檔案同步備份與保密安全系統之設計與實現

Design and Implementation of a Secure File Synchronization and Backup System

汪宏峻　　曾守正*

國立高雄第一科技大學

資訊管理研究所

高雄市

imfrank@ccms.nkfust.edu.tw

Rix Wang and Frank S.C. Tseng

Department of Information Management

National Kaohsiung First University of Science and Technology

Kaoshung City, Taiwan

imfrank@ccms.nkfust.edu.tw

Received 14 February 2006; Revised 11 May 2006 ; Accepted 19 May 2006

摘　要

現代化的企業營運愈來愈依賴資訊系統，使得採用磁碟儲存設備的相關應用也急速地擴張，進而引發了儲存設備管理與安全備份解決方案的需求快速增長。在IDC的「2004年上半年亞太儲存軟體追蹤半年報」報告中也指出了台灣地區儲存軟體市場在今年上半年度較去年同期成長了52%，而其中成長的主要力道來自於儲存資源管理軟體。然而目前企業組織所採用的備份機制，仍然是以鏡像停機的備份管理居多，卻鮮少以檔案即時備份的架構來進行直接備份管理。本論文以目前使用較爲普遍的NTFS檔案系統作爲檔案資料備份工作環境，實際建構了一個檔案同步備份與保密安全系統，該系統可以在指定目錄資料夾下，不論是本機或網路端的位置，均能對檔案做即時監看與同步安全備份。在此系統程式中，使用者可以透過事先設定好的資訊，將監看目錄資料夾內的檔案資料利用檔案加解密演算法與檔案驗證的方式產生並複製一份具備保密安全功能的備份至指定的目錄位置中，達成檔案複製備份與資料保密之目的。我們也對整個系統做了實驗性的效能評估，發現即使在檔案公用的情況下，只要採用一般大眾的個人電腦基本配備，就可以依系統實際執行的評估數據，在檔案數量低於500個的情況下，達到100%的安全複製能力。這讓我們的系統很適合台灣中小企業在備份作業上的實務運作需求。

關鍵詞：儲存資源管理，備份管理系統，加解密演算法，檔案驗證

* 通訊作者

ABSTRACT

As the daily business operation of enterprises gradually rely on information systems, the applications of common enterprise storage proliferate rapidly, which caused the demands of a secure management solution for backing up useful files. One of the surveys in IDC regarding "Asia/Pacific Semiannual Storage Software Tracker, H1 2004" has pointed out that on the first half of the year in 2004, the growing of the market in Taiwan for storage management solution has been lifted up to 52%, when comparing to the same period of last year. The growth mainly comes from the market of storage management software. However, the commonly used backup mechanism for most of the enterprises still needs to be accomplished by shutting down systems with image mirroring management. They rarely adopt on-line and real-time approaches to backup files directly. In this paper, we propose a solid system design on top of the popular NTFS file system, to practically implement a backup environment for critical files. Besides, we have implemented a practical and secure file backing up system, which backs up monitored files or directories with encryption to a dedicated destination. According to our experiment, the result shows that the system can safely back up files without loss by using only ordinary personal computers, when the monitored files are less than 500. That makes our system a very practical and feasible solution for most of the small and middle enterprises in Taiwan.

Keywords：Storage resource management, File back up, Encryption algorithm, File verification.

一、前 言

1.1 研究動機與目的

根據國際數據資訊 (IDC，International Data Corporation) 在 2004 年底發表的「2005 中國 IT 產業 6 大市場機會」研究報告中提出：2005 年全球 IT 市場將持續在積極性的市場震盪中調整，而中國 IT 市場將在這個震盪中擁有巨大商機和挑戰。其原因在於 IDC 評估 2004 年亞太地區市場較 2003 年成長約 10%，其中 50%以上的成長量是來自於中國。而亞太地區於 IT 投資的比率上，儲存管理可以說是名列第一，預計金額約達美金 40 億元左右 [1]。由此可見，企業儲存與備份安全管理在近代資訊科技發達的影響之下益形重要，而這也激發了本研究所要探討的主題。

在現今資訊系統儲存環境不斷改變的情況下，分散式的作業使得儲存設備的管理或操作都將更爲複雜。在 IDC 的另一份報告中也明白指出：企業資訊系統耗費在分散式儲存產品、環境中的人力成本，已佔總費用中高達 65% 的比例。由於目前企業組織所採用的備份機制，仍然是以鏡像停機的備份管理居多，卻鮮少以檔案即時備份的架構來進行直接備份管理，若加上與日俱增的作業系統漏洞、病毒肆虐，以及人爲疏忽等威脅因素，則即時且動態製作備份檔案以應緊急之需，使企業減少災害的損失，加速復原速度，就成了非常重要的課題。當然，整體環境尚需對檔案實施有效的存取安全管理，預防內部人員侵越存取權限違法攫取資訊，且亦不能影響大量系統執行效能的目標。因此，本論文的貢獻在於：提供一套安全、實用而可靠的線上檔案備份機制，讓企業得以在保有檔案的安全保護機制之外，還能夠主動彙整企業的檔案文件，提供整體人員一個檔案匯集儲存庫，做爲企業知識管理與分享的基礎。

1.2 研究範圍與限制

針對這個研究課題，我們應用了傳統檔案與資料夾異動的訊息特性，結合檔案的加、解密和驗證方法，實作了一個檔案與目錄資料夾的同步備份保密安全系統，並以許多不同格式的檔案與目錄資料夾加以監看。透過作業系統的即時異動訊息通知，以多重執行緒的方式對這些檔案資料加密，或檢核驗證之後再加以複製備份，使得這些資料內容在安全、即時，且不致過分影響作業系統效能的狀況下，達成同步安全備份的結果。也就是說，建立一個以目錄檔案即時安全備份的系統工具，以強化目前鏡像備份工具的不足。

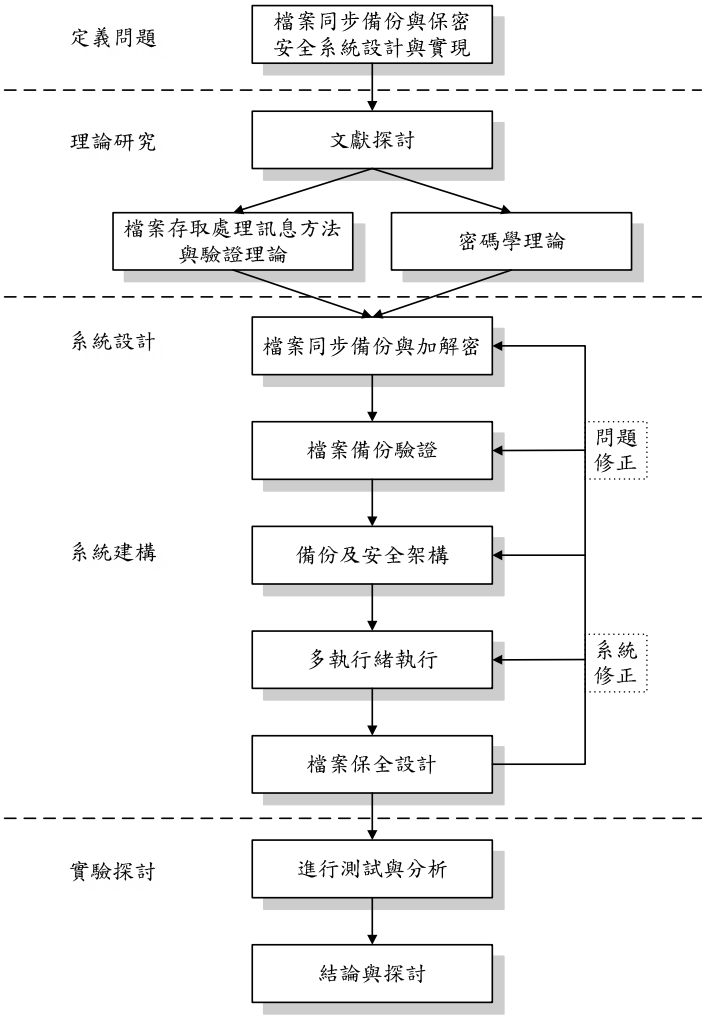
然而，限於設備與環境的條件，本研究具有如下列幾點的研究限制需要說明：

1. 必須採用微軟 Windows 2000 Server (或以上) 的作業系統，且限於使用 NTFS (New Technology File System) 系統檔案格式。
2. 無法允許過多的應用程式常駐在伺服器作業系統記憶區塊內，因爲這會影響到整體的備份效能與效果。

