國立台北大學資訊工程學系專題報告

相機來找碴

專題組員:鄭星佑
指導老師:張仁俊老師
專題編號: PRJ-NTPUCSIE-100-003
執行期間:100 年 3 月 至 101 年 1 月

1. 摘要

運用一套密碼與編碼之演算法，對圖片進行運算，將資訊植入圖片進行保護，於日後可用來對圖片進行竄改的偵測與定位；而被植入保護的圖片與原始圖檔能達到肉眼上看不出差異，且其檢驗結果讓人可以很輕易的辨識出差異,並將這套技術與 Android 智慧型手機的相機鏡頭作結合寫成一個 APP,達到一個軟體就能做照相、植入保護、竄改的偵測與定位等功能。
關鍵詞：密碼、編碼、Android

2. 簡介

近年來智慧型手機與平板電腦是一股新的趨勢，而其中主要有 APPLE 跟 Android 兩大陣營，後者是所謂的 open source，開發者可以輕易的拿到一些系統相關資訊，並且彼此之間進行交流。

而且智慧型手機上有很多功能，像是 G-sensor、溫度感應、相機、語音、無線網路等等；開發者可以運用自己的程式能力加上創意，搭配智慧型手機上的功能去做結合，來寫出屬於自己的應用程式，不但可以提高使用智慧型手機的方便性跟娛樂性，也能上傳到 market 與人分享，甚至藉此來獲利，也因此專題題目才會想要著手於 Android APP 開發。

而現代人使用智慧型手機的習慣，除了撥打跟接聽電話外，拿來聽音樂、上網、照相、甚至玩一些別人所開發出來的 APP 做遊戲用途等等，也都是很頻繁的使用動作；而我的目標主要是想做影像這一塊，因為大多數人拍完照片以後，下個動作不外乎就是上傳到網路上，也許是個人相簿或是像 FACEBOOK 這種平台，而這也就會衍生出個人資訊安全的議題，比如說照片會不會被盜用拿去做宣傳廣告，或是遭人移花接木等等。

所預期想要達到的目標，就是開發出一個能夠對圖片植入保護的照相軟體，並且也能做竄改的偵測與定位；也就是說，透過此 APP 拍照，照片會被植入保護，當然也能夠選取其他特效照相軟體所拍成圖片對其植入保護；而被植入保護的圖片與原始圖檔要達到肉眼上看不出差異，且所植入的資訊就是將來能讓我們拿來對圖片進行竄改的檢驗與定位。

簡單來說只要一個 APP，就能夠進行拍照、植入保護、竄改的偵測與檢驗等動作。

3. 專題進行方式

- **實作平台**

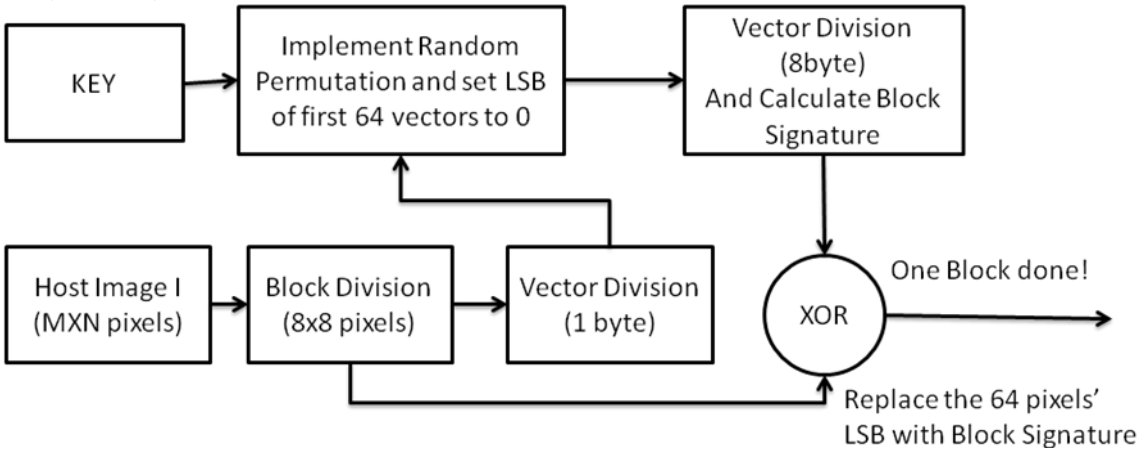
先以 Windows xp 為實作平台，用 MS Visual C++ 2005 寫一個視窗程式出來，確認演算法的理論可行之後，再以 ASUS T101 android 3.0 為平台，用 Eclipse 去寫程式做成 Android APP，後續的實機測試都是直接拿 ASUS T101 來進行，並且不斷 try and error 去做 debug 跟修改及微調。

