以安全執行環境限制之內容保護機制

林宗伯
中央研究院資訊所
台北市南港區研究院路
（02）27883799-1459
lancelot@iis.sinica.edu.tw

洪偉能
中央研究院資訊所
台北市南港區研究院路
（02）27883799-1463
wnhung@iis.sinica.edu.tw

黃世昆
中央研究院資訊所
台北市南港區研究院路
（02）27883799-1808
skhuang@iis.sinica.edu.tw

摘要

數位內容保護機制可分成三方面，一是以資料形式的實體限制（copy restriction），另一方面是以加密的方式虛擬隔離（cryptographic seal），第三方面是程式控制形式的軟體包覆隔離（software wrapper）。實體隔離限制是傳統版權發行控制的方法，卻無法有效防止數位內容經網路等形式的大量複製散佈與非法傳播。本系統嘗試整合資料與程式流程控制(data and control flow)來進行內容保護與產權管理，並提出模擬實體內容改變複製品質的調整模式(content degradation)。有別於傳統數位產權管理，我們採行資料權限與控制權限分離的機制，使權限規格與描述更具彈性，並可避免惡意程式的干擾與竄改。

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Modules and interfaces*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *Invasive software and Unauthorized access*

General Terms

Management, Security, Legal Aspects.

Keywords

Self Executable Content, Sandbox, Content Degradation, Content Protection, Rights Management.

1. 前言

數位內容有別於傳統有形著作，以檔案形式存在、容易快速複製。因為這樣的特性，數位內容的複製動作容易觸犯著作權法中的重製過失，同時也可能侵害著作權人的權益[1]。正因為數位資料的使用極易有疏忽而產生侵權行為，為了保障與維護著作權人的權益，數位內容的權利管理機制（Digital Rights Management，DRM）就顯得格外重要。

數位內容供應的組成份子有三方面。第一是內容提供者，第二是內容服務提供者，第三是消費者。此三者形成數位內容的供需鏈。成功的 DRM 系統，必須保護三方的權益，並確保三方利益不受損害。

首先，為了使數位內容提供者得到利益，必須根據內容被使用的情形，讓創作者取得合理的報酬。為了達此目的，必須使數位內容本身僅能在合法的情形下執行所授予的動作。本系統採用資料加密與軟體包覆(software wrapper)達成此目的。

其次，數位內容服務提供者並非真正的創作者，而是提供入口(portal)，讓創作者能方便地提供作品，使用者也能快速找到所要的內容。

1.2 有待 DRM 系統解決的問題

網際網路的發達，打破了幾已達平衡的著作權市場[2]，其中關鍵在於電腦與網路快速複製與傳播的能力。一般人複製自己檔案是很普遍的行為，但若複製的不是自己的檔案，而是有版權的電子書，雖然一樣是複製動作，卻發生了重製的行為。這兩種狀況，同樣的動作，有不同的法律責任。第二種狀況有時可視為合理使用，但因為電子書複製後，仍保有同樣品質。所以數位內容的合理使用仍很難界定[3]。

數位權利管理雖能有效地達到限制複製的目的，但同時也剝奪消費者應有的權利。著作權法是為了保護著作人權益，調合社會公共利益，促進國家文化發展[4]，並非限制消費者的使用行為。有些人認為數位內容的複製是必然而有其必要性，甚至認為沒有得到著作權人同意的公開散播才算違反著作權法[5]。因此無論是消極的限制使用、或積極地促進文化發展，都必須顧及創作者與消費者的權益，並能控制數位內容的複製及散播。

以下章節分別介紹本系統採用相關技術、系統架構、平台管理、Rights center 與 client reader 的互動，最後並探討利用 Detours[12] 來保護數位內容的相關實作設計。

2. 內容保護平台相關技術

本系統所採用技術的介紹，包括 Sandbox 機制，Self Executable Content 與 Content Degradation 等。分別介紹如下：

2.1 Sandbox

Sandbox 在不修改原來行為流程，能對此程序加上安全性的限制，通常應用於網路安全、或防範病毒等領域，例如可針對作業系統進行防護、以防止未知病毒攻擊。若針對數位內容保護方面，可在使用者透通的情況下，確保數位內容本身被合法使用。