3. 所謂的『同步』(Synchronization) 是指：針對所監看的單一來源檔案資料在經過作業系統異動通知 (例如：檔案的新增或修改) 時程約一百八十秒鐘以內開始進行複製備份，若超過異動時間區間即可能造成備份複製動作的失效，並且超過此一時間則就失去了所謂『同步』的意義。
4. 所欲備份或執行加、解密和驗證的檔案總容量不得超過 4 GB，否則將因檔案過大而可能發生檔案漏失的情形。
5. 當有其他程式需要開啓被監看的備份檔案，必須以共享模式開啓，否則若任何程式以專用模式開啓，將會影響最後備份與保密安全的正確性。

1.3 研究架構

本論文的研究架構共分爲四個主要部分，如圖一所示。



圖一：研究架構圖

第一部分爲「定義問題」，主要說明研究背景、研究動機與目的，和研究的範疇及限制等。第二部分爲「文獻探討與理論研究」，此部分對檔案存取訊息同步備份處理、檔案稽核驗證理論方法以及密碼學設計、檔案加解密應用理論做進一步的瞭解，以訂定本文

的研究系統建立方法。第三部分爲「系統建構」，重點乃基於先前定義的問題與理論的探討，實際建構一個有效並能執行檔案同步備份與檔案加解密或檢核驗證與複製的系統架構。其中當然包含了系統模型設計、決定備份及安全架構、多執行緒執行、檔案保全設計等。第四部分爲「實驗探討」，就是在此系統建立完成後，選取測試樣本，進行實際的系統安全備份測試，並評估分析實施結果的資訊，若測試效果不佳則持續找尋問題所在，再修正系統建構方法，以求達到最理想的同步備份與保密安全效果。

爲確保資料的完整性實施備份所可能遇到的技術問題，基本上可分爲：對支援高度一致性共享資料物件的同步存取以及忽略系統，以及應用程序或使用者對復原資料物件所造成的失敗因素等關鍵技術 (WH Kohler, 1981)。而這些技術中還包含了資料物件的鎖定、時間區隔、交互重疊、衝突與保留狀態等更細部的技術問題 [7]。當代所流行的許多資料備份軟體，如：CA ARCserve、HP OpenView、Omniback Ⅱ、IBM ADSM 等，也都應用了上述的關鍵技術架構了自動定時備份管理、備份介質自動管理、資料庫即時備份管理等功能。這些產品有些注重於對各種作業系統和資料庫平臺的支援，而有些則注重於對該公司軟、硬體產品的支援。但是對於檔案資料即時備份的軟體產品，卻所見不多。因此，在本文中，我們將根據部份上述的理論與技術，就檔案系統即時複製安全備份的層面與角度加以探討，以期能彌補現有市場上可能的不足之處。

二、相關文獻探討與理論研究

2.1 檔案相關技術

在電腦的儲存設備裡，動輒就有成千上萬個檔案是很常見的事情，而這成千上萬個檔案，就如同圖書館裡有上萬本書，如果不分類整理，將會很難收藏、保存、管理與使用。因此爲了達到有效率且可靠的檔案存取並加以分類儲存，作業系統必須同時由下而上的運用「適當的目錄結構」、「檔案系統」與「儲存設備驅動程式」三者互相配合來有效的管理檔案資料的存取，執行檔案備份工作的道理亦相同 [6]。在 NTFS 檔案系統結構中，對儲存設備中的檔案存取是按叢集 (Cluster) 進行分配：一個叢集必需是實體物理儲存設備磁區的整數倍數，並且總數必須是 2 的整數次方。一般的磁區是 512 位元組，但叢集大小實際上會由格式化 (Format) 程式根據磁軌容量大小自動進行分配。其系統結構的每個檔案都有一個稱爲檔案索引號 (File Reference Number) 的 64 位元 (bits) 唯一標記。檔案索引號由兩部分組成：一個是檔案編號，另一個則是檔案順序編號。檔案編號爲 48 個位元，是對應該檔案在 MFT（Master File Table）中的位置，而檔案順序號則隨著每次檔案記錄的重覆使用而增加，主要是爲了協調 NTFS 進行內部一致性檢查所採用的設計[8]。對於檔案與目錄存取的規則，NTFS 則採用了下面的幾個原則 [5]：

1. 對檔案資料內容較小檔案，作業系統會應用 MFT 檔案紀錄組來紀錄儲存檔案的內

容，如：大小、位置及屬性等，並使用常駐狀態 (Resident) 紀錄。

2. 若檔案內容較 MFT 紀錄組長度大許多，檔案的屬性部分則被轉爲非常駐狀態 (Non-Resident)，且用一個檔案屬性指標指向一個外部資料延伸 (Data Runs Or Data Extents) 叢集，該延伸叢集用來存儲那些 MFT 內存儲不下的檔案或目錄資料夾的屬性。
3. 對於大型的檔案其屬性值也是非常駐型態的格式，會在建立主要的 MFT 紀錄組檔案中存在一個至數個指向另一次級 MFT 紀錄組檔案的指標，而次 MFT 紀錄組檔案則內含如上述般的檔案屬性指標指向外部延伸的叢集屬性。
4. NTFS 檔案系統對超大型的檔案則反覆使用並延伸前述的彈性步驟來對檔案進 行紀錄，也就是說檔案越大，其存取的架構越形複雜。

由此可知：NTFS 檔案系統對儲存設備檔案的資料存取，是採用檔案輸出、入 (File I/O) 資料串流 (Stream) 的方式來進行，也就是利用 MFT 中的檔案屬性索引、資料配置緩衝區並配合二元樹，來對檔案串流進行索引與存取 [9]。

因此，微軟作業系統中提供了使用者應用程式一些具有對儲存設備元件進行檔案資料串流存取以及訊息處理的功能與介面 (Application Program Interface, API)，大致可區分爲表一中的幾大類 [10]：