- **技術介紹**

這部分要講解的算是專題的核心，也就是利用密碼與編碼來對圖片植入保護的演算法部分，會搭配流程圖一步一步來進行解說。

核心演算法流程

（流程圖）



主要 Input 有兩個，也就是 KEY(user-defined password)，跟要做植入保護程序的 Host Image(透過相機或是選取指定路徑取得)，首先我們會將 host image 以每 64 個 pixels 為單位切成好幾個 Block，每次只抓一個 Block 來進行處裡；Block 抓出來以後，先以 1 個 byte 為單位去做 vector division，分出 64 vectors 之後，以 KEY 為 seed 做 Random permutation，將 vector 的順序打亂並且把每個 vector 的 LSB 清空，設為 0。

接下來則以每 8 個 bytes 為單位再做一次 vector division，分出 8 個 vectors 後做疊加運算，最後就會得到一個 8 byte 的 vector，也就是算出 Block Signature。

這個 Block Signature 是 8*8，也就是 64Bits，剛好可以跟 Block 裡的 64 個 pixel 去做 XOR 的運算，分別把 Block Signature 的每個 Bit 填到已經被我們清空的、每個 pixel 的 LSB 裡。

這樣子就算完成一個 Block，接下來則是進行下一個 Block，直到整張圖跑完為止，即完成對圖片植入保護的部分。

而由於我們動的是每個 pixel 的 LSB，因此對於人類肉眼的視覺上來說，並不會感覺到被植入保護資訊的圖片與原始檔案有何差別，但那些植入的資訊對於我們之後做竄改的偵測與定位確相當重要。

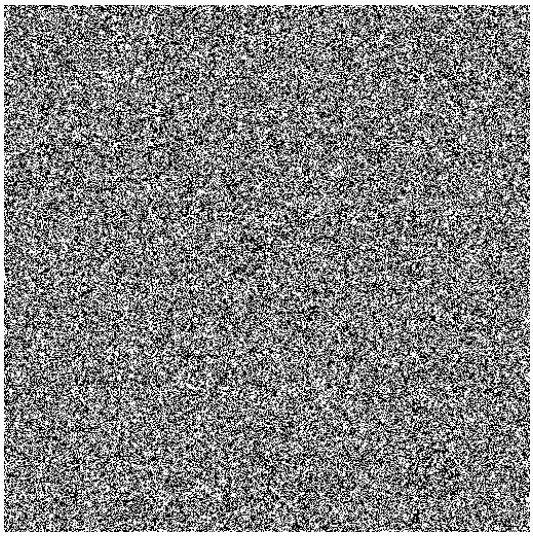
接下來介紹竄改的偵測與定位這部分的流程，事實上其方法與植入保

護的程序並沒有很大的差異，一樣都是先把圖片以每 64 個 pixels 為單位切割成好幾個 Block，再搭配 KEY，用同樣的程序去做運算得到 Block Signature；不過每個 pixel 的 LSB 要清空做運算前，會另外用一個矩陣把值都存起來，目的就是用來與算出來的 Block Signature 做比對，進而能想達到去檢驗圖片上哪些相對應的 pixel 曾被修改過，而比對結果若一樣則回填白色，而假如遭竄改，就回填黑色，因此就能很明顯的辨識出遭竄改的範圍。(見圖 1-1、1-2)

但前提是圖片得有植入保護過才行，假如我們拿未經此程序處理的圖片進行檢驗，或是 KEY(user-defined password)輸入錯誤，得到的結果都會是一張雜訊圖。(見圖 2)

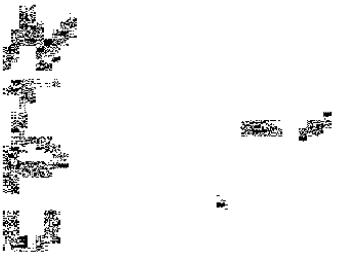
這麼做的目的是為了保障個人，因為每個人的 KEY 應該都只有自己曉得，用什麼 KEY 去做植入保護的程序，相對的也就要用同樣的 KEY 來做竄改的偵測與定位；因此別人不能對你的圖片進行檢驗，而相對的我們也無法去檢驗別人的圖片。

(圖 2：未植入保護或 KEY 打錯的結果)



(圖 1-1：遭竄改的圖片)

