國立交通大學

資訊學院 資訊學程

碩士論文

有效改善網路檔案儲存安全性之架構設計

Effective architecture design on network files access security improvement

研究生　　：戚得郁

指導教授　：黃俊龍 博士

中 華 民 國 一 〇 三 年 六 月

有效改善網路檔案儲存安全性之架構設計

Effective architecture design on network files access security improvement

研 究 生：戚得郁　　Student：Te-Yu Chi
指導教授：黃俊龍 博士　　Advisor：Dr. Jiun-Long Huang

國 立 交 通 大 學
資訊學院 資訊學程
碩 士 論 文

A Thesis
Submitted to College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science
June 2014

Taipei, Taiwan, Republic of China

中華民國一〇三年六月

有效改善網路檔案儲存安全性之架構設計

學生：戚得郁　　指導教授：黃俊龍 博士

國立交通大學　資訊學院　資訊學程碩士班

中文摘要

隨著科技的快速發展，網路技術從早期的撥接上網，到 98 年的單向 Cable，以至現今的 ADSL，網路頻寬不斷以躍進的方式爆炸性成長。

網路原就充滿許多大量的資料交換，依賴著硬體技術的不斷進化，更多的功能、服務都可以輕而易舉的實做，應用層面亦更為廣泛。藉由網路存取資料已成為各種裝置必備之功能，許多軟硬體供應商也涉足雲端儲存技術，提供雲端存取服務；然而在享受這些服務的過程中，資料安全的疑慮卻也在無形中增加暴露風險。

目前一般網路安全著重在使用者的檔案加密，藉以防止其他使用者或是駭客的竊取，但卻忽略了提供儲存資料的服務供應商，有機會更輕而易舉的取得使用者資料。以著名的相片儲存服務軟體 Instagram 為例，就曾於 2012 年 12 月宣佈修改隱私權政策及服務條款，可在不通知使用者的情況下，將用戶的照片提供給廣告商。使用者資料被窺視的情況層出不窮，如何防範這種情況發生，即是本研究所關注的重點。

本研究以提昇使用者資料儲存之安全性為出發點，希望防止提供儲存空間的管理者對於資料的窺視及濫用；藉由設計各種資料儲存模組，並導入適宜的加密演算法，同時測試、比較模組的效能，有效的提昇資料安全性及資料隱私性。

研究結果指出，本研究所設計的虛擬資料儲存架構，整合 AES Rijndael、DES 等對稱性加密演算法，以 Seek 的方式快速取得所需檔案片段，提昇讀取檔案的速度，且有效強化資料安全性及隱私性，防止資料暴露的風險，降低管理者獲取使用者檔案資訊的可能性。

Effective architecture design on network files access security improvement

Student：Te-Yu Chi Advisors：Dr. Jiun-Long Huang

Degree Program of Computer Science
National Chiao Tung University

Abstract

With the rapid development in technology, the network technology has transformed from the early one way Cable to the current ADSL system. The broadband speed also follows an explosive growth path along with the newest innovation.

Internet consist plenty of information exchange. As the hardware standard improved, more functions and services can be easily performed. As a result, the applications on these technology are everywhere in our daily life. Accessing data through internet become a necessary function on digital devices. Many software providers also started developed such products to provide cloud service. However, the information safety issues arise while people enjoy the convenient services provided by internet technology.

The current internet safety focuses on the file encryption by the user to avoid other users or hackers to access personal data. However, users often neglect the easy access to data from the service providers. Taking Instagram as an example, Instagram declared the edited term of use which allows them to use client's photo for commercial purposes without notice. The focus of this study is to prevent such unwitting events from the users.

This study started from users' data security and further discusses the prevention on illegal use of data by the service providers. Using different type of data saving module along with appropriate encrypted algorithm to test out the most efficient and privacy-preserving one. The result of this research shows that the architecture of data warehouse with synchronous encryption algorithm AES Rijndael, DES can significantly improve data security and privacy and avoid the risk of data leakage, furthermore, to lower the possibility of administrator accessing users' file information.

致　　謝

兩年的交大研究所，即將隨著論文的結束告一段落，終於順利地走到這一步，回顧過去兩年的辛苦，一切值得。這一路上陪伴在左右的人很多，每一個重要的時刻，有你們在身邊，是一件幸福的事。

第一個要感謝的是我的家人，包括我的父母與我的姊姊，不論在任何時刻，總是默默在我背後給予最大的支持與肯定，能夠從交大畢業的這份榮耀，不僅屬於我，更屬於你們，我愛你們。

第二個要感謝的是我的指導教授黃俊龍教授；非常慶幸能夠有您擔任我的指導教授，雖然相處的時間不多，但是您總是能在我需要被提醒的時候拉我一把，給我一個方向，並且給了我很多啟發，謝謝您。

第三個，我要感謝彭艷平小姐，這一路辛苦妳了，不論在工作上、課堂上甚至是口試時，總是能看到妳的身影，謝謝妳一路都陪在我的身邊，陪我度過我的各種難關，忍受我的情緒失控，謝謝妳。

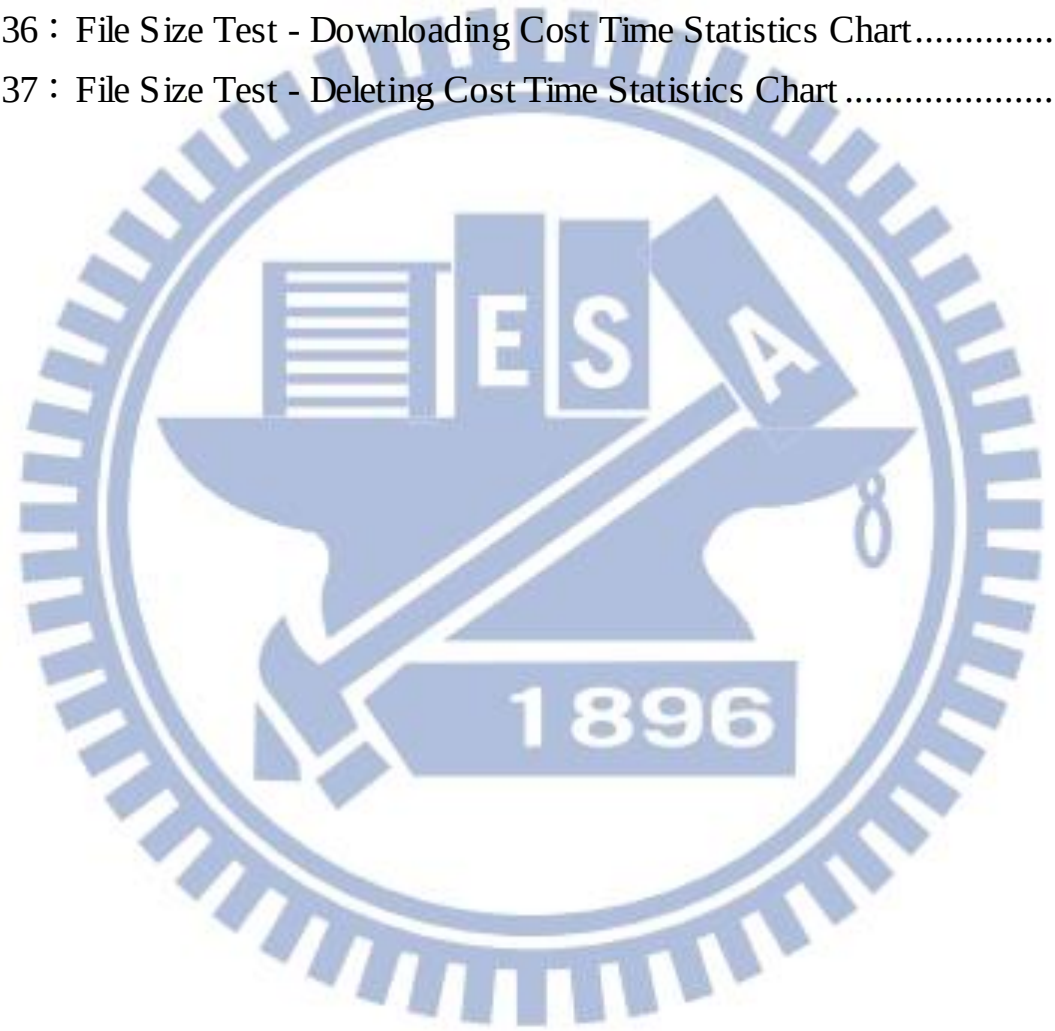
第四個，我要感謝幾位我最重要的好朋友，陳建中、李航、李致廷、周彥儒，我的生活，一切的喜怒哀樂因為有你們而豐富，如果沒有你們，將沒有今天的我，謝謝你們。

最後，很開心能夠順利地如期畢業，交大是一所非常棒的學校，我將會以身為交大生為驕傲，往下一個目標繼續向前努力。

目錄

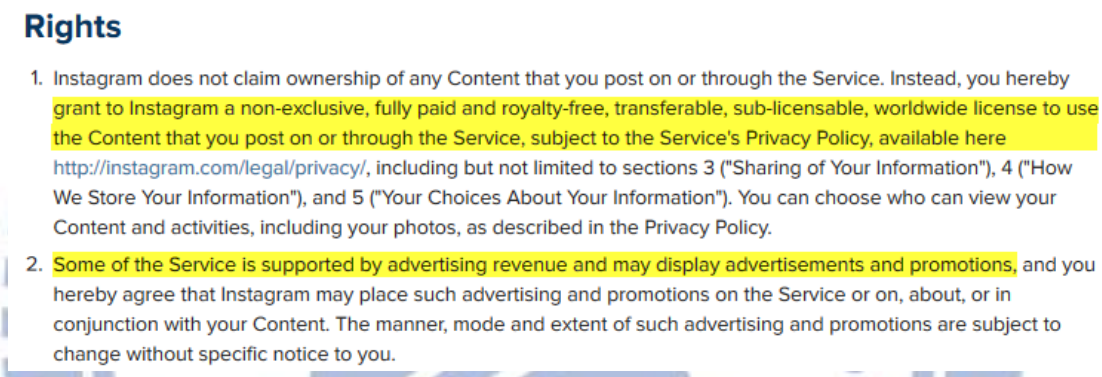
表目錄

圖目錄



一、 緒論

網路儲存技術已漸趨成熟，舉凡 Microsoft、Dropbox、Google、Synology 等軟硬體廠商，皆跨足雲端網路儲存領域，提供使用者完善的網路儲存服務；然而便利服務的背後充斥的安全性的疑慮。目前一般網路安全著重在使用者的檔案加密，藉以防止其他使用者或是駭客的竊取，但卻忽略了提供儲存資料的服務供應商，有機會更輕而易舉的取得使用者資料。以著名的相片儲存服務軟體 Instagram 為例，就曾於 2012 年 12 月宣佈修改隱私權政策及服務條款，可在不通知使用者的情況下，將用戶照片提供給廣告商。圖 1 為 Instagram 公告的隱私權政策節錄。



Rights

1. Instagram does not claim ownership of any Content that you post on or through the Service. Instead, you hereby grant to Instagram a non-exclusive, fully paid and royalty-free, transferable, sub-licensable, worldwide license to use the Content that you post on or through the Service, subject to the Service's Privacy Policy, available here http://instagram.com/legal/privacy/, including but not limited to sections 3 ("Sharing of Your Information"), 4 ("How We Store Your Information"), and 5 ("Your Choices About Your Information"). You can choose who can view your Content and activities, including your photos, as described in the Privacy Policy.
2. Some of the Service is supported by advertising revenue and may display advertisements and promotions, and you hereby agree that Instagram may place such advertising and promotions on the Service or on, about, or in conjunction with your Content. The manner, mode and extent of such advertising and promotions are subject to change without specific notice to you.

圖 1：Instagram 隱私權政策節錄

資料來源：Instagram 官方網站（http://instagram.com/about/legal/terms/#）

部份使用者皆有使用 NAS 作為檔案備份的經驗，NAS 設備廠商為跟進雲端服務領域，亦開始提供使用者雲端儲存服務的功能。當使用者扮演管理者的角色，深刻的體會到管理者的權限是強大的，所有資料對於管理者而言，皆是容易被存取的。

使用者將重要資料上傳至雲端磁碟，卻忽略了資料可能已被管理者窺視使用的可能。因此開始思考解決方式，以目前現有的策略中，大多以檔案加密為解決方案，在用戶端將檔案進行加密，並將加密後的檔案上傳至雲端空間。以商業軟體 Boxcryptor 為例，其提供使用者在本機端掛載一顆虛擬硬碟，並將檔案於本機端進行加密，完成後再將加密後的檔案上傳至雲端空間。圖 2 為 Boxcryptor 官方網站對於軟體的說明。

What is Boxcryptor?

Boxcryptor is an **easy-to-use encryption software optimized for the cloud**. It allows the secure use of cloud storage services without sacrificing comfort. Boxcryptor supports all major cloud storage providers (such as Dropbox, Google Drive, Microsoft SkyDrive, SugarSync) and supports all the clouds that use the WebDAV standard (such as Cubby, Strato HiDrive, and ownCloud). With Boxcryptor your files go protected to your cloud provider and you can enjoy peace of mind knowing that your information cannot fall into the wrong hands.

Here is how it works:

Boxcryptor creates a virtual drive on your computer that allows you to encrypt your files locally before uploading them to your cloud or clouds of choice. It encrypts individual files - and does not create containers. Any file dropped into an encrypted folder within the Boxcryptor drive will get automatically encrypted before it is synced to the cloud. To protect your files, Boxcryptor uses the AES-256 and RSA encryption algorithms.