Sandbox 技術可以用在限制對系統資源的存取，如對檔案的讀取、寫入、或執行，或限制對網路連線的使用，以避免惡意程式盜取資料或建立後門。Sandbox 的技術有下列幾種型態：

(1) Virtual-Machine Sandbox

這一種型態最典型的是 Java[6]的 JVM。Microsoft 的.Net Platform[7]也採此類型架構。

(2) Application-Level Sandbox

這類的 Sandbox 通常是針對特定 Application 行為進行安全性的限制。因此以能否取得 Source Code 又分成兩類 Sandbox 技術。

若能取得 source code，可在 compile 或 link 的時候，加入額外的訊息或 link 到 Sandbox 所提供的 library，將來 Application 在執行的時候就能被保護。如 StackGuard[8]可以保護 Application 避免產生 buffer overflow 的問題，Microsoft 的 VisualStudio.Net[9] 也提供了類似的機制。

若無法取得 source code，則利用 binary rewriting[10][11]，或由作業系統所提供的 dynamic linking 機制，在 Application 執行前先載入 Sandbox，或於 Application 執行時，利用系統提供的 API 以 Sandbox 機制包覆。Detours 就是一種程式包覆機制[12]，其提供 Win32 平台的 API interceptor，可以利用所提供的 API 攔截並修改原來 Win32 API 的行為，或加入安全性的限制。在 Linux 與 Solaris 平台上則可利用 injectso[13]，其目的類似 Detours library，同樣也是在平台上提供攔截 System API 的功能。

(3) Kernel-Level Sandbox

Kernel-Level Sandbox 是將 Sandbox 轉變成

kernel 模組，此類 Sandbox 在 kernel mode 中執行，任何的 API 呼叫、或 Application 行為都可被隨時監視，並依據不同的安全性需求而設定相對應的安全條件。

目前多數作業系統的 kernel 都可以自行撰寫 kernel module，在 kernel 執行的時候隨著 kernel 一起執行，在 Win32 平台上是以 VxD 的方式存在，Linux 或 Solaris 上則是以 kernel module 的方式存在，如 GSWTK[14]同時提供 Win32、Linux、Solaris、FreeBSD 等 kernel-Level 的 sandbox 機制。

Kernel-Level 與 Application-level sandbox 之優缺點互異。kernel-level sandbox 不易被發覺，由於所有的 System API 呼叫都能被監視，因此也很難被移除或規避。但缺點是開發困難，由於又是在 kernel-mode 下所執行，如果安全條件未正確設定，將會嚴重影響系統效能。

2.2 Self Executable Content

如同軟體產權的保護，新的保護方法不久就有相對應的破解技術。在一個無法保證不被破解的環境下，必須有補救的機制，使得一旦發生破解行為時，可以在損失最少的情況下，提供補救的方法。如同軟體病毒的防治技術或軟體的 patch 也是這樣的觀念，一旦有新的病毒，或發現軟體本身在設計初未發現的 bug，必須提供新的病毒定義檔或提供 patch 來做適當的補救。Self execute content 就是基於這樣理念的設計。數位內容本身不在是被動、靜止的，而是一種可被執行的個體，一旦數位內容被發現被侵權行為時，可以馬上再發行新版本的數位內容以防止侵權範圍的擴大。

數位內容包含了可在某種平台上執行的執行格式，其執行碼包含數位內容所需要的撥放器，並且當數位內容被發行到使用者端後，可透過事先安裝的執行平台取得被使用的環境資訊，包括數位內容是如何被撥放、撥放平台的資訊、輸出平台資訊等。數位內容可依照這些資訊，自行判斷在此環境播放是否合法。一旦發生侵權行為，後續發行的數位內容將依此類資訊進行管控，禁止數位內容的再次播放。

