國立交通大學

資訊科學與工程研究所

碩 士 論 文

SecCap：自動化個人文件安全系統・使用智慧卡認証

SecCap：Automatically Securing Personal Documents
with Smartcard Authentication

研 究 生：陳癸夫

指導教授：楊 武　教授

中 華 民 國　九 十 七　年　七　月

SecCap：自動化個人文件安全系統，使用智慧卡認証
SecCap：Automatically Securing Personal Documents with Smartcard Authentication

研 究 生：陳癸夫　　Student：Kuei-Fu Chen

指導教授：楊 武　　Advisor：Wuu Yang

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

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

Computer Science

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

SecCap：自動化個人文件安全系統・使用智慧卡認証

學生：陳癸夫　　指導教授：楊 武 博士

國立交通大學資訊科學與工程研究所

摘要

文件安全是現今個人電腦上一項很重要的課題。根據統計數字顯示[25]，全球高達 92%的終端使用者皆使用Microsoft Windows作業系統，其中亦有90%的使用者不會將電腦內部的私人機密文件進行加密保護。超過 80%的安全威脅來自企業內部[20]，若有心想竊盜資料的攻擊者可取得該電腦實體，所有未加密的文件將不保。

使用者不將私人機密文件進行加密的原因，主要是他認爲手動對其加解密需多幾道繁雜的手續，或是不了解加密保護的重要性。

有鑑於此，在這篇論文當中，我們提出了一個軟體系統：SecCap，建構於 NTFS 檔案系統、Windows 使用者與金鑰管理之上，透過智慧卡做個人身份辨認與驗証，在可接受的效能負擔下對私人機密文件進行自動化加解密。主要是希望能解決以上問題，讓使用者能更容易地達到個人文件安全的目地。

SecCap：Automatically Securing Personal Documents with Smartcard Authentication

Student: Kuei-Fu Chen Advisor: Dr. Wuu Yang

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract

Document security is now an important issue on personal computer. According to the statistics show[25], as high as 92% of end-users are using Microsoft Windows operating system of the world. In which, 90% of users will not encrypt to protect their confidential documents in their computers. More than 80% of security threats are issued from internal of company[20]. If the attacker, who think to steal data, can obtain the computer entity, all unencrypted documents are exposed.

The user will not encrypt his confidential documents because he thinks it needs more complex procedure to encrypt them manually, or does not understand the importance of encryption protection.

In view of this, in this paper, we propose a software system: SecCap, which is based on NTFS file system, Windows user and key management. SecCap authenticates the user by smartcard, and automatically encrypts the confidential documents with acceptable overhead. This is the main hope to solve above problem, so that users can more easily reach to goal of personal document security.

致謝

首先，誠摯地感謝我的指導教授楊武博士。這幾年來老師的耐心、不時地討論並指點我正確的方向，不論是靈感的啓發或難題的解決，都詳細的給予指教，使我在這些年中獲益匪淺。老師對於學問的嚴謹亦是我輩學習的典範，若非您的諄諄教誨，學生是難以順利完成本論文的。

三年的日子裡，實驗室共同生活的點滴、學術上的討論、難熬的夜晚、昏睡的下午，還有考試前的不安……感謝眾位學長、同學、學弟們的互相砥礪，你們的陪伴讓研究生活變得絢麗多彩。感謝學長信達、宸瑋、和博曄，不厭其煩的與我討論實驗上的問題。也感謝一起打拼的同學們文均、志誠、還有冠志，有了你們的幫忙，使我更順利通過這場考驗。炒熱實驗室氣氛的學弟們，帥維、奕圻、禮君、俊宇、冠旭、德發、有倫、耀崙、培翔，也要感謝你們的關心與問候，讓我熬過了緊繃的時光。

女友智玉在背後默默的支持更是我前進的動力，沒有妳的體諒與包容，相信我的研究生活將是很不同的光景。前女友菡珍、明玉，也由衷感謝妳們曾經陪著我渡過這段時光。最後，謹以此文獻給我摯愛的雙親，感謝您們不時給予打氣，您們是我心靈上的支柱，讓我能夠心無旁騖的專心完成我的論文。感謝所有陪伴我走過這三年研究生涯的人，謝謝你們。

目錄

圖目錄

第一章　序論

1.1 研究動機

隨著電腦相關產業與學術的快速發展，資訊安全愈來愈被受重視。諸如網路安全、系統安全、文件安全等等，皆有許多成果應用與研究正在進行。對於大部分的終端使用者而言，他們可能對資訊安全不太了解、甚至對電腦不太熟悉的情況下，便時常使用這些方便的工具。這將產生許多的安全性問題，如病毒肆虐、駭客入侵等等。

不論是大小型企業、政府機關、或是個人終端使用者，其中一項愈來愈被重視的問題－文件安全，已是目前資訊爆炸時代裡所避免不了的。近年來有許多民眾，因使用電腦不當，如安裝了來路不明的軟體、電腦中毒等等，導致個人私密資料外洩，甚至政府機關、警備單位等等，同樣都有這些問題[27]。這些都是使用者忽略了個人私密文件安全的重要性。

文件安全是我們及身邊的人將會面臨的問題。對一般使用者而言，他們可能不太想要去了解文件安全，卻又需要依賴電腦做爲生活、工作等的工具。我們認爲這個問題必須想辦法解決，因此便開始著手於這篇論文、這項作品。

1.2 研究目標

一般而言，使用者不會將私人文件做加密的原因，不外乎如下：

- 使用者認爲文件即使不加密，放在本機電腦也是安全的
- 使用者不了解文件安全的重要性
- 使用者不清楚如何將文件加解密
- 使用者認爲手動將文件加解密很麻煩

- 使用者認爲直接使用內建的或目前即有的軟體做檔案加解密，並沒有眞正保護到文件的效果

上述的原因裡，一般的檔案加解密軟體，大都只使用一個使用者定義的通關密語作爲身份認証與檔案加解密金鑰的依據。對於大多數使用者的不良習性，他們所設定的通關密語很可能是簡單的、容易猜測的。我們認爲，單憑這些資訊應用於文件安全是不夠的。並且因爲上述的原因，我們必需提供一個簡單、易上手的使用者介面，自動地爲文件做加解密。

因此，我們規劃了一個軟體系統：SecCap，用以解決這些問題。透過智慧卡的身份認証、與使用者定義之通關密語，對所存取的檔案進行自動化即時加解密的動作。透過 SecCap，使用者僅須像平常一樣，開啓文件編輯軟體存取檔案，即可擁有一定程度的文件安全，不必再手動介入進行加解密動作。

由於一切的加解密動作均爲自動化執行，在保持文件安全的情況下，我們將使用者的不便盡量減至最低，以增加使用者使用此系統的意願。

SecCap 的主要貢獻如下：

- 結合與應用智慧卡之身份認証。
- 依據處理程序爲單位，自動化加解密文件。
- 提供圖形化使用者介面 (Graphic user interface, GUI)，以減少使用者爲達到文件安全所需要的手動操作。

SecCap的詳細系統架構圖如Fig. 25。各模組當中，GUI/Shell、FileSpy user-mode communicator爲SecCap實作內容；myOpenSSL (myRSA) 係修改自OpenSSL函式庫；Smartcard authenticator爲SecCap調用內政部憑証管理中心提供的函式庫；FileSpy kernel-mode driver是由IFS-DDK (Installable file system Driver development kit) 所提供之File system filter driver範例之一；其它如EFS (Encrypting file system)、憑証與金鑰管理等係使用了作業系統提供之函式。

1.3 相關背景

SecCap主要依智慧卡做爲驗証工具。1.3.1與1.3.2分別介紹智慧卡與目前所使用的自然人憑証，以及其所信賴的基礎建設。1.3.3介紹SecCap所使用的主要加密模組EFS，它是一個系統設備驅動程式，能即時地對檔案做加解密的動作，對於本系統而言是個很好應用的工具。1.3.4與1.3.5介紹SecCap用於監聽磁碟檔案存取之工具FileSpy，與其一些關於File system filter driver的基本知識。

1.3.1 Smartcard

智慧卡，簡單而言，就是一張有如名片大小的塑膠片，內嵌一顆晶片。目前在其上的應用非常廣泛，如銀行服務、健保卡、大眾運輸車票、門禁系統等等。它就像一台具體而微的電腦，通常有它自己的CPU、RAM、EEPROM、ROM、I/O port等。ISO7816[6]對智慧卡的規格有更詳細的規範。

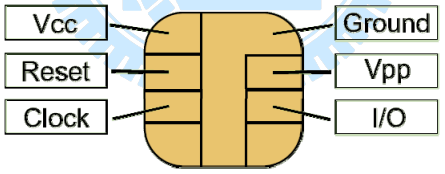

Fig. 1　接觸式晶片卡連接腳位

CPU 負責處理外界下達給它的指令 (I/O、運算)，RAM 作爲暫存資料所用 (計算過程)，EEPROM 有如 Flash 記憶體儲存卡片資訊 (檔案系統、序號、憑証等)，ROM 包含製作時已燒入的作業系統 (Card operating system, COS)。我們可以透過讀卡機，下達指令 (Command) 給智慧卡，並且智慧卡也會有所回應 (Response)。

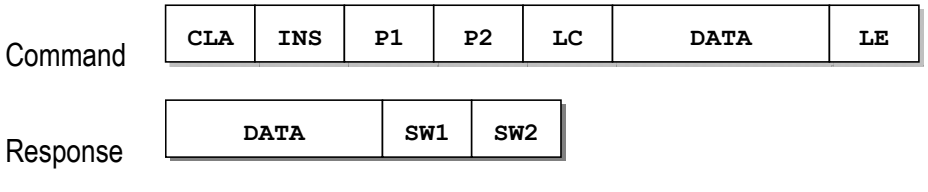

Fig. 2　智慧卡通用指令與回應格式

智慧卡分爲接觸式 (contacted) 與非接觸式 (contact-less)。接觸式智慧卡 (Fig. 1) 的電源，必須由外界讀卡設備來供應；非接觸式智慧卡的電源來源，則必須透過外界讀卡設備的無線載波，來感應智慧卡內的線圈以產生電流，推動智慧卡內的晶片。

智慧卡強調的，不僅是它的運算能力，最重要的是它的安全功能。由於它提供一個對外的介面、保護內部資料的安全、不容易被僞造，是個安全的資料儲存載具。也因此許多的應用，許多的公司皆投入其中。

由於智慧卡的起源，並沒有一個很好的標準組織迅速的對其做規範，使得智慧卡已經應用至市面上同時，許多的組織才著手進行其標準化的內容。這使得每一種卡片皆定義自己的指令代碼與功能，甚至有其自己的檔案系統、檔案、作業系統，其規格封閉、皆不對外公開，使得一般應用程式無法通用於所有種類的卡片。

SecCap需要智慧卡做爲其認証依據。這張卡片最好是大家都已經擁有的，而且格式公開，或是有API與文件提供給研究與開發人員使用。目前健保卡、金融卡，爲目前民眾使用最爲廣泛，但其格式與API等文件皆不公開。次普遍的自然人憑証符合了上列需求，因此本論文當中，選擇了自然人憑証[19]作爲示範。

1.3.2 PKI與自然人憑証

1.3.2.1 公開金鑰基礎建設

1976 年，Whitfield Diffie、Martin Hellman與Ron Rivest、Adi Shamir、Leonard Adleman等人相繼公佈了安全金鑰交換 (Key exchange) [4]與非對稱金鑰演算法 (Asymatric key algorithms) [21]後，整個通訊方式爲之改變。隨著高速電子數位通訊的發展，使用者對安全通訊的需求越來越強。密碼協定在這種訴求下逐漸發展，造就新的密碼原型。全球網際網路發明與擴散後，認證與安全通訊的需求也更加嚴苛。當時在網景 (Netscape) 工作的Taher ElGamal等人發展出SSL / TLS (Secure sockets layer / Transport layer security) [1]協定，包含了金鑰建立、伺服器認證等，公開金鑰基礎建設(Public key infrastructure, PKI) 的架構因此浮現。

密碼學上，公開金鑰基礎建設藉著憑証管理中心 (Certificate authority, CA) 將使用者身份與其公開金鑰串接在一起。每個憑証中心所簽發的使用者憑証與其公開金鑰是唯一的，並且不可僞造。這個建設所依賴的是公開加密演算法與數位簽章，以及通訊期間所使用的金鑰交換技術，對於目前的科技發展來說，很難在有效時間內破解它而得出私密金鑰內容。因此對於資訊安全，其應用非常的廣泛。

1.3.2.2 自然人憑証

內政部憑証管理中心，以自然人憑証[19]實現了政府公開金鑰基礎建設(Government public key infrastructure, GPKI) 。自然人憑証是一張智慧卡，可以說是一個「網路上的身份証」，內存有卡片持有者的個人資訊、加解密與數位簽章用的公開金鑰 (Public key) 與私密金鑰 (Private key) ，並且私密的部分需透過PIN碼驗証後才可取得。

自然人憑証可以用於資訊系統上做爲身份鑑別 (Authorization) 與認証(Authentication) 、資料完整性 (Integrity) 、私密性 (Confidentiality) 、

與不可否認性 (Non-repudiation) 等。其內部數位憑証格式採用X.509，係由國際電信聯盟通信標準化組織 (International telecommunication union for Telecommunication standardization, ITU-T) 爲公開金鑰基礎建設所制定的標準。由於許多憑証應用軟體及程式庫皆支援X.509 格式[7]，並且內政部亦提供其API與相關文件，使得自然人憑証可以很容易地被做爲驗証的工具。

Fig. 3　自然人憑証

1.3.3　Encrypting File System與LSA

SecCap使用了加密檔案系統 (Encrypting file system, EFS) [15]做爲主要加密模組。它建構於NTFS檔案系統之上，並能即時對檔案進行加解密動作。同時，它也是一個File system filter driver。

1.3.3.1　Encrypting File System

加密檔案系統是由 Windows 系列所提供的檔案加密服務。它提供了檔案私密性，與 Windows 使用者、公開與私密金鑰、NTFS 檔案系統做整合。但其未提供其他如完整性與認証保護，並且目前只支援 NTFS 檔案系統。

Operating system	Default algorithm	Other algorithms
Windows 2000	DESX	(none)
Windows XP	DESX	3DES
Windows XP SP1	AES	3DES, DESX
Windows Server 2003	AES	3DES, DESX
Windows Vista	AES	3DES, DESX
Windows Server 2008	AES	3DES, DESX

Fig. 4　EFS 加密演算法與作業系統支援

加密檔案系統混合應用了對稱式加密演算法 (Symmetric encryption algorithm) 與非對稱式加密演算法 (Asymmetric encryption algorithm)。它使用系統內 X.509 憑証作爲主要存取權限的來源，且憑証內之「用途」欄位必須包含有「Encrypting file system」(1.3.6.1.4.1.311.10.3.4) 的權限。每一個已加密的檔案，係由隨機產生的對稱加密金鑰 (Symmetric encryption key) FEK (File encryption key) 做加密保護。EFS 將此 FEK 使用憑証裡的公開金鑰加密後，直接存放於此檔案的 EFS 屬性內。當使用者欲存取已加密的檔案時，他必須擁有對應至此憑証的私密金鑰，才可取得針對此檔案的正確 FEK，而對已加密檔案進行解密。

上述，使用者的公開金鑰存放於「%APPDATA% / Microsoft / System Certificates / My / Certificates」，而私密金鑰經由主金鑰 (Master key, MK) 加密後存放於「%APPDATA% / Microsoft / Crypto / RSA / SID」，並且 Windows 對 MK 本身進行加密 (其加密金鑰由使用者帳號密碼等資訊雜湊產生) 後，存放於「%APPDATA% / Microsoft / Protect / SID」。