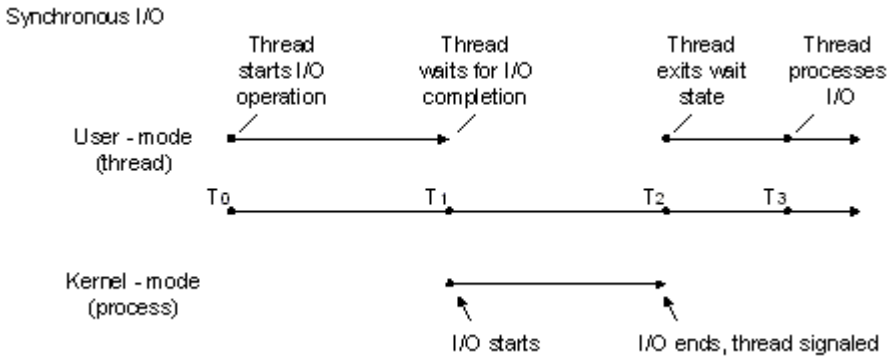
微軟作業系統針對儲存設備檔案的存取，提出了兩種模式 (Microsoft Platform SDK：Synchronous and Asynchronous I/O, 2005)：同步檔案存取 (Synchronous File I/O) 以及非同步檔案存取 (Asynchronous File I/O) 模式，後者也稱爲「覆疊式存取」(Overlapped I/O)。當任何執行緒 (Thread) 對檔案系統提出同步檔案存取的要求時，這個執行緒必須進入等待狀態，直到檔案存取作業執行完成。若執行緒提出的是非同步檔案存取要求，則執行緒可以繼續其他的工作，直到檔案存取作業執行完成後，核心檔案作業系統會中斷並通知執行緒動作已經完成，此部份的同步功能在我們設計備份系統時相當重要。圖二以及圖三分別解釋了上述兩種現象。

2.2 密碼學與資料驗證

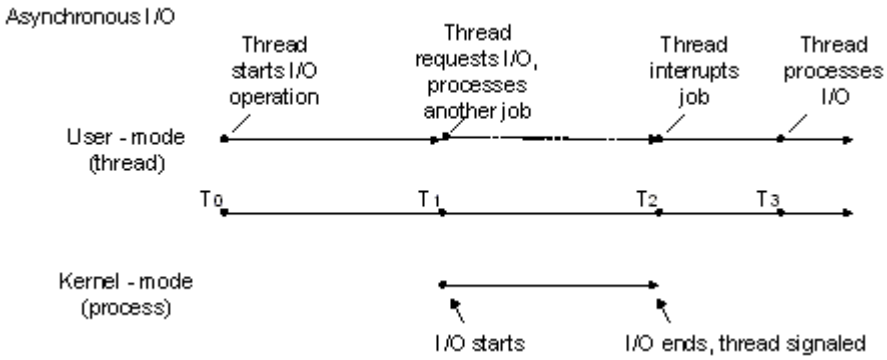
密碼學自西元前 2000 年前發展至今，其基本過程不外乎是對原來稱爲明文(Plaintext)的檔案或資料，依某種加密演算法 (Encryption Algorithm) 進行編碼處理，使其成爲一段不可讀的密文 (Ciphertext)，而在輸入相應的密鑰 (Secret Key)之後才能藉由解密演算法 (Decryption Algorithm) 顯示出原本內容的不同方法。其中主要可分爲三大類的密碼系統：第一類爲稱爲秘密金鑰匙 (Secret Key) 密碼系統或稱爲對稱金鑰匙 (Symmetric Key) 密碼系統；第二類爲公開金鑰匙 (Public Key) 密碼系統或稱非對稱金鑰匙 (Asymmetric Key) 密碼系統；第三類爲信託金鑰匙式 (Key Escrow) 密碼系統，或金鑰匙回復式 (Key Recovery) 密碼系統[3][13]。在設計本系統的保密安全部份時，首先應該考慮到的因素是安全，其次則是效率，以及加、解碼的難易程度和實用性。由於所採用的方法與需處理的位元組數的不同，以及系統實際上需要的複雜程度並不高，故我們認爲：採用對稱式演算法應該是較合適的選擇。

表一：WIN32 File Management API Functions 資料表

管理功能類別	說　　明
檔案管理類	負責應用程式對作業系統的檔案或週邊設備執行增、刪、複製等管理指令，如：CreateFile、CopyFile、DeleteFile 等函式。
檔案串流控制類	負責應用程式對作業系統的檔案串流或週邊設備執行實際讀寫的函式庫，如：LockFile、ReadFile、WriteFile 等。
檔案映射類	負責應用程式對作業系統的檔案串流或週邊設備執行映射 (Mapping) 的功能，如 CreateFileMapping、OpenFileMapping 等函式。
檔案編碼類	負責應用程式對作業系統的檔案串流或週邊設備執行編碼 (Encrypting) 的功能，如 EncryptFile、DecryptFile 等函式。
檔案壓縮類	負責應用程式對作業系統的檔案串流或週邊設備執行壓縮 (Compressing) 的功能，如：LZOpenFile、LZCopy、LZInit、LZRead、LZClose 等函式。

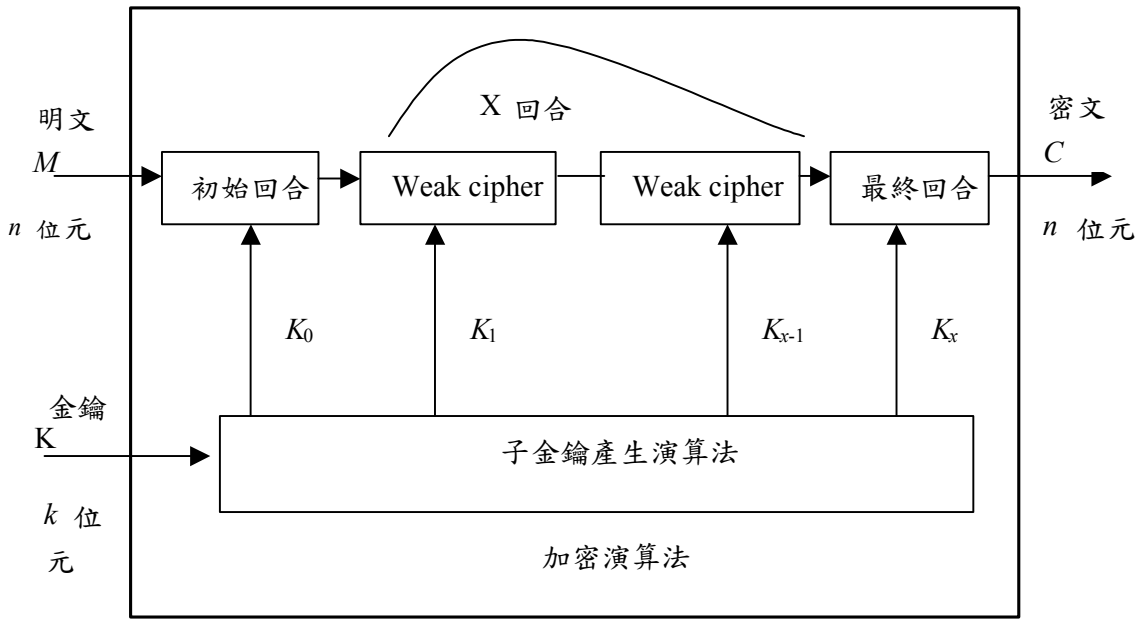


圖二：同步檔案存取 (Synchronous File I/O) 模式



圖三：非同步檔案存取 (Asynchronous File I/O) 模式

在選擇演算法之前，我們必須先瞭解塊狀密碼器 (Block Cipher) 這種元件，它是資訊安全中一個非常重要的密碼元件，主要的特點是加解密的速度非常快，並可達成資訊的隱私性、資訊完整性，和資訊鑑別性等要求。常用的運算方法有「替換」(Substitution)、「換位」(Transposition)、「乘法」(Multiplication)，以及「互斥或」(XOR) 等，並反覆執行類似作用的程序結構，以達到加、解密的效果 [2]，如圖四所示。