不管是防毒軟體病毒碼的更新、或軟體的 patch，為了補救軟體發生不可預期的錯誤，都必須有及時反映的追蹤機制，以清楚問題發生點。例如各防毒軟體的病毒監視中心、微軟線上錯誤報告服務等，都有提供追蹤機制，以立即反應軟體所發生錯誤的時機及問題。因此在 self executable content 的概念中，也需要此類追蹤機制，以反制數位內容可能被侵權的行為。在所規劃的實作系統中將使用 image based 與 message based watermark 的訊息來提供追蹤機制，若以圖片為例，image based watermark 在圖片中藏入了發行單位與數位圖片本身的資訊，以 message based watermark 來進行 image based watermark 的管理與追蹤、並加入使用者的身份訊息。

2.3 Content Degradation

在影像處理的領域裡，有很多 degradation model 被提出來，為的是要改善影像經掃描後的品質[15][16]。在影印機、掃描器等光學掃描機器的使用過程中，每一次光學掃描過後的品質，一定會比原來的品質差。這些可以透過 degradation model 的處理，使得掃描過後影像更清楚，也讓之後的影像分析、辨識更為精確[17]。簡言之，這些應用在影像處理的 degradation model 是為了使較差的影像品質變得較好。

然而，在 DRM 系統裡，我們也可以利用這些 degradation model 來反向操作，把品質較好的數位影像內容變成較差的影像，以此來控

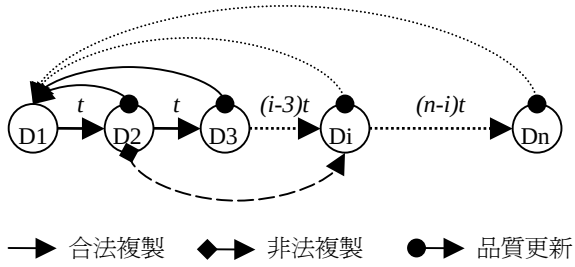
制數位內容的散播。亦即，在每一次複製行為發生時，就降低品質到某種程度，以至於在複製多次之後的品質已經低劣到不堪使用的狀況。

同樣的想法，不只可以應用在影像，任何的數位內容都可以用降低品質的方式來遏止數位內容的散播。

2.3.1 Time Schemes

可以利用時間當作品質劣化的因素。當消費者拿到數位內容後，在合法的使用裡，品質劣化的速度較為緩慢；相反的，如果是非法取得的數位內容，則很快地嚴重劣化，導致無法觀看或閱讀等。

如圖一所示，假設以同一種 model 做線性劣化的方式。D1 為原始的數位內容，經過一單位時間 *t* 之後，品質變為原來的 (n-1)/n，以 D2 表示，D3 以後以此類推。在合法的使用情



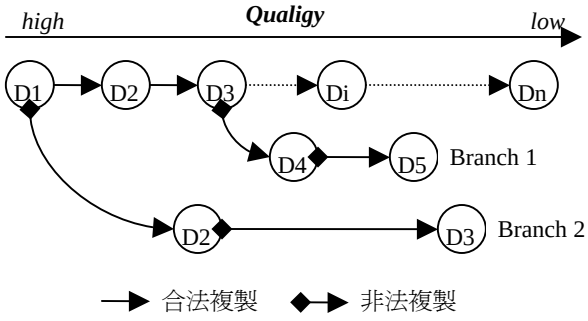
圖一 Content Degradation with Time Scheme

形下，數位內容的品質劣化的很慢。但是如果是非法使用，品質劣化的速度就很快，如圖一中非法複製的路徑。雖然實際的時間是介於 *t* 到 *2t* 之間，但是，品質確是如同經過 *(i-3)t* 的時間一樣。

2.3.2 Copy Scheme

除了時間的做法以外，也可以使用不同的 model、不同劣化的程度來造成品質劣化程度的不同。

如圖二，Di 為經過 i-1 次複製後的數位內容，如果是非法複製，則劣化的程度會因為不同的 model 而產生不同的品質。圖二中，Branch 1 的 D3, D4, D4 與 Branch 2 的 D1, D2, D3 皆為非法複製的路徑，然而 Branch 1 的 D5，也就是經過四次複製的數位內容可能會比 Branch 2 中，才經過三次複製的數位內容品質好一點。如此可以增加品質的混亂程度。