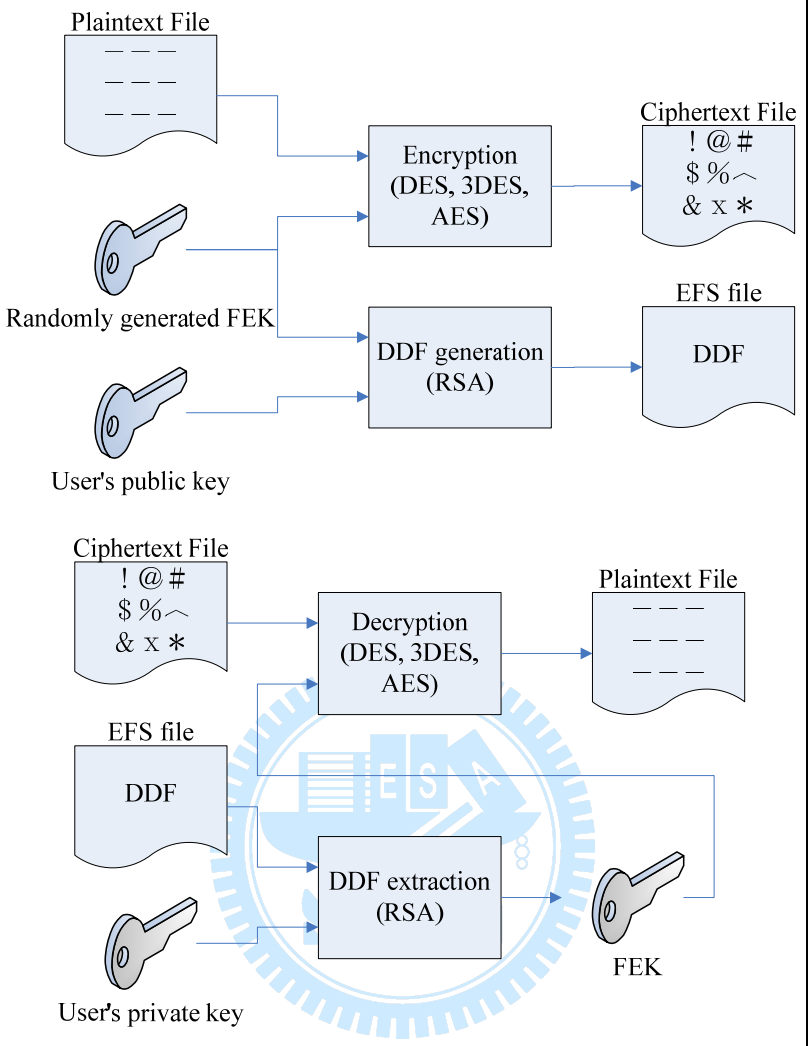

Fig. 5　EFS 加解密流程圖

1.3.3.2　Local Security Authority

LSA (Local security Authority) [26]是一個Windows系統服務，控制與管理系統的安全性原則。它用以驗証Windows使用者登入、處理使用者更改密碼、與建立存取令牌 (Access token) 。並且將系統安全事件寫入記錄檔。

對於加密檔案系統而言，LSA 主要是執行 EFS 的 user-space 指令。LSA 隨機產生一個新的 FEK，並且透過使用者公開金鑰加密，再將兩者回應給發出請求的 EFS (Kernel) ，隨後 EFS 便可使用其 FEK 做檔案加密。反之，LSA 也進行解密 FEK，回傳給 Kernel 以做解密檔案之用。

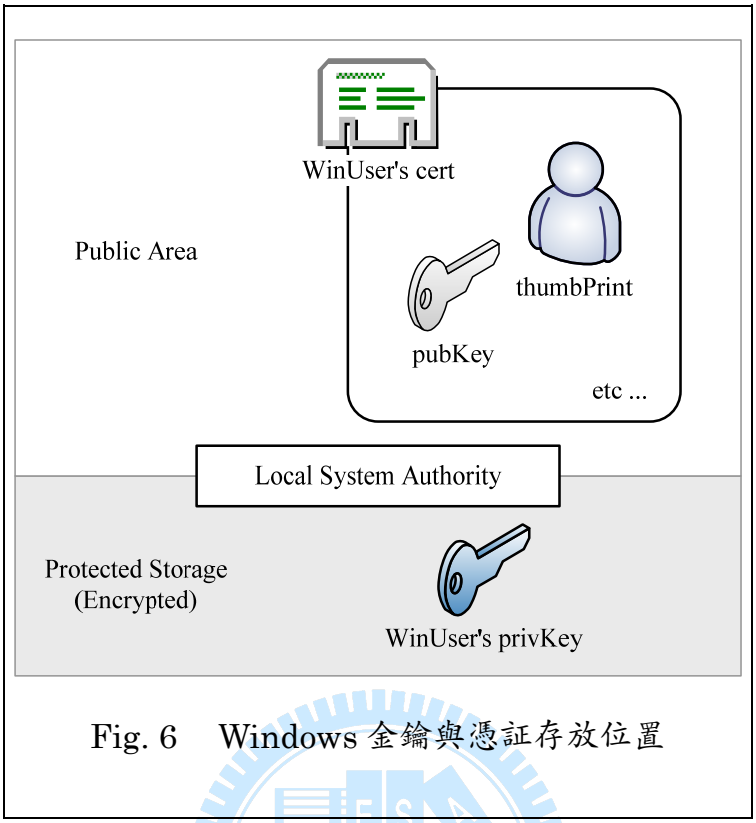

Fig. 6　Windows 金鑰與憑証存放位置

在 LSA 對 FEK 進行加解密的過程中，它會確認目前使用者的 EFS 憑証是否被更改過，是否應重新加密 FEK。爲了增加效率，LSA 亦會快取使用者私密金鑰，並且暫存在一個安全的位置。

1.3.4　File System Filter Driver

File system filter driver (檔案系統過濾驅動程式) [16]是一個系統裝置驅動程式，係屬Windows作業系統中最複雜的驅動程式種類之一。Windows作業系統沒有提供一個完整的介面，讓應用程式可以直接介入檔案存取，因此我們需要另外引用一支File system filter driver (即FileSpy) 來與Windows作業系統合作。若欲發展一個檔案系統或File system filter driver，一般需使用WDK (Windows driver kit) 所提供之IFSDDK (Installable file system Driver development kit) [17]。雖然IFSDDK提供了Sfilter與FileSpy做爲File system filter driver之示例，但其文件對於很多重要的部分與細節僅僅輕描淡寫帶過。關於File system filter driver開發的相關書籍與參考文件也不多。

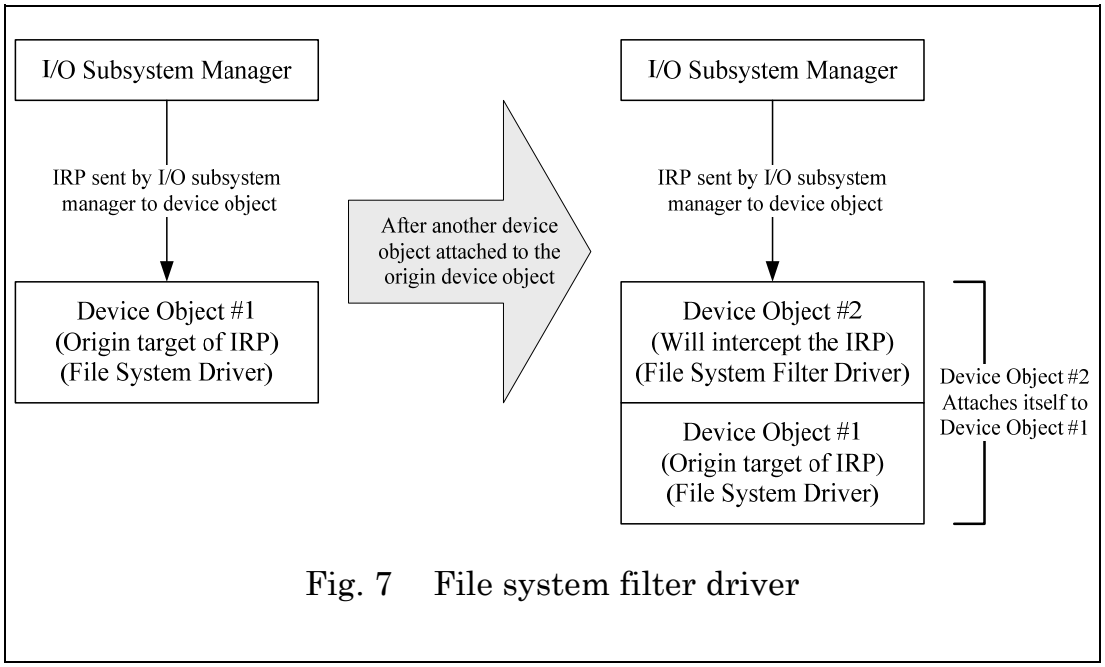

Fig. 7　File system filter driver

一般而言，File system filter driver是界於File system driver與I/O subsystem manager之間，如Fig. 7。File system filter driver介入截取所有目標至檔案系統（或另一個File system filter driver）的請求（Request）事件，如：開啓（Open）、新增（Create）、讀取（Read）、寫入（Write）、關閉（Close）等。介由請求事件到達原先的目標裝置之前，Filter driver可以延伸或代換原先目標裝置所提供的功能。它可以將此請求內容做增修更動，再將其送至原先目標，或記錄複本至另一裝置，甚至將此請求丟棄。通常它是一個選擇性（Optional）的驅動程式，用來新增或修改原先檔案系統的行爲。

根據 Microsoft WDM (Windows driver model) 所定義，File system filter driver 需實作兩類的分發函式 (Dispatch functions)：

- IRP (I/O request packet) 系列函式：如 IRP_MJ_CREATE、IRP_MJ_SET_INFORMATION、IRP_MJ_WRITE、IRP_MJ_READ 等等約 28 項
- Fast I/O 系列函式：如 FastIoWrite、FastIoDeviceControl、MdlReadCompleteCompressed 等等約 22 項

上述 IRP 係由 I/O subsystem manager 所發送，Fast I/O 係由 Cache

subsystem manager 所發送，而且根據作業系統版本不同，Filter driver 必須定義或實作的函式亦有所差異。它們的功能基本上相同，只是提供不同的 API 供驅動程式使用，效率亦有些微差別。一般而言，這是驅動程式與 Windows 作業系統之間的溝通方式，其它的作業系統 (如 Linux) 並無對驅動程式做此定義與要求，但亦有工具可對這些特定的 Windows 驅動程式做轉換，使其能使用於其它作業系統。

通常需要撰寫 File system filter driver 的目地，不外乎有下列幾種：

- 用於防毒引擎。它希望能在系統存取檔案的時後，捕獲或攔截其讀寫的內容，然後檢測其中是否有病毒程式碼的存在。
- 用於檔案系統的附加功能。例如希望在檔案寫入過程中對內容進行加密，或是內容個性化等等。或是針對特殊的過程進行處理、增加檔案系統效率等等。
- 某些安全性相關軟體使用 File system filter driver 進行檔案讀寫的控制，作爲防止訊息洩漏等應用。
- 某些文件安全商家用來進行文件備份與恢復。

下一小節 FileSpy 即介紹 File system filter driver 的其中一種應用。

1.3.5 FileSpy

FileSpy 是一個 File system filter driver，由 IFSDDK 提供的一個範例程式。它可以監視系統上正在執行中的所有本地與網路磁碟上的 IRP 與 Fast I/O，並讓使用者指定監視哪些分割區 (Volume)。它分爲兩個部分：User-mode 與 Kernel-mode。Kernel-mode 部分監視並記錄磁碟存取活動，並且將此資訊格式化後，傳送給 User-mode 端的部分。User-mode 部分將這些資訊輸出至螢幕或是記錄至檔案。

FileSpy kernel driver與EFS driver皆屬於File system filter driver，連接在File system driver之上，如Fig. 8所示。在本論文中，SecCap運用了

FileSpy，爲全域檔案存取做監聽並取出其檔案名稱以及其他相關資訊。

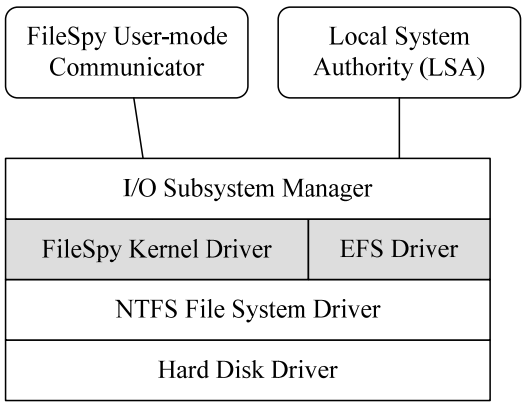

Fig. 8　FileSpy kernel driver 與 EFS driver

1.4 論文架構

本論文一共分爲五個章節。

第一章敘述本論文之背景、研究動機、研究目標，介紹本論文與作品的來由與初步的想法，並說明此作品之貢獻與引用的相關模組簡述。另外介紹本論文所應用到的相關知識與背景，以及相似的應用方法。包含了 SocksCap 軟體、智慧卡、公開金鑰基礎建設 (Public key infrastructure, PKI) 與自然人憑証、EFS (Encrypting file system) 、檔案系統過濾驅動程式 (File system filter driver) 、以及 FileSpy。

第二章爲相似的軟體系統做簡要介紹，包含了 PGPDisk、TrueCrypt、Decart、與 Maya 等軟體，並且對其功能做比較。

第三章詳述此作品之功能需求與分析，並對我們希望達成的目標做圖形化的描述做爲操作流程之參考，以及針對系統各部分的設計與實作細節做介紹。

第四章針對此系統之實驗設計與其環境敘述、與各項功能與特色的測試，

並且做相關的安全性分析、加密內容結果分析、與系統的效率等等。

第五章爲本論文之總結，以及其未來展望。

第二章　相關研究

本章介紹目前現有與SecCap類似、相關功能的系統。基本上，SocksCap (2.1) 是自動化地將網路應用軟體內之所有網路指令做Socks封裝，以符合 Socks protocol。本論文參考此系統與其介面做爲主要想法。PGPDisk、TrueCrypt、Dekart、與Maya (2.2) 爲目地相似於SecCap之軟體系統，我們在2.3做相關功能比較。

2.1 SocksCap

SocksCap是由NEC公司所開發的一個非商用免費軟體，目前它已經停止繼續研發，並且有新的版本取代：FreeCap[14]，在GNU GPL (General public license，通用公共許可証) 下發佈。

SocksCap可以讓某些沒有提供Socks[13]等防火牆設置之客户端軟體，使其具有Socks功能，並連接網際網路。就好像一個帽子一樣，可以蓋住客户端軟體，補捉他們的所有網路連接指令與內容，取代爲具有Socks協定的版本，並且轉向至Socks伺服器。

它的功能簡單、易於使用，並且幾乎可以支援所有 Windows-based TCP/IP 應用軟體，包含 Internet Explorer、MSN Messenger 等等。其基本的運作原理，是針對所有處理程序，使用全域 DLL 注入 (Global DLL injection)，將 Winsock API 進行掛勾程序 (Hook process)，來達成它的目地。

SecCap 與 SocksCap 的目地有異取同工之相似之處：SocksCap 將沒有 Socks 版本的網路指令，取代成具有 Socks 版本的指令；而 SecCap 將未經加密的檔案存取指令，邏輯上取代成具有加解密版本的指令。盡管其實作過程與運作原理不同，但其目地是相似的。