圖四：塊狀密碼器結構圖

近年來，由於資訊科技技術大幅進步，使得電腦運算能力大幅提昇，因此美國國家標準與技術協會 (NIST) 於 1997 年 4 月正式公告徵求下一代的塊狀密碼器的先進加密標準 (Advanced Encryption Standard, AES)，來保護具有敏感特性的聯邦政府資料。此新一代的塊狀密碼器安全性設計必須在三重資料加密 (Triple Data Encryption Standard, 3-DES) 標準之上，並且需擁有更高的效率、更合理的成本、更長的金鑰匙位元數，以及更強的反破解能力，以便讓資料的安全性受到更長時間的保護。在參考我國國家檔案管理局於民國九十年完成的「電子檔案儲存之安全認證研究」研究報告中所列舉的各種加、解密演算法優、缺點內容後 [1]，我們瞭解了：非對稱式的密碼系統太過於繁複，且處理效能較慢，而 DES 密碼系統則安全性較低且金鑰匙過短，因此我們選擇了對稱性先進加密標準演算法 (AES) 中的 Rijndael 以及 BlowFish 加密演算法，以滿足本系統在保密安全、效率和實用性的設計需求，而這兩種演算法也均運用到塊狀密碼器密碼元件的原理。

爲了保證資料的完整性，我們必須在系統同步備份執行動作完成後，進行檔案資料正確性的驗證，因此，從設計此同步檔案備份安全系統的觀點上來看，必須兼顧資料的完整性與安全性，以及執行效能與穩定性。我們利用循環冗餘校驗技術 (Cyclic Redundancy Check, CRC)，提供了一個合理而迅速的方法來檢驗兩個檔案的相似程度。其檢驗過程簡單描述如下：在某一端根據要校驗的 X 位元二進制碼的資料序列，以一定的規則產生一個檢驗用的 n 位元循環冗餘校驗碼，而在另一端也根據資料序列內容和同樣

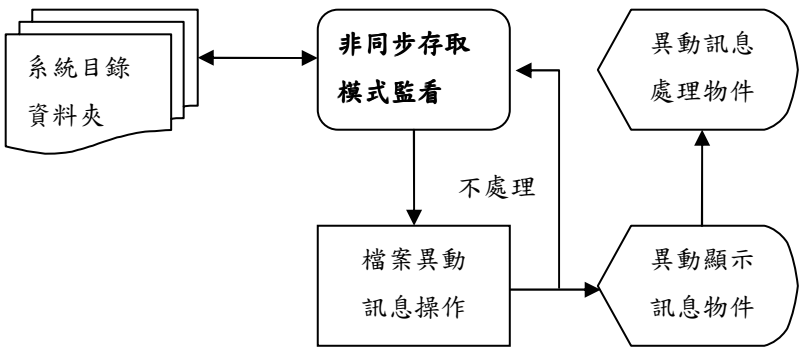
的規則產生另一個檢驗碼後進行比對檢驗，以確定這兩個資訊序列內容是否相同。因此，在資料檢驗時，可以作爲檔案資料比對檢驗的參考方法。計算方式是將待檢驗的區塊視爲一堆連續位元所構成的一整個數值，通常以二進位表示，並將此數值除以一特定的除數，此除數又稱爲衍生多項式 (Generation Polynomial)。此除數一般皆由設計硬體或軟體的廠商所提供，而除數值位元數目則視欲得到的 CRC 位元數目而定 [4]。除了運用循環冗餘校驗 (CRC) 對備份檔案的完整性做驗證之外，也可以利用其他的方法例如資訊摘要校驗演算法 (Message-Digest Algorithm 5, MD5) 以及逐段位元比對等方式來執行驗證的工作。

由於我們在考慮驗證方式時，也同時應該考慮兩個問題：一個是系統本身的效能問題，另一個則是系統運作即時性的同步問題。然而，若必須在效能和驗證的即時與完整性之間作出選擇時，速度應該放在較優先的選項。因此，考量本節所提及的兩種除循環冗餘校驗之外的檔案驗證方法、和 NTFS 檔案系統的互動性因素，以及驗證互補性因素等，我們選擇了 CRC32 這種檔案驗證機制，以求達到運用兩種驗證方法互相補強，並確保檔案完整性的目的，並符合效能的要求。

三、檔案同步備份與加解密設計

3.1 檔案同步備份

在前幾節中曾提到對儲存設備檔案的存取，有同步以及非同步檔案存取兩種模式。經過查閱微軟系統開發網路（Microsoft Developer Network，MSDN）中，對儲存設備檔案系統異動訊息相關的資料後，決定採用先『**監看**』（Monitor Watch）來源目錄檔案資料的方式，並獲得異動的訊息才執行同步備份。但如此所面臨的問題就是，要採取同步檔案存取監看模式或是非同步檔案存取監看模式。若採用同步檔案存取監看模式，則系統必須在檔案異動訊息全部執行完成之後才能執行備份動作，這樣不符合系統設計的目的，因此應採取非同步檔案存取監看模式，並加上適當的補強方式來進行監看與執行即時同步備份的功能。圖五顯示了基本監看程式模組的架構，以及訊息處理的流程。



圖五：基本監看程式模組架構圖

3.2 檔案共享安全限制

當我們設計的系統開始監看目錄資料夾與檔案後，可能會有其他應用程式與執行緒來存取我們監看的目錄資料夾與檔案資料，因此將產生目錄資料夾與檔案的異動訊息，此時會面臨『沒有存取權限』以及『共用違規』的這兩個大問題。尤其當系統管理者未授予安全備份系統存取的權限，或者是若其他執行緒正以專用讀寫方式開啓檔案時，將會產生共用違規的現象。以下就針對這兩個問題來找出解決之道：

1. 沒有存取權限：根據 MSDN 中的文獻中指出，這個問題可以利用 CreateFile 這個 API 執行時所傳回的傳回值來解決。技術文章中說明，當使用這個函式執行對儲存設備的存取時，若執行成功函式會傳回指定目錄資料夾與檔案的檔案操作識別碼。而若此函式執行失敗，則會傳回作業系統預先設定好的錯誤值，我們可以利用所傳回的錯誤值來顯示這沒有存取權限的錯誤訊息提示介面。而這個取得作業系統預先設定好的錯誤值函式的原型意義如下：

   DWORD GetLastError(VOID)

   其中應用這個函式所取得的錯誤值是一個雙精準度的數值，而這個數值可以在 MSDN 技術文件中找到所代表的錯誤原因後顯示訊息通知界面。

2. 共用違規：要解決這個問題，首先要了解這個問題的造成原因以及其解決方案有可能是底下幾種情形：

a. 防毒軟體程式的保護。

b. 作業系統或檔案系統最佳化的執行保護。

c. 作業系統或檔案系統對使用者的檔案存取策略重覆存取或發生衝突，例如：作業系統或檔案系統正在對使用者所存取的檔案資料進行安全性權限修正或磁碟區塊重新配置等現象。

d. 其他應用程式與執行緒正以專用模式開啓檔案資料。

表二說明了在設計此系統時，在所監看的來源與進行同步複製之間，可能遇到的檔案共享安全限制中的幾種情形，與可進行處理的方式。

運用多重執行緒的原因通常是爲達成提高效能，以及經過系統自動排程優先順序後再執行以達到近似同步進行的效果。一般而言，在多重執行緒中的每一個執行緒，所執行的工作大多是需要耗費大量時間與反覆執行的例行性工作 (Routine)。因此我們爲了解決系統在監看或複製時可能遭遇到共享違規的問題，也採取這種作法。系統中具體的實現說明如下：在備份複製檔案資料遇到共享違規時，我們在主系統程式建立監看物件成功的同時，亦自動建立一個呼叫複製備份多重執行緒的事件，以及一個對應此呼叫的多重執行緒來執行任務。以下是系統部份程式碼：