(圖 1-2：遭竄改的圖片之檢驗結果)



4. 主要成果與評估

- **研製成果**

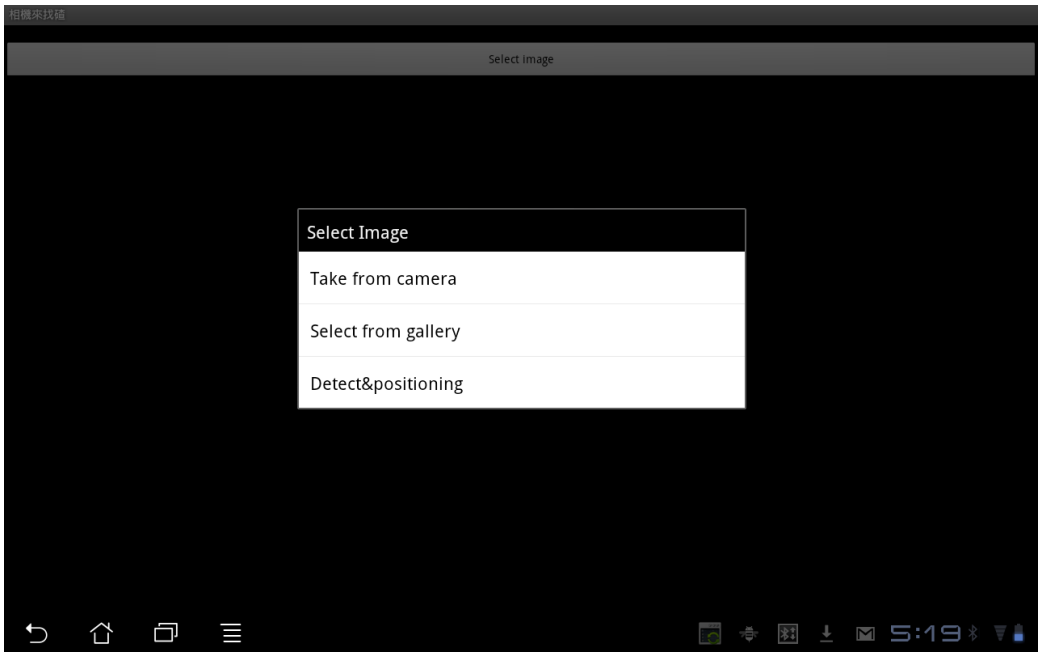
由於我是自己一組，所以單獨包辦了所有的事情，包括此 APP 想法的構思與程式開發，以及後續的實機測試跟 debug 等等。

而一開始在 Windows xp 上開發出視窗程式，確定理論可行之後，我就開始轉移到 ASUS T101 這個硬體上，改以 Eclipse 配合 Android SDK 的套件作開發，用 JAVA 改寫並且結合手機的相機功能來完成這個特別的照相軟

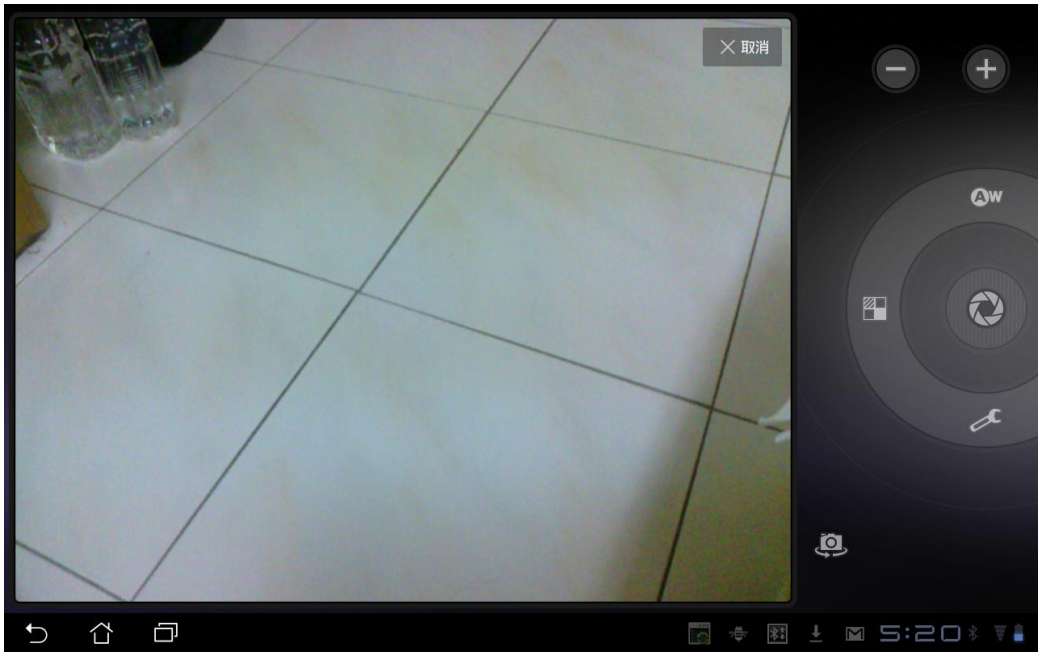
體，事實上前面所舉例的幾張圖片範例，都是用該軟體成功檢驗出來的圖片，證明有確實達到最初設定的目標：「一個軟體就能做照相、植入保護、竄改的偵測與定位等功能。」