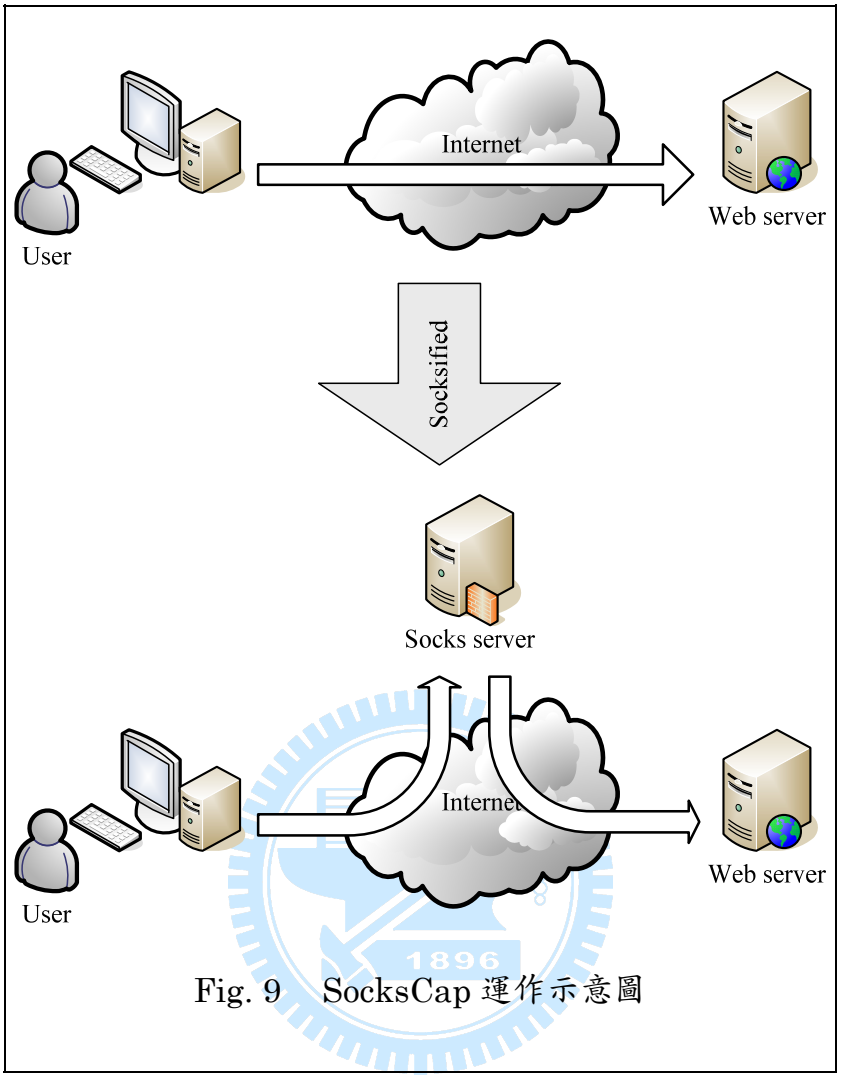

Fig. 9　SocksCap 運作示意圖

2.2　PGPDisk、TrueCrypt、與Maya

PGPDisk 是一個即時加解密(On-the-fly encryption, OTFE)的虛擬磁碟(Virtual disk)系統，並將一加密後的虛擬磁碟存入一個檔案內。其較舊的版本爲免費下載，但已不繼續維護。現在 PGP virtual disk 爲 PGP desktop 系列產品的一部分，並在 Windows 2000/XP/Vista 與 Mac OS X 上運行。新版本的 PGPDisk 支援 RSA 公開金鑰加解密，並且計劃適用於其它常見的作業系統如 Linux。

TrueCrypt 的功能與 PGPDisk 雷同，它不僅可將一加密後的虛擬磁碟存入一個檔案，它亦是一個檔案系統(File system)，可直接由作業系統掛載，並

且支持整個磁碟或分割區(Partition)的加密。TrueCrypt 爲開放原始碼的免費軟體系統，並適用於許多常見的作業系統，如 Windows、Mac OS X、以及 Linux。TrueCrypt 使用 AES、Twofish 等對稱式加解密做爲主要加密演算法，以及 SHA-512、Whirlpool 做爲雜湊函式，不支援非對稱式加解密。

Maya 檔案加密工具提供離線的檔案加解密，它可加密檔案或資料夾，並提供人性化的介面、易於上手。它使用一自訂的通關密語 (Passphrase) 之雜湊值對檔案內容進行 3DES 對稱式加解密，是個簡單方便的工具。目前 Maya 提供免費版本與付費版本，並且僅支援 Windows 作業系統。

2.3 功能比較

Fig. 10列舉出市面上某些與SecCap類似的系統異同之處。其中OTFE爲即時 (Real-time) 透明 (Transparent) 加解密。若欲針對同一磁區的各別檔案做加解密，卻又需要即時透明處理，此時必需使用檔案系統層級加密 (File system level encryption)，如SecCap所使用的EFS。

	SecCap	PGPDisk	TrueCrypt	Dekart	Maya
OTFE (On-the-fly encryption)	v	v	v	v	
Whole disk/partition		v	v	v	
File encryption	v				v
Public key encryption	v	v		v	
Smartcard authentication	v			v	
User defined passphrase	v	v	v		v
Free/open source			v		

Fig. 10 類似系統功能比較

一般的檔案加解密軟體，通常是針對整個磁碟或磁碟分割區做加密保護，亦可支援上線存取。某些軟體提供針對個別的檔案加密，但通常必須是離線作業，無法上線存取，也就是說此檔案無法在應用程式存取的同時，對它即時加

解密。若使用者希望能對個別檔案加密，同時又享有即時透明的上線存取，此時有兩種選擇：使用具檔案層級即時加解密功能的檔案系統，或使用具檔案加密功能的 File system filter driver。目前上述軟體並不多見，且 EFS 僅支援 NTFS 檔案系統。

不論是磁碟加密軟體，或離線檔案加密軟體等等，大部分的認証與加解密金鑰皆僅有使用使用者定義的密碼 (通關密語) 。部分智慧卡公司提供智慧卡認証與加解密軟體，一般也屬於離線檔案加密軟體。SecCap 結合了上述各項優點，提供即時透明的檔案加解密、智慧卡認証與使用者定義之通關密語認証等，是目前類似功能軟體裡功能較爲特別的。

第三章　系統設計與實作

3.1　功能需求與分析

一般使用者對於使用電腦的習性，通常是希望軟體系統能夠讓使用者介面直覺化、簡單化、容易學習與使用，亦期望能達到一定程度的文件安全。而對於受保護的文件，合法使用者將能夠輕易地直接存取、解密、攜出。本系統：SecCap 當中，使用者介面的直覺簡單化將會是一重要考量，配合智慧卡認証、Windows 使用者金鑰管理、與 EFS (Encrypting File System) ，加以整合應用，來達到使用者的需求。

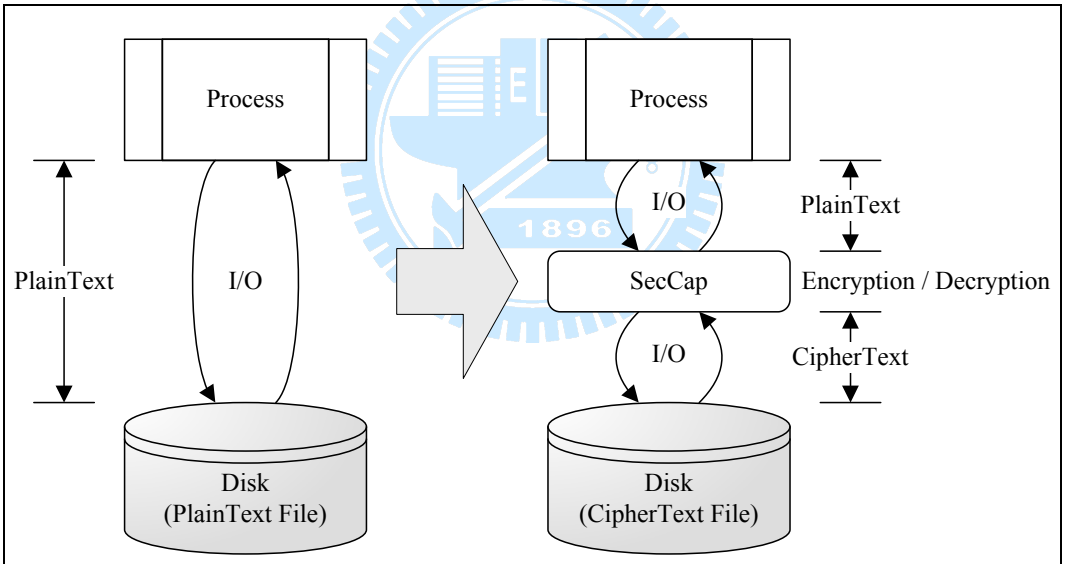

Fig. 11　SecCap 的角色

邏輯上，本系統SecCap所扮演的角色，可由Fig. 11表示。一般情況，使用者處理程序所儲存的文件，將會即時加密存放於硬碟；同理，使用者處理程序所讀取的文件，將會從硬碟即時解密取出。某些非一般情況，例如使用者剛新增的文件，或初次存取的文件，在硬碟中可能保持未加密，必須等待此次行程結束、由使用者確認此文件需要加密，此後便皆以密文方式儲存。

依照目前的設計，SecCap 提供以下的功能與相關特色：

- 直覺的圖形使用者介面
- 智慧卡驗証使用者身份、登入與登出
- 個別的使用者之應用程式記錄與其設定檔
- 由使用者決定自動與手動加密之檔案類型
- 個別的使用者工作環境
- 以處理程序做爲監視磁碟存取之單位
- 加密文件可於不同電腦上存取，且保有私密性
- 建構於 Windows 使用者與金鑰管理上

3.2 操作流程

本節根據1.2與3.1對於SecCap使用上的需求，將簡單敘述SecCap之一般操作流程。

首先，我們執行SecCap應用程式，如Fig. 12。

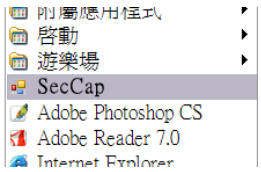

Fig. 12 SecCap 操作流程 1

待SecCap主視窗出現後，便可進行登入動作，如Fig. 13。

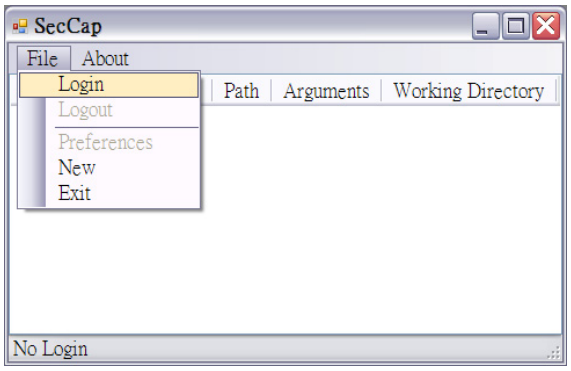

Fig. 13 SecCap 操作流程 2

此時若未正確插入「自然人憑証」智慧卡，將出現錯誤訊息，並且登入失敗，如Fig. 14。

Fig. 14 SecCap 操作流程 3

因此我們必需將「自然人憑証」正確插入智慧卡讀卡機當中，如Fig. 15。

Fig. 15 SecCap 操作流程 4

首先，SecCap使用「內政部憑証管理中心」之憑証 (MOICA.CER) 作爲CA，對卡片內之憑証進行驗証程序。待驗証通過，確認此憑証確實由「內政部憑証管理中心」所核發，接下來便進行PIN碼驗証動作。使用者輸入正確的PIN

碼後，便可登入SecCap，如Fig. 16。

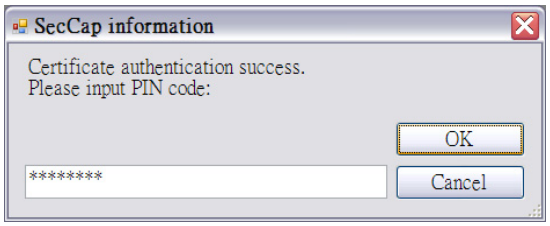

Fig. 16 SecCap 操作流程 5

於本機電腦上，一個新的使用者將會出現初始化的偏好設定。使用者可以設定通關密語 (密碼，由使用者自行定義，不同於Windows使用者密碼，不同於卡片PIN碼) ，以及預設加密的檔案類型。這些偏好設定日後亦可以再行更改，如Fig. 17。

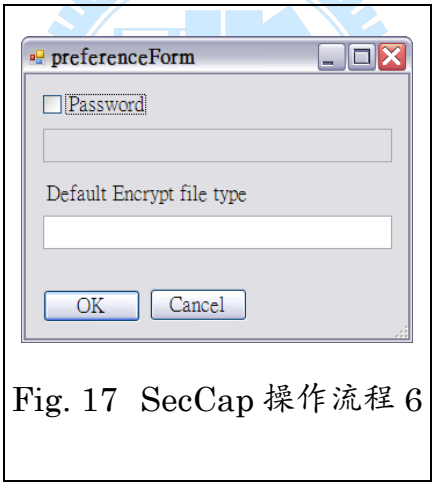

Fig. 17 SecCap 操作流程 6

於SecCap主視窗的應用程式清單內，我們可以經由滑鼠拖曳或手動新增的方式，加入與執行應用程式。此例中，我們透過SecCap來執行Microsoft Word 2003，如Fig. 18。