表二：檔案共享安全限制與處理方式摘要表

限制情況	處理方式	備註
沒有存取權限	使用 GetLastError 函式傳回值。	顯示沒有存取權限的錯誤訊息提示。
防毒軟體程式保護	選擇顯示訊息界面通知使用者自行修正。	或在使用手冊中加註說明。
作業系統或檔案系統最佳化執行保護	同上。	屬於無可避免的例外情形之一。
檔案存取策略重覆存取或發生衝突	同上。	亦屬於無可避免的例外情形。
執行緒正以專用模式開啓檔案資料	**利用多重執行緒的方式進行克服**。	設定最佳等待時間。

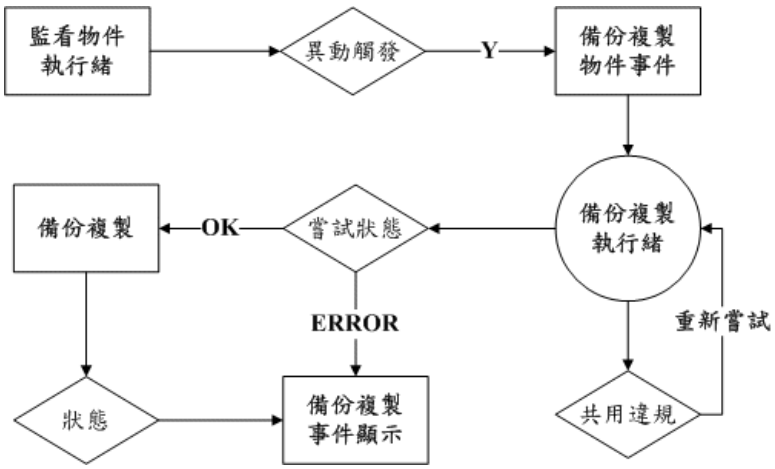
```
void CmnDirChangeHandler_ListBox::OnAdd_Action()
{   int x_thi = 0; // 執行緒測試計數器
    CWinThread *pThread = AfxBeginThread(ThreadRoute,(LPVOID)this ,
                    THREAD_PRIORITY_NORMAL,10000000,0);
// 建立並呼叫執行緒，並將自己當作參數傳遞給執行緒 }
```

而在對應的多重執行緒，則執行檔案資料的開啓測試，若失敗則重試數十次，並在最後一次執行失敗後指示顯示錯誤訊息。

```
UINT ThreadRoute(LPVOID lparamobj)
{    // This Thread Will Cover When ERROR_SHARING_VIOLATION
CDirectoryChangeHandler_ListBox* pWnd =
(CDirectoryChangeHandler_ListBox*)lparamobj;
    CString lpszFile = pWnd->m_AddCopyFileStr;
    CFileFind sth_FindActFile;
    sth_FindActFile.FindFile(lpszFile);
    ....................................................
    pWnd->OnWatch_Action(lpszFile); // 開始執行備份複製
    AfxEndThread(0,true);
#define MAXRETRIES  68   // 重試次數
#define RETRYDELAY  2800 // 重試時間(以毫秒計算)
HANDLE  hFile    = INVALID_HANDLE_VALUE;
```

```
DWORD   dwRetries = 0;
BOOL    bSuccess  = FALSE;
DWORD   dwErr   = 0;
do { hFile = CreateFile( lpszFile,
    GENERIC_READ,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL ,
    NULL); // 這裡使用此函式進行共用違規偵測
    if ( INVALID_HANDLE_VALUE == hFile )
      {  dwErr = GetLastError();
          if ( ERROR_SHARING_VIOLATION == dwErr ) {
            dwRetries += 1;
            Sleep(RETRYDELAY);
            continue;       }
        else { // Another error occurred.
            pWnd->On_OutputErrStr(_T("非共用式違規正在重新嘗試執行........"));
            dwRetries += 1;
            Sleep(RETRYDELAY);
            continue; }
    }
    bSuccess = TRUE;
    break;
} while ( dwRetries < MAXRETRIES );
if  ( bSuccess ) { // You Succeeded in Opening the File.
    CloseHandle(hFile);  // Close The File
    pWnd->OnWatch_Action(lpszFile); }
else    { // Failure Occurs. Do Graceful Error Handling.
    MessageBox(_T("Tried to update data file but it was already in use"),
        MB_OK | MB_ICONSTOP );  // 顯示錯誤訊息 }
AfxEndThread(0,true); // 釋放執行緒
return; }
```

簡而言之，解決在複製備份時所發生的共用存取違規問題，並同時建立自動呼叫複製備份多重執行緒，不失爲一個良好的解決方案如圖六所示。

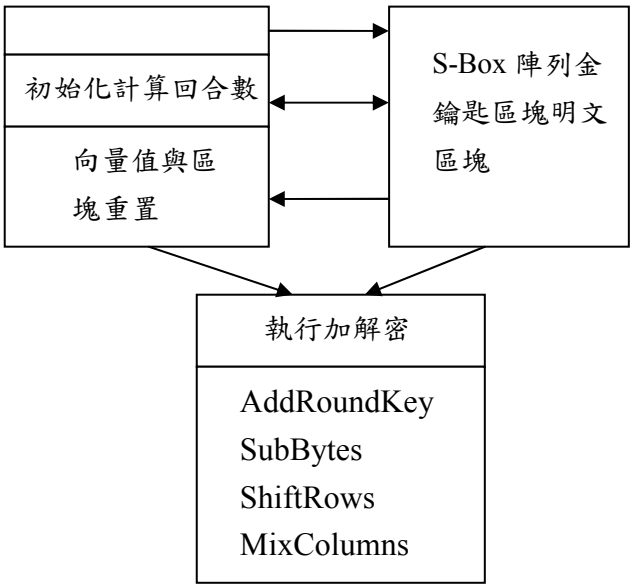


圖六：複製備份多重執行緒執行流程圖

3.3 檔案加解密

本小節說明：如何將檔案資料，以檔案資料串流方式設計系統，以實現先前所提到的先進加密演算法與 BlowFish 演算法這兩種加、解密動作的設計原則。首先我們對這兩種演算法應如何設計進行分析：

1. Rijndael 先進加密演算法部份：根據定義與數學公式，首先以陣列定義 S-Box 值與金鑰匙區塊，以及明文加密區塊的大小，然後進行塊狀密碼器的初始化（Initialize）。接著要初始化每個計算階段的向量值加解密區塊，最後再執行加入回合金鑰匙（AddRoundKey）、位元組取代轉換（SubBytes）、移列轉換（ShiftRows）置換以及混行轉換（MixColumns）等加解密的計算。整個架構設計如圖七所示。