圖 2：Boxcryptor 功能說明

資料來源：Boxcryptor 官方網站（https://www.boxcryptor.com/en/boxcryptor）

實際測試 Boxcryptor 後，其操作流程及結果如下：

（1）於本機模擬虛擬磁碟（磁碟機標籤為 X）

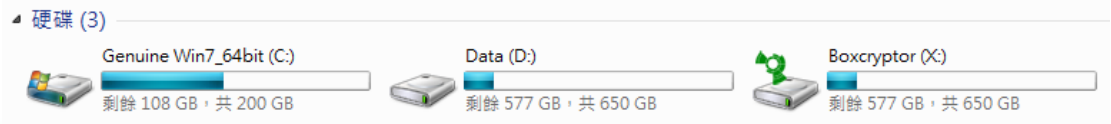


圖 3：Boxcryptor 掛載虛擬磁碟顯示圖

（2）將欲上傳之檔案拖移至磁碟中，跳出以下訊息，確認是否加密

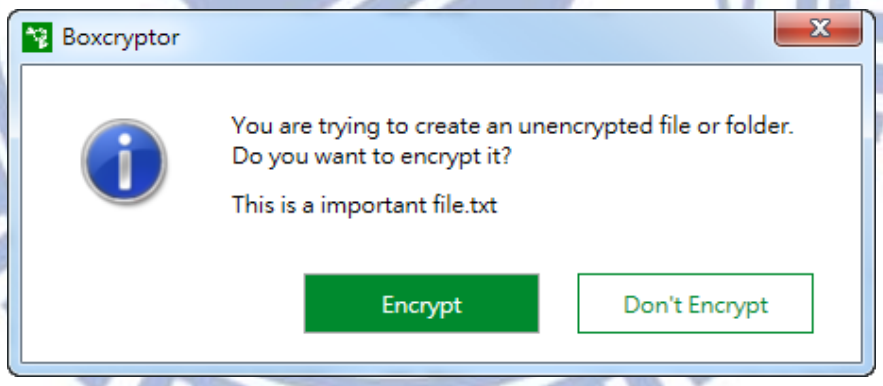


圖 4：Boxcryptor 確認視窗，詢問使用者是否進行檔案加密動作

（3）加密完成後則產生加密檔(.bc)

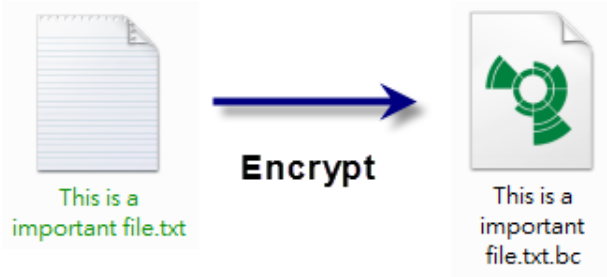
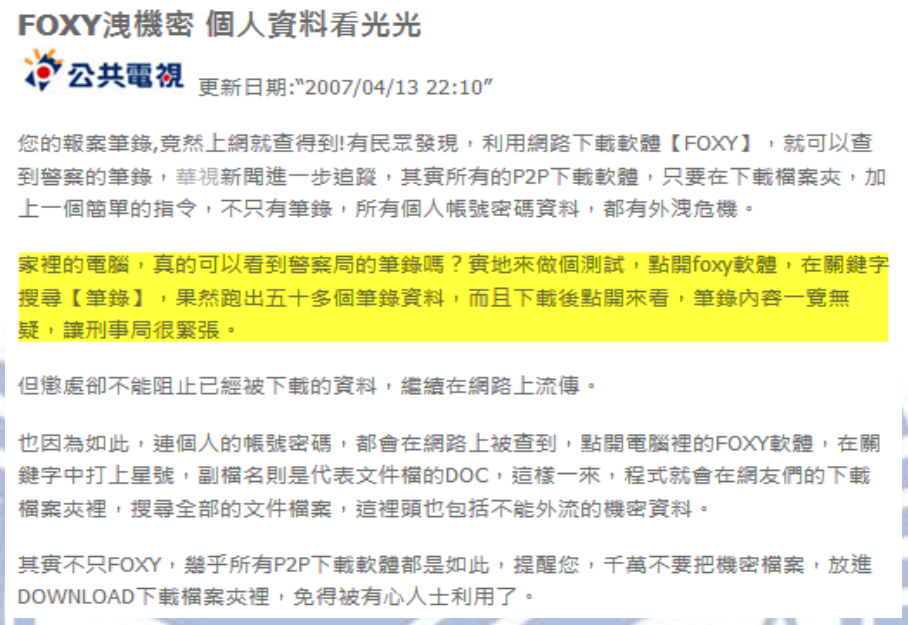


圖 5：Boxcryptor 輸出檔案結果

從測試結果可知檔案已被正確的加密，但值得思考的是檔名的資訊仍暴露在外，仍可能有資訊洩漏的疑慮。過去曾有新聞指出，知名 P2P 軟體 Foxy，即發生過數次因檔名資訊而導致資料外洩的紀錄，由新聞事件得到教訓，雖檔案被個別加密，但敏感的檔名資訊，仍可能吸引有心人士注意而遭到破解。



FOXY洩機密 個人資料看光光

公共電視 更新日期:"2007/04/13 22:10"

您的報案筆錄,竟然上網就查得到!有民眾發現，利用網路下載軟體【FOXY】，就可以查到警察的筆錄，華視新聞進一步追蹤，其實所有的P2P下載軟體，只要在下載檔案夾，加上一個簡單的指令，不只有筆錄，所有個人帳號密碼資料，都有外洩危機。

家裡的電腦，真的可以看到警察局的筆錄嗎？實地來做個測試，點開foxy軟體，在關鍵字搜尋【筆錄】，果然跑出五十多個筆錄資料，而且下載後點開來看，筆錄內容一覽無疑，讓刑事局很緊張。

但懲處卻不能阻止已經被下載的資料，繼續在網路上流傳。

也因為如此，連個人的帳號密碼，都會在網路上被查到，點開電腦裡的FOXY軟體，在關鍵字中打上星號，副檔名則是代表文件檔的DOC，這樣一來，程式就會在網友們的下載檔案夾裡，搜尋全部的文件檔案，這裡頭也包括不能外流的機密資料。

其實不只FOXY，幾乎所有P2P下載軟體都是如此，提醒您，千萬不要把機密檔案，放進DOWNLOAD下載檔案夾裡，免得被有心人士利用了。

圖 6：公共電視 2007 年 4 月份歷史新聞節錄

資料來源：公共電視台新聞網 網站（http://news.pts.org.tw/detail.php?NEENO=60489）

以歷史事件為借鏡，希望能研究出一個架構用以加強檔案於網路環境下的隱私及安全性，防止各方的窺伺。此架構的模型應以保險櫃的概念實做，使用者的所有檔案被鎖在保險櫃中，需要時才利用鑰匙打開保險櫃，取出所需的檔案。對於管理者而言，僅能看到一個保險櫃的檔案格式而無法得知保險櫃內有任何形式的檔案及資訊。

本研究所能應用的範圍，以資料儲存服務為範疇，舉凡本機、區域網路、NAS 所提供的網路儲存功能、雲端儲存等皆為此研究可應用的範圍。主要將以以下兩個方向作為研究的重心：

（1）檔案加解密機制。

（2）檔案儲存架構。

為了完成研究目標，將研究重點區分為三個部份進行研究及實驗：

（1）研究檔案加解密演算法效能及加密強度，找出適合的加解密法並導入至架構設計中。

（2）根據現有檔案儲存模組進行測試、設計。

1. 壓縮檔：運用壓縮檔可將檔案上鎖的特性進行架構的實做。
2. File System：嘗試於儲存空間建立虛擬磁碟，並將其視為檔案存放的目標。

（3）將上述的結果進行分析、比較，歸納其效能或安全性，找出可能發生的問題，並重新設計一個架構以改善效能及安全性上的問題。並符合研究目標的需求。

最後，本論文結構共分為五個章節。從第二章開始，將探討可能運用的相關技術研究；以加解密演算法機制、檔案系統為研究重點，進行技術的了解及分析、比較。第三章進行系統架構的實做，並予以說明其程式架構。第四章利用完成的系統架構進行實驗比較，證明架構的優勢。第五章則進行本研究的總結，以及未來展望。

二、 相關研究

2.1 加解密機制

本研究為考量檔案的安全性，檔案的加密是必然需要的。以下根據加解密演算法進行分析並獲得適宜的加密法以導入設計中。檔案的加解密機制主要分為對稱式密鑰加密法（Symmetric encryption）和非對稱式密鑰加密法（Asymmetric encryption）兩種，其定義[1]如下：

（1）對稱式密鑰加密法：

加解密皆使用同一把金鑰，並透過換位、替換、互斥、乘法等運算進行加密動作。資料於發送端使用此金鑰進行資料的加密運算獲得密文，接著將資料送至接收端。接收端若須解密則利用相同的此金鑰進行解密運算重現明文。對於其他沒有獲得金鑰的使用者而言，則無法進行密文的解密。此作法確保了資料的安全性及機密性。常見的加密演算法包括：DES、3DES、RC2、AES 等。下圖為對稱式密鑰加密法的架構。

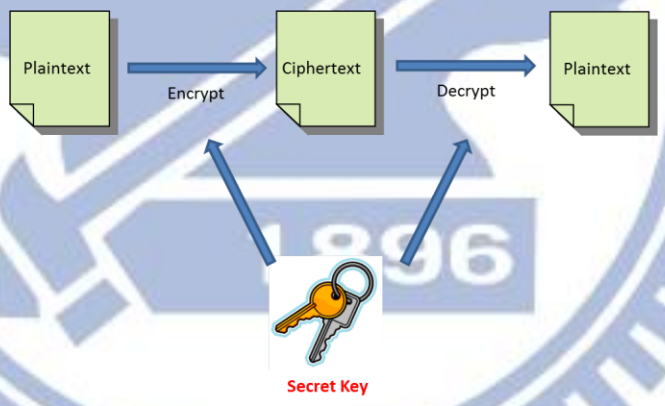


圖 7：對稱式密鑰加密法架構圖

（2）非對稱式密鑰加密法：

利用兩把不同的金鑰（分別為公開金鑰及私密金鑰）交互進行加解密作業。資料於發送端使用私密金鑰進行資料的加密運算獲得密文，接著將資料送至接收端。接收端若須解密則利用公開金鑰將密文進行解密運算以重現明文。此方式除可提昇資料安全性，亦提供了資料的鑑別性能力。由於公開金鑰無法推斷出私密金鑰，又必須利用兩把不同的金鑰進行加解密，使得此類加密法需要較對稱式密鑰加密法更複雜的演算法以達成目的，因此效能上也緩慢許多。常見的演算法如：RSA。下圖為非對稱性密鑰加密法的架構。

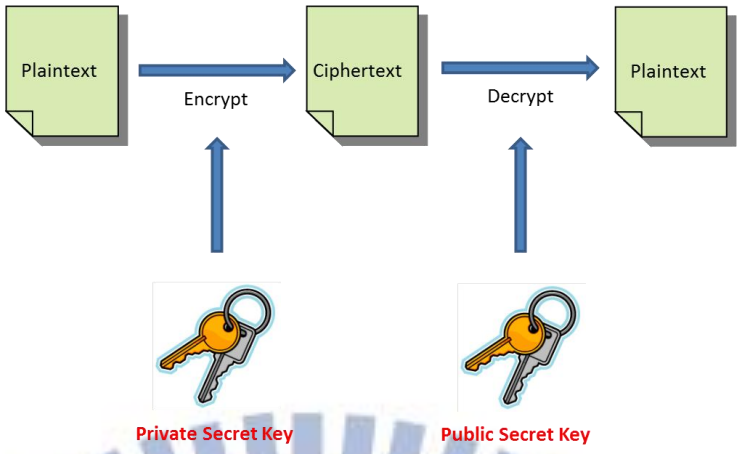


圖 8：非對稱性密鑰加密法架構圖

研究[2]指出，對稱性加密演算法相對於非對稱性是擁有較高效能的，本研究為考量使用者對於效能影響的觀感，將採用對稱性加密演算法作為本研究的加密方式。又以研究[3]中的實驗結果而論，3DES 雖具高度的安全性，但不論加解密時間成本皆與其他的演算法差異過多，故不考慮；而 Rijndael[1]雖相較於 DES 須花費較多的時間成本，但卻可提供較為安全的資料加密，適合應用在重要的資料，例如後續架構設計中提及的 Metadata 設定檔。而以速度較快的 DES 則可應用在後續架構設計中的文件加密，以此提昇文件加密的效能。

2.2 檔案儲存模組

本研究的目的在於如何在遠端的儲存空間建立一個安全且隱密的資料架構，此資料結構可將檔案包含於此，對他人而言，僅能看到一個封裝的結構而無法了解內容，本節將藉由兩個現有且可能達成目標的儲存模組進行說明，分別為資料壓縮技術及檔案系統。

2.2.1 資料壓縮技術

資料壓縮(Data compression)的主要目的在於將一資料來源盡可能的減少其不需要或是多餘的訊息（Data redundancy），進而達到位元數降低，並仍能足以表達其資訊內容的結果。以影像資料為例，透過降低編碼多餘量(Coding Redundancy)、像素間多餘量(Interpixel Redundancy)、心理視覺多餘量(Psychovisual Redundancy)，以減少不必要的位元數但仍能顯示正確的影像資訊。而一般所討論的資料壓縮技術共有兩個演算法，分別為壓縮演算法與重建演算法[4]：