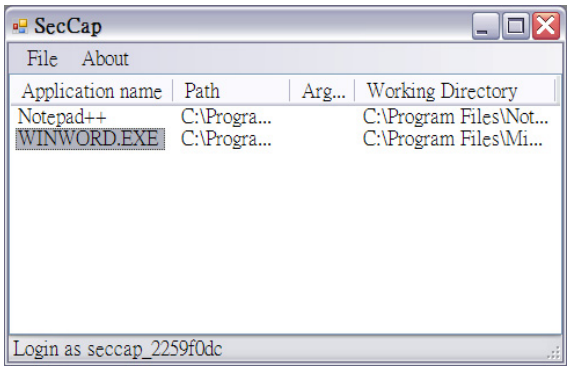

Fig. 18 SecCap 操作流程 7

由SecCap執行的Word視窗內，我們建立一份新的Word文件，並進行編輯，如Fig. 19。

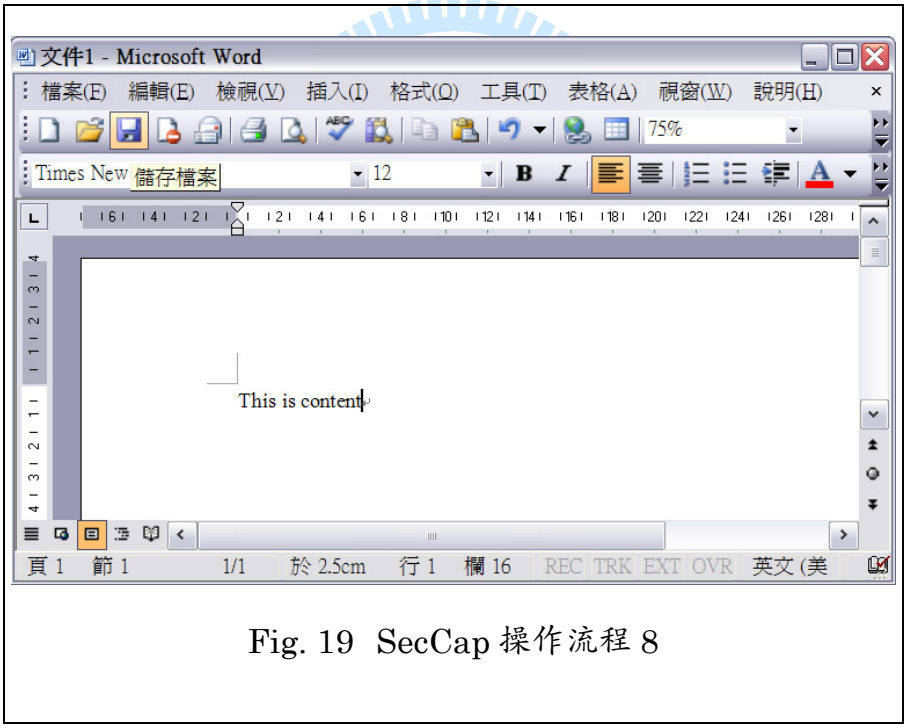

Fig. 19 SecCap 操作流程 8

編輯完畢後，再行存檔，如Fig. 20。

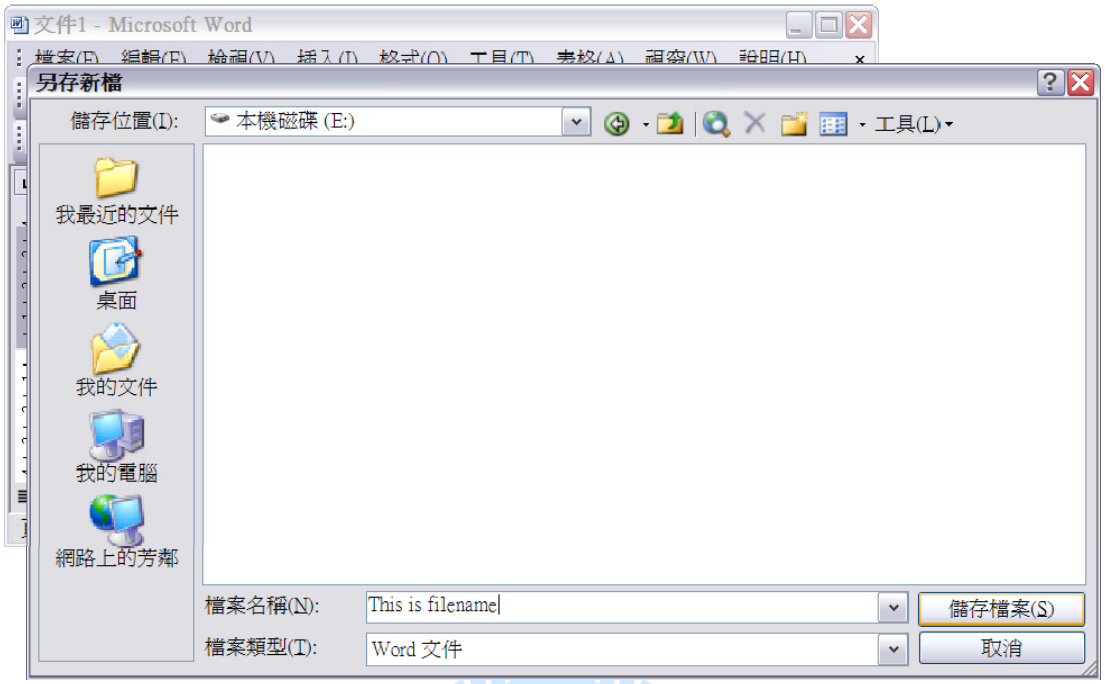

Fig. 20 SecCap 操作流程 9

將Word關閉後，SecCap將會彈出一列表視窗。此視窗列舉出Word所有曾寫入、更名等的檔案清單。其中灰色字體表示此檔案無法被加密，綠色字體表示此檔案爲已加密，黑色字表示此檔案可被加密。我們將欲加密的檔案勾選起來、欲解密的檔案取消勾選之後，點選「Encrypt / Decrypt」，便可將檔案直接進行加解密，如Fig. 21。

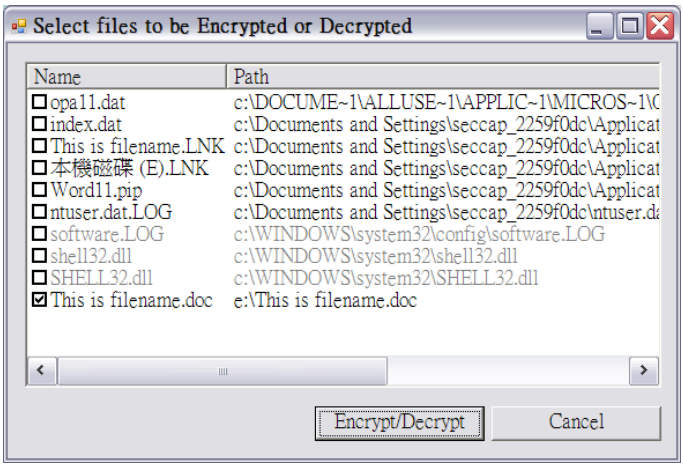

Fig. 21 SecCap 操作流程 10

檔案加密完畢後，我們可以開啓「檔案總管」瀏覽我們所儲存的已加密檔案，會發現其呈現綠色，表示此檔已經由EFS加密，如Fig. 22。

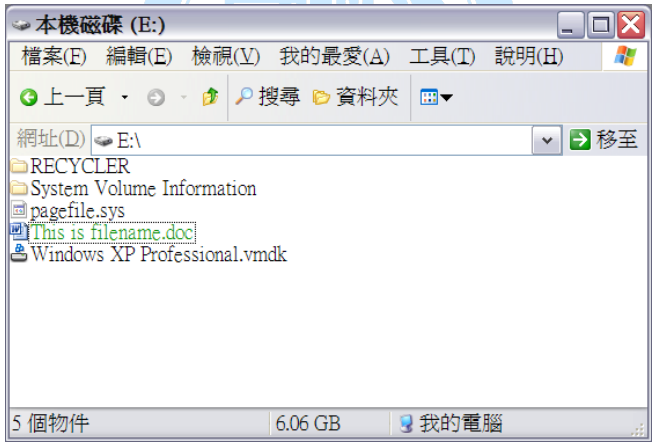

Fig. 22 SecCap 操作流程 11

在檔案的進階內容中，我們可以查詢其加密所使用的憑証資訊，如Fig. 23。

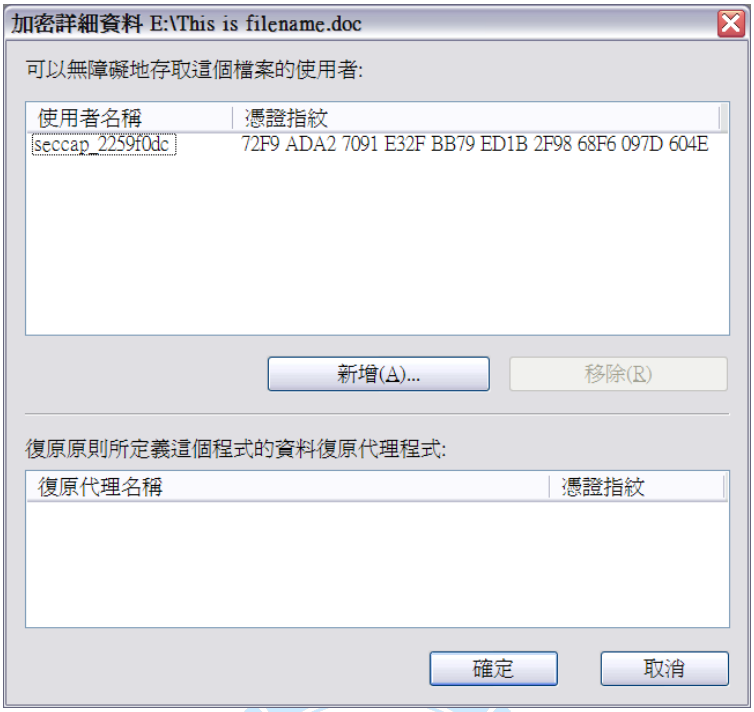

Fig. 23 SecCap 操作流程 12

以上，爲SecCap操作的基本流程。更詳細的使用與實驗，將於第四章繼續做深入討論。

3.3 系統架構概述

Fig. 24爲SecCap系統架構概念圖，基本上可以將SecCap區分爲 4 塊模組：GUI shell、Smartcard authenticator、File encryption manager、FileSpy user-mode communicator。以下爲各區塊做簡要說明，各細節於3.4做更詳細的敘述。

GUI shell 主要爲處理使用者所觸發的事件，以及與其他模組做溝通，並且初始化 SecCap 之環境等等，做爲使用者介面。

Smartcard authenticator 執行智慧卡相關的身份認証。當使用者登入 SecCap 時，此部分將被啓動。目前此部分係使用「內政部憑証管理中心」所

核發的「自然人憑証」做爲身份憑証，未來亦可與其他 PKI 進行合作。

File encryption manager 負責將檔案加解密的動作。當它接收到來自 FileSpy user-mode communicator 的檔案資訊，並且通過加密條件測試之後，便使用當前 SecCap 使用者之身份，利用 EFS 將此檔案加密。

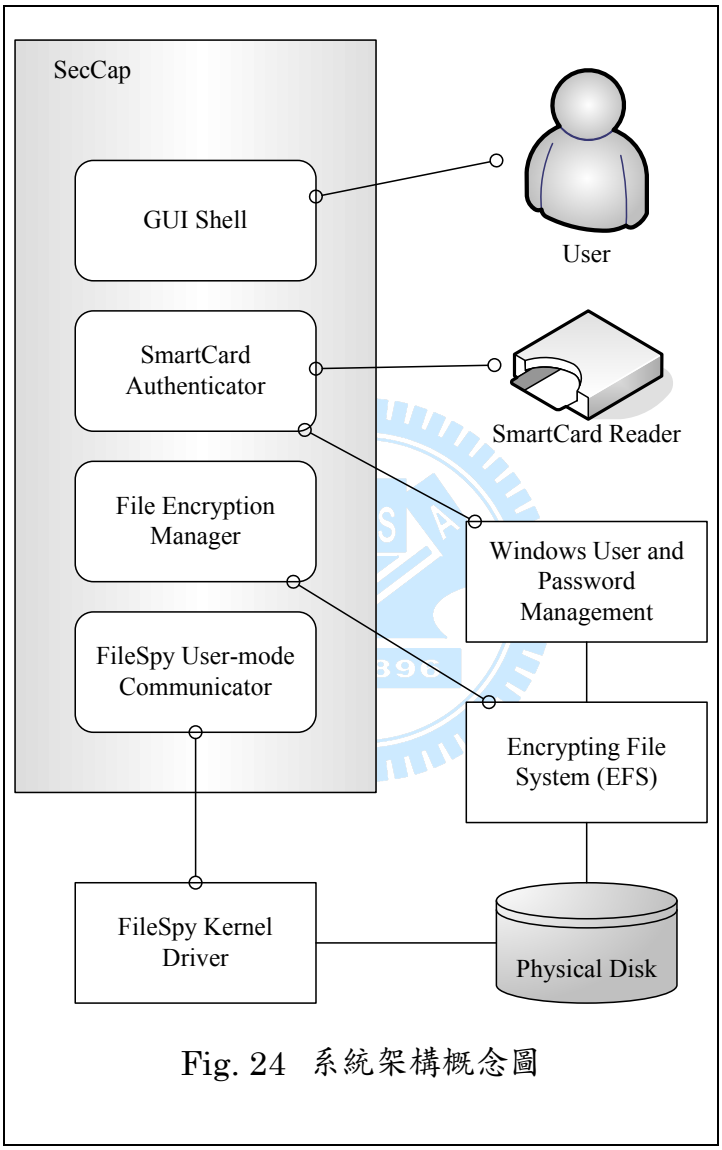

Fig. 24 系統架構概念圖

FileSpy user-mode communicator 負責接收來自 FileSpy kernel-mode driver 所送出的檔案資訊，並且經篩選過濾後，傳送給 File encryption manager。同時，它也接收來自 GUI shell 所送出的 Process ID 資訊，將其加入監視列表當中。

3.4 模組設計細節

本節詳述SecCap各模組之設計與實作細節。Fig. 25為本系統更詳細的架構圖，其中，灰底的區塊為SecCap實作之內容，粗框的區塊屬於Windows kernel的部分，而虛線箭頭為模組間實際通訊之方式。

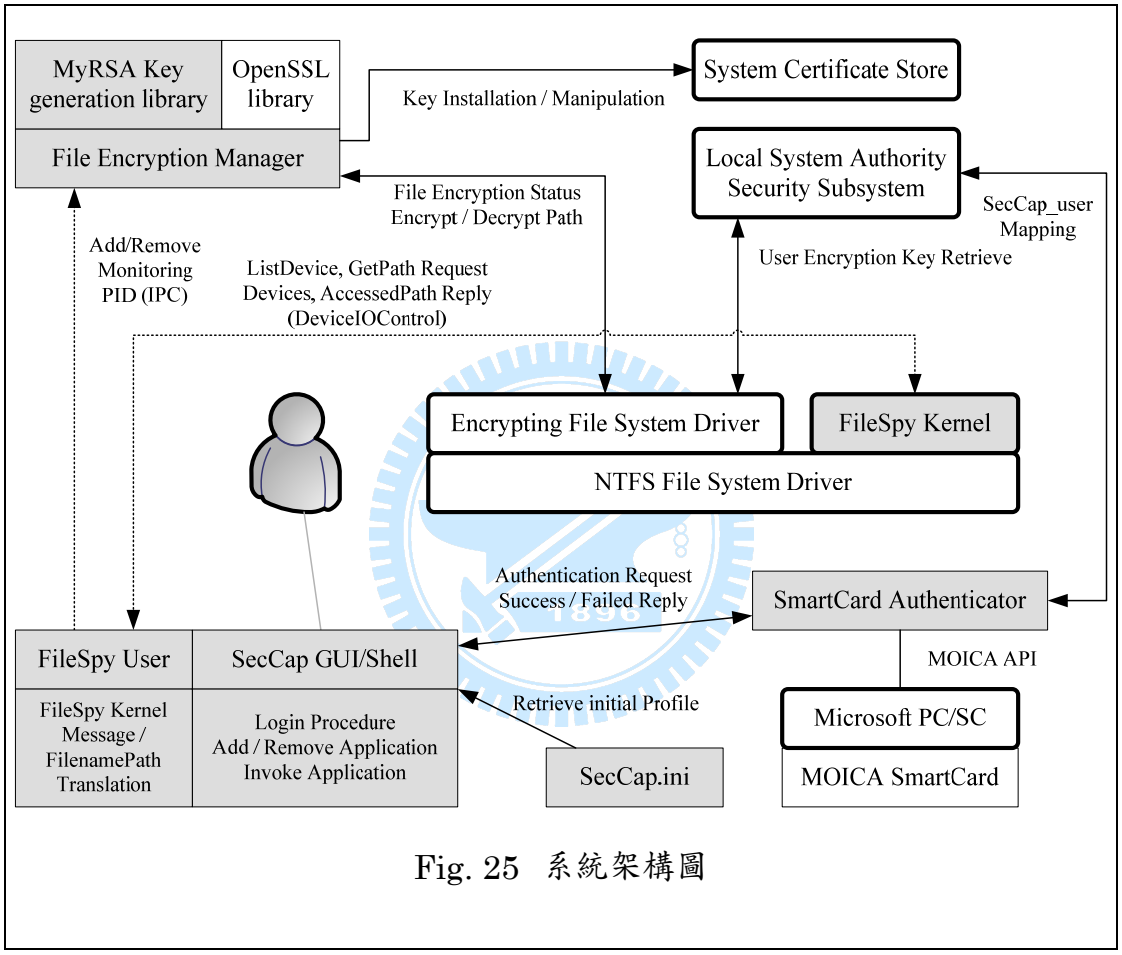

Fig. 25 系統架構圖

3.4.1 GUI shell

Windows 作業系統，表現傑出的其中一項即為其優秀的圖形化使用介面，讓使用者更容易親近。Event-driven 成為 GUI 程式的一大特點，程式平時處於 Message loop 當中等待訊息、觸發事件。

SecCap 也不例外，這部份主要為處理使用者所觸發的事件，以及與其他模組做溝通。它的工作內容如下：

- 初始化工作視窗環境。
- 處理新增刪除應用程式、拖曳捷徑、維護與修改應用程式清單、排序。
- 使用者登出、登入介面，與其偏好設定。
- 以 SecCap 身份執行清單中的應用程式。
- 使用者登入後，喚醒 FileSpy user-mode communicator 與 File encryption manager。
- 當應用程式被執行與結束時，將其 PID (Process ID) 等資訊送至 FileSpy user-mode communicator 與 File encryption manager。

根據目前定義，使用者之偏好設定的結構如Fig. 26。userCode爲自然人憑証內主體之雜湊值。userName爲「seccap_xxxxxxxx」，其中x部分即爲userCode之 16 進制表示法，此欄位也做爲Windows使用者帳號名稱，而其登入之密碼爲{userCode, Secret_value}之雜湊值。其餘 3 個欄位便爲使用者偏好設定視窗之內容，如Fig. 17。

```
class UserProfile
{
     ulong     userCode;
     String^   userName;
     bool      passwordEnable;
     String^   password;
     String^   encExt;
};
```