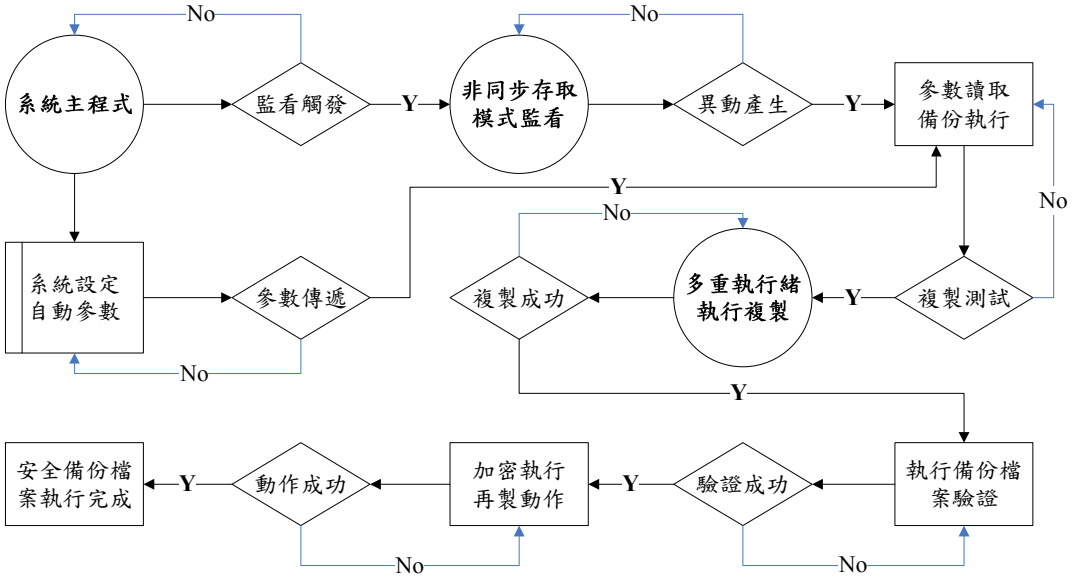
圖七：Rijndael 先進加密演算法類別架構圖

2. BlowFish 演算法：必須先定義陣列 S-Box 值與金鑰匙區塊以及明文加密區塊的大小，然後計算金鑰匙值。當金鑰匙值計算完成後，就以相同的模式初始化每個計算階段的向量值和加、解密區塊，並開始進行加、解密的計算。但是與上一個演算法不同的是，此演算法多了對 P-Arrary 的定義和 f 功能函式的計算，但並不在每回合經過四個階段函式的計算中。整個架構與圖七相當類似，故不再贅述。

以上這兩個演算法在架構上非常類似。所以基於模組化的原則，我們除了將以類別的方式設計他們的內容外，亦可以設計一標準化的物件類別，延伸繼承並覆寫這兩種演算法的物件類別和其中的一些初始化事件與程序。如此一來，不但在主系統中可以隨時宣告及呼叫這個標準化的物件類別來進行操作，甚至也可以利用傳遞參數的不同，而直接使用不同的演算法對檔案資料進行加、解密動作。

3.4 系統模型設計

根據前面的許多研究結果，將其轉換成應用系統的整體系統模型架構與流程可以如圖八來表示。



圖八：系統整體模型架構流程圖

我們的程式使用了 VC++中的視覺化基礎類別 (Microsoft Foundation Class, MFC) 進行設計，整體架構是從建立一個主要視覺化程式開始後，再載入一個主要的系統視窗程式。在建立完成之後，除了主視窗與其物件類別各事件程序需要撰寫程式加以控制之外，監看程式同樣以類別方式建立，以呼應主視窗類別的指令。而我們在系統中共設計了三個主要類別 CmnDirWatch、CFileNotifyInformation 以及 CMnDirChangeHandler 進行對來源資料夾的監看，並且各自負責處理執行監看與異動產生的事件。其中：

1. **Class CmnDirWatch**：負責系統整體監看動作的執行，並由類別事件將監看的異動訊息取得後做適當的處理。

2. Class CfileNotifyInformation：負責傳遞處理與紀錄監看類別所傳回來的異動訊息，並可讓其他公有類別進行訊息的存取。
3. Class CmnDirChangeHandler：負責對監看異動訊息育狀況的實際掌握合處理操作，例如：顯示訊息或追蹤訊息的傳遞等。

在 Class CmnDirWatch 類別當中，我們運用了在前小節中所提到的 API 監看函式 ReadDirectoryChangesW，進行對來源目錄檔案資料的監看。部份原始程式如下：

```
if( ReadDirectoryChangesW( pdi->m_hDir,   // 傳入異動訊息紀錄通知類別
        pdi->m_Buffer,   // 指定異動訊息所儲存的緩衝區位置
        READ_DIR_CHANGE_BUFFER_SIZE, // 指定爲異動訊息變動的緩衝區
        pdi->m_bWatchSubDir,      // 指定是否要監看來源的子目錄
        pdi->m_dwChangeFilter,   // 指定是否要過濾條件
        &pdi->m_dwBufLength,      // 在非同步模式監看時需要的緩衝區長度
        &pdi->m_Overlapped,       // 指定非同步模式監看物件旗標
        NULL) )
{ // 監看初始化成功，開始正常執行監看
  pdi->m_RunningState = CDirWatchInfo::RUNNING_STATE_NORMAL;
  pdi->m_dwReadDirError = ERROR_SUCCESS; // 設定監看傳遞狀態
  if( pdi->GetChangeHandler() )
        pdi->GetChangeHandler()->On_WatchStarted(ERROR_SUCCESS,
pdi->m_strDirName ); // 正常執行則將操作識別碼傳遞給訊息類別 }
else   // 若監看初始化未成功
{       // 取得錯誤發生的原因並傳遞給訊息類別
  pdi->m_dwReadDirError = GetLastError();
  pdi->GetChangeHandler()->On_WatchStarted(pdi->m_dwReadDirError,
pdi->m_strDirName); }
```

而在檔案驗證部分則建立如下：

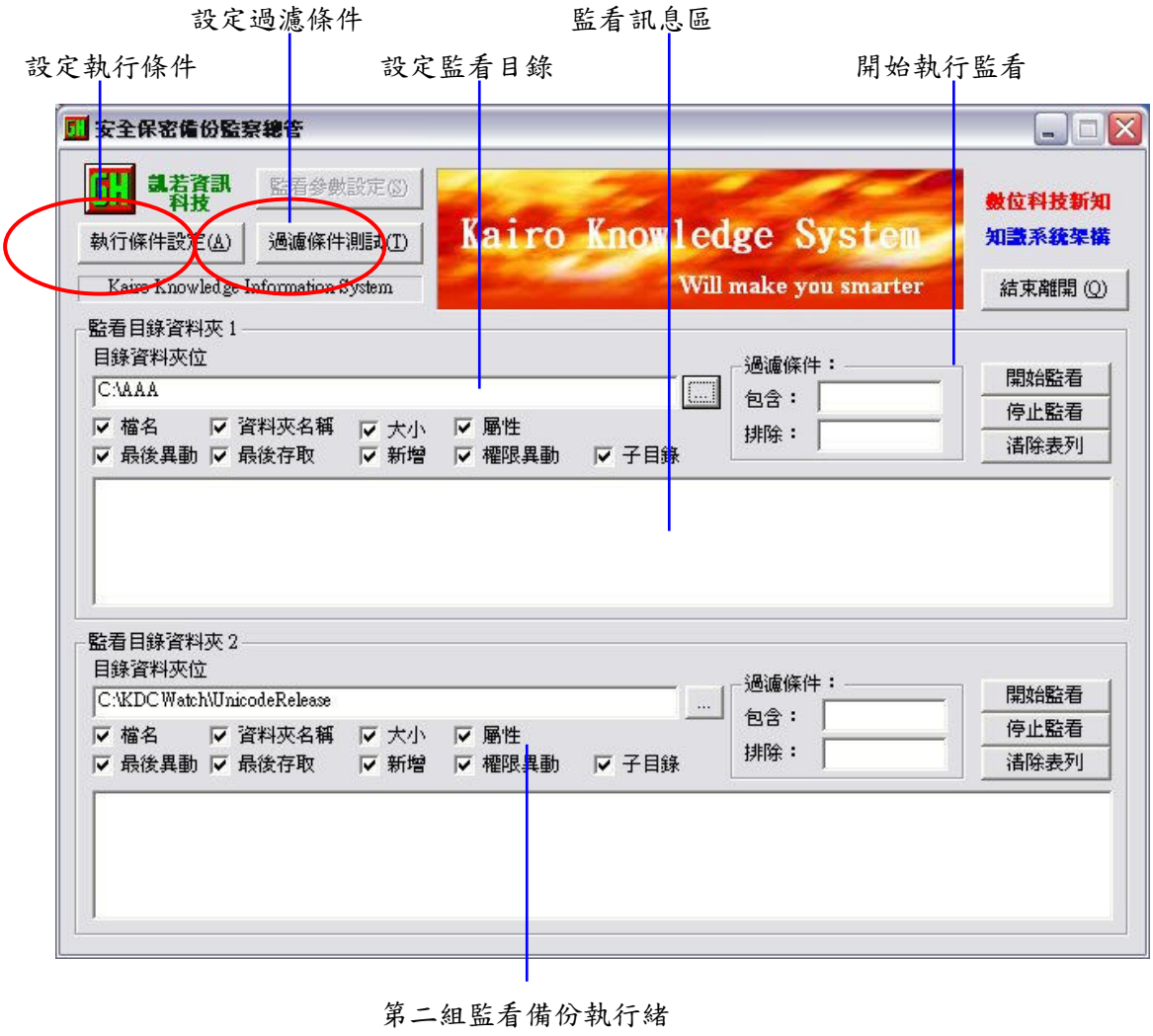
```
file.open((char*)szFilename, ios::in | ios::nocreate | ios::binary, filebuf::sh_read);
if(!file.is_open()) dwErrorCode = file.fail();
else
{       while(nCount)
  { CalcCrc32(buffer[nLoop], dwCrc32); // 計算 CRC 值 }
file.close();
```