[1] Rijndael 為 NIST 所篩選出的 AES 加解密演算法代表。本研究後續皆以 Rijndael 作為 AES 的敘述。

（1）壓縮：將資料來源 X 透過壓縮演算法輸出一個資料位元較小的 Xc。
（2）重建：將 Xc 透過重建演算法產生 Y，Y<=X。

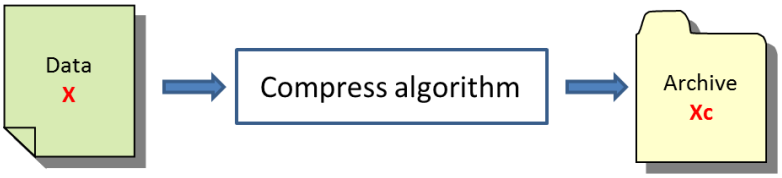


圖 9：資料壓縮架構圖

圖 10：資料重建架構圖

現今廣為人知，較為流行的幾個壓縮格式有 ZIP、RAR、TAR 等，然而 TAR 格式多應用於 Linux 的作業系統，對於大多數使用者的作業系統環境為微軟的前提下較為不適合作為本研究的實驗對象，反觀 ZIP，微軟於作業系統中已內建對於 ZIP 壓縮格式的支援[5]。又以 RAR 作為比較，雖研究[6]中了解 RAR 與 ZIP 互有其壓縮率及壓縮時間上的優點，但壓縮技術對於本研究的價值在於其能達到多個檔案封裝的特色，且格式必須相容更多使用者的作業環境，壓縮率則非研究所討論的重點。綜合上述論點，本研究將以 ZIP 作為研究對象之一。

2.2.2 檔案系統

檔案系統的目的在於提供使用者大量的資料儲存空間，並將資料以檔案的形式存放。而檔案則是由眾多資料所產生的一個集合，將資料以有規則的被儲存，其包含檔案本身、檔案名稱、檔案屬性及其目錄結構。檔案名稱進行檔案的區分，副檔名則提供檔案的類型，檔案屬性提供其相關資訊，包括型態、大小等。

下表根據目前常為人使用的檔案系統格式進行比較[7][8]。由表中得知，NTFS 格式的磁碟架構可應用於較大的檔案格式且其高階資料結構可以提昇效能及空間的使用率，故本研究將以 NTFS 作為研究對象之一。

表 1：FAT16、FAT32、NTFS 比較表

	FAT16	FAT32	NTFS
作業系統	Win 95, Win 98, Win ME		Win XP, Win Vista 及其之後的版本
磁區限制	4GB	2TB（受微軟分割限制，僅支援至 32GB）	2TB
單檔限制	2GB	4GB	16TB（可能受 MBR 限制）
安全性	無	無	提供檔案安全性控管
特色	(1) 讀取速度最快 (2) 大容量的磁碟支援性最差 (3) 結構簡單	(1) 結構簡單 (2) 速度次之 (3) 支援熱拔插	(1) 最大容量的支援 (2) 修復磁碟能力 (3) 高階資料結構，可有效提昇效能，增加空間使用率

2.2.2.1 NTFS

New Technology File System（NTFS）是 Microsoft 目前作業系統主要使用的檔案系統。以物理結構而言，磁碟（Disk）的最小單位為磁區（Sector），磁區大小則視作業系統而定。檔案的儲存最小單位，則是以叢集（Cluster）為單位；所謂叢集，即一組磁區的集合。所有檔案皆儲存於叢集中，即使檔案大小小於叢集大小，仍須佔用一個叢集的空間。而叢集的大小取決於磁碟的分割（即切割為 Partition），當分割越大時，叢集的 Size 也會隨著增加。

切割多個 Partition 的用意在於方便資料的管理（不同 Partition 儲存不同類型的資料，進行資料的分類）及增加資料的安全性（同磁碟的其中一個 Partition 毀損不會影響其他 Partition 資料）。

相對於 Partition，另一種切割則是 Volume；Volume 的主要特色為可以進行跨磁碟的容量擴充，其共有五種不同的儲存方式，分別為：

（1）Simple Volume：一顆磁碟時使用，具擴充能力，但無加速及容錯能力。

（2）Spanned Volume：當需要進行跨磁碟擴充容量時使用，其限制為至少兩顆的磁碟至上限為 32 顆磁碟。但其功能僅於跨磁碟的空間擴充，架構與 Simple Volume 相同，無加速及容錯能力。

（3）Striped Volume：至少需要兩顆磁碟（至多 32 顆磁碟）進行資料的儲存。將資料進行 64KB 的切割，並依序儲存至磁碟中。此模式由於可同時進行多磁碟的存取，可大幅提昇存取效能，但因資料被切割分散在各磁碟，很有可能因其中一顆磁碟的毀損及造成資料的損壞。

（4）Mirrored Volume：至少需要兩顆磁碟（至多 32 顆磁碟）。此種模式主要用意在於將多磁碟切割出一塊相同的大小，並在寫入一份資料時，同時寫入到多顆磁碟中，用以確保資料的安全容錯能力。但此種方式亦會導致儲存空間的減少，且無法對資料存取有加速的效果。

（5）RAID-5：至少需要三顆磁碟（至多 32 顆磁碟）。此種模式同時使用了類似 Striped 及 Mirror 的模式。以三顆磁碟為例，同時切割相同大小進行資料的儲存，當資料進行儲存時，以不固定的方式選擇兩顆磁碟進行資料依序的寫入（類似 Striped 模式），並由剩下的一顆磁碟進行資料的校驗值儲存。以不固定方式的好處是每顆磁碟不會僅是扮演相同角色，而當其中一顆磁碟故障時，仍能由另外兩顆磁碟進行資料恢復的作業。此種方式確保了資料的安全性，並也提昇資料寫入的效能。惟須增加硬體成本。

在 NTFS 磁碟格式中，利用 Logical Cluster Number（LCN）及 Virtual Cluster Number（VCN）進行檔案的相對位置及絕對位置的定位，藉由 LCN 及 VCN 可對檔案進行存取的動作。檔案的資訊及格式儲存於檔案配置表中，而 NTFS 的檔案配置表稱為 Master File Table（MFT），其作用為佔用磁碟最前面的一塊空間，用以紀錄最重要的檔案資訊，例如檔名、檔案大小、相關時間以及檔案的位置索引等。下圖說明 NTFS 在進行檔案存取時的實際情況。

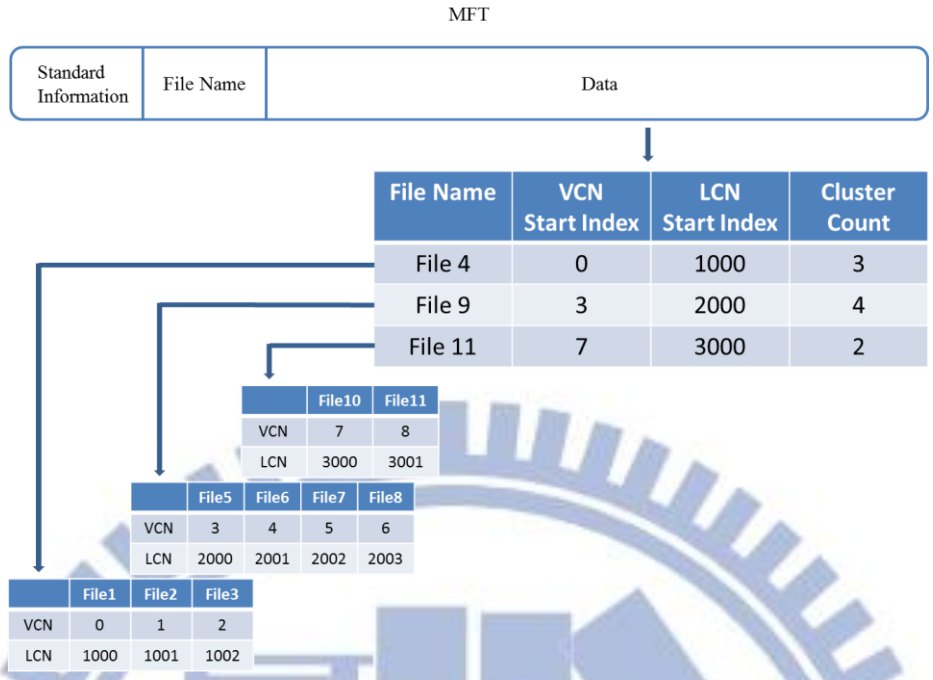


圖 11：NTFS 檔案結構

2.2.2.2 NFS

Network File System（NFS）由 Sun Microsystems and IBM 於 1984 年所發展。其功能主要透過網路方式及使用者的帳號及密碼進行資料夾/檔案的權限控管，在各平台上分享、共用其檔案內容。憑藉著 NFS 對於檔案存取的權限能力強大，恰符合本研究所須的檔案處理技術，本研究將以 NFS 架構作為 NFS 作為資料傳輸、處理的方式。下圖說明 NFS 架構。

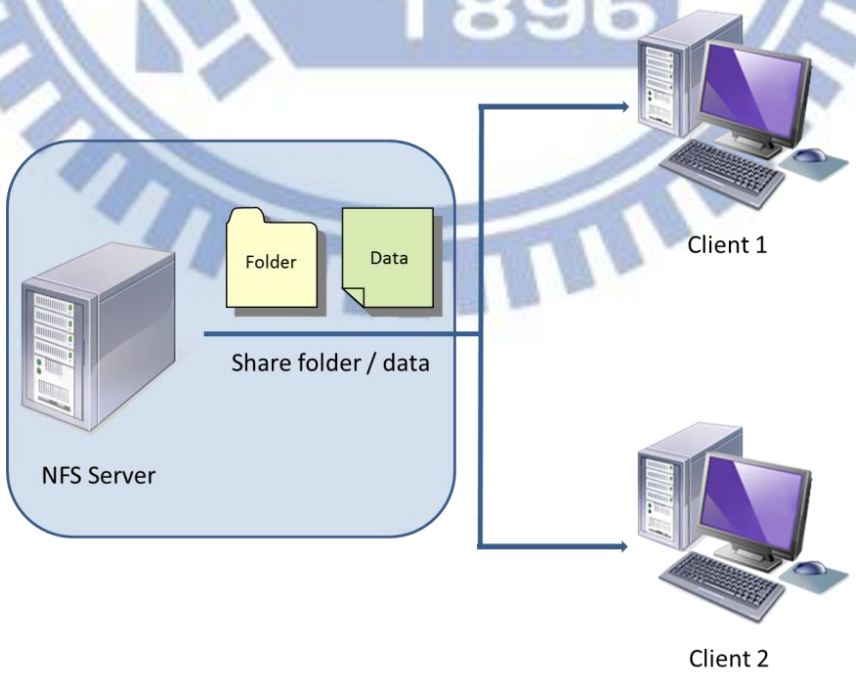


圖 12：NFS 架構

NFS 支援非常多的檔案存取功能，所以需要使用到 Port 作為溝通，需透過註冊 Remote Procedure Call（RPC），可將每個功能的 Port Number 給予對應，並回傳至用戶端以進行雙方的連結，進而達到資料的傳輸、使用等功能。首先，當 Client 對 Server 提出程序或指令的執行要求時，會將指令及程序傳送至 RPC 進行分析及對 Server 的 Port Number 對應，對應完成後，Server 會進行使用者的權限管理，確認主機是否有此使用者，利用使用者的 UID、GID 及使用者密碼是否正確，是否擁有權限操作此類的行為，若授權完成即可開始進行檔案系統的操作（指令及程序的執行）。

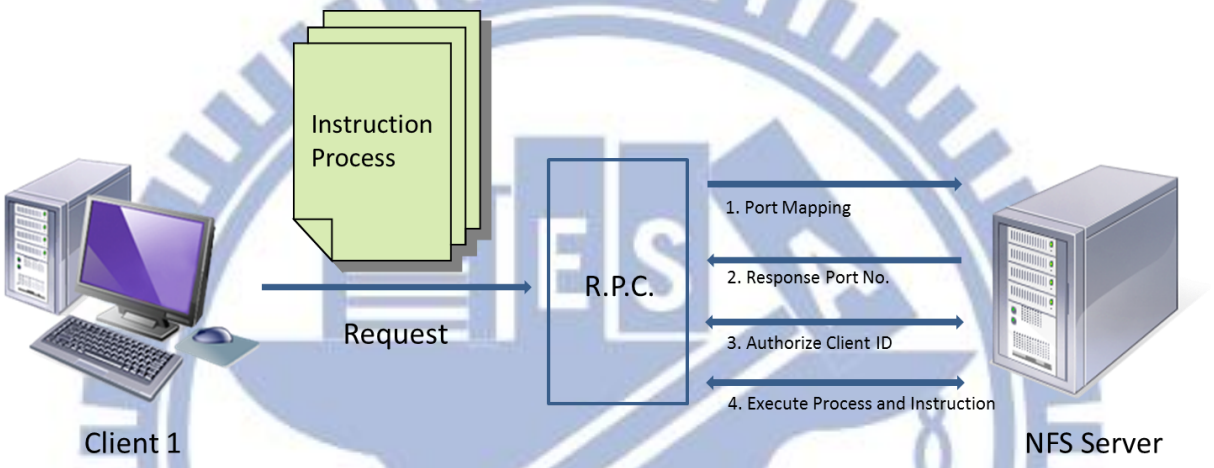


圖 13：Client 透過 RPC 與 NFS Server 溝通方式

三、 系統設計與實作

3.1 功能需求與分析

基於存取檔案的便利性、節省硬體空間的成本、跨區域及跨平台的整合性，使用者利用遠端的空間，進行檔案的存取；然這樣的行為卻可能造成使用者的資料遭到竊取因而增加風險。在享受服務的當下能確保資料的安全性是使用者的重要考量，要如何利用遠端空間的服務設計出一個安全、隱私的架構，且此架構是不會增加使用者過多的操作負擔是此次需求的重要思考方向。

（1）確保檔案在遠端儲存服務的安全性、隱私性。

（2）使用者能簡單地操作，將檔案進行上傳、下載、刪除作業。

觀察一般進行檔案上下傳的流程，通常是直接透過本機端應用程式的安裝或是網頁方式進行操作行為，使用者透過服務供應商提供的 AP 進行檔案維護作業。為求使用者不須額外的操作負擔，應將此功能建構在服務供應商之上，透過另外一個中介的軟體或網頁服務，與供應商所提供的儲存空間進行溝通及檔案操作。下圖根據上述需求及分析進行架構的規劃設計。