Fig. 26 UserProfile 結構

SecCap 針對每個使用者皆獨立自行的設定檔，檔名爲「seccap_xxxxxxxx.ini」，其中x部分即爲userCode。其內容範例如Fig. 27，「User Settings」爲使用者偏好，而其中Password是使用者自定之通關密語經由secret_value加密後儲存。「Applications」即爲應用程式清單，內容可經過手動 (File→New) 或是將*.exe或*.lnk檔案拖曳至SecCap主視窗內新增，

亦可在清單項目上點選右鍵做修改或刪除，如Fig. 28。清單列表將其轉爲純文字後儲存。未來若有安全考量，可以考慮直接將此設定檔加密後，再行儲存至磁碟上。

```
[User Settings]
Password Enable=True
Password=KbnGwcJACJMjBjHn
Default Encryption Extension=*.txt, *.doc

[Applications]
Application Name=遠端桌面連線
Path=C:\Program Files\Remote Desktop\mstsc.exe
Arguments=
Working Directory=C:\Program Files\Remote Desktop\

Application Name=WINWORD.EXE
Path=C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE
Arguments=
Working Directory=C:\Program Files\Microsoft Office\OFFICE11
```

Fig. 27 SecCap 使用者設定檔範例

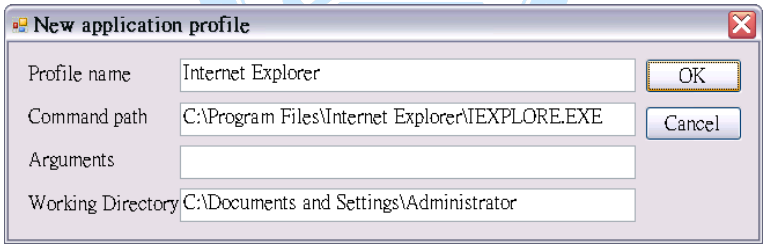

Fig. 28 新增應用程式視窗

當使用者點選一應用程式，欲執行之時，SecCap將行程資訊設定完畢後(包含使用者名稱、登入密碼、應用程式路徑與參數、工作目錄等)，便呼叫Process::Start執行它，並增加Exited事件處理函式，隨後將其PID送至FileSpy user-mode communicator，以便將此處理程序增加至監聽項目清單，而監聽項目清單之結構示意圖如Fig. 29。當SecCap收到Exited事件時，同樣地將離開的PID送至FileSpy user-mode communicator，以便將其從監聽項目清單中

移除。

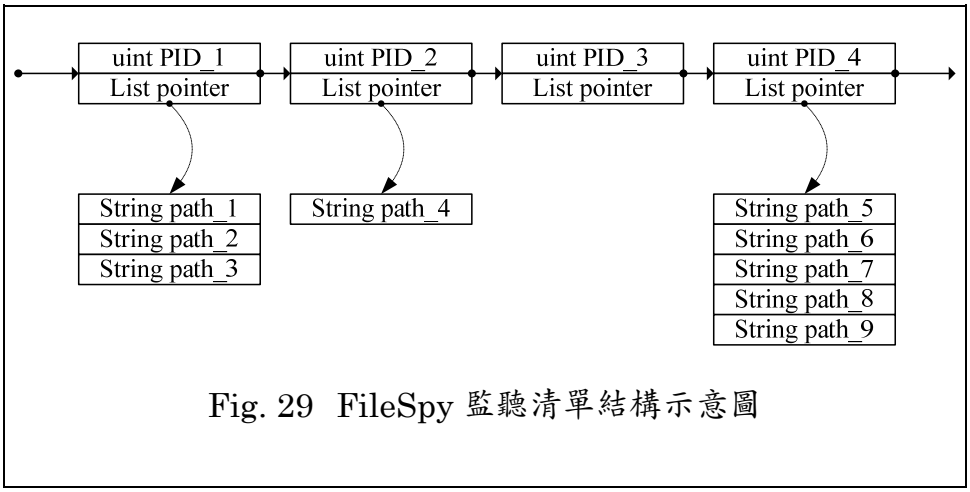

Fig. 29 FileSpy 監聽清單結構示意圖

3.4.2 Smartcard authenticator

爲了確認卡片合法持有者爲目前的使用者，此部分獨立出來，做爲使用者身份驗証之用。它於使用者進行 SecCap 登入時被啓動，其工作內容如下：

- 讀取「自然人憑証」內之憑証資料。
- 利用「內政部憑証管理中心」之憑証，對讀出的憑証進行驗証。
- 驗証卡片之 PIN 碼。

我們透過自然人憑証 API，將卡片之 X.509 格式憑証內容讀出。首先使用「內政部憑証管理中心」之憑証 (MOICA.cer) 做爲 CA，對此憑証進行驗証，驗明它確實是由此 CA 所核發。然後，彈出視窗，請使用者輸入卡片之 PIN 碼。最後將驗証成功與否的結果傳回給 SecCap，若成功，同時將憑証之主體名稱回傳給 SecCap 以做金鑰產生之用。

3.4.3 File encryption manager

此部分與其他模組的主要不同點爲：此部分由獨立的行程「FEncMgr.exe」做處理，且行程之 Windows 執行身份爲「SecCap_xxxxxxxx」，其中 x 部分即爲 userCode；而其他部分之處理程序名稱爲「SecCap.exe」，Windows 執行身份爲當前桌面之使用者。此部分的工作內容如下：

- 使用者私密金鑰、公開金鑰與憑証之產生。
- 使用者私密金鑰、公開金鑰與憑証之安裝。
- 接收來自 SecCap 的 PID (Process ID) 與存取之檔案名稱資訊。
- 列舉存取檔案之視窗。
- 執行存取檔案之加解密。

這邊有關私密金鑰、公開金鑰與憑証之產生有兩項訴求：我們希望同一個使用者，能夠在不同的電腦上使用相同的私密金鑰、公開金鑰；並且，使用者之「自然人憑証」卡片可能會毀損、更換，我們必需針對同一使用者、不同卡片，產生相同的私密金鑰。而現有的私密金鑰、公開金鑰產生模組，其產生方式需參考許多以外在的雜訊做爲亂數產生器 (Pseudo-random number generator, PRNG) 所提供之數值。因此我們使用修改過後的OpenSSL[3]程式庫：myOpenSSL，做爲金鑰產生的模組。下列是我們所有修改、提供之函數系列：

- myRSA_generate_key 系列
- myRAND_meth 系列
- myBN_rand 系列
- myBN_generate_prime_ex 系列

以上，我們將所有會因外在環境因素 (時間、設備等) 的而變動的敘述刪除，進而修改爲常數版本。針對 SecCap 而言，我們只需要某幾項資訊做爲產生亂數之種子參考，再根據這些亂數產生金鑰。因此 SecCap 需要直接使用到 myRAND_meth 與 myRSA_generate_key。而其所參考之函數 myBN_rand、myBN_generate_prime_ex 亦一拼做修改。

私密金鑰、公開金鑰與憑証之產生，與將其安裝至系統之流程如Fig. 30。其中，「myRSA private/public key generating」執行私密金鑰、公開金鑰之產生，「PKCS #10 request」[22]至「PKCS #12 encapsulated certificate」[23]

執行憑証之產生。「Add to system certificate store」至「Done」將憑証安裝至作業系統，並且將其設定爲SecCap使用者的EFS檔案加密金鑰。

我們使用智慧卡憑証中的「主體名稱」、使用者自訂的「通關密語」、及SecCap 內定的「秘密字串」，做爲亂數產生器的種子，也就是亂數產生的依據。如此一來，便可在不同電腦上產生相同的亂數序列，進而可產生相同的私密金鑰、公開金鑰。

金鑰產生完畢後，我們必需產生EFS憑証，才可將其安裝至作業系統。Windows目前支援X.509[7]之DER (Distinguished encoding rules) [8]格式憑証，並且此憑証之金鑰使用功能必需包含Microsoft EFS (1.3.6.1.4.1.311.10.3.4) [18]等等。我們使用OpenSSL的X.509 系列程式庫建立EFS憑証，並且自行簽名。Fig. 31爲我們產生的憑証示例。

取得私密金鑰與 EFS 憑証之後，透過 OpenSSL 的 PKCS12 系列程式庫，將它們封裝爲 PKCS#12 結構，再交由下一步做處理。

透過Win32 API，我們將此PKCS#12 結構資料（包含了私密金鑰與EFS憑証）安裝至Windows憑証存放區。我們可以使用主控台 (Microsoft management console, MMC) 瀏覽憑証存放區，查看已安裝的憑証，如Fig. 32。