而我把 APP 的 Layout 設計得很簡潔(見圖 3-1、3-2、3-3)，因為畢竟是個照相軟體，畫面上不適合有太多的按鈕，觸發程序的按鈕我都以對話框選單的方式呈現，點擊後就用 Intent 的方法去叫出相機鏡頭或內建的檔案管理員，得到圖片來源後對其進行植入保護的程序或著是檢驗之。

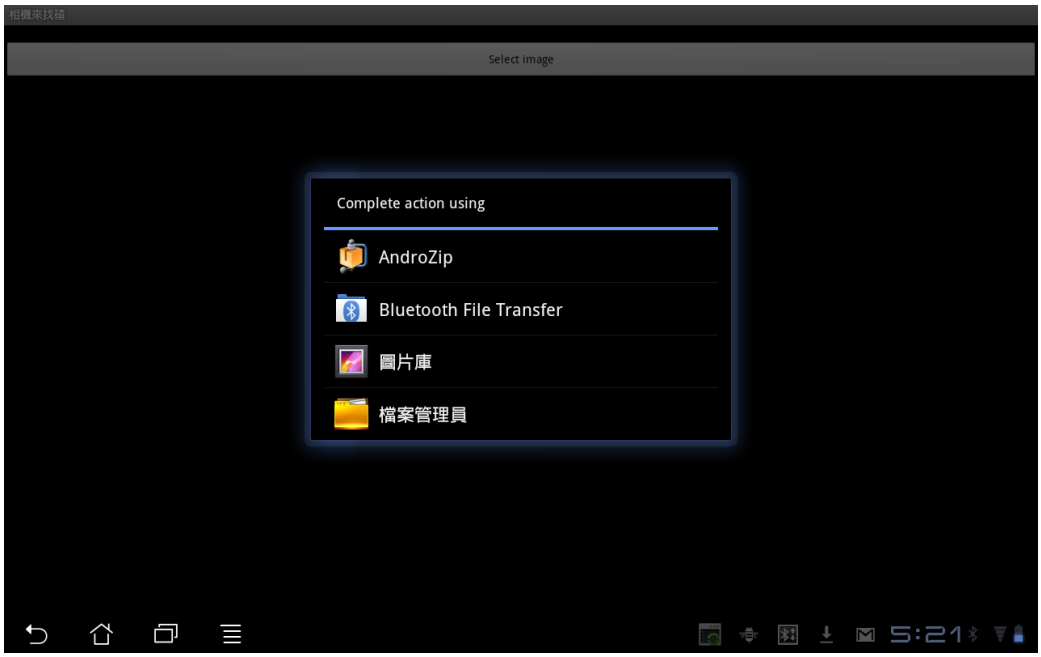
（圖 3-1：程式主畫面）



（ 圖 3-2：點選「Take from camera」後，即開啟相機鏡頭畫面 ）



（ 圖 3-3:點選「Select from gallery」或「Detect&Positioning」對選取之圖片植入保護或進行檢驗 ）



不過起初剛轉移到做 Android APP 開發時遭遇了不少困難，首先 Android 是以 JAVA 去做開發，對於以往只學過 C 跟 C++的我花了不少時間去熟悉 JAVA，但還好我所想要寫的東西並不會接觸到 Android 太底層的東西，而且有學過電腦語言，要去學習別種語言到堪用的程度，不會花太多時間。

再來就是 Android 一些程式開發的特色，像 Intent 跟 Activity 這些概念，是以前寫視窗程式沒接觸過的。

所幸 Android 是 open source 的系統，對開發者來說滿友善的，除了自己找書來看以外，網路上也有 Android 官方的開發者網站，除了對 SDK 的介紹外、也有一些給開發者的導覽跟教學，算是滿詳盡的。