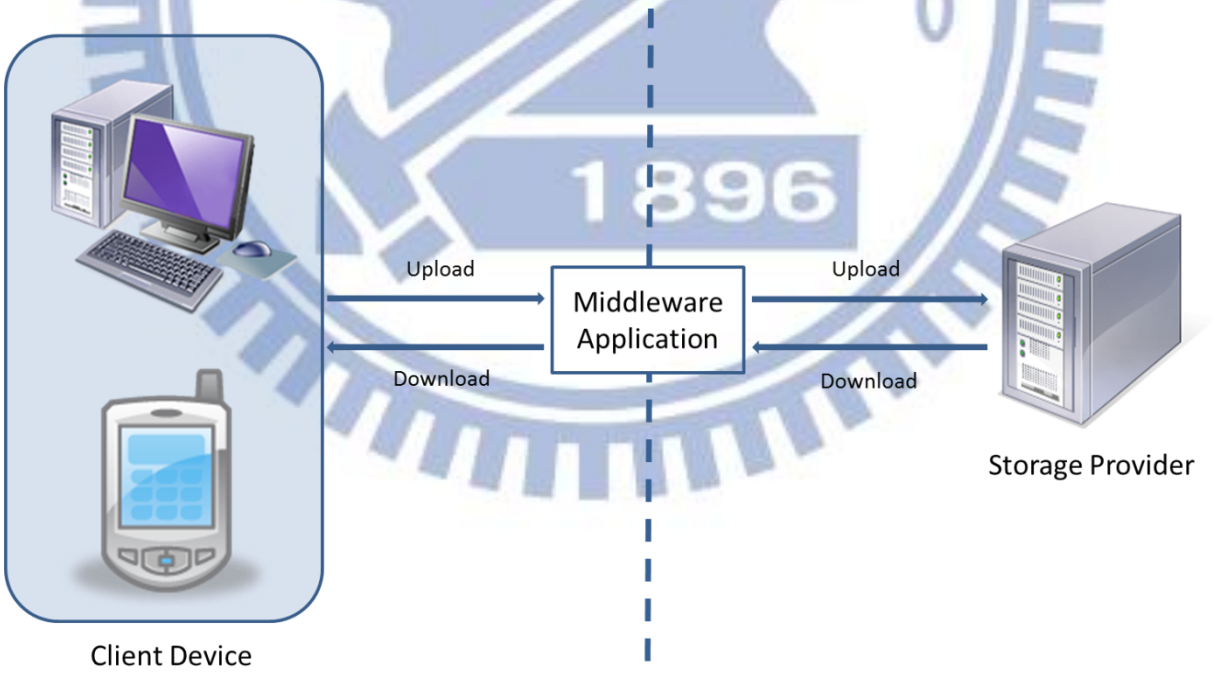


圖 14：需求分析架構圖

3.2 系統架構

依據 3.1 節的架構為雛型討論，要達成研究最終目的，設計重點將是中介軟體的開發與設計。而要如何有效保護使用者資料、隱藏檔案資訊，防止一般駭客及管理者的檔案窺視及竊取將是此中介軟體所應扮演的角色。其功能必須能完善的進行檔案的加解密、資訊的隱藏以及一般性的維護功能（包括上傳、下載、刪除）。

第二章的相關研究中曾提及檔案系統的架構；對於檔案的操作，共有三種模式，分別是：循序存取、隨機存取及索引存取。其中透過循序存取的檔案系統可以快速的獲得檔案，透過索引存取的方式可以藉由索引快速取得檔案的位置；其存取架構分別為圖 15、圖 16。本研究的設計將會同時運用這兩種存取模式，嘗試使設計的磁碟效能提昇。另外，NTFS 架構中提及，檔案資訊是利用檔案配置表（MFT）進行儲存。藉由上述特色，發展一個簡單、快速的虛擬檔案磁碟，名為 Simple Virtual Disk（以下簡稱 SVD）。此架構的概念以一個副檔名為.svd 的檔案格式呈現於任何遠端的儲存服務空間；使用者於本機建置一個 svd 的檔案結構，並將其放至遠端的空間，對於使用者、管理者或是其他第三方而言，僅能看到此檔案的存在，檔案的擁有者則直接對此檔案進行資料存取，所有的文件將置入於此檔案結構中以受到資訊隱藏的保護。

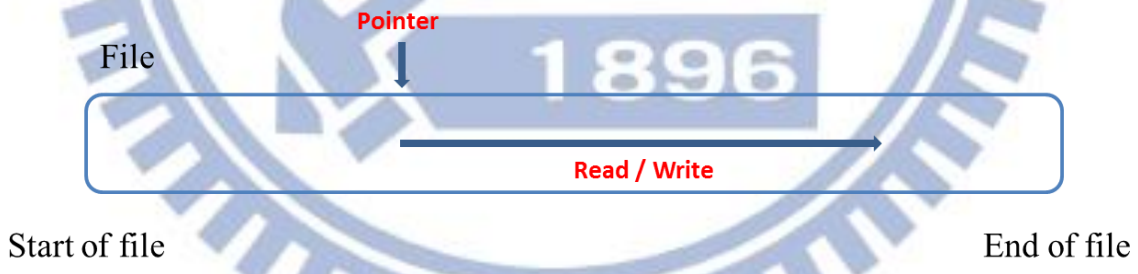


圖 15：檔案循序存取架構圖

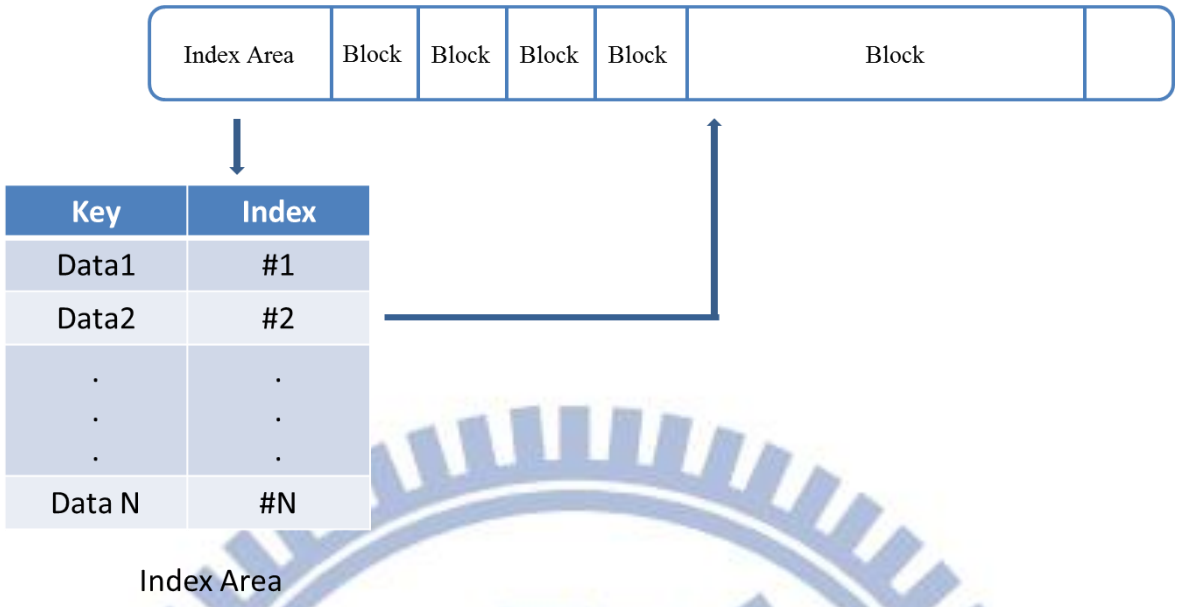
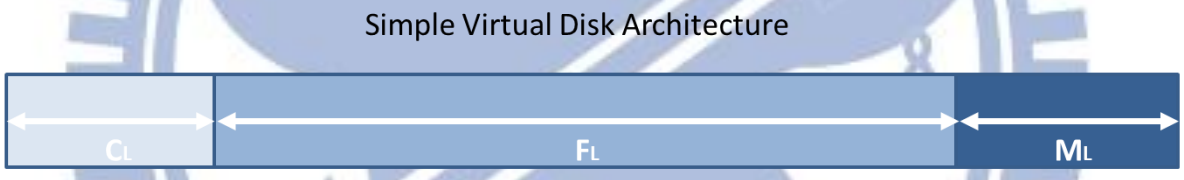


圖 16：檔案索引存取架構圖

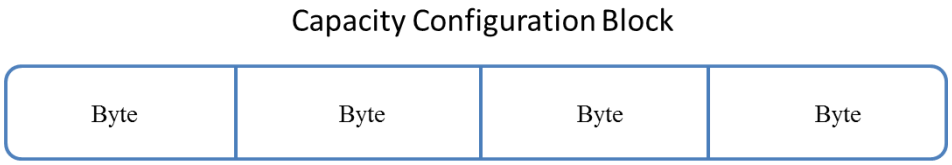
SVD 結構如圖所示，共分為三個區塊，分別為 Capacity Configuration、Encrypted File Collection、Metadata Configuration。以下進行各區塊說明。



名詞定義	說明
CL	容量長度（Byte）
FL	加密檔案集合總長度（Byte）
ML	設定檔長度（Byte)

圖 17：Simple Virtual Disk 架構

Capacity Configuration Block 位於 SVD 的開頭。SVD 中，實際存放檔案的大小由使用者所訂定，又 Metadata 因檔案的多寡而增減，故其長度是變動的。因此為取得檔案區塊的實際長度，設定一個固定長度的區塊，以 MB 為單位定義 SVD 容量的大小。



CL	Min Size（MB）	Max Size（MB）
4	0	2^{4*8}-1

圖 18：Capacity Configuration Block Architecture

Encrypted File Collection Block 緊接 Capacity Configuration Block 之後，以位元組為單位進行實體檔案的儲存。因文字檔案容易由位元組進行 ASCII 的轉換而取得文字內容導致文件暴露風險。為考慮檔案的安全性，此區塊不僅將檔案進行存放，並進行檔案的加密，再將加密後的檔案存放於此區間。以循序存取方式為主，將已加密的檔案依序進行存放作業。

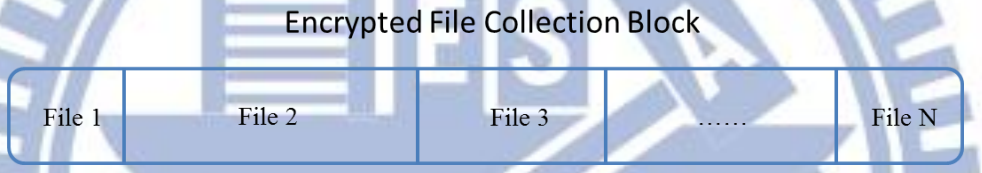


圖 19：Encrypted File Collection Block Architecture

Metadata Configuration Block 為 SVD 的最後一個區塊，以索引及 NTFS 的 MFT 概念進行設計；此區塊負責紀錄檔案的重要資訊，包括 SVD 磁碟資訊、檔案名稱、類型、長度及位置等。此區塊以 XML 形式儲存，所有的檔案操作都會先透過讀取此 XML 的內容，獲取其資訊後再進行實際的檔案存取。由於 XML 格式內容也可能有被竊取的可能性，所以在進行檔案操作時，仍會先將其進行加密動作，其格式還原如下圖。