憑証安裝完成後，將其設定爲 Windows 使用者之 EFS 加解密金鑰。如此使用者私密金鑰、公開金鑰與憑証之產生與安裝便完成。

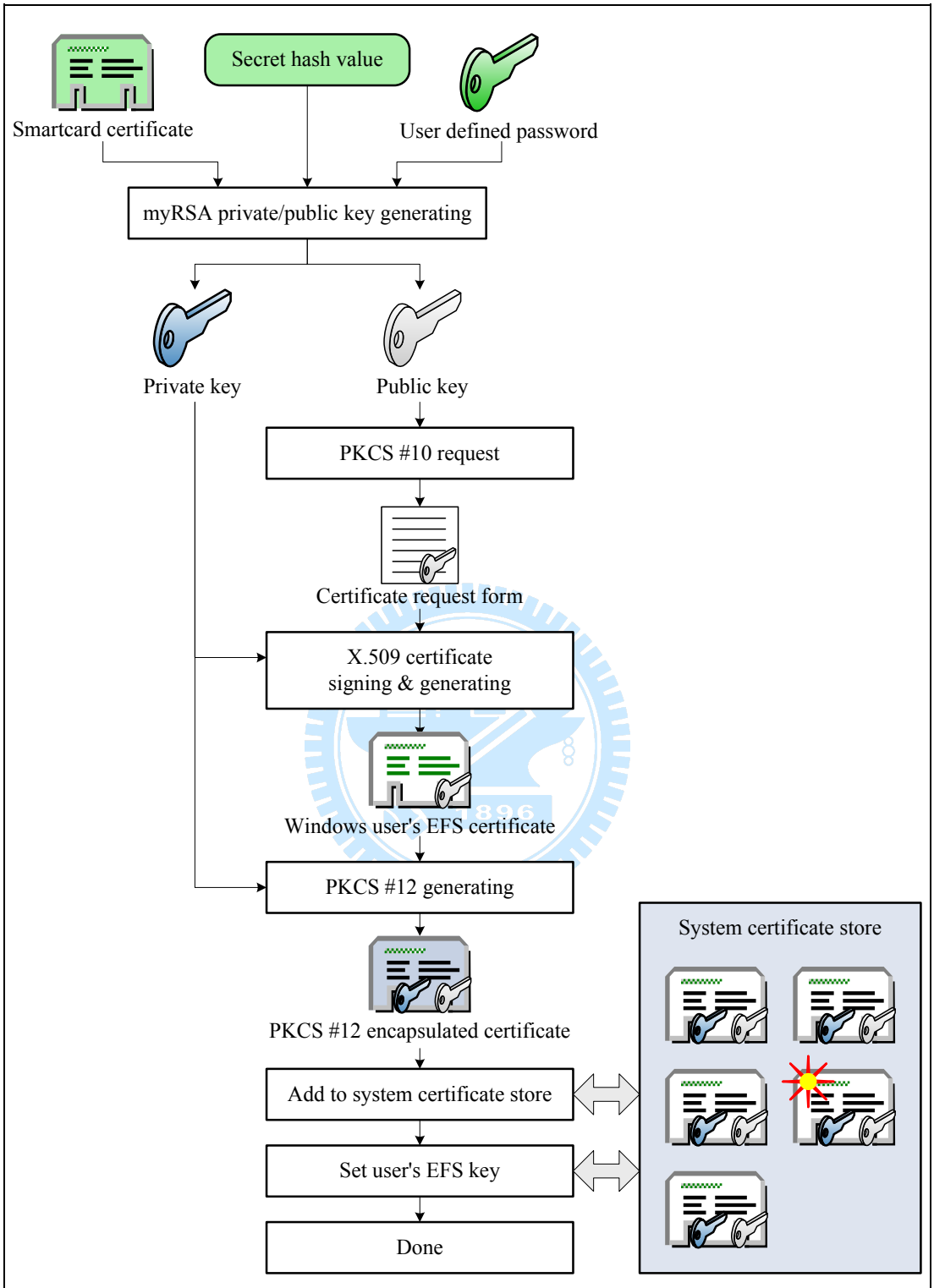

Fig. 30 私鑰、公鑰與憑証之產生與安裝流程

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      92:9f:e2:ce:a6:0e:c8:32
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: CN=seccap
    Validity
      Not Before: Jan 1 00:00:00 2000 GMT
      Not After : Jan 1 00:00:00 2020 GMT
    Subject: CN=seccap_xxxxxxxx
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:d0:c5:dd:2b:d0:12:8d:67:57:c6:a6:e2:ff:73:
          ......................
          88:2a:0b:04:ea:3b:ef:6b:37
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      X509v3 Extended Key Usage:
        Microsoft Encrypted File System (1.3.6.1.4.1.311.10.3.4)
  Signature Algorithm: sha1WithRSAEncryption
    c3:6d:4b:67:ab:27:b9:60:59:09:38:ea:e1:f9:35:b3:61:c6:
    ......................
    6e:13
```

Fig. 31 EFS 憑証

File encryption manager 會透過 IPC 接收來自 SecCap 之資訊：

- 由 SecCap 所執行之應用程式的 PID。
- 已結束之應用程式（當初由 SecCap 所執行）的 PID。
- 應用程式（當初由 SecCap 所執行）所存取檔案之路徑，與其 PID。

上述所有資訊將會記錄至存取清單結構，其示意如Fig. 29。若收到執行之應用程式的PID，則新增一項PID結構標頭至存取清單；若收到結束之應用程式的PID，則將其PID結構標頭從存取清單中移除。

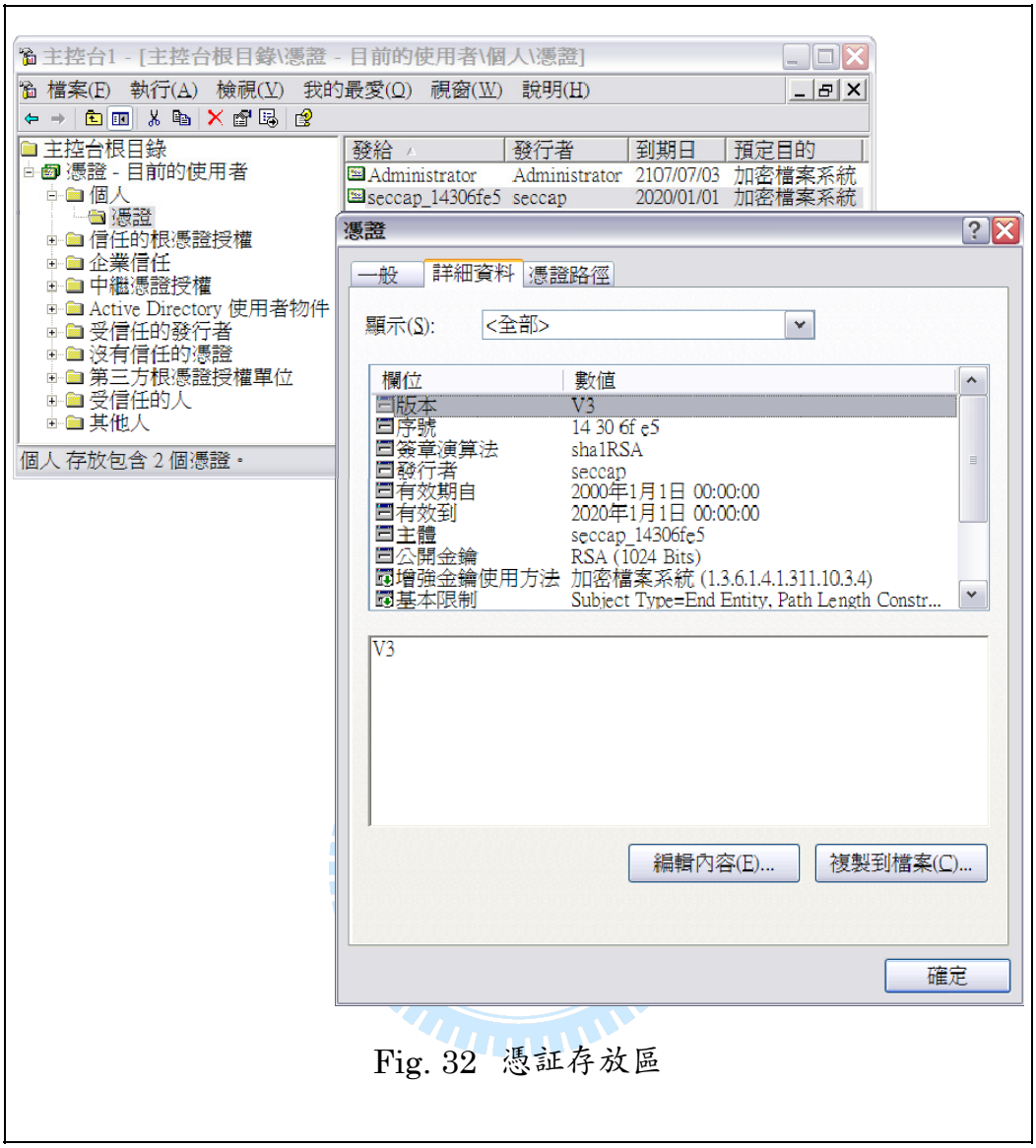

Fig. 32 憑証存放區

當應用程式（當初由SecCap所執行的）結束後，SecCap會彈出此行程所存取之檔案列表視窗，如Fig. 33。其中綠色字代表目前此檔案已加密；灰色字代表此檔案無法被加密（如系統檔、不存在或無權限的檔案等）；黑色字代表可加密。我們可將檔案勾選起來進行加密，或將其除去進行解密。所有的檔案加解密方式皆使用EFS之API。

當 SecCap 使用者進行登出時，此行程（即 File encryption manager）便結束。待下一次登入時，此行程再重新被啓動。

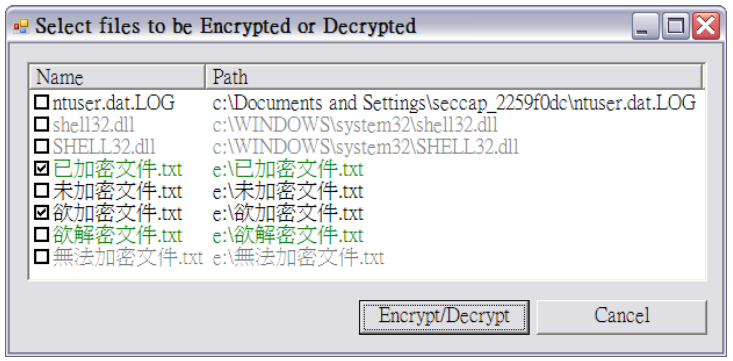

Fig. 33 使用者勾選加解密視窗

3.4.4 FileSpy user-mode communicator

此部分的主要工作內容如下：

- 透過系統服務管理員，啓動 FileSpy kernel-mode driver。
- 停止 FileSpy kernel-mode driver。
- 請求 FileSpy kernel-mode driver 綁定所有的 NTFS 分割區。
- 一般檔案路徑與 IRP (I/O request packet) 檔案路徑之轉換。
- 接收來自 FileSpy kernel-mode driver 的檔案名稱等資訊。
- 接收來自 SecCap 的被執行和已結束之 PID 資訊。
- 發送應用程式所存取之檔案路徑至 File encryption manager。

首先，我們將 FileSpy kernel-mode driver 啓動後，透過 DeviceIOControl 函式，取得目前系統上所有分割區 (Volume) 列表，並選定所有 NTFS 分割區，同樣透過 DeviceIOControl 要求 FileSpy kernel-mode driver 綁定這些分割區。此時，FileSpy kernel-mode driver 便已不斷地將全域存取之檔案資訊發送至 FileSpy user-mode communicator，包含所有的 IRP、FastIO 等，而我們有興趣的僅有 IRP_MJ_WRITE (檔案寫入) 與 IRP_MJ_SET_INFORMATION (更改檔名等) 的檔名與其存取行程之資訊。同時，我們接收來自 SecCap 的 PID 資訊，將被執行和已離開之 PID 加入或

刪除自監聽清單。

若一 IRP 資訊上述之 IRP 條件，並且其 PID 亦存在於監聽清單內，則我們將此 IRP 之檔案路徑（如：\Device\HarddiskVolume2）轉換爲一般檔案路徑（如：E:\tmp.txt）後，將其連同 PID 透過 IPC 機制，一起發送給 File encryption manager。

當 SecCap 使用者進行登出時，此部分便會要求關閉 FileSpy kernel-mode driver 服務，並且釋放與其相關的資源。

第四章　實驗設計與結果分析

SecCap 的主要目地在於，提供使用者方便的環境與智慧卡驗証，來達到個人文件的安全。因此本論文的主軸，將著重在此實驗的結果與分析。本章將進行 SecCap 之各項操作測試，並對於實驗結果做相關的討論與分析，以下各節爲各個實驗細節做詳述。

4.1　實驗設計

此節當中，我們介紹實驗相關的環境與測試資料 (4.1.1) ，以及欲實驗的各項功能與特色 (4.1.2) 。除了進行此兩小節內的各項實驗，我們亦對加解密之流程加以敘述與其安全性分析，以及對已加密的文件內容進行分析，最後與其他類似軟體之功能與效率比較。

4.1.1　實驗環境與測試資料

本實驗使用數台電腦，分別由 SecCap 執行數個不同的應用程序，並使用不同使用者的自然人憑証智慧卡做交叉測試，同時嘗試加解密不同大小之檔案以及其它雜項，以利分析比較且達到實際運用的要求。

	CPU	RAM	Operating System
PC1	AMD Athlon 3000+ 1.80GHz	1472M	Windows XP Pro SP2
PC2	AMD Athlon 1800+ 1.54GHz	480M	Windows XP Pro SP2

Fig. 34　實驗設備

本實驗使用數台電腦進行測試，實驗設備與作業系統等等環境列表於Fig. 34，並分別由SecCap執行數個不同的應用程序。每台電腦皆使用數張不同使用者之自然人憑証智慧卡進行測試，使用者列表於Fig. 35。各次測試皆執行Fig.

36之應用程序。

	姓名	卡片序號
USER1	陳癸夫	GP00000001343063
USER2	張志誠	GP00000001342839

Fig. 35 使用者與其自然人憑証序號

	應用程序名稱及版本
APP1	Microsoft Notepad 5.1
APP2	Microsoft Office Word 2003
APP3	Microsoft Office Excel 2003
APP4	Microsoft Office PowerPoint 2003
APP5	Notepad++ 4.9.2
APP6	Adobe Acrobat 8 Professional

Fig. 36 測試應用程序

4.1.2 各項功能與特色

經過上一小節的交叉測試應用程式與自然人憑証，我們接下來測試SecCap之其它功能與分析，主要如下：

- 應用程序清單之記錄
- 直覺的拖曳方式，新增應用程序設定檔
- 使用者之登入、智慧卡驗証、登出機制
- 自動、客製各類型的檔案之加解密
- 加密文件可於不同電腦上存取，且保有私密性

以上的各個項目主要爲使用者介面之操作細項。另外爲了測試SecCap系

統之耐用性，除了測試以上應用程式與自然人憑証、使用者介面操作之外，我們也進行以下實驗：

- 加解密 1 KB、1 MB、100 MB 之文件
- 兩個應用程式同時讀取相同的文件
- 針對同一份已加密文件，進行重複增修存取已加密文件數個小時以上，並且插入其它已加密或未加密文件

下一節將實際操作上述所有測試做爲展示。

4.2 實驗結果

根據上一節的實驗設計內容，我們將於實際操作每一個測試項目(4.2.1)，與對金鑰產生、加解密流程加以敘述以及其安全性分析 (4.2.2)，並且將加密內容相關資訊進行分析 (4.2.3)，最後針對SecCap做效率比較(4.2.4)。

4.2.1 實例操作

此小節我們針對4.1.2所列舉的測試項目做一一呈現。

- 應用程序清單之記錄

我們分別使用USER1 與USER2 之自然人憑証登入SecCap，並新增應用程式至清單。SecCap將會各別儲存其資訊至檔案「seccap_xxxxxxxx.ini」，其中x部分爲userCode，它係由自然人憑証內之主體名稱所雜湊計算出。此例中USER1 之userCode爲「2259F0DC」，USER2 之userCode爲「14306FE5」，其內容如Fig. 37、Fig. 38、。這些記錄檔案，將會在使用者下次登入時，自動讀入SecCap。

```
[User Settings]
Password Enable=False
Password=
Default Encryption Extention=*.txt, *.doc

[Applications]
Application Name=WINWORD.EXE
Path=C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE
Arguments=
Working Directory=C:\Program Files\Microsoft Office\OFFICE11

Application Name=小畫家
Path=C:\WINDOWS\system32\mspaint.exe
Arguments=
Working Directory=C:\WINDOWS\system32

Application Name=記事本
Path=C:\WINDOWS\system32\notepad.exe
Arguments=
Working Directory=C:\Documents and Settings\Administrator
```

Fig. 37 seccap_2259f0dc.ini 之內容

```
[User Settings]
Password Enable=False
Password=
Default Encryption Extention=*.txt

[Applications]
Application Name=Notepad++
Path=C:\Program Files\Notepad++\notepad++.exe
Arguments=
Working Directory=C:\Program Files\Notepad++
```

Fig. 38 seccap_14306fe5.ini 之內容

- 直覺的拖曳方式，新增應用程序設定檔

我們可以直接使用滑鼠拖曳方式來新增應用程式 (Fig. 39) ，或是手動透過功能表「File」→「New」來新增。當然，我們僅能增加執行檔類型 (EXE) 或捷徑類型 (LNK) 指至執行檔，如Fig. 40。每當我們新增時，便會彈出視窗 (Fig. 28) ，使用者可在此對其相關欄位做編輯後確認。

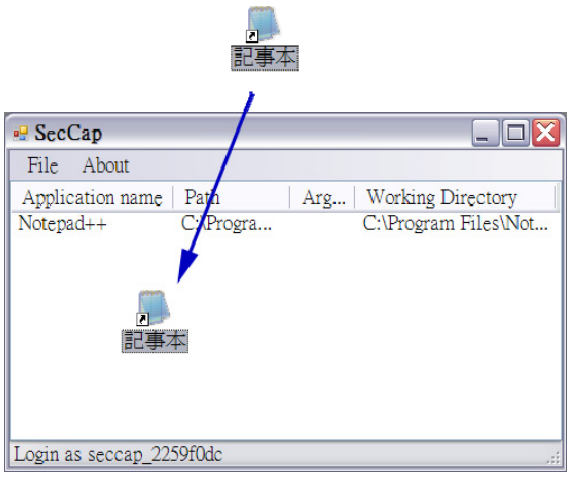

Fig. 39 滑鼠拖曳新增應用程式

Fig. 40 增加應用程式之錯誤訊息

- 使用者之登入、智慧卡驗証、登出機制

使用者必需先使用自然人憑証進行登入並輸入卡片PIN碼，若登入失敗，將會顯示錯誤訊息及代碼，如Fig. 14。使用者登出時，即會將所有資源（包含File encryption manager、FileSpy user-mode communicator等）與使用者資訊釋放。我們進行USER1 與USER2 重複登出、登入 10 次，所有操作皆保持正常。

- 自動、客製各類型的檔案之加解密

我們可以讓使用者自訂SecCap預設加密的檔案類型，亦可讓使用者手動勾選需加密的檔案。如Fig. 17，我們可在「Default encrypt file type」欄位填

上「*.doc, *.txt」，此後SecCap執行之應用程式（如：Word）所存取的檔案，只要其檔案名稱符合上述條件，皆會被自動化地被加密，便不需要使用者手動勾選它，如Fig. 41。

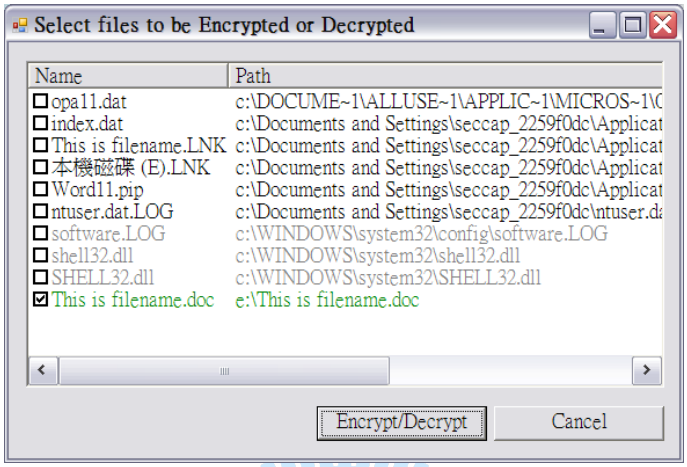

Fig. 41 預設加密的檔案類型

- 加密 1 KB、1 MB、100 MB 之文件

此項目我們用以測試加密上述各大小檔案之EFS加密所花費的時間，其結果顯示於4.2.4小節當中。

我們使用作業系統提供之函式「clock_t clock (void)」作爲測量時間的工具，它會回傳目前行程的 CPU 時間。即使此函式不是切確的精準，如磁碟存取時間在作業系統之多工環境下爲一大變數，但它已排除如 CPU 排程、其它行程與環境等影響，因此使用此函式之測量結果仍具有一定的參考價值，並且，我們使用多次存取測量，做爲緩和變數的方式。

- 兩個應用程式同時讀取相同的已加密文件

我們使用 SecCap 執行之「記事本」與「Notepad++」應用程式對已加密與未加密之檔案「Test_document_1K.txt」同時進行存取。此部分實際測試項目如下：

– 以「記事本」與「Notepad++」共同存取已加密文件

– 以「記事本」與「Notepad++」共同存取未加密文件，並且將「記事本」關閉後對其進行加密，而「Notepad++」繼續編輯此文件

某些應用程式在開啓檔案時，會對檔案做「寫入保護」動作（如 Word），其它程式若開啓此檔案時，便只能以唯讀方式開啓。此處有關唯讀的問題，與 SecCap 並無直接相關，因爲 SecCap 只在應用程式進行寫入檔案的時候進行加密轉存，所以我們並不對此情況做特別討論，而根據實驗，其特性也相同於未加密的情況。

上述兩項實驗之結果，其表現狀況皆與存取一般未加密檔案的情形相同，其不同之處僅在於實際儲存在磁碟上的檔案爲密文文件。此爲 EFS 實現的透明且即時加解密之功效。

- 加密文件可於不同電腦上存取，且保有私密性

我們直接將已加密文件存入隨身碟，將其複製到另一台電腦。其電腦上同樣顯示爲密文，並且僅有合法之 SecCap 使用者才可以將其內容正確讀取出，其餘存取模式如同原先的本機電腦。

- 針對同一份已加密文件，進行重複增修存取已加密文件數個小時以上，並且插入其它已加密或未加密文件

爲了測試SecCap之耐用性，我們徹底地實驗此系統之相關應用。我們使用Word對本論文之複本進行一般操作。我們的隨機操作包含了增修內文、插入已加密及未加密圖片檔、插入已加密及未加密Word檔、插入已加密及未加密物件檔案、不定時儲存檔案等等，如Fig. 42。

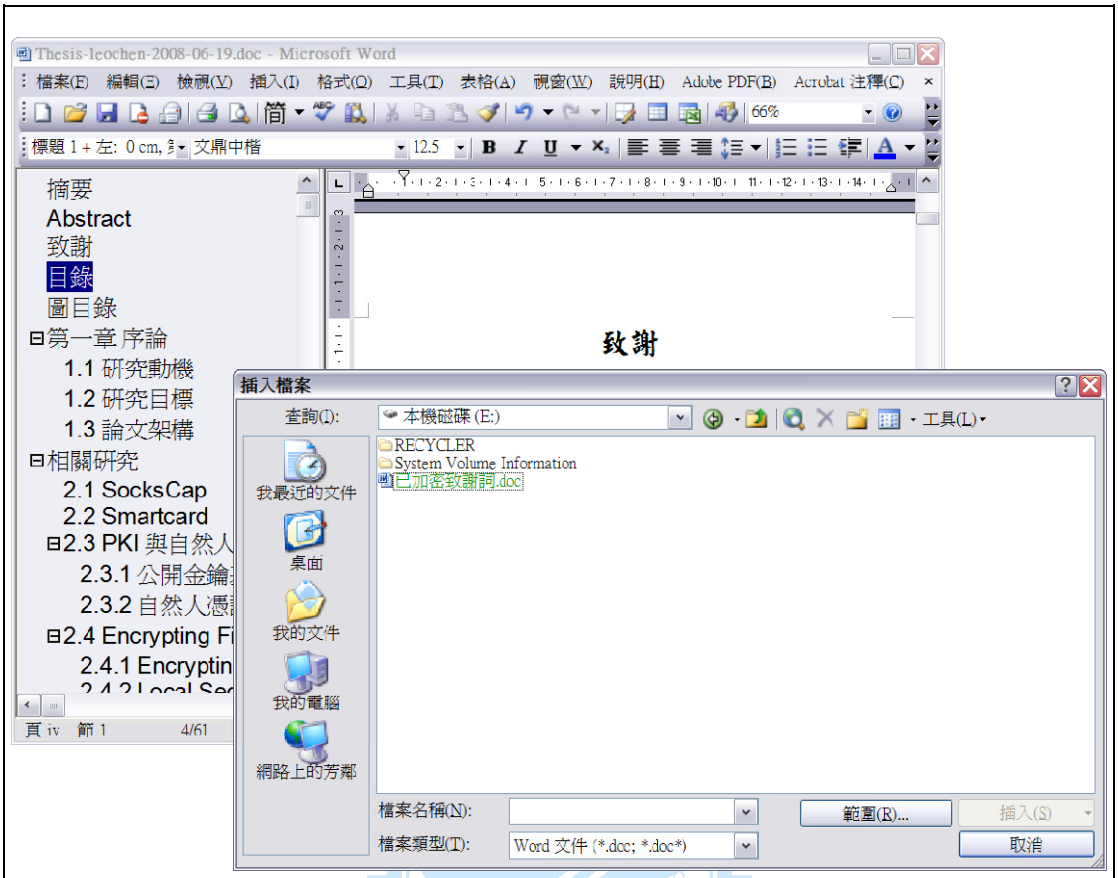

Fig. 42 插入已加密文件

4.2.2 安全性分析

此小節我們針對「非對稱加解密金鑰產生」與「加密流程」做安全性相關的分析。SecCap使用修改過後的OpenSSL程式庫「myOpenSSL」做爲金鑰產生模組，其相關介紹與金鑰安裝流程於3.4.3。另外我們使用EFS做爲主要的檔案加密工具，相關介紹於1.3.3.1。

爲了在不同電腦上產生相同的非對稱加解密金鑰，我們使用myRSA_generate_key_ex函式產生金鑰，它與OpenSSL的RSA_generate_key_ex函式的不同，主要爲我們移除了所有因外在環境因素(時間、設備訊息等)的而變動的敘述刪除，進而修改爲常數版本。非對稱加密金鑰的資料結構中，重要的成員如Fig. 43。myRSA_generate_key_ex函式初

始化區域變數後，首先產生大質數p和q，再計算n（值爲pq）與d，其中d滿足de≡1 mod (p-1)(q-1)，e爲函式參數之一，通常爲 65537、17 或 3，最後計算dmp1 (值爲d mod (p-1))、dmq1 (值爲d mod (q-2)) 與iqmp (值爲q^{-1} mod p)。