圖二 Content Degradation with Copy Scheme

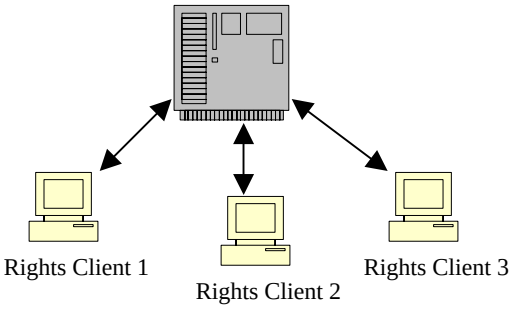
3. 內容保護平台系統架構

DRM 系統可以分成三大部份：(1) 數位內容的製造。(2) 數位內容的散播。(3) 數位內容的運用。

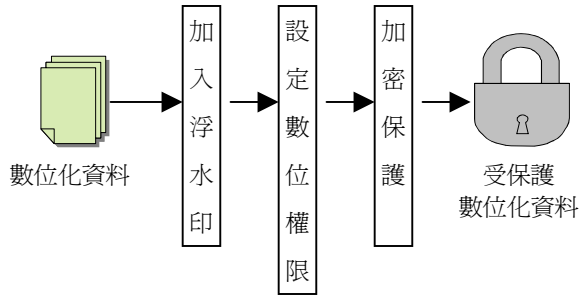
3.1 數位內容的製造

針對目前數位內容提供者的建議，整個數位產權管理平台採用分散式的架構，各數位內容提供者都有專屬平台來完成有關數位化內容的保護與權限控制等步驟，如圖三。而在每一數位內容提供者的 Rights Client 則會進行以下的流程，如圖四。

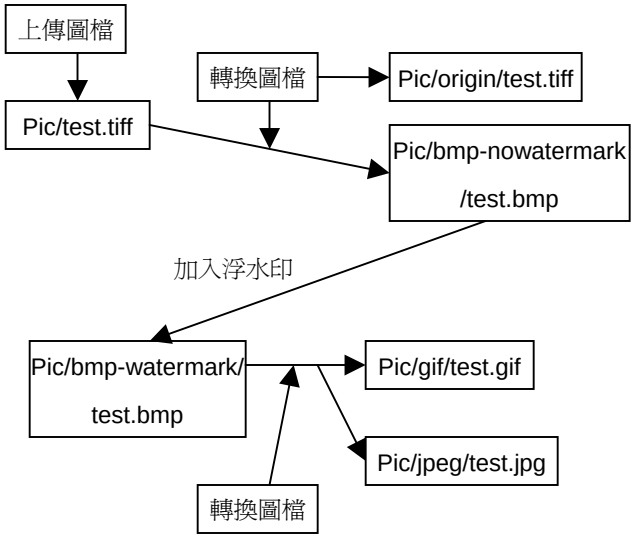
在內容提供者製作數位內容之後為了未來的舉證以及內容保護作用，必須先為數位內容加上浮水印後再繼續後續的處理程序。圖五是加上浮水印的流程。



圖三



圖四 數位內容設定權限及加密



圖五 加入浮水印

在數位內容加入浮水印之後，數位內容提供者則可以透過 Rights Client 來設定數位內容的相關權限，此數位權限可限制以下的使用範圍：備份、複製、刪除、編輯、出借、播放（使用）、列印、讀取、交換等。透過 Rights Client 設定完數位內容相關的使用權限之後，Rights Client 即產生符合 XrML 2.0 規範的 XML 檔案。一個符合 XrML 2.0 規格的文件會如表一所列檔案。

表一 XrML Licence file

```
<license>
       <grant>
              <cx:play/>
              <cx:digitalWork>
              </cx:digitalWork>
              <validityInterval>
                     <notAfter>
                     2002/07/01
                     </notAfter>
              </validityInterval>
       </grant>
</license>
```