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<Config>
  - <SVDInfo>
        <Info CreationDate="2014/05/03 14:34:53" Capacity="10485760"/>
    </SVDInfo>
  - <FileInfo>
        <Info End="1048594" Start="11" Length="1048584" FileName="file1.txt" DirName="Dir1"/>
    </FileInfo>
  - <PositionInfo>
        <Info End="10485770" Start="1048595"/>
    </PositionInfo>
</Config>
```

圖 20：Metadata Configuration 實際內容

（1）SVDInfo：定義 SVD 資訊；紀錄磁碟建立時間（CreationDate）、磁碟容量（Capacity）。

（2）FileInfo：定義檔案資訊，每個檔案皆會產生一筆 Info 的 Record。檔案資訊包括開始位置（Start）、結束位置（End）、檔案大小（Length）、檔案名稱（FileName）、檔案資料夾路徑（DirName）。

（3）PositionInfo：紀錄目前可用空間區塊，每個可用區間都會有一筆 Info 紀錄。位置資訊主要紀錄可用空間的起始位置（Start）及結束位置（End）。

要以上述三個區塊的結合為架構設計 SVD，將碰到一個困難的課題，即如何藉由檔案位置及檔案長度從區塊中取得所需的檔案片段。若無法克服此問題由檔案的某個位置開始讀取檔案僅能從頭開始讀取，則對於 SVD 這類容量可能達到 GB 等級的架構設計即為空談，因每次存取檔案都必須從頭開始，將會浪費非常多的效能。為克服上述問題，SVD 的實做需要仰賴 File Seek 的技術。所謂的 File seek 即提供讀取檔案串流時，可指定檔案位置及長度以進行檔案片段的讀取，運用此法可加速讀取大檔案中片段檔案的效能，進而解決上述問題。

此架構能夠被實做的最後一個關鍵在於需要利用加解密技術來增強其安全性，包括檔案本身及 Metadata。由 2.1 節中了解，DES 演算法的加密強度雖較弱，但卻擁有較快的加解密速度；而 AES 雖加解密速度較慢但卻可提供較高的安全性強度。在希望盡量不影響使用者操作花費的時間成本為前提，同時搭配此兩種加解密模式可減輕許多成本負擔。

SVD 的架構中最為重要不可被破解的部份，不外乎 Metadata 的內容；因 Metadata 中包含檔案的位置、名稱等重要資訊，又其檔案的使用空間較小，適合花費較高的成本獲得更高的安全性。所以 Metadata 的部份將採用 Rijndael 的加解密演算法，以確保設定檔的高安全性。而檔案本身雖也有其加密的必要性，但因各檔案的資訊已被隱藏包裝在 Metadata 中，且以序列的方式串連，駭客或其他第三方管理者無法得知個別檔案的大小及內容，更無法得知檔案的起始位置及結束位置。因此檔案的加解密方式，將利用加解密速度較快的 DES 進行檔案的加解密處理，提高安全性之餘也能節省許多時間成本。

3.3 功能操作流程

本節將對 SVD 架構所能提供的功能進行實際流程說明，包括 SVD 的建置、檔案的上傳、下載、刪除。SVD 建置流程主要分為三個部份：Metadata 的

初始化及加密、Capacity 區塊的設定、Encrypted File Collection 區塊的初始化及設定。依據流程，詳細說明如下：

（1）產生 Metadata設定檔，並對此檔進行初始化設定；包括磁碟建立時間、容量等設定。

（2）將初始化後的 Metadata 設定檔以 Rijndael 加密法進行加密作業。

（3）依照使用者所設定的磁碟容量進行 Capacity Block 的設定。

（4）因初始的 SVD 磁碟尚未擁有檔案，故以亂數方式填入位元組，填滿磁碟空間。

（5）將上述完成的三個區塊進行 SVD 檔案的寫入整合即完成 SVD 的建置作業。

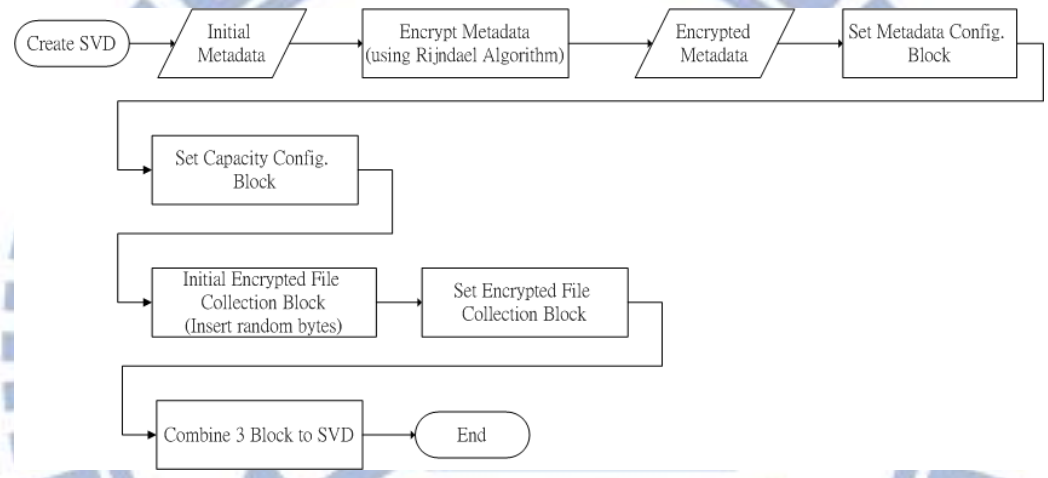


圖 21：SVD 建置流程圖

SVD 上傳檔案的流程如下圖，流程主要分為三個部份：Metadata 的取得、檔案的加密與上傳、Metadata 的更新、加密與上傳，詳細流程說明如下：

（1）確認本地端檔案是否存在，若存在才繼續上傳作業。

（2）因 Metadata 的長度不固定，故須藉由 Capacity Block 取得磁碟長度後，再加上 Capacity 本身的位元組長度，即可正確獲得 Metadata 的起始位置，又 Metadata 設計於最後一個區塊，故將檔案由起始位置讀至最後即可獲得 Metadata 的完整加密檔案。

（3）由 Metadata 取得檔案資訊以確認是否有重複檔名。取得空間資訊以確認是否有充足容量可供檔案上傳。

（4）空間與檔案確認後，將檔案以 DES 加密演算法進行加密運算。

（5）將 SVD 磁碟以 Seek 的方式指向存放位置並開始進行檔案的上傳。

（6）重新計算可用空間資訊並新增檔案資訊至 Metadata。

（7）將 Metadata 再度以 Rijndael 加密演算法進行加密並上傳至 SVD。

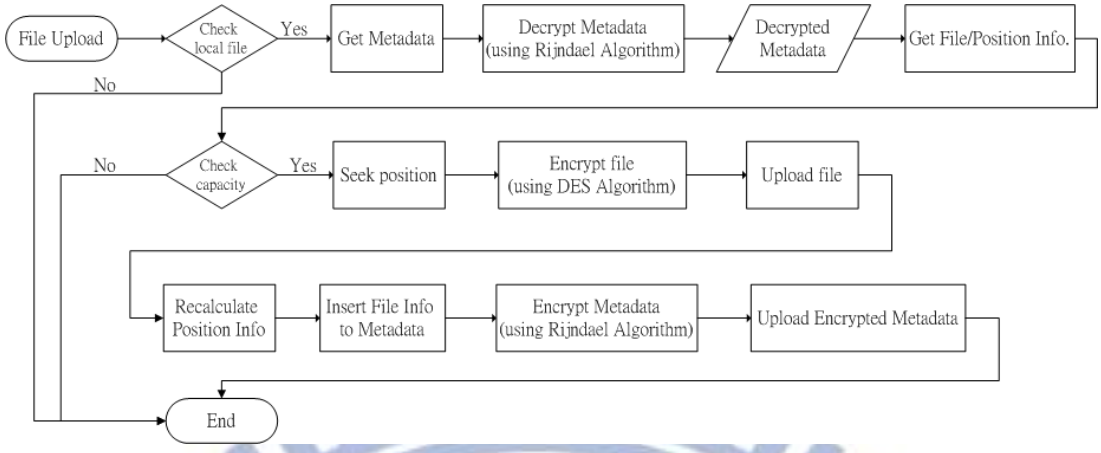


圖 22：SVD 檔案上傳流程圖

SVD 進行檔案下載的流程如下圖，主要分為四個步驟：Metadata 的取得、取得加密檔案、解密、檔案下載。其詳細說明如下：

（1）取得 Metadata 的方式與上傳相同，故不再贅述。

（2）以檔案名稱作為關鍵字，從 Metadata 取得檔案資訊後，以 Seek 方式獲得加密檔案。

（3）以 DES 演算法進行解密的動作，獲得原始檔案。

（4）檔案下載。

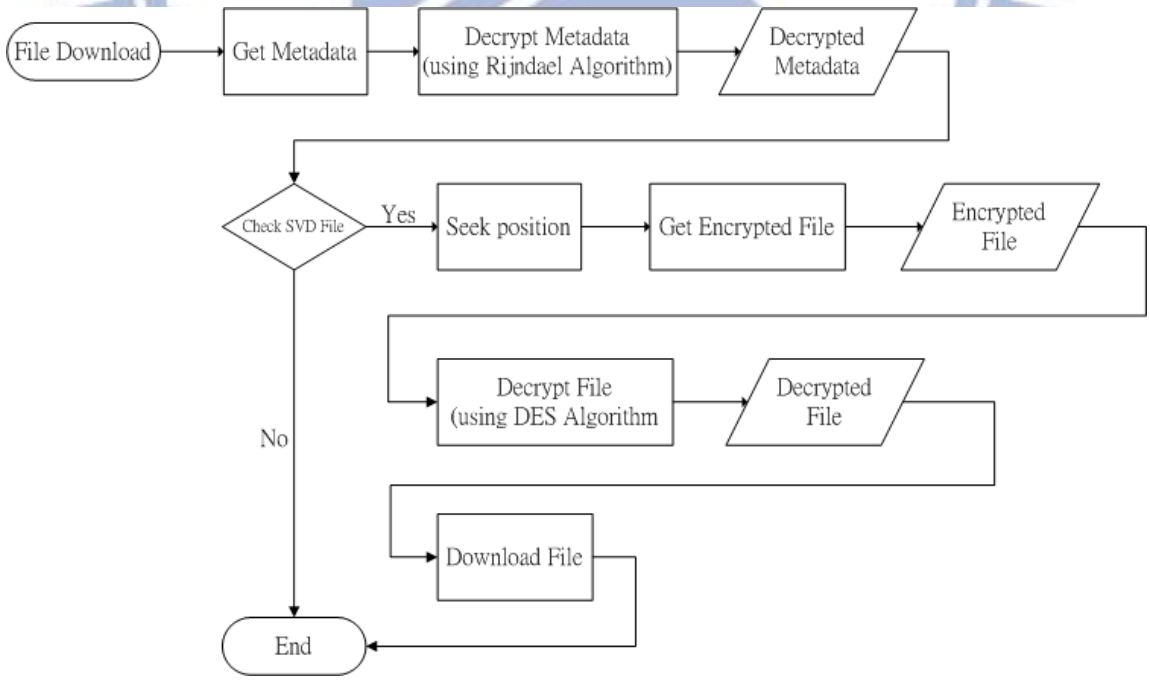


圖 23：SVD 檔案下載流程圖

SVD 刪除檔案的流程如下圖。由於 SVD 的 Encrypted File Collection 區塊中的檔案，原就已進行 DES 的加密，所以當需要進行檔案刪除時，僅須