以上系統程式即在以檔案資料串流方式將檔案資料載入並計算 CRC32 值以供驗證比對檔案資料完整性的部份。其他的模組詳細架構由於篇幅有限，就不再詳細列舉系統的原始程式內容。

四、系統設計結果分析

4.1 系統設計結果

系統在設計完成後，其主畫面與操作流程中各畫面如下列各圖所示。



圖九：同步備份與保密安全系統主畫面圖

圖十：同步備份與保密安全系統執行畫面圖例

系統操作流程爲：先開啓主系統畫面後，設定備份監看的過濾條件，接著開啓設定執行動作的對話視窗，選定備份複製的目的位置，其中系統監看來源與目標位置皆可以指定爲網路磁碟機，接著再設定執行動作的驗證與加解密參數，確定之後回到主系統畫面按下開始監看按鈕即可。

4.2 測試資料分析

在效能測試的部份，我們訂定了幾個主要的實驗數據項目來收集必要資訊，並依據這些項目中的資訊來達成先前說明的同步複製備份與加、解密安全性的成功率，以及實施時間上的執行效能。各數據指標的意義如下：

1. 次數：爲實施實驗的次數，爲求系統在實際執行時的真正效果與數據的正確性。
2. 檔案數量：由於系統是以檔案複製加密的方式進行安全備份，因此投入的檔案數目不宜太小，且應逐步增加檔案的投入。
3. 總容量大小：這項數據說明了軟、硬體方面，包含：系統中央處理器、隨機存取記憶體、儲存設備速率，以及系統本身的處理速度限制或可能的缺失原因。
4. 執行效能：可以說明系統在執行前幾項不同的基本條件時，所花費的執行效能與時間，提供我們對系統架構與模組改進的參考。
5. 達成率：是指經過系統的訊息監控、處理、複製和驗證加密後，實際真正成功的檔案數量與來源檔案數量的百分比。

接著，我們以設計好的系統，投入約 500 至 1000 個檔案，檔案個別大小並不固定，並且以總容量大小在 2G Byte 以內進行系統的成果測試。以下爲數次系統的實施成果分析表。

表三：同步備份與保密安全系統測試成果表一

次數	檔案數量	總容量大小	執行效能	達成率
1	約 500 個檔案	300 MB	380 Sec	100%
2	約 600 個檔案	600 MB	700 Sec	99.90%
3	約 700 個檔案	800 MB	1010 Sec	98.95%
4	約 800 個檔案	1.2 GB	1583 Sec	94.88%
5	約 900 個檔案	1.5 GB	2422 Sec	92.45%
6	約 1000 個檔案	1.8 GB	3122 Sec	90.66%

(包含加密與檔案驗證)

表三顯示了使用系統時包含設定執行加密與檔案驗證動作時的檔案數量、總容量大小、效能、達成率等實施數據。爲求提升效能與進行不同的多樣化表現，因此我們取消檔案驗證的部份再做提升效能的測試，如表四。

表四：同步備份與保密安全系統測試成果表二

次數	檔案數量	總容量大小	執行效能	達成率
1	約 500 個檔案	300 MB	206 Sec	100%
2	約 600 個檔案	600 MB	421 Sec	99.94%
3	約 700 個檔案	800 MB	625 Sec	98.96%
4	約 800 個檔案	1.2 GB	1283 Sec	95.11%
5	約 900 個檔案	1.5 GB	1985 Sec	92.69%
6	約 1000 個檔案	1.8 GB	2628 Sec	91.33%

(包含檔案加密但不包含驗證)

在上面兩表可看出不使用檔案加密功能時，應可以增加部份的效能與達成率。繼續再以設計好的系統，增加檔案資料的投入約 1000 個以上的檔案，並且總容量大小在 2G Byte 以上進行系統的成果測試，以下表列爲數次系統的實施成果分析表。

表五：同步備份與保密安全系統測試成果表三

次數	檔案數量	總容量大小	執行效能	達成率
1	約 1000 個檔案	2.1 GB	3562 Sec	85.47%
2	約 1100 個檔案	2.5 GB	3988 Sec	78.78%
3	約 1200 個檔案	2.8 GB	4685 Sec	76.66%
4	約 1300 個檔案	3.0 GB	5298 Sec	75.28%
5	約 1400 個檔案	3.5 GB	5764 Sec	72.69%
6	約 1500 個檔案	4.0 GB	6247 Sec	70.28%

(包含加密與檔案驗證)

在我們投入更多，且更大總容量的測試檔案資料，執行加密與檔案驗證動作後的檔案數量、總容量大小、效能、達成率等實施數據如表五。基於同樣的理由，也取消檔案加密的部份再做效能測試，如表六。

表六：同步備份與保密安全系統測試成果表四

次數	檔案數量	總容量大小	執行效能	達成率
1	約 1000 個檔案	2.1 GB	3144 Sec	86.26%
2	約 1100 個檔案	2.5 GB	3645 Sec	78.99%
3	約 1200 個檔案	2.8 GB	4077 Sec	77.59%
4	約 1300 個檔案	3.0 GB	4822 Sec	76.83%
5	約 1400 個檔案	3.5 GB	5184 Sec	73.80%
6	約 1500 個檔案	4.0 GB	5552 Sec	72.79%

(包含檔案加密但不包含驗證)