```
struct rsa_st
{
     BIGNUM     *n;
     BIGNUM     *e;
     BIGNUM     *d;
     BIGNUM     *p;
     BIGNUM     *q;
     BIGNUM     *dmp1;
     BIGNUM     *dmq1;
     BIGNUM     *iqmp;
     …
};
```

Fig. 43 非對稱加密金鑰之資料結構

一般而言，我們產生大質數 p 和 q 的方式，通常是隨機選擇一個大數，再根據一些規則重複測試其是否爲質數。若測試不通過，便再隨機選擇大數再行測試，若通過，則我們便完成產生大質數。這裡「隨機選擇大數」將是我們欲在不同電腦上產生相同的非對稱金鑰之關鍵，也就是說，我們必須可以控制隨機產生的數值，希望它不會因我們無法預知的外界因素來影響隨機數值的產生。

因此，我們同樣地實作 myRAND_meth 系列函數做爲亂數產生器、myBN_rand 系列函數做爲大數亂數產生器、myBN_generate_prime_ex 系列函數做爲大數質數產生器。以上系列亂數產生器 (Pseudo-random number generator, PRNG) 皆移除了時間、CPU 時鐘、設備資訊、EGD (entropy gathering daemon) 等等參考，我們僅參考智慧卡內憑証之主體名稱、SecCap 秘密字串之雜湊值、與使用者自訂之密碼 (通關密語) 之雜湊值等等，做爲產生亂數之依據，用以在不同電腦上產生相同的非對稱加解密金鑰，以及解決更換卡片的問題。

根據現今非對稱加解密技術的應用如此發達 (如 PKI)，我們相信此技術

對於目前的科學而言具有一定的安全程度。SecCap 對於此部分，較於不同的是「加解密金鑰的產生」其中的亂數產生器。若攻擊者可同時取得憑証之主體名稱 (SubjectDN) 、SecCap 秘密字串 (SecCap secret value) 、使用者自訂之通關密語 (Passphrase) 、以及我們所使用的雜湊方法 (Hash method) 等數項資訊，則我們產生出來的非對稱加解密金鑰便爲可預期的 (Predictable) ，此則爲不可靠、有安全疑慮的方法。但我們認爲攻擊者或一般使用者，不易於同時取得上述全部的資訊，並且爲了解決不同電腦上金鑰產生的問題，SecCap 選擇此擇中辦法。

EFS加密流程簡圖如Fig. 5，其中「Plaintext File」即爲欲加密之明文檔案，「FEK」爲隨機產生的對稱式金鑰，用於加密檔案，「User's public key」爲使用者之非對稱公開金鑰，「Encryption」爲對稱式加密方法，以FEK爲金鑰，對Plaintext File做加密動作，「DDF generation」爲Data decrypting field (EFS屬性之其中一欄位，如Fig. 44所示) ，其中包含了以User's public key爲非對稱金鑰，對FEK加密的資料。

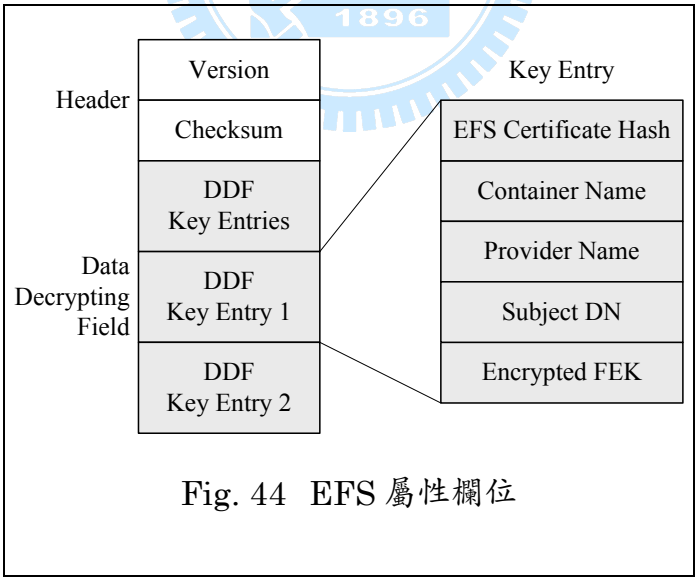

Fig. 44 EFS 屬性欄位

EFS 對於金鑰管理與存放，可說是層層把關。一使用者若欲解密文件，首先需使用 MKEK (Master key encryption key，由使用者密碼與其它資訊等雜湊而成) 對已加密的 MK (Master key，位於%APPDATA% / Microsoft /

Protect / SID) 做解密動作，再以 MK 對已加密的使用者私密金鑰 (位於 %APPDATA% / Microsoft / Crypto / RSA / SID) 解密。取得私密金鑰後，便可對已加密的 FEK (位於 EFS 屬性之 DDF 欄位內) 做非對稱解密。最後，再以此 FEK 爲金鑰，對密文檔案解密，便可取得明文檔案。

對於一個攻擊者而言，由於每個密文檔案皆由不同的對稱金鑰 FEK 加密，通常他希望能夠取得使用者私密金鑰，便可對每一個密文檔案做解密。若攻擊者可取得下列資訊之一，則私密金鑰便可能遭受破解：

- Master key (MK)
- Master key encryption key (MKEK)
- Windows 使用者密碼、與 MKEK 之產生方法

上述產生一系列金鑰之源頭爲使用者密碼與 MKEK。所幸，Microsoft 並未說明 MKEK 之產生方法，或其所參考的相關資料與雜湊函數。攻擊者在未知其 MKEK 產生方法的情況下，應難以取得使用者私密金鑰。

4.2.3 加密內容分析

我們使用檔案大小爲 1024 Bytes的「Test_document_1K.txt」對其做加密，其明文爲UTF-8 編碼，內容如Fig. 45，密文內容如Fig. 46，而其EFS屬性內容及註解如Fig. 47。此節利用ntfsprogs[2]、WinHex[24]將密文與EFS屬性之內容進行獲取。

SecCap：自動化個人文件安全系統．使用智慧卡認証　學生：陳癸夫 指導教授：楊 武 博士國立交通大學資訊科學與工程研究所　摘要 文件安全是現今個人電腦上一項很重要的課
(cropped)
於 NTFS 檔案系統、Windows 使用者與金鑰管理之上，透過智慧卡做個人身份辨認與驗証，在可接受的效能負擔下對私人機密文件進行自動化

Fig. 45 Test_document_1K.txt 之明文內容