刪除 Metadata 的內容，將其檔案長度、起始位置等進行刪除，無須再抹除已加密檔案。失去資訊的加密檔案，則就像亂碼般已無作用也無法再被獲得，此舉能大幅提昇刪除檔案的速度。檔案刪除的實際流程說明如下：

（1）取得 Metadata（與上傳方式相同）。

（2）重新計算可用磁碟空間並刪除檔案資訊。

（3）重新將 Metadata 加密並更新上傳回 SVD。

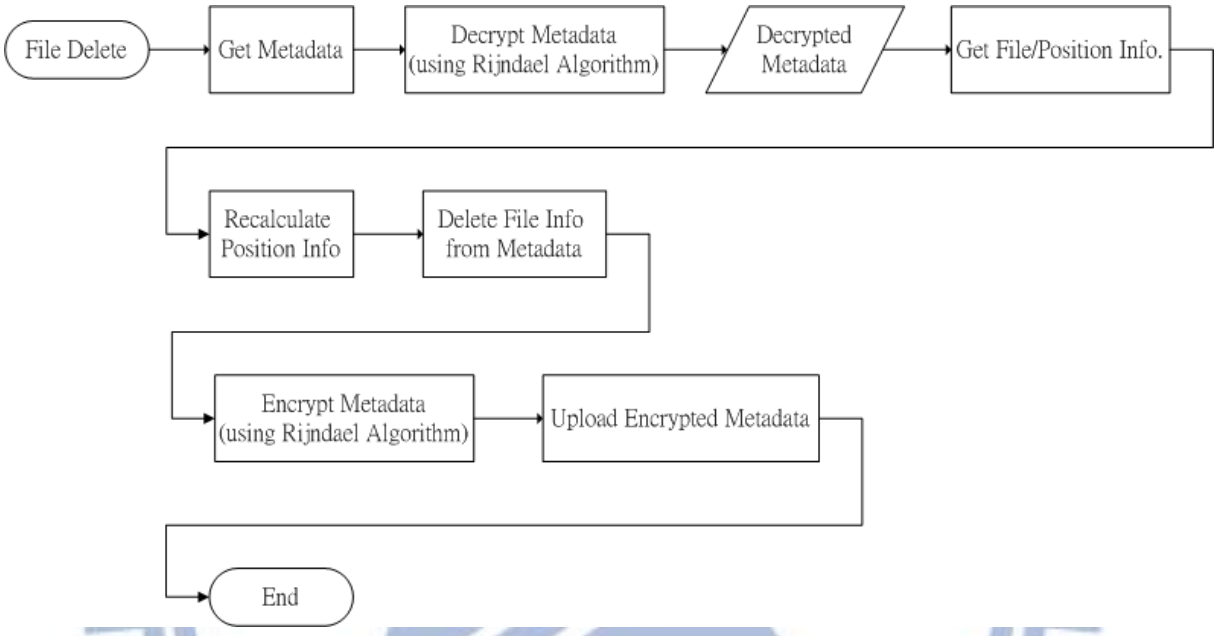


圖 24：SVD 檔案刪除流程圖

3.4 設計細節

本節將以 ASP.NET C#作為開發工具，詳述 SVD 的實作內容。實作的重點將分為三個部份：加解密功能、使用者認證授權功能、SVD 架構。

加解密功能作用在於提供外部程式呼叫，將所需的檔案輸入後進行各種加解密演算法的處理，並輸出加解密結果。藉由給定加解密演算法的類型，給予不同的金鑰長度進行運算；由於每個演算法所需金鑰長度不同，故利用 PasswordDeriveBytes[1]類別改變金鑰長度。判斷演算法類型後，利用微軟提供的 CryptoStream[2]類別，輸入明（密）文及延伸金鑰（Key）及 IV，進行加解密演算並輸出至應用程式中。

第二章提及存取 NFS 需要於主機端設定帳號，並設定權限。當使用者需要存取檔案時，利用帳號、密碼進行驗證，確認是否有權限進行資料的存取。由上述可知，需要一個模擬本機使用者帳號的功能，即 Impersonate

[1] PasswordDeriveBytes 為微軟提供，使用 PBKDF1 演算法進行密碼衍生金鑰。

[2] CryptoStream 為微軟提供，將資料流進行加密/解密的資料流轉換。

機制。運用 Impersonate 機制，可以讓使用者在模擬期間扮演所需的帳號，獲得其角色的權限。由於模擬角色的功能需要底層的支援才能達到，此部份呼叫系統的 dll，以進行角色的模擬使用。由外部進行 Imperonate 的呼叫，輸入所需模擬的角色帳號及密碼，通過驗證後即可於遠端模擬此帳號的權限。

對於 SVD 的實作，為考慮不同應用程式呼叫的便利性，將 SVD 的操作功能包裝於 SVDManager 類別；此類別提供 SVD 建置、上傳、下載、刪除、資訊取得等功能。其類別架構如下圖。以下將針對較為重要的函式進行說明。

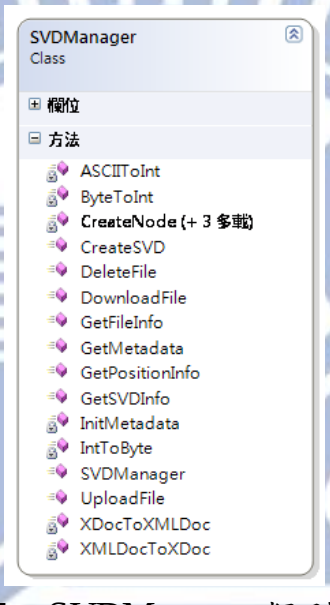


圖 25：SVDManager 類別架構

（1）GetMetadata Function

此方法功能為取得 SVD 中 Metadata Block 的內容。透過給定 Capacity Configuration Block 的長度 CapacityDigit，計算出 Metadata Block 的位置，並利用 Rijndael 演算法進行解密而獲得 Metadata。

（2）Get Information Functions

程式中共有三項資訊是需要被取得的，分別是 SVD 的磁碟資訊、檔案資訊以及空間位置資訊。程式將 Metadata 輸入，利用 .Net framework4.0 提供的 Linq 查詢整合語言，快速的藉由條件篩選獲得所需要的資源。

（3）CreateSVD Function

提供建置 SVD 虛擬磁碟的功能。透過遠端儲存空間、容量、金鑰及 Capacity Info. Block 的長度等變數輸入，建立一個完整的 SVD 結構。

方法中個別初始化三個區塊的物件，並於最後利用 FileStream 的方式將三個物件進行寫入的動作。

（4）UploadFile Function

為 SVDManager 提供了檔案上傳的功能。外部應用程式提供欲存取的遠端 SVD 磁碟位置、金鑰及本地端檔案，進行檔案的上傳作業。過程中，將檔案進行 DES 的加密，並將 Metadata 進行更新及 Rijndael 的加密，最後將檔案及 Metadata 分別更新至 SVD，完成上傳的作業。

（5）DownloadFile Function

為 SVDManager 提供了檔案下載的功能。外部應用程式輸入 SVD 磁碟位置、金鑰及欲下載的檔案路徑及名稱以進行檔案的下載。實際的下載流程如下，主要藉由 File Seek 快速的指向欲下載的檔案位置，讀出檔案後進行 DES 的解密作業，最後進行檔案的下載動作。

（6）DeleteFile Function

提供 SVD 檔案刪除的功能。外部應用程式輸入 SVD 位置、欲刪除的檔案路徑及名稱後，進行 Metadata 的資訊抹除作業。刪除檔案的流程為取得 Metadata 的資訊，並確認是否有檔案紀錄存在，若有則開始進行 Metadata 的更新。

3.5 實作範例

本節將上述所設計的 SVD 架構進行實際的運用操作；本研究測試了數個的雲端儲存平台，包括 DropBox、GoogleDrive、Microsoft OneDrive[1]等，由於 DropBox、GoogleDrive 未能提供 NFS 模式的存取服務，無法進行實際操作。而 OneDrive 則利用了 WebDAV 的協定提供 NFS 的服務，Web-Based Distributed Authoring and Versioning(WebDAV)[9]是 HTTP/1.1 的延伸協定，主要功能與 NFS 相同，可進行遠端的網路資料交換及存取，使網路資料擁有可讀寫性；本研究將以 OneDrive 作為 SVD 實際的案例測試。然而實際操作的期間，發現實際效果不如預期，雖 SVD 能夠正確的被存取但效率不佳，為確認問題癥結，本研究針對檔案及 SVD 的大小做了相關的測試。藉由網路測試軟體確認提供下載 60Mbps、上傳 15Mbps 的網路能力。

第一個實驗以 SVD 大小作為變數進行檔案存取，測試中在不同的 SVD 中上傳 1MB 的檔案，確認其時間成本的花費。圖 26 可得知，雖皆進行 1MB

[1] Microsoft OneDrive 為微軟所提供之雲端儲存服務，官方網站為：https://onedrive.live.com/about/zh-tw/。

的檔案上傳，但卻因 Server 的 SVD 大小不同，而產生不同結果（圖中的 Simple Upload 為對照組，顯示上傳 1 至 10MB 所須花費的時間）。且上傳 1MB 的檔案至 SVD 的時間與上傳一個與 SVD 相同大小的檔案時間花費是相似的。第二個實驗以檔案的大小作為變數進行檔案存取的效能測試，於 OneDrive Server 上建置一個 10MB 的 SVD，並對此 SVD 進行 1MB 至 9MB 共計九個檔案的上傳，其結果如圖 27。由結果得知花費時間並不與上傳的檔案有太大的影響。