在產生 XrML 數位內容敘述後，Rights Client 將記錄以下資訊在本地端的資料庫中：(a)原始數位內容、(b)入浮水印數位內容、(c)格式轉換內容、(d)浮水印解密鑰、(e)數位內容權限描述、(f)圖檔名稱。同時把除了數位內容外的資訊（大部分都是文字檔），透過網路傳輸到 Rights Center，以便將來作為數位資料交換權限使用。

3.2 數位內容的散播與運用（播放）

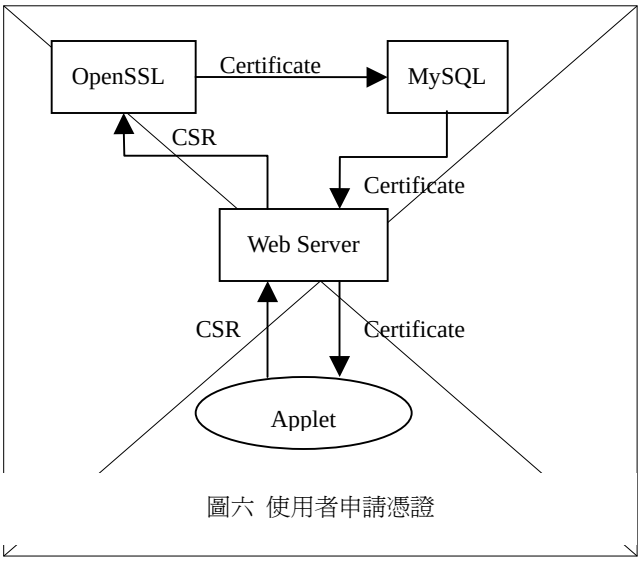
我們使用 WEB Server 作為數位內容的窗口，同時利用 Java Applet 技術，讓一般使用者能方便地透過瀏覽器來取得並運用數位內容。

使用者在取得數位內容之前，必須先通過認證，證明此使用者有權利使用。通過認證之後，才能授權此使用者有該內容相關使用權限。

3.2.1 Authentication

我們使用公開金鑰系統(PKI)來進行使用者認證。首先，如圖六，使用者在使用之前必須先申請憑證(Certificate)。

Applet 將於 Client 端產生一組 KeyPair，並以 Java KeyStore (JKS) 的格式儲存在 Client 端。接著產生 CSR，回送給 Server，利用 OpenSSL 進行 Certificate 加簽，再儲存於 MySQL 資料庫。完成後，Server 將



圖六 使用者申請憑證

Certificate 送回 Client 端，儲存於 Client 端。

使用者登入時，即可以用申請過的 Certificate 來做身份的認證。在登入的時候，必須輸入帳號及密碼。在此，我們以 E-Mail 帳號來做識別。(可以把 E-Mail 的資料放在 Certificate 中) 以 E-Mail 帳號做為索引，以讓 Server 取得該使用者相對應的 Certificate，進而取得該使用者的 Public Key。以 Challenge / Response 的檢查方式來確認使用者擁有與 Public Key 相對應的 Private Key。流程如下：

Step 1: 輸入 E-mail 及密碼

Step 2: Server 根據所輸入的 E-mail 從資料庫中取得 Certificate。

Step 3: Server 產生一亂數，並用此 Certificate 中的 Public Key 加密此亂數。

Step 4: Server 將加密過後的亂數送給 Client。

Step 5: Client Applet 使用密碼取得 JKS 內的 Private Key。

Step 6: Client Applet 把從 Server 來的訊息解開，並傳送回 Server。

Step 7: Server 驗證此亂數數值，是否與一開始所產生的相對應。若正確，Server 確認為合法使用者。

3.2.2 Authorization

認證後，就能取得授權的數位內容。Client 取得授權有兩部份，一部份是數位作品(Digital Work)；另一部份是對數位內容所擁有的權利(License)，以 XrML 格式描述。