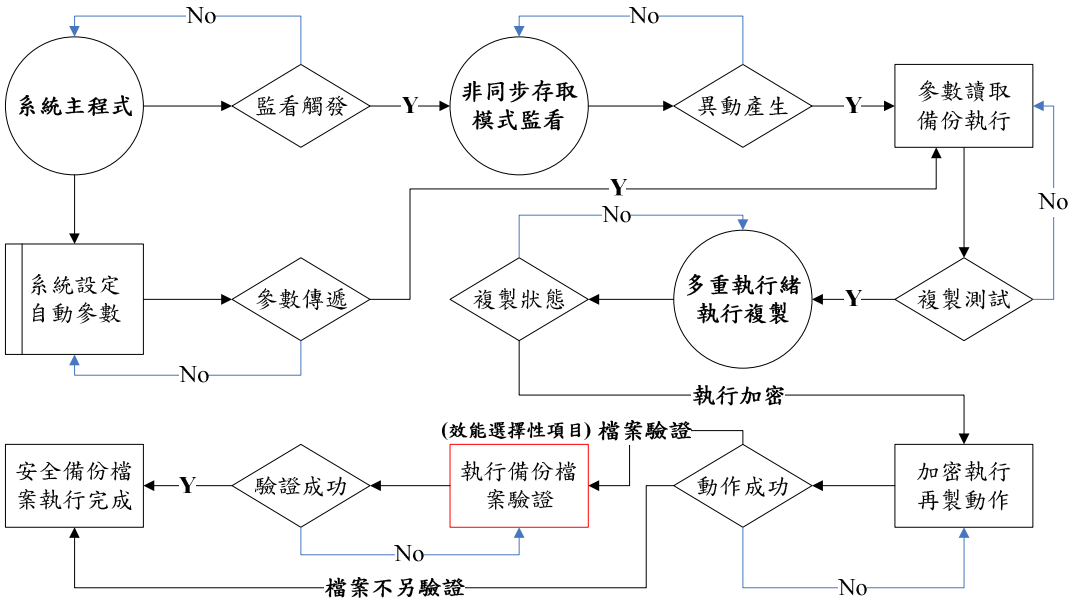
根據以上幾個表列的資料我們得知，系統的效能如果不經過檔案驗證部份可以提升更佳的效能與備份達成率，其中達成率是指檔案總數與總容量。

4.3 系統功能修正模型

經由上小節實施的成果分析之後，我們知道可以調整修正系統的模型架構，並針對這兩個方向來進行：

1. 調整執行緒延遲時間：經過許多次的測驗實施，在一般的檔案大小的情況下，約單一檔案在 5MB 以內，我們得出最佳的執行緒延遲時間約爲 180 秒。在這樣時間下執行緒可以將違規共用的情形解決的達成率較佳，若太長則失去同步備份的意義，若太短則執行達成率會容易偏低。
2. 調整整體效能與達成率：由實驗結果得知，預設值若不實施檔案加密將可提升系統整體的執行效能，因此必須修改系統的整體架構與流程。最後修正的系統架構流程圖如圖十一所示。

經修正過後的模型，經過我們再次測試，所得出的數據資料果然如預期般的略微改進部分的效能。當然，以上這些數據會依不同的硬體配備而有所不同，測試用的配備使用 Celeron 2.4G 的中央處理器，768 MB 的隨機存取記憶體，以及一部 120G - 7200rpm 與一部 40G - 7200rpm 的硬碟機，提供作爲參考。



圖十一：修正後之系統架構流程圖

五、結論

最後，在經過以上的理論探討研究與實際系統的設計實施之後，我們實作出相對應的成果，並經凱若資訊科技公司的實際應用後，得到了以下幾點結論，以作爲未來繼續研究發展的方向：

1. 想要從儲存設備檔案系統的異動訊息獲得相關資訊，可以採用監看來源目錄檔案資料的方式，並且在獲得其異動訊息之後實施執行同步備份。
2. 系統可以利用 CreateFile 這個 API 執行時所傳回的傳回值，解決沒有存取權限的問題，並運用多重執行緒的執行來達成提高效能與近似同步進行和解決共享違規的方案。
3. 基於模組化的原則，應儘量以類別的方式設計檔案加解密演算法的內容，亦可以設計標準化的物件類別，以延伸繼承並覆寫這些演算法內容的物件類別和其中的初始化事件與程序。
4. 預設的系統流程可以將檔案驗證列爲選項，確保檔案的保密與完整，但是若針對較不敏感性的資料，爲求系統最佳的效能，可以直接以預設值實施備份。
5. 系統經過測試可以對所有網路磁碟機以及本機磁碟機進行來源監看，也可以作爲備份的目的位置。若備份目的地爲遠端的網路磁碟機，則端視網路頻寬大小而定。
6. 適當的運用此同步備份與保密安全系統，應可以增進儲存資源管理的效益，降低企業資訊系統耗費在分散式儲存產品，與環境中的人力成本總經費的目的。

雖然，經過本文的研究與探討之後，已經初步驗證了儲存管理與備份的實作技術與困難點，但是仍有許多值得繼續研究發展與努力的空間。以下是我們就此同步備份與保密安全系統而言，所提出的幾個值得繼續研究並且改善的重點：

1. 如何使系統在常用的範圍內，讓大容量複製的安全備份達成率達到 100%。
2. 如何能夠使用在更大容量，甚至無容量限制的安全備份執行上正常運作。
3. 如何加入系統排程，甚至實施整體安全備份計畫，達到更精簡資源、更安全以及更有效的目標。
4. 如何整合資料庫與知識庫系統，或更多方向例如嵌入式硬體 (Embedded System Hardware)，或不同的檔案與作業系統平台 (如：Linux 檔案系統) 的實際應用，以延伸系統整體的價值。

相信後續若能夠朝這幾個方向繼續研究與努力，除了可以更增加本同步備份與保密安全系統的實用價值之外，對整個儲存管理的科技進展，也提供了一部份的幫助。

誌　謝

本研究承蒙國科會計畫 (編號：NSC 94-2416-H-327-009) 的部分贊助，特此感謝。

參考文獻

[1] 黃明祥等，“電子檔案儲存之安全認證研究”，國家檔案管理局，http://www.archives.gov.tw/internet/c_institute_classroom_detail_internet.aspx?tmp=10,1，朝陽科技大學資訊管理系，2001。

[2] 賴溪松等，“AES的簡介與發展”，中華民國資訊安全學會，AES成果發表會，http://www.ccisa.org.tw，November, 2000。

[3] 謝續平，“交大資工所網路安全課程：密碼系統簡介，Chapter 3”，http://dsns.csie.nctu.edu.tw/course/intro-security/2005/，2005。

[4] A. L. Roginsky et al., “Efficient computation of packet CRC from partial CRCs with application to the Cells-In-Frames protocol,” Computer Communications, Vol. 21, 1998, pp.654-661.

[5] C.M. Kozierok, “New Technology File System: NTFS Files and Data Storage,” The PC Guide, http://www.pcguide.com/ref/hdd/file/ntfs/files.htm, Apr. 2001.

[6] D.A. Solomon and M.E. Russinovich, Inside Microsoft Windows 2000,” Third Edition (Microsoft Programming Series), Microsoft Press; 3rd, Sept. 16, 2000.

[7] D.B. Lomet, “High Speed On-line Backup When Using Logical Log Operations,” Proc. ACM SIGMOD: International Conference on Management of Data, 2000, pp. 34-45.

[8] D. Mikhailov, “NTFS File System,” 2000, http://www.digit-life.com/articles/ntfs/

[9] M. Russinovich, “Inside Win2K NTFS, Part 1,” Windows & .NET Magazine, http://www.windowsitpro.com/Article/ArticleID/15719/15719.html, 2000.

[10] Microsoft Platform SDK: Storage, “File Management Functions,” http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/base/file_management_functions.asp, Jan. 2005.

[11] Microsoft Platform SDK: Storage, “Synchronous and Asynchronous I/O,” http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/base/synchronous_and_asynchronous_i_o.asp, Jan. 2005.

[12] W.H. Kohler, “A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems,” ACM Computing Surveys, Vol. 13, June 1981, pp. 149-182.

[13] W. Stallings, Cryptography and Network Security: Principles and Practice, 3rd Ed., Prentice-Hall, Inc. 2002.