依結果顯示，雖 WebDAV 可進行 NFS 模擬，但其協定是架構在 HTTP 之上，對於檔案的存取仍須讀取整個檔案才能操作，造成效能的下降；故若須進行 SVD 的實際應用，仍應以 NFS 協定進行存取才能提供較好效能。

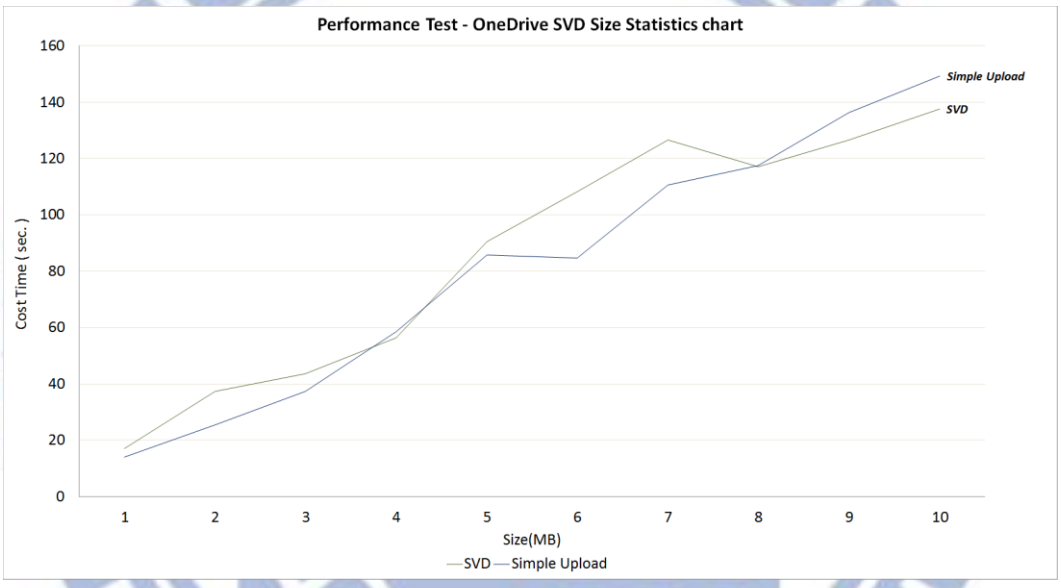


圖 26：Performance Test – OneDrive SVD Size Statistics Chart

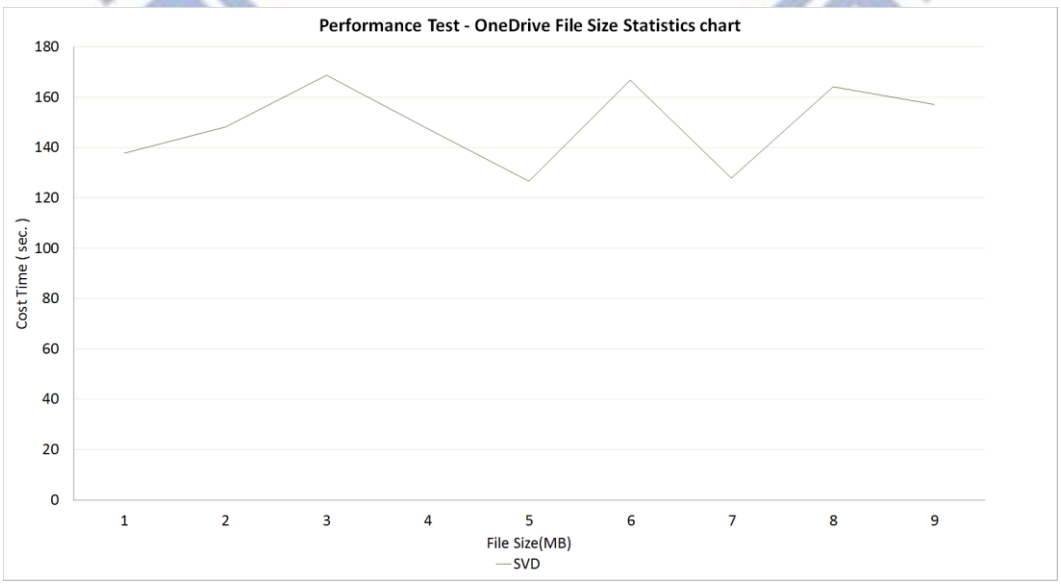


圖 27：Performance Test – OneDrive File Size Statistics Chart

四、 實驗設計與結果分析

4.1 實驗設計

本研究的實驗以兩個方向進行:穩定性及效能。以現有的壓縮技術(ZIP)及檔案系統(NTFS)與研究中所設計的 SVD 架構作比較。由於本研究重點在於檔案的安全性及隱私性，故每種儲存機制皆會導入加密機制進行分析研究。以 ZIP 而言，為達到安全性及隱私性，須利用本身的 Lock 機制，將檔案進行上鎖，才能防止檔案的獲得。而 NTFS 檔案系統本身雖可將檔案包裝在此系統中，但沒有進行加密，所以需要導入加密演算法，為求實驗的公平性，亦於 NTFS 中導入與 SVD 相同的 DES 加密演算法。

根據實驗所需要的三種儲存機制，同樣地利用 C#作為開發工具進行開發。為求 ZIP 及 NTFS 的實作，同時導入了 DotNetZIP[1]以進行 ZIP 的實作及.Net DiscUtils[2]以進行 NTFS 的 Virtual Hard Disk(以下簡稱 VHD)實作。

本研究共設計了三個實驗，分別如下：

(1) 以反覆的上傳、下載、刪除進行壓力測試，確認磁碟的穩定性及時間成本效益。

(2) 根據不同的檔案大小進行效能實驗。

(3) 根據不同檔案數量進行效能實驗。

本實驗為建置 NFS 環境，對於環境需求的角色共需兩個，分別為 Server 及 Client。假設 Server 為遠端儲存空間，而 Client 則扮演使用者端的操作行為角色。為求兩主機間的網路環境穩定，不受到其他可能因素的干擾，利用網路分享器的方式建立了一個簡易的區域網路如下圖。本實驗所測試的兩台主機皆為一般坊間能購入的主機，而分享器傳輸速度的規格則為 100Mb。最後，因實驗中會大量反覆的運算，故實驗測試將以批次作業方式，於 Client 端進行反覆操作。

[1] DotNetZIP 為 OpenSource，其官方網站為：http://dotnetzip.codeplex.com/，提供 ZIP 的建置及操作功能。

[2] .Net DiscUtils 為 OpenSource，其官方網站為：https://discutils.codeplex.com/，提供虛擬磁碟建置及操作。

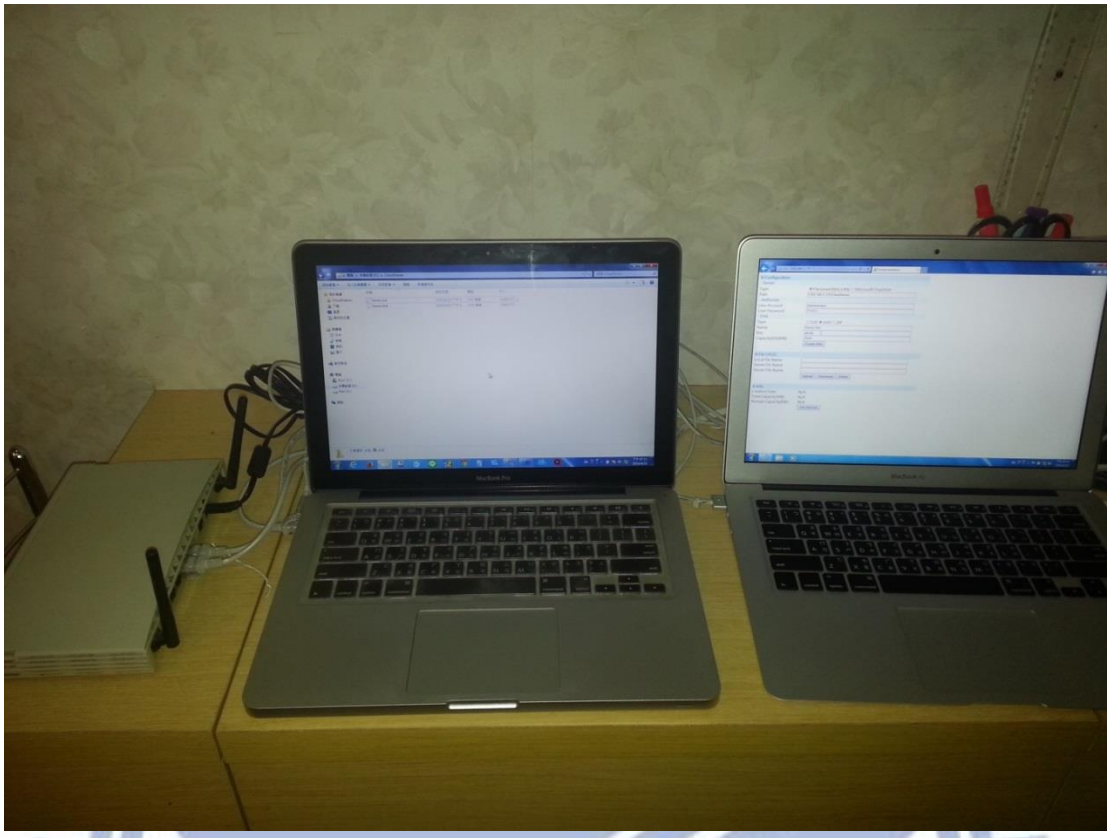


圖 28：區域網路架設

4.2 穩定性及時間成本實驗

此實驗進行各種儲存機制的穩定性及所耗費的時間成本比較。以大量且反覆的上傳、下載、刪除作業，提高結果的準確性。其實驗對象以壓縮檔、檔案系統、SVD 三種儲存形式作為比較的項目，同時測試其加密及未加密的結果，共有六個實驗對象。實驗將以一個固定的檔案作為實驗的操作物件，根據六個實驗對象進行反覆的檔案行為操作，進行一千次的上傳、下載及刪除的作業，最後將結果紀錄、繪製圖表。圖 29 至 31 顯示實驗結果。

由實驗結果得知，不論何種儲存機制必然會因加密而導致時間成本的增加，這是為求增加安全性而必須負擔的犧牲。綜觀上傳及下載而言，VHD 在進行此類操作時，因其結構的複雜，須先開啟 Disk 及 Volume，才能對檔案作操作，此舉導致時間成本的增加。而以線圖來看，ZIP 最不穩定，且加密後需要花費過多的時間成本。相較於 ZIP 及 VHD，不論上傳、下載，SVD 皆能會得最好的效能；且以線圖的起伏來看，SVD 在不斷反覆的壓力測試中，相較於其他的儲存機制，有更好的穩定性。以刪除而言，三者皆有很好的表現，而 SVD 因只進行 Metadata 的資訊抹除，所以速度更是在三者中獲得最佳成績。

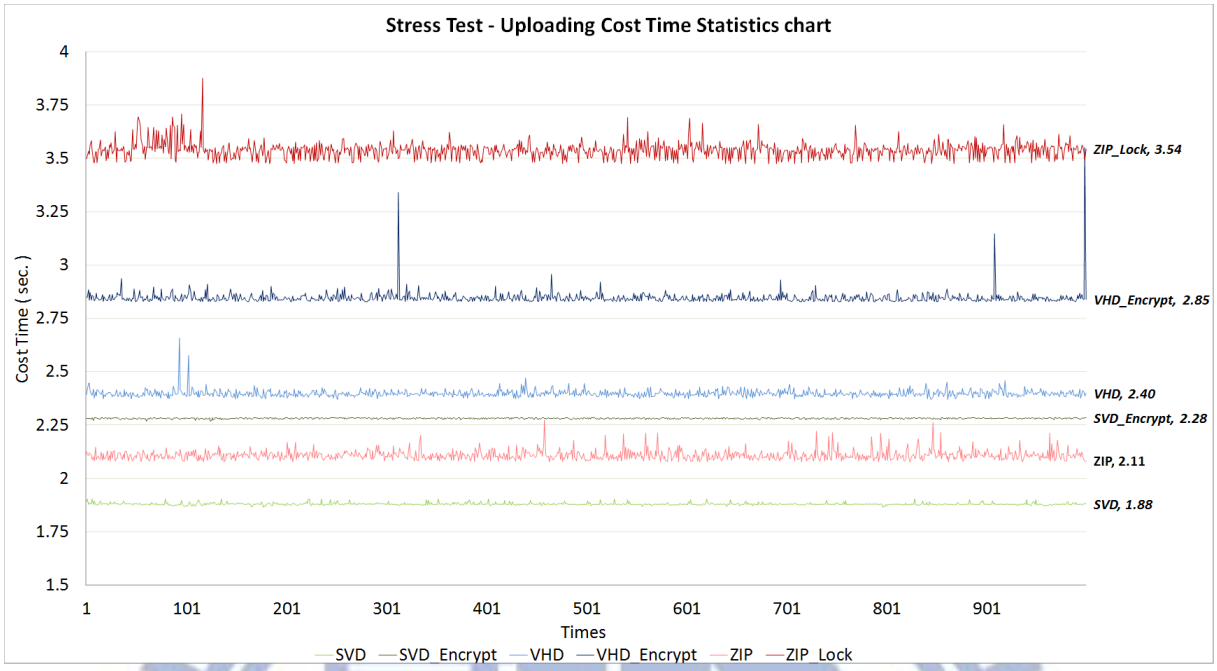


圖 29：Stress Test – Uploading Cost Time Statistics Chart

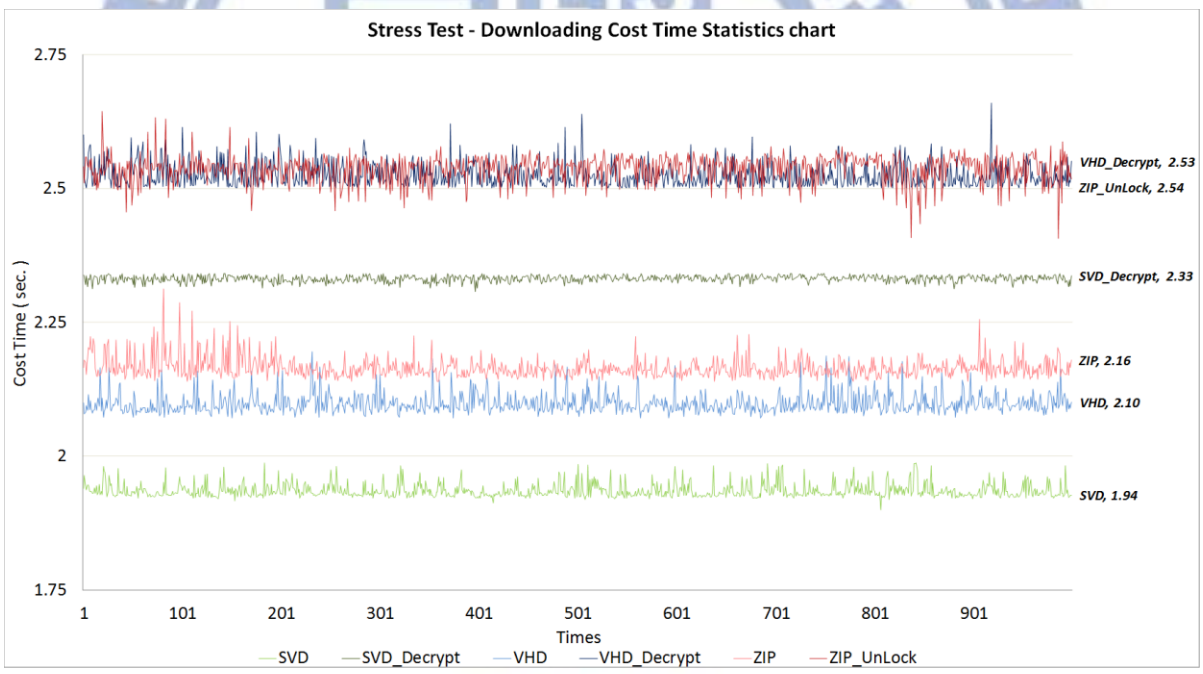


圖 30：Stress Test - Downloading Cost Time Statistics Chart

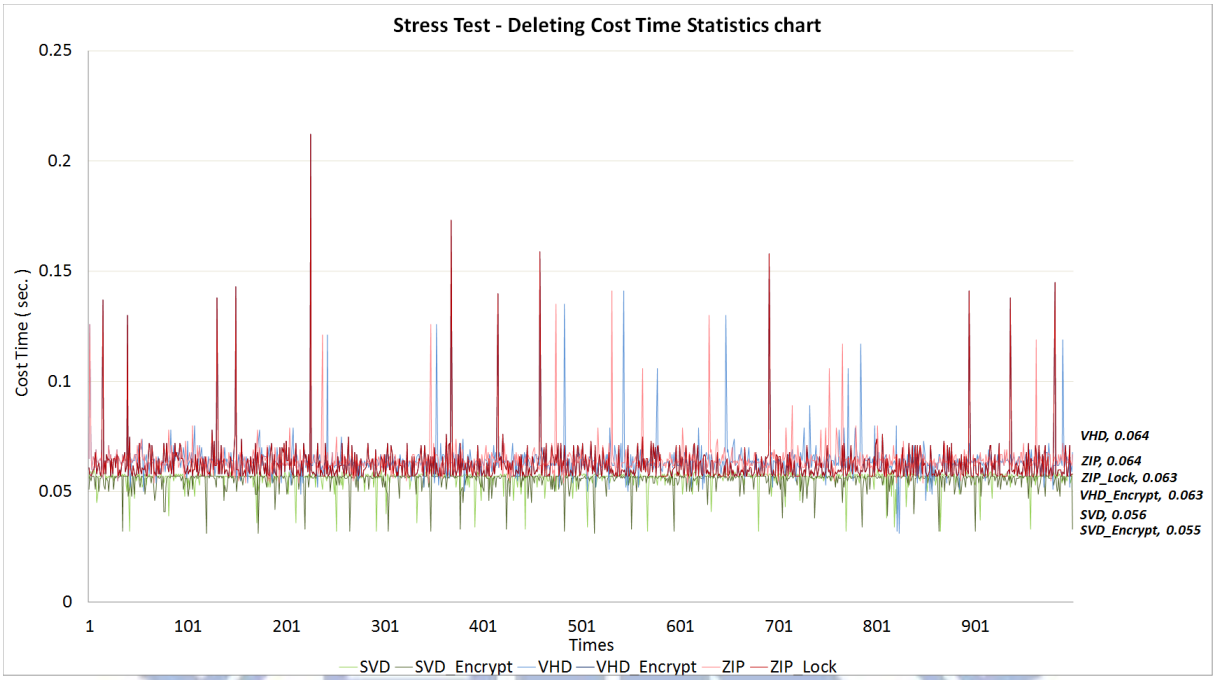


圖 31：Stress Test - Deleting Cost Time Statistics Chart

4.3 檔案數量效能實驗

此實驗目的在於研究儲存機制是否會受到檔案數量而影響效能。由 4.2 節實驗結果了解不論何種儲存機制皆會受到加解密的動作增加時間成本。然而此成本是必須具備的前提下，此實驗的實驗對象將設定為加密後的儲存機制，共計三個對象：壓縮檔、檔案系統及 SVD。實驗將以一組相同檔案大小的檔案作為操作物件，進行上傳、下載、刪除的動作，判斷當儲存機制在不同的檔案數量時的時間耗費成本。圖 32 至 34 顯示實驗結果。

由實驗結果了解，SVD、VHD 不論在上傳、下載、刪除對於檔案的數量皆沒有任何影響，而 ZIP 雖在上傳、下載時不受檔案數量影響，但其刪除時因為需要進行實際檔案的抹除，而導致刪除速度隨著檔案的數量增加而遞減。

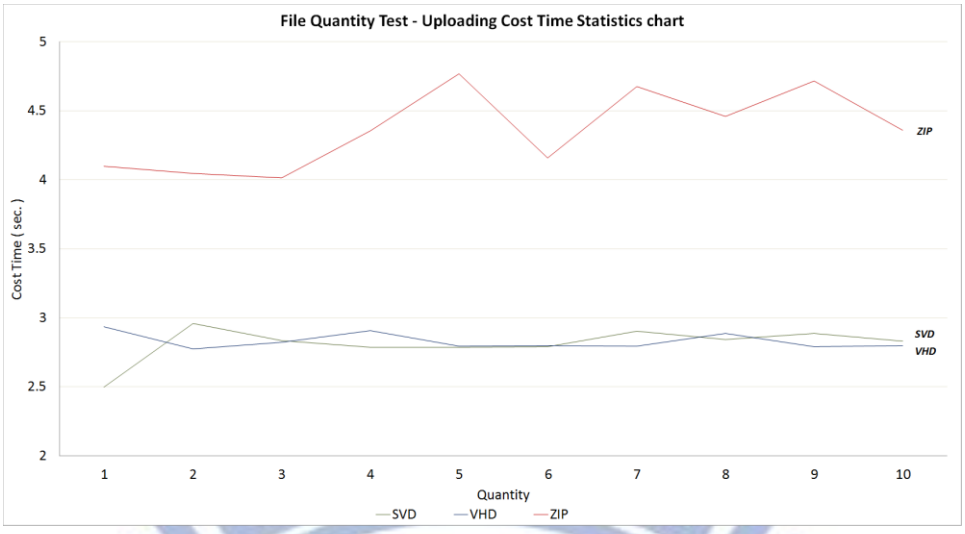


圖 32：File Quantity Test – Uploading Cost Time Statistics Chart

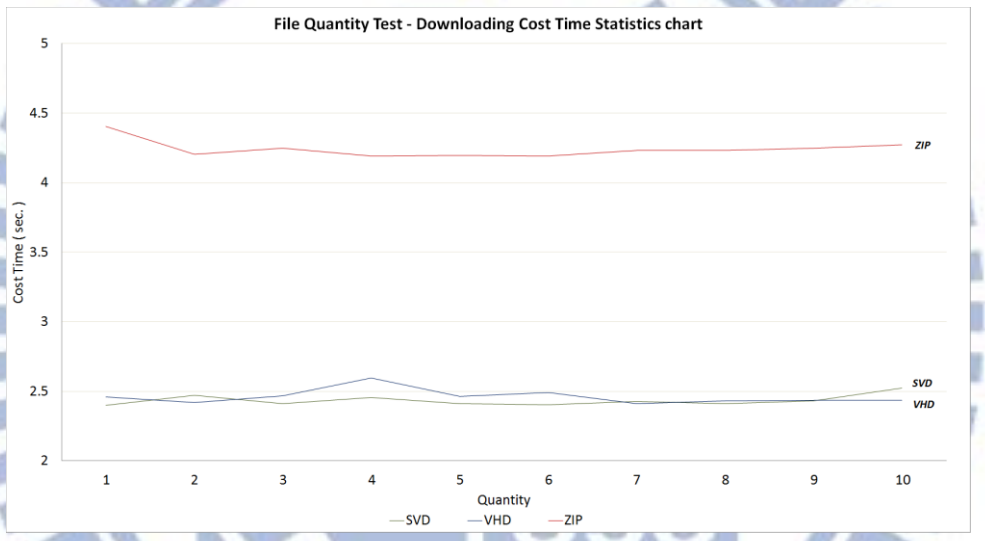


圖 33：File Quantity Test – Downloading Cost Time Statistics Chart