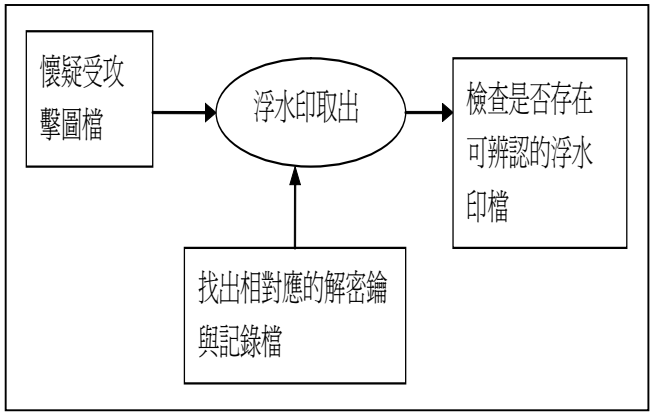
由於非對稱性的加解密系統，必須花費較多的時間計算。通常針對訊息量較少的資訊做加解密，例如登入資料、信用卡號碼等，而數位作品本身，如圖檔、聲音檔，則使用對稱性加解密方式處理。

Server 將動態產生 Random Key，並用此 Random Key 加密數位內容。之後，再把 Key 與 License 以使用者的 Public Key 加密。最後再將兩種加密後的資料合在一起，送至 Client 端。

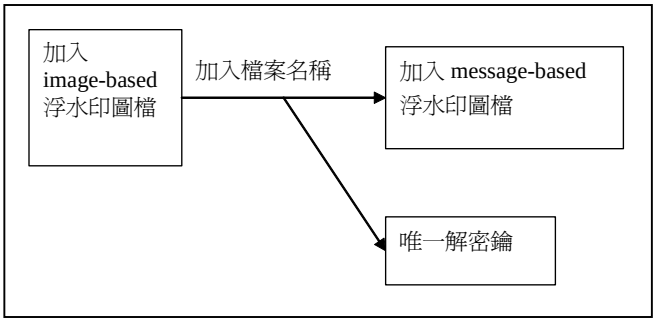
當 Client Applet 收到加密後的數位內容，將分成兩部份處理：一部份是加密後的 License，另一部份是加密後的數位內容。首先，必須先用 Private Key 解出 License 部份內容，同時取得用來解數位內容的 Random Key，再以 Random Key 解出數位內容。

3.3 Rights Center 與 client reader 互動、侵權管理

我們所使用的 image-based 浮水印機制會產生：(1) 浮水印解密鑰、(2)浮水印記錄檔，這兩個檔案都是將來要取出浮水印所必須的檔案。從數位內容中取出浮水印的程序為：



若只有少量的數位內容，可以用手動方式找出數位內容所對應的：(1) 浮水印解密鑰、(2)浮水印記錄檔。但對於大量圖檔，人力比對勢必不可行。為了解決此問題與有效地自動產權追蹤，我們使用 message-based 的浮水印機制，利用此機制將圖檔名稱藏入此圖中，利用這樣的關係以便在將來能找出浮水印與解密鑰之間的關連。以下為加入 message-based 浮水印的程序。



目前在我們的機制中都是由同一密鑰來處理 message-based 的浮水印

因此，每一個欲保護的數位內容都會有兩種不同的浮水印機制：(1)image-based、(2)message-based。Image-based 浮水印用來藏舉證用途使用的額外圖檔，message-based 用來記錄取出 image-based 所需要的資訊。

在加完浮水印以及設定數位內容權限之後，如果這一份數位內容是將發行的版本，仍必須經過加密與封裝的步驟。利用數位內容提供者產生的 key 將數位內容加密，防止數位內容在網路傳輸中被擷取，並將加密保護的數位內容與 XrML 數位權限敘述封裝，封裝過的數位內容將被視為可以發行的最基本的單位，而使用者的閱讀器也必須能夠支援此種格式，目前我們也開發了一個實驗用的閱讀器，能夠解開並讀取以 XrML 所描述的權限資訊。

在 Rights Client 與 Rights Center 之間的傳輸，目前支援兩種協定的傳輸：HTTP 與 FTP，在 Rights Client 完成對數位內容的保護動作後，可以選擇是否將數位內容傳送到 Rights Center 來做發行的動作。在 Rights Client 與 Rights Center 之間的實體檔案傳輸時的安全性，目前都是使用 Openssl 來實作 HTTP 與 FTP 安全傳輸。