- **擴展方向**

在做專題的過程中也發現到一些未來也許能再擴展的部分，像是演算法的改進，因為我只動到 pixel 的 LSB，意思也就是說，0~255 中我只動了 1，所做的動作對圖片顏色呈現的約只有 0.4%的影響程度而已，因此是否能對圖片植入更強的保護，藏一些資訊至倒數第 2、甚至第 3 個 bit 裡頭，且依然能讓圖片看不出與原始圖有和差異，也許是能繼續研究的；當然也或許沒這個必要性或視覺效果會不好等等；這些都是之後可以去做延伸探討的部份。

再來則是類似這種概念能不能應用到實體相機上，因為數位相機拍出來的照片都會有 EXIF 訊息，紀錄像是拍攝時間、相機型號、解析度、曝光時間諸如此類的訊息等；如果這種植入保護的概念能應用到數位相機上，再配合製造商的出貨號或條碼之類的來當作 KEY，或許能達到同樣效果。

5. 結語與展望

這次的專題接觸了以往所沒碰過的 Android APP 開發，不但如此，JAVA 語言跟 Eclipse 這個開發軟體也是第一次接觸，因此開始轉移到做 APP 開發真的是碰了不少壁，原本覺得理論都已經寫出實際的視窗程式來了，開發成智慧型手機的 APP 不過就是結合相機功能來完整它而已，豈知剛起頭就沒原先所想的那麼順利，最後只好回歸最原始，買了幾本介紹 Android 應用程式開發的書來看，也去借了幾本講 JAVA 的書來學習，最後才慢慢掌握到 Android 一些重要的特色，像是 intent 跟 activity 的概念。

這些應該要會的基本概念有了以後，開發軟體其實就比較容易了，最主要就是因為 Android 是 open source，很多系統資訊都是公開的，甚至每個版本多出哪些功能、要怎麼用，官方的開發者網站上也都會揭露；所以如果有學過電腦語言再加上有一些開發程式的經驗，要轉到 Android 這一塊，到達上手、寫一些堪用的 APP，門檻其實不會太高。

當然最重要的還是要有基本的 JAVA 語法概念，以及對 Android 程式特色要有所認知；這些觀念建立起來，就可以運用自己的創意跟程式能力，結合智慧型手機或平板電腦這些載具上的功能，來寫出屬於自己 APP；甚至也能針對某種目的或族群去做開發，像是遊戲娛樂或著是跟生活息息相關的議題等等；開發出來的 APP 如果很有趣或引起的共鳴多，想要藉此來獲利也不是不可能的事情，畢竟現在市面上有不少公司或是以個人名義來進行各種手機應用程式的開發。

6. 銘謝

首先要感謝張仁俊老師，這段時間真的是增長了不少，不管是資訊相關知識或是程式能力、甚至做人處事的道理等等。

老師其實都不會給我們這些專題生什麼壓力，每個禮拜就是固定開會回報自己的進度，做了哪些事情、進行的怎麼樣、有沒有遇到什麼困難等等；但我覺得這種反而是最高境界，因為你不曉得到底做到怎麼樣的程度老師會覺得 OK，所以你至少會做到讓自己放心，才敢去開會、報告時才不會因為沒料而心虛；也因此，不知不覺中反而會自動自發的去 push 自己。

而當然我們還是會有出包，或是偷懶開天窗等很誇張的時候，你可以察覺到老師的表情不太對，不過老師都不曾用尖銳的語言或情緒化的方式來表達其不滿，這點很讓我敬佩。

還有就是經驗的傳授，像是簡報的技巧跟一些要領，因為有些東西就是你沒經驗，但是只要別人稍微點你一下，改過來之後，對於整體呈現出來的感覺就會差很多。

總之這一年下來覺得老師是個很正派、有氣度的人，相當感謝他!

另外也要感謝兩位朋友，涂仲蔚跟吳孟倫，因為我是自己一組的關係，有時寫程式會出現盲點，雖然編譯過得去，但就是會有一些 runtime error，有時自己再怎麼 debug 還是有點鬼打牆，一直在原地打轉，所以如果真的瓶頸卡很久，我都是找他們兩個討論，因此很感謝他們都願意花時間跟我討論，給我一些意見跟幫助，甚至是教我一些 debug 的〝眉角〞！

7. 參考文獻

[1] Gasolin, Google！Android 2 手機應用程式設計入門第三版

[2] 林城, Google Android 2.X 應用程式開發實戰 第二版

[3] Android Developers, http://developer.android.com