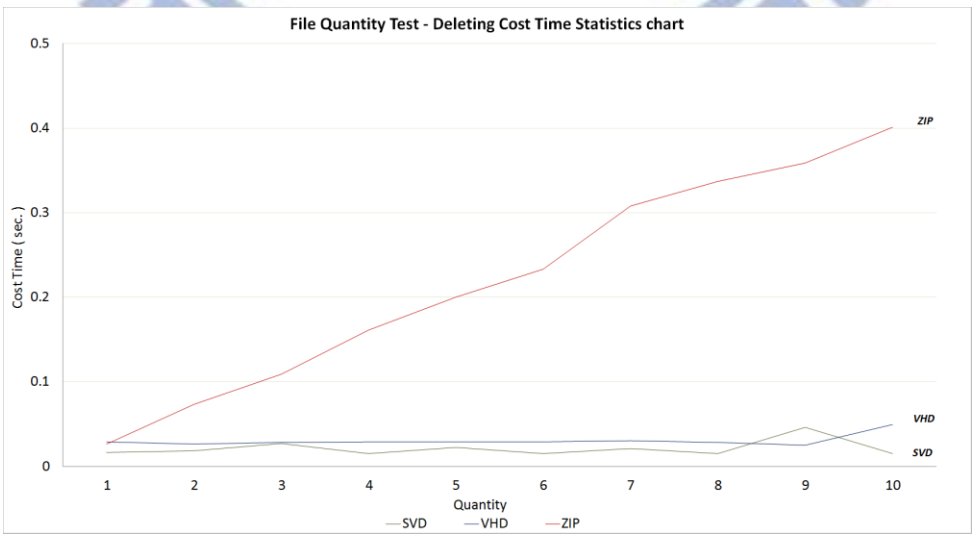


圖 34：File Quantity Test – Deleting Cost Time Statistics Chart

4.4 檔案大小效能實驗

此實驗的目的仍在進行效能的比較，但與 4.3 節不同的是，此實驗改以檔案大小作為變數進行效能的比較。與 4.3 節相同，以加密過後的壓縮檔、檔案系統、SVD 為對象進行實驗。由檔案大小作為實驗變數的考量，以 10MB 為單位遞增產生檔案。藉由一組不同大小的檔案進行上傳、下載、刪除等操作，以獲得實驗數據。圖 35 至 37 顯示實驗結果。

由實驗結果得知檔案大小與時間成正比，以上傳、下載而言，SVD 的表現最為優異，ZIP 的效能最差；以刪除來看，三者皆不受檔案大小的影響，可以在短時間內完成刪除作業。

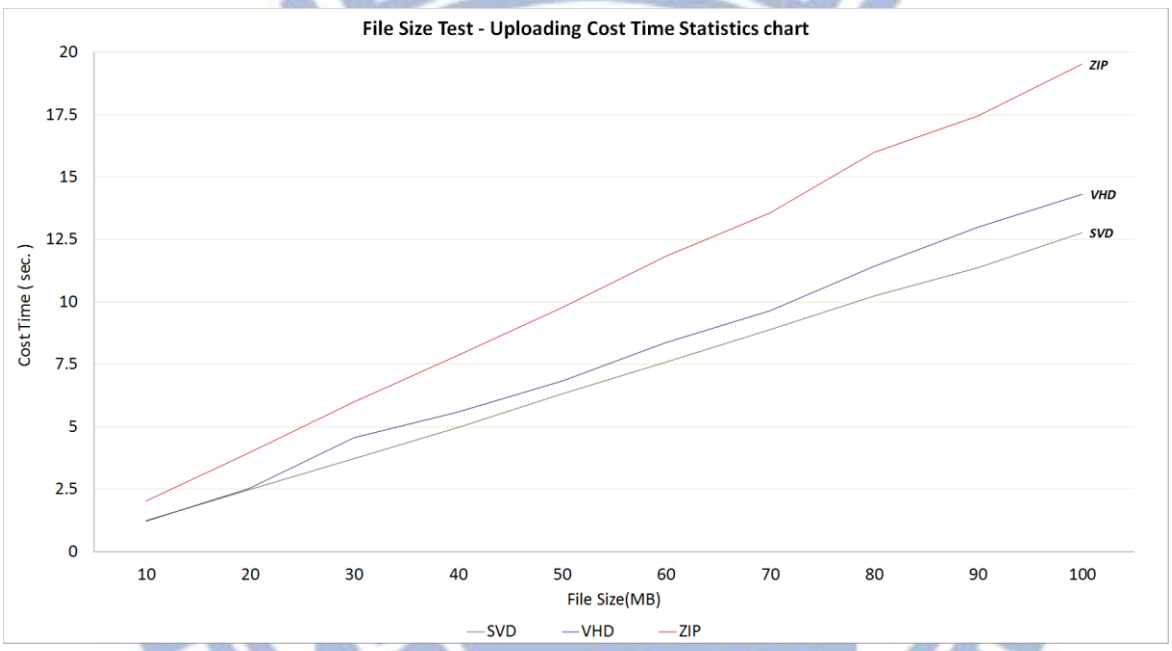


圖 35：File Size Test - Uploading Cost Time Statistics Chart

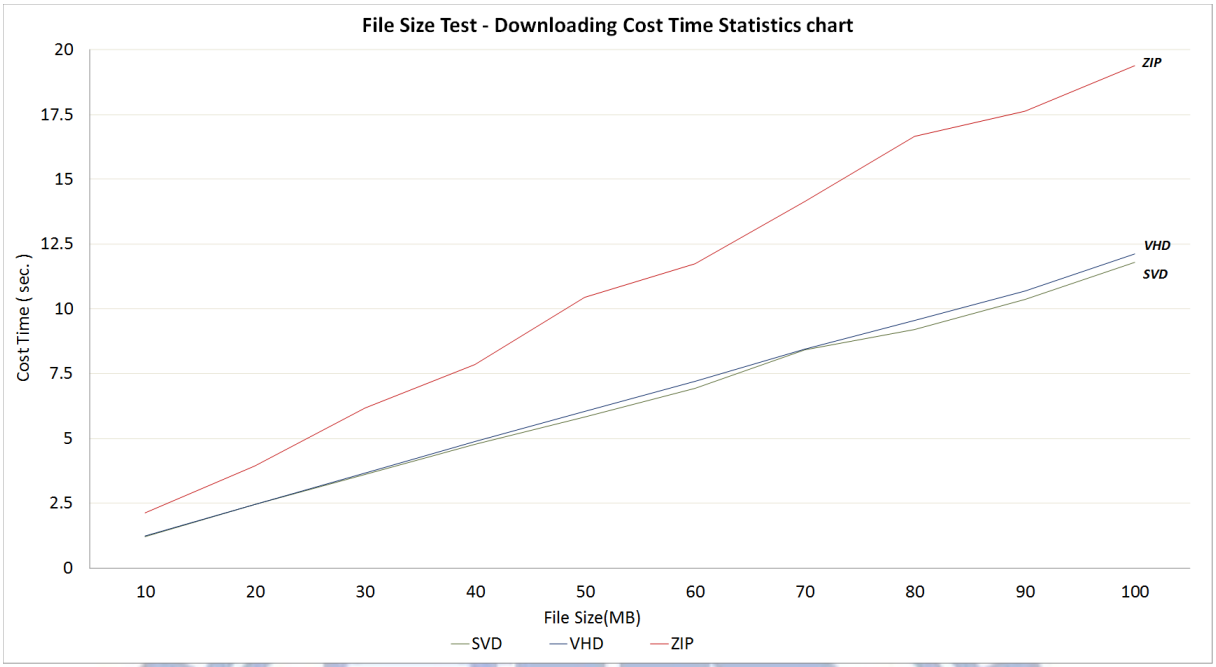


圖 36：File Size Test - Downloading Cost Time Statistics Chart

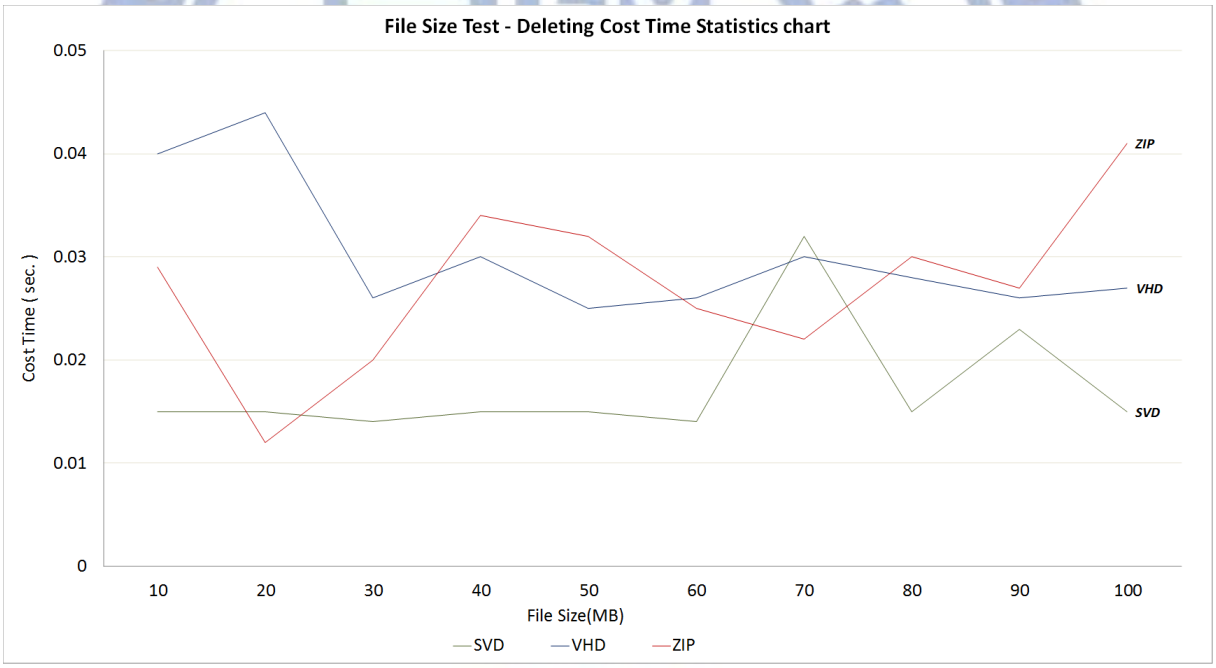


圖 37：File Size Test - Deleting Cost Time Statistics Chart

五、 結論

以往的網路儲存機制皆以防止第三方駭客或是使用者的窺視作為防範對象，而疏忽了對於管理者的資料防範，因而增加了資料的暴露風險。文中結合了不同的技術研究，包括檔案系統、加解密機制、NFS 等，設計出 Simple Virtual Disk 的架構，以簡單卻不失安全性及隱私性的結構，在各項實驗中取勝。此架構除效能外，最重要的，是能夠有效的提昇使用者資料安全性及隱私性，進而達成本研究的目標，防止資料的窺視。

本研究主要以 NFS 作為研究環境，利用 File Seek 的技術，實做了 SVD 的結構。然而網路的應用非常廣泛，舉凡 HTTP、FTP 等皆為網路應用的範疇，未來應思考其他的傳輸協定，是否也能透過類似 Seek 的方式進行檔案的處理。另外，SVD 的設計重點於本研究中主要進行了資料的操作行為，雖對於 SVD 的有效空間使用率有作過一定的優化，每次進行文件維護時會去計算可使用的空間片段，並挑出適合的空間去存放文件，但仍可能產生 defragment 的情況，然而本研究為考量進行 SVD 重整搬移資料可能會花去過多的效能，故此部份未進行實作，此部份可於未來進行進一步的研究。

參考文獻

[1] Amritpal Singh, Mohit Marwaha, Baljinder Singh and Sandeep Singh, "Comparative Study of DES, 3DES, AES and RSA," International Journal of computers and Technology, Vol9, No3, July 2013.

[2] Orner K. Jasim Mohammad, Safia Abbas, El-Sayed M. El-Horbaty and Abdel-Badeeh M. Salem, "A Comparative Study between Modern Encryption Algorithms based on Cloud Computing Environment," Proceedings of the 8th International Conference for Internet Technology and Secured Transactions, pp. 531-535, December 2013.

[3] O P Verma, Ritu Agarwal, Dhiraj Dafouti and Shobha Tyagi, "Performance Analysis of Data Encryption Algorithms," Electronics Computer Technology, Vol5, 2011.

[4] Khalid Sayood, Introduction to Data Compression, 3rd ed., Morgan Kaufmann, San Francisco, 1995.

[5] Microsoft, "Zip or unzip a file," http://office.microsoft.com/en-us/help/zip-or-unzip-a-file-HA001127690.aspx.

[6] Konecki, M., Kudelic, R. and Lovrencic, A., "Efficiency of lossless data compression," Proceedings of the 34th International Convention on Information and Communication Technologies, Electronics and Microelectronics, pp. 810-815, 2011.

[7] Microsoft, "NTFS compared to FAT and FAT32," http://msdn.microsoft.com/en-us/library/cc779002%28v=ws.10%29.aspx, Jan 2005.

[8] Microsoft, "Comparing NTFS and FAT file systems," http://windows.microsoft.com/en-us/windows-vista/comparing-ntfs-and-fat-file-systems.

[9] Fredj Dridi and Gustaf Neumann, "How to implement Web-based groupware systems based on WebDAV," Proceedings of the 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 114-119, 1999.