在 Rights Center 會把要發行的數位內容轉換成 self executable content，再由使用者下載此類可執行內容。使用者在使用此類數位內容之前，必須先安裝一個提供此數位內容執行的平台。目前我們所設計的 self executable content 是使用修改過的 windows PE 格式，這種格式無法直接被 windows 作業系統的 loader 執行，只能在我們所提供的執行環境中執行。不提供直接執行檔是考慮到執行檔可能被惡意修改、以避免數位產權保護機制失效，或可能被病毒感染、甚而被加上後門等足以危害使用者系統的惡意程式。

在所規劃的實作中，安全執行環境與平台是使用 detours，運用其 sandbox 軟體包裹技術。在此安全執行環境中，可攔截到執行數位內容所必須使用的 windows system API 與其參數，因此能限制可執行數位內容的環境，包括限制在某個目錄下執行、或只存取某些系統資源等，只要能用 windows system API 所描述的行為都可被監視攔截。即使可執行數位內容在

傳輸過程中被修改，使用者的執行環境仍被保護，並能認證數位內容的來源是否安全。

4. 結論

本論文探討數位內容保護平台的設計，將保護的層面分成內容使用者(client) 與內容保護平台(server)兩個主要角色，在保護層面同時採行資料虛擬區隔與控制流程隔離兩方面的限制，在權限控管上更具彈性。

我們將內容與權限控管封裝為可執行內容。這些內容在安全執行環境中執行，取得使用者系統的資訊，認證使用者的身份後判斷是否數位內容本身可以在此環境中執行，並使用 XrML 來描述數位內容本身的使用行為模式，如執行次數、可執行的時間限制等。XrML 與 SandBox 技術相輔相成，在不同的層次限制可以存取的範圍，以提供數位內容及使用者雙方面的保護。

5. 參考資料

[1] 劉晉豐,〈從網路科技看著作權法之重製權〉書苑季刊 49 期（90 年 07 月）第 23－37 頁 http://public.ptl.edu.tw/publish/suyan/49/23.htm

[2] 賴文智，〈著作權交易型態演變〉益思科技法律事務所

[3] 葉乃瑋・賴文智，〈數位圖書館與著作權〉益思科技法律事務http://www.is-law.com/專文發表/IPR/數位圖書館與著作權.htm

[4] 中華民國著作權法第一條

[5] Emnest Miller and Joan Feigenbaum. Taking the Copy Out of Copyright. Security and Privacy in Digital Rights Management, ACM CCS-8 Workshop DRM 2001, Philadelphia, PA, USA, November 5, 2001.

[6] http://java.sun.com/

[7] http://www.microsoft.com/net/

[8] http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/

[9] http://msdn.microsoft.com/vstudio/

[10] Romer, Ted, Geoff Voelker, Dennis Lee, Alec Wolman, Wayne Wong, Hank Levy, Brian Bershad, and J. Bradley Chen. Instrumentation and Optimization of Win32/Intel Executables Using Etch. *Proceedings of the USENIX Windows NT Workshop 1997*, pp. 1-7. Seattle, WA, August 1997. USENIX

[11] Larus, James R. and Eric Schnarr. EEL: Machine-Independent Executable Editing. *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 291-300. La Jolla, CA, June 1995

[12] http://research.microsoft.com/sn/detours/

[13] http://www.securereality.com.au/

[14] http://opensource.nailabs.com/wrappers/

[15] Henry S. Baird. Document Image Defect Models and Their Uses, Proceeding, IAPR 2nd Int'l Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan, 1993

[16] Henry S. Baird. The State of the Art of Document Image Degradation Modeling. Xerox Palo Alto Research Center.

[17] T. Kanungo, R. Haralick, and I. Phillips. Global and Local Document Degradation Models, Proceedings of the Second International Conference on Document Analysis and Recognition ICDAR-93, Tsukuba, Japan, 1993.