且快取額滿的時候，第一種情況，上傳檔案且快取額滿時，系統會先進資料庫內的 Cache 資料表尋找該使用者現有 Cache 其 counter 的最小值，即取出時間戳記為最久以前的，將其以新上傳的檔案取代，並將新上傳檔案的 counter 設為當前時間的 Timestamp。第二種情況是下載時，由於本系統內的所有下載要求都會先經過快取系統，這時候快取系統會先尋找要求的檔案是否存在於伺服器內，若是存在則為快取命中，會先更新 Cache 資料表內該檔案的 counter 為當前的 Timestamp，若是不存在則為快取失效，一樣會去找出 Cache 資料表該使用者現有 Cache 其 counter 的最小值，接著連線至 Dropbox 將要求的檔案下載回來把該檔案替換掉，並且將 counter 設為當前的 Timestamp。

4.5 任務排程

在客戶端上會有一個任務排程系統，使用者所有上傳以及下載的任務都會透過任務排程系統，任務排程裡採用的是 Queue 來安排上傳下載的執行順序，在使用者要進行上傳或是下載時，會先新增一個 Job，然後送進 JobQueue 裡的 waitingQueue 去等待被執行，runningQueue 內的數量即執行中的 Job 數量小於傳輸數量限制時，則會從 waitingQueue 取出第一個 Job 來執行，並且將該 Job 從 waitQueue 移出放入至 runningQueue 裡面。當 Job 執行完畢時，則會將該 Job 從 runningQueue 裡移除，如此才能讓下一個 Job 開始執行。

5. 系統分析與比較

現行類似的系統有：HopeBay ArkExpress, Riverbed SteelStore，本章節將會先簡單介紹這兩個系統，再與本系統進行比較。

5.1 Hope Bay ArkExpress

ArkExpress 是和沛科技所推出的產品，這項產品主打的特色有：可擴充性、可相容性、存取效能、安全加密、彈性安裝。這套系統有一個最大的特色是，它將檔案上傳至後端雲端儲存空間的設計，該系統在使用者將檔案透過 CIFS、NFS、rsync 等將檔案傳輸至伺服器時，不會立即將該檔案傳輸至後端的雲端儲存空間，而是等到網路的離峰時刻才將檔案傳輸至後端的雲端儲存空間，這樣的行為可以避免系統在平常佔用網路頻寬，也可以利用到離峰時刻的網路頻寬。

5.2 Riverbed SteelStore

SteelStore 則是由 Riverbed 所推出的產品，其主打特色與 ArkExpress 是非常類似的，主打特色有三點：Efficient、Proven、Secure，Efficient 方便，SteelStore 強調的是會去壓縮檔案，所以傳輸量會大幅減少，速度也因此增加。其餘部份皆與 ArkExpress 大致相同。

5.3 簡單比較

接下來就 ArkExpress、SteelStore 與本系統進行簡單的比較，項目如表 5-1：

	ArkExpress	SteelStore	本系統
加密	AES-256	○	AES-256
加密處	伺服器	伺服器	客戶端
LAN 傳輸經過加密	X	X	○
快取	○	○	○
客戶端界面	CIFS, NFS, rsync	CIFS, NFS	Web
後端儲存	Dropbox	AmazonS3	Dropbox

表 5-1　　各系統之比較

從表 5-1 我們可以看到 ArkExpress 與 SteelStore 在使用者界面上是與本系統有最大差別的地方，這兩個系統採用的皆是 CIFS 與 NFS，這種傳輸方式對使用者會是較方便的一種做法，使用者通常不需要另外安裝額外的軟體，只需要經過一些簡單的設定就能使用，但是在一般情況下，這兩種界面在進行傳輸的過程中，其資料通常不會經過加密，加密的動作都是在伺服器上執行，即使是在同一個區域網路下，資料沒有經過加密就傳輸也是非常危險的，因為一個檔案的存取權限也不會是開放給所有使用者的，所以當沒有權限的使用者去竊聽區域網路，就有可能取得非他權限所能取得的資料。在本系統上，所有的加密都是在客戶端上進行的，所以在區域網路的傳輸過程中，資料也是經過加密的，也因此其他客戶端和系統都完全無法得知資料內容，甚至是加密資料用的金鑰，這點是我認為本系統較為安全的部份。

6. 結論

本論文主要實作一個安全的雲端儲存閘道器，可以將檔案加密後上傳至雲端儲存空間，並且將常用的檔案在本地端預留一份作為快取，避免耗費大量資源自行建置私有雲端儲存空間，快取系統還可以提昇存取效能。本系統也可以將檔案分享給系統內的使用者，使得使用者在操作上更加便利及安全。另外，本系統還提供 Docker Image，能夠讓用戶快速安裝此應用至雲端服務平台。

在未來本論文希望先朝支援多種雲端儲存空間邁進，目前後端的儲存空間只有支援 Dropbox，但是因為目前雲端儲存空間的應用程式界面大多都是採用 RESTful Web API，所以可以在不需要耗費太多功夫的情況下就大大提昇便利性。其次，若能有像目前 Dropbox、Google Drive 等在客戶端系統上的應用程式，能夠自動同步且加密特定資料夾的運作方式，以此達到 CIFS、NFS 的便利性，並且提昇安全性。

參考文獻

[1] Huijun Xiong, Xinwen Zhang, Danfeng Yao, Xiaoxin Wu, and Yonggang Wen. 2012. Towards end-to-end secure content storage and delivery with public cloud. In Proceedings of the second ACM conference on Data and Application Security and Privacy (CODASPY '12). ACM, New York, NY, USA, 257-266.

[2] Idilio Drago, Marco Mellia, Maurizio M. Munafo, Anna Sperotto, Ramin Sadre, and Aiko Pras. 2012. Inside dropbox: understanding personal cloud storage services. In Proceedings of the 2012 ACM conference on Internet measurement conference (IMC '12). ACM, New York, NY, USA, 481-494.

[3] Huijun Xiong, Xinwen Zhang, Danfeng Yao, Xiaoxin Wu, and Yonggang Wen. 2012. Towards end-to-end secure content storage and delivery with public cloud. In Proceedings of the second ACM conference on Data and Application Security and Privacy (CODASPY '12). ACM, New York, NY, USA, 257-266.

[4] Min-Yu Chen, Chi-Wei Liu, and Min-Shiang Hwang. 2013. SecureDropbox: a file encryption system suitable for cloud storage services. In Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference (CAC '13). ACM, New York, NY, USA, Article 21.

[5] Seny Kamara and Kristin Lauter. 2010. Cryptographic cloud storage. In Proceedings of the 14th international conference on Financial cryptograpy and data security (FC'10), Radu Sion, Reza Curtmola, Sven Dietrich, Aggelos Kiayias, Josep M. Miret, Kazue Sako, and Francesc Sebe (Eds.). Springer-Verlag, Berlin, Heidelberg, 136-149.

[6] Jianying Zhou. 2014. On the security of cloud data storage and sharing. In Proceedings of the 2nd international workshop on Security in cloud computing (SCC '14). ACM, New York, NY, USA, 1-2.

[7] Barry Leiba. 2012. OAuth Web Authorization Protocol. IEEE Internet Computing 16, 1 (January 2012), 74-77.

[8] Antonio Garrote Hernandez and Maria N. Moreno Garcia. 2010. A formal definition of RESTful semantic web services. In Proceedings of the First International Workshop on RESTful Design (WS-REST '10), Cesare Pautasso, Erik Wilde, and Alexandros Marinos (Eds.). ACM, New York, NY, USA, 39-45.