```
000  43 4a 4c 73 cb 69 cf fe 94 1e f3 4b 1a 28 50 c1 CJLs.i.....K.(P.
010  86 7e b3 3f 0c de 9e 4d 43 d1 84 c2 7c c4 26 75 .~.?...MC...|.&u
020  66 27 ca e3 0e a7 2f 88 a8 b9 38 55 c9 fa 34 72 f'..../...8U..4r
030  0e 5e 95 20 2f 2d 18 53 0c 3f b8 52 ad c6 2a cf .^. /-.S.?.R..*.
040  44 cc bc 8b 9d 28 d0 83 15 94 35 2a 72 a2 6b f1 D....(....5*r.k.
                          (cropped)
3b0  2a bc 3e 16 13 3c 09 d7 99 2e 1f ae b5 4d d5 0b *.>..<.......M..
3c0  96 a2 8d 52 d8 b0 e1 6c a7 a0 ae 82 24 9b 2a 71 ...R...l....$.*q
3d0  90 90 dc b0 34 4b 56 37 88 29 c3 cd a2 8b 92 69 ....4KV7.).....i
3e0  f4 4b fe 12 6d 09 71 db 81 b3 fd ab 44 4c ab 73 .K..m.q.....DL.s
3f0  a5 37 ff 49 93 86 f8 35 01 dd fc f4 f3 95 45 4b .7.I...5......EK
```

Fig. 46 Test_document_1K.txt 之密文內容

Fig. 46當中，其內容大不同於明文內容，且無法閱讀，在此我們便判定它確實是經過加密後的文件。

Fig. 47當中，「EFS attribute size」指出此屬性的長度，即爲 528 (0x210) Bytes。「Public key thumbprint」爲公開金鑰之姆指紋。「Private key GUID (Globally unique identifier) 」，亦稱作Container name，是私密金鑰之一雜湊值，用來從CryptoAPI取得所需資訊。「Cryptographic provider name」爲 Microsoft Base Cryptographic Provider 1.0。「User name」即爲EFS憑証之主體名稱（非自然人憑証），在SecCap中即爲使用者名稱。「Encrypted FEK (File encryption key) 」爲已使用公開金鑰加密過後的FEK，其FEK是一把對稱加密金鑰，眞正用來加密檔案使用的。

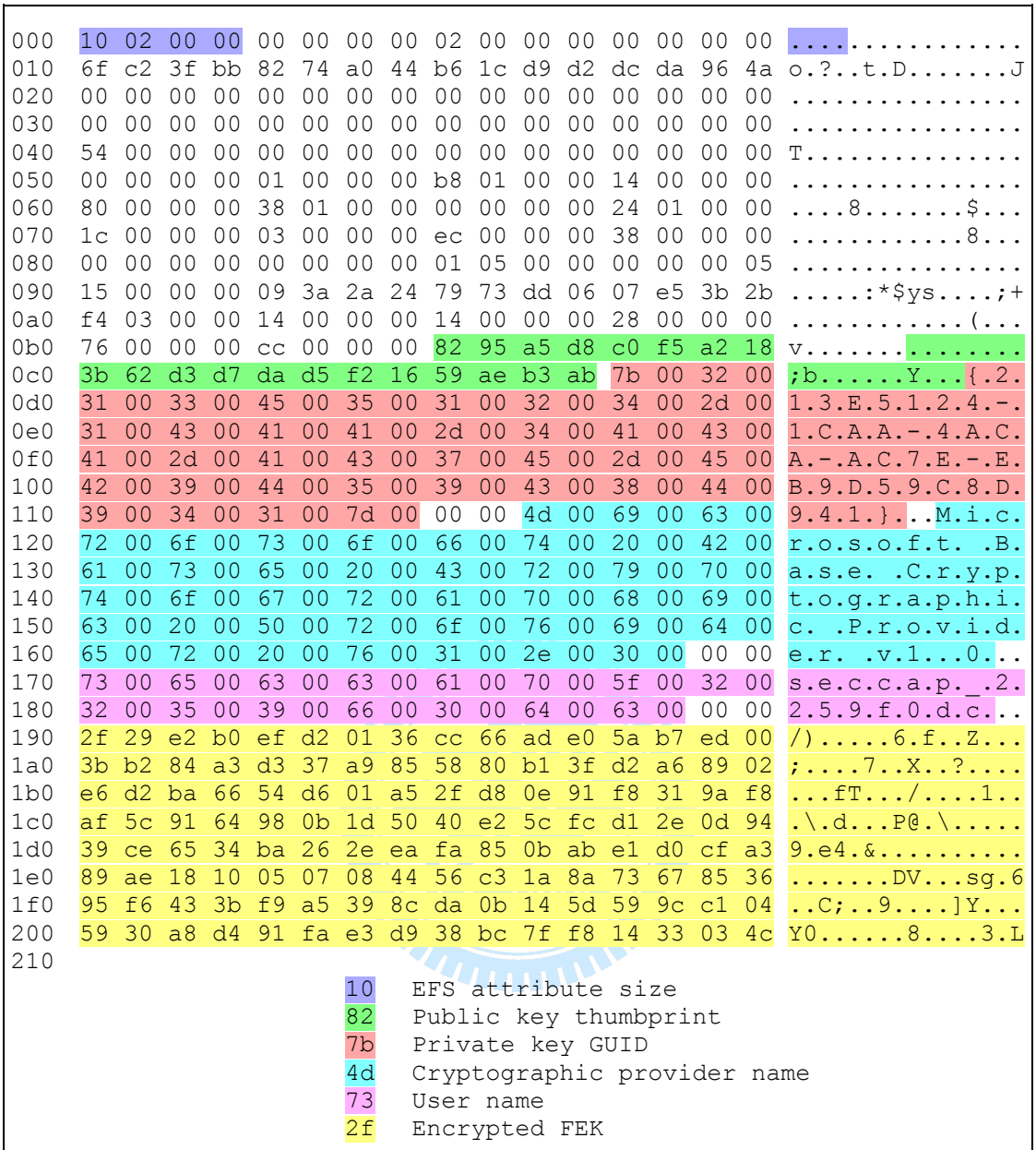

Fig. 47 Test_document_1K.txt 之 EFS 屬性內容與註解

4.2.4 效率比較

Fig. 48、Fig. 49、與Fig. 50爲PC1 與PC2 執行SecCap加解密 100 Mbytes、1 Mbytes、與 1Kbytes檔案所需的時間，其顯示4.2.1中的「加密 1 KB、1 MB、100 MB之文件」的結果，實際加解密模組係由EFS實作，其數據僅供SecCap加解密之負擔做爲參考。

Fig. 51為USER1 與USER2 在PC1 與PC2 上之非對稱加解密金鑰產生所需花費的時間。由於USER1 與USER2 之主體名稱不相同，因此將會產生不同的亂數序列（Random number sequence），進而影響金鑰產生所需的時間。

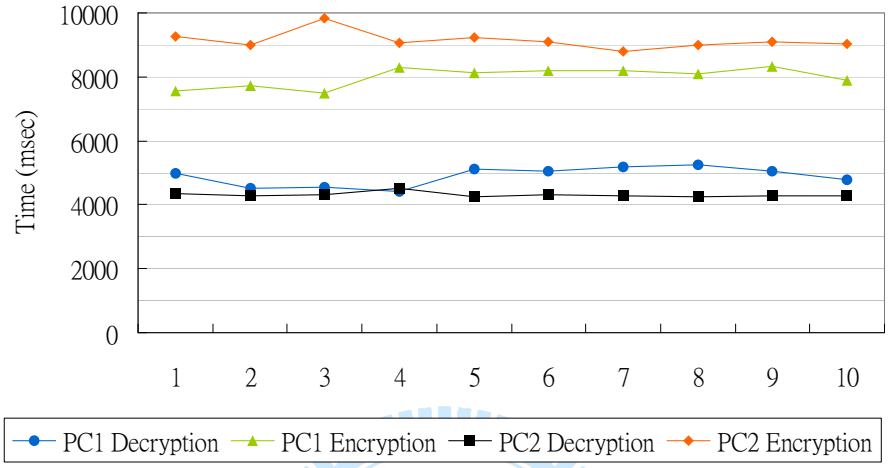

Fig. 48 加解密 100 Mbytes 的時間

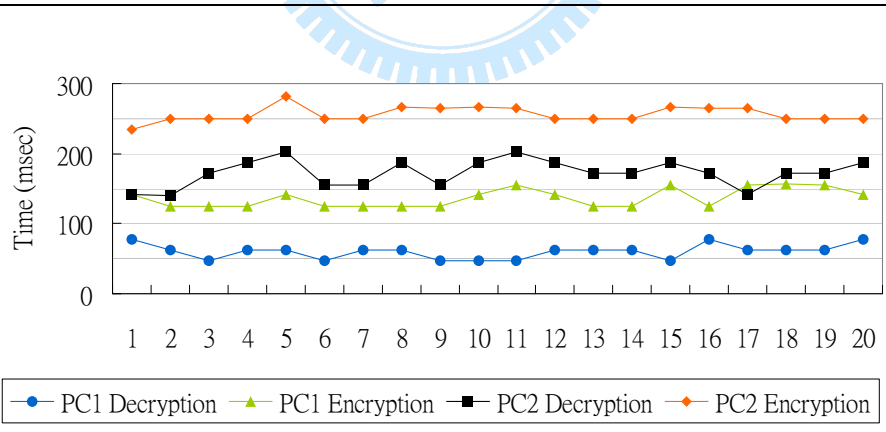

Fig. 49 加解密 1 Mbytes 的時間

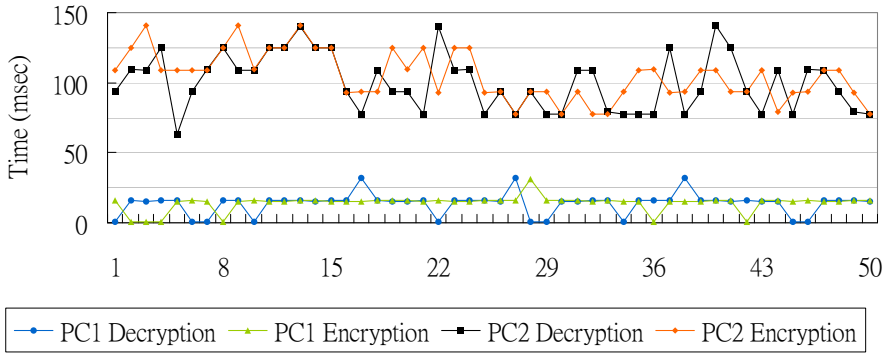

Fig. 50 加解密 1 Kbytes 的時間

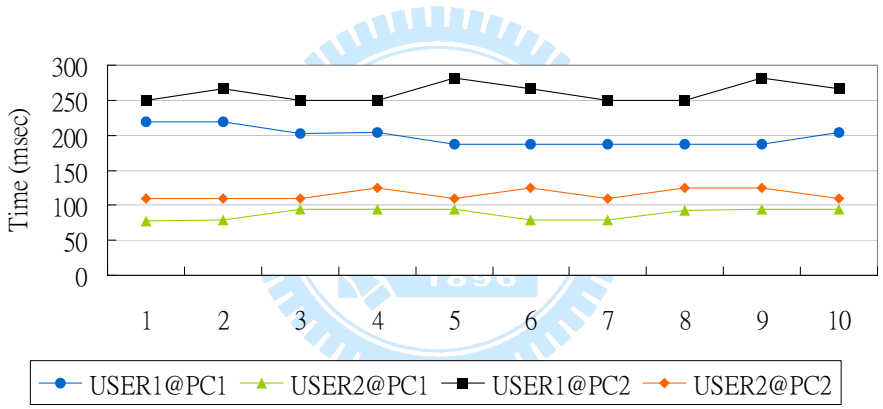

Fig. 51 非對稱加解密金鑰產生的時間

我們另外針對一般檔案加解密軟體、與 SecCap 做操作次數分析。欲使用文件編輯軟體配合一般檔案加解密軟體，首次所需要的步驟如下：

- 開啓文件編輯軟體
- 編輯增修文件
- 儲存文件
- 關閉文件編輯軟體
- 使用檔案加解密軟體加密文件

而第二次之後的存取，所需要的步驟如下：

- 使用檔案加解密軟體解密文件
- 使用文件編輯軟體開啓文件
- 編輯增修文件
- 儲存文件
- 關閉文件編輯軟體
- 使用檔案加解密軟體加密文件

欲使用文件編輯軟體配合 SecCap 進行檔加解密，首次所需要的步驟如下：

- 開啓並登入 SecCap
- 使用 SecCap 開啓文件編輯軟體
- 編輯增修文件
- 儲存文件
- 關閉文件編輯軟體
- 使用 SecCap 加密文件

而第二次之後的存取，所需要的步驟如下：

- 使用 SecCap 所開啓之文件編輯軟體開啓文件
- 編輯增修文件
- 儲存文件
- 關閉文件編輯軟體

由於使用SecCap，第二次之後的存取皆少了手動加解密的步驟，因此在多次存取而言(如Fig. 52)，我們確實減少了使用者爲了達到文件安全約三成的操作。

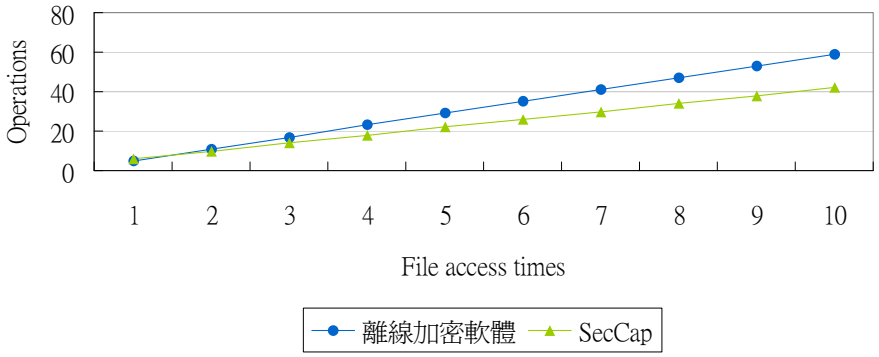

Fig. 52 SecCap 所減少的手動操作

第五章　結論

5.1　結論

現今資訊爆炸的時代裡，隨著各項技術的發展，個人私密文件的安全性已愈來愈被受重視。不論是各大小企業、政府機關或警備單位、甚至個人使用者，皆發現了文件安全的重要性。而使用者身份認証的方式，從純粹密碼認証，發展至指紋、視網膜辨識，其中目前公認較爲普遍且安全的方式，智慧卡認証便爲其中之一。

一般電腦使用者，大都習慣了已往對於電腦的操作方式。若欲保持個人私密文件的安全性，使用者必須多學習一些工具、以及手動執行一些操作來達成。這些步驟通常較爲繁鎖，且根據一般使用者的惰性，他們很可能認爲這些動作已造成了他們的麻煩，甚至認爲這些保護文件的動作是非必要的。

爲了解決上述問題，本論文－SecCap 系統－示範了自動化個人文件安全與智慧卡認証的結合應用。由於現今許多類似的軟體，大都僅提供離線加解密，或是只有一道通關密語做爲使用者身份認証的依據。使用者必須在編修文件時，不斷的手動對其做加密解密，我們認爲這是相當不便的。另外僅以一道通關密語做爲身份之辨認，亦是不甚妥當。本論文提出之 SecCap 系統，使用智慧卡做爲認証，並且可選擇性再使用通關密語認証做爲第二道關卡，最主要的，使用者不必每次對私密文件做手動加密解密，便可直接存取編修該文件。

5.2　未來展望

資訊安全課題愈來愈受重視，不少的學界業界皆投入其中，並且根據各項的需求，有許多相關的草案或標準已紛紛提出。本篇論文雖然能藉由設計實作與程式的撰寫，來實現我們的想法，但在有限時間內完成的研究，其成果亦是

相當有限。SecCap 結合了智慧卡認証、EFS 加密模組、與 FileSpy 驅動程式等，提供圖形化使用者介面，但是尚有許多部分是可以再行加強的。目前的電腦的平台與作業系統可說是多而繁雜，而 SecCap 僅針對 Windows 作業系統與.NET framework 所設計，並且許多功能都有所局限，不容易做其它的延展。

SecCap 目前僅針對磁碟上的檔案進行私密性保護，並無做到認証性、不可否認性等更進一步的安全性保護。此處需仰賴於規模更大的系統設計才可達成。另外，現在已知有某些惡意軟體、病毒等等，並不針對磁碟上的資料做窺視，進而直接對記憶體內容進行竊取。因此亦有一項議題－對存於記憶體之內容做保護－是個更多更複雜的討論。

我們希望未來能將 SecCap 做較佳的模組化，讓使用者可選用內建或自訂的加解密演算法，或是可選擇其它的認証方式。定義完整的介面，讓程式開發者更容易在其上頭增加其它功能，如加密電子郵件等。對於透明即時加解密而言，必須針對不同的作業系統，另外撰寫一支 File system filter driver，與應用程式與認証程序做溝通，如此便可在其它的檔案系統與平台當中使用。

參考文獻

[1] Alan O. Freier, *"The SSL Protocol Version 3.0"*, Transport Layer Security Working Group, http://wp.netscape.com/eng/ssl3/draft302.txt, 1996.

[2] Anton Altaparmakov, *"ntfsprogs: tools for doing neat things with NTFS"*, http://www.linux-ntfs.org/, 2007.

[3] Eric A. Young and Tim J. Hudson, *"OpenSSL: The Open Source toolkit for SSL/TLS"*, http://www.openssl.org, 2008.

[4] Eric Rescorla, *"Diffie-Hellman Key Agreement Method"*, Request for Comments, http://www.ietf.org/rfc/rfc2631.txt, 1999.

[5] National Institute of Standards and Technology, *"Announcing the Advanced Encryption Standard"*, Federal Information Processing Standards Publication 197, http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[6] International Organization for Standardization, *"ISO 7816: Smart Card Standard Overview"*, http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx, 2005.

[7] International Telecommunication Union Telecommunication Standardization Sector Study Group 17, *"ITU-T Recommendation X.509"*, published as ISO/IEC 9594-8, http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.509-200508-I!!PDF-E&type=items, 2005.

[8] International Telecommunication Union Telecommunication Standardization Sector Study Group 17, *"X.690 ASN.1 encoding rules: Specification of Basic Encoding Rules (BER) , Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) "*,

published as ISO/IEC 8825-1, http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf, 2002.

[9] International Telecommunication Union Telecommunication Standardization Sector Study Group 17, *"X.680 Abstract Syntax Notation One (ASN.1) : Specification of basic notation"*, published as ISO/IEC 8824-1, http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf, 2002.

[10] J. Daemen and V. Rijmen, *"AES Proposal: Rijndael"*, AES Algorithm Submission, 1999.

[11] Kazumasa Omote and Kazuhiko Kato, *"Protection and Recovery of Disk Encryption Key using Smart Cards"*, 5th International Conference on Information Technology: New Generations, 2008.

[12] Marc Witteman, *"Advances in Smartcard Security"*, Information Security Bulletin, http://www.riscure.com/1_general/articles/ISB0707MW.pdf, 2002.

[13] Marcus Leech, *"RFC 1928: SOCKS Protocol Version 5"*, Request for Comments, http://www.ietf.org/rfc/rfc1928.txt, 1996.

[14] Max Artemev aka Bert, *"FreeCap"*, General Public License, http://www.freecap.ru/eng, 2006.

[15] Microsoft corporation, *"Encrypting File System"*, http://www.microsoft.com/taiwan/technet/security/guidance/protect_data_EFS.mspx, 2006.

[16] Microsoft corporation, *"File System Filter Drivers"*, http://www.microsoft.com/whdc/driver/filterdrv/default.mspx, 2008.

[17] Microsoft corporation, *"Installable File System Driver Development Kit"*, http://www.microsoft.com/whdc/devtools/ifskit/default.mspx, 2004.

[18] Microsoft corporation, *"Object IDs associated with Microsoft cryptography"*, http://support.microsoft.com/kb/287547/en-us, 2007.

[19] MOICA內政部自然人憑証管理中心, http://moica.nat.gov.tw.

[20] Robert Richardson, *"The 12th annual Computer Crime and Security Survey"*, CSI, http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf, 2007.

[21] RSA Laboratories, *"PKCS #1 RSA Cryptography Standard"*, RSA Security Inc. Public-Key Cryptography Standards, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf, 2002.

[22] RSA Laboratories, *"PKCS #10 Certification Request Syntax Standard"*, RSA Security Inc. Public-Key Cryptography Standards, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf, 2000.

[23] RSA Laboratories, *"PKCS #12 Personal Information Exchange Syntax Standard"*, RSA Security Inc. Public-Key Cryptography Standards, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-12/pkcs-12v1.pdf, 1999.

[24] Stefan Fleischmann, *"WinHex: Computer Forensics & Data Recovery Software, Hex Editor & Disk Editor"*, X-Ways Software Technology AG, http://www.x-ways.net/winhex/, 2007.

[25] W3 Schools, *"W3 Schools OS platform statistics"*, http://www.w3schools.com/browsers/browsers_os.asp, 2008.

[26] Wikipedia, *"Local Security Authority Subsystem Service"*, http://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service, 2008.

[27] 周桂田, *"生物特徵辨識作爲全球鐵的牢籠－全球在地化之風險典範衝突"*, 全球化時代的公民與國家暨台灣社會變遷基本調查第十次研討會, http://www.ipsas.sinica.edu.tw/image/ipsas/1/531.pdf, 2007.