OpenDReaMS: 數位產權交換、管理系統

林宗伯 洪偉能 黃世昆
中央研究院資訊所 中央研究院資訊所 中央研究院資訊所
lancelot@iis.sinica.edu.tw wnhung@iis.sinica.edu.tw skhuang@iis.sinica.edu.tw

ABSTRACT

我們所發展之 OpenDReaMS (Open Digital Rights exchange and Management System) 系統, 利用 wrapper 技術, 整合 MS Media、Office, Acrobat Reader 等現行流 通 DRM 管理系統, 達成三層次的數位產權交換、控管機制。第一層次是網路自由交換、強制單一產權 的單一流通性, 適合於 Peer-to-peer 網路環境應用。第二層次是 Wrapper 存取原則控制, 達成 控制 agent 與內容瀏覽環境的互不信任管理。第三層次則是進行授權 (license) 管理, 將相關授權原則經由第 一層與第二層的解譯與強制實行後, 轉爲流通 DRM 系統 (如 Acrobat reader) 的管理指令 (directive)。經由此三層次的協調控制, OpenDReaMS 將可應用於 P2P 環境, 並與現行流通 DRM 系統相容並行。

在 OpenDReaMS 伺服器端的部份可以應用於一般以 Client-Server 爲基礎的網頁架構上, 並在不影響原有架構下, 提供需要嚴謹數位產權保護之數位內容發佈系統。此系統也可以應用於數位圖書館、數位博物館, 提供數位內容提供者一個安全的架構, 讓具有價值的數位內容可以安全的以網頁的型態發佈, 使有被授權的使用者可以使用此數位內容, 而不用擔心發布之數位內容可能被盜用的情況。OpenDReaMS 伺服器端的延伸架構也可以提供數位內容提供者獨立產權授權伺服器, 並且與獨立之付費機制結合。提供數位內容提供者單純的環境、以專注於數位內容的製作與發佈。

1. INTRODUCTION

現行針對內容敏感的網頁保護, 大部分的解決方案都僅止於起始階段的身分認證[5], 只要網頁瀏覽者通過身分認證後, 網頁伺服器端或網頁發行者就無法對其發佈的網頁做任何的控管, 其問題在於目前的瀏覽器架構會把網頁內容存在本地端、以作爲下一次網頁存取的快取之用。因此, 網頁瀏覽者在通過網頁伺服器的認證之後, 即會在本地端取得此網頁的一份複製, 因此也間接取得此網頁的所有使用權限。

在一般以自由發行爲目的之網頁架構下, 此行爲並不會造成任何困擾, 本地端快取更可以加快使用者瀏覽網頁的效果。但是, 在 HTML 格式成爲越來越多文件格式的選擇時, 許多公司或組織內部機密文件採用 HTML 格式保存, 文件瀏覽架構也會以 Intranet 網頁瀏覽的形式呈現, 這種使用者通過認證後便可以郵寄、列印、修改此文件的架構, 顯然無法確實保護內部機密文件。

此外, 這樣的架構也限制了發佈在網頁上的數位內容的價值性。由於網頁瀏覽者會在本地端有一份此網頁的複製, 對於數位容提供者, 如數位博物館、網路電子書等, 經由網頁發佈的數位內容管道便無法安全地保護其數位產權, 大部分的內容提供者僅利用網頁型態發佈數位內容的索引或以預覽格式呈現, 使用者若要取得有保存價值之數位內容時, 仍必須透過其他的發行管道。

因此,OpenDReaMS 計畫在伺服器端的目的即爲提出一個在網頁伺服器端保護之架構, 讓網頁發行者可以在使用者通過一般的身分認證, 取得網頁內容後, 可以進一步的限制使用者使用此網頁內容的權限, 以保護數位內容以網頁型態呈現時之數位產權。

目前 DRM 機制在業界中沒有一定的標準。因此多數內容開發廠商會依據其數位內容的格式、特定的授權機制、數位內容的散佈方式等各種因素來發展符合自己需求的 DRM 系統, 以下稱 COTS DRM。

OpenDReaMS 的目的並不在於開發一個新的 DRM 機制, 而在於整合現有的 COTS DRM 系統。透過此系統, 數位內容不再侷限於原有 COTS DRM 的內容發行方式, 而能隨易地透過網路來散佈, 包括 P2P, WWW 等。在播放平台端, OpenDReaMS 加強原本 COTS DRM 的權限管理機制, 讓數位內容的散佈更靈活、權限管理更安全。

對數位內容開發者而言, 此系統增加對數位內容的保護、保障開發者的利益; 對數位內容消費者而言, 增加使用上的便利性。

2. OPENDREAMS ARCHITECTURE

目前主流的 COTS DRM 方法仍屬於 Client-Server 的架構。內容提供者扮演 Server 的角色; 消費者則爲 Client, 若要存取數位內容時必須連上網路, 線上取得授權後才能使用。

圖1 爲 OpenDReaMS 的主體以及與 COTS DRM 的整合架構。由圖所示 OpenDReaMS 是介於原本 COST DRM 的 Server 與 Client 之間, 但獨立於 COTS DRM 外。OpenDReaMS 的整合, 對 COTS DRM 並不需做任何改變, 並提供自訂管理數位內容產權的方法。

主體架構可分爲對 Server (Content Provider) 與對 Client (Content Consumer) 的整合。在 Server 端, 可整合各不同格式的數位內容及權限描述檔, 加以一般化 (Generalize) 爲被保護的數位內容, 及權限描述檔。在 Client 端, 根據數位內容的格式, 判定所屬 COTS DRM, 再將此內容個別化 (Specialize) 成符合其格式的檔案。

整個架構又可分爲三個部份:

- **Proxy:** 數位內容提供者所提供的原始數位內容, 在 Proxy 經過加密處理成爲受到保護的數位內容檔案 (Rights Protected Digital Content)。
- **Authorization:** 數位內容提供者所給的權限設定, 在 Authorization 子系統中加以處理成權限檔案。
- **Wrapper:** Wrapper 將被保護的數位內容解密之後, 再根據權限檔案所描述的設定、產生符合 COTS DRM 格式的檔案, 再由對應的播放平台來播放。

3. INTEGRATION

爲了準確地管理數位內容, 必須能針對不同的使用者賦予不同的使用權限; 甚至對同一位使用者, 在不同的授權條件下, 所賦予的權限亦有所不同, 如圖2。因此在整合時, 必須把數位內容本身與權限控制分開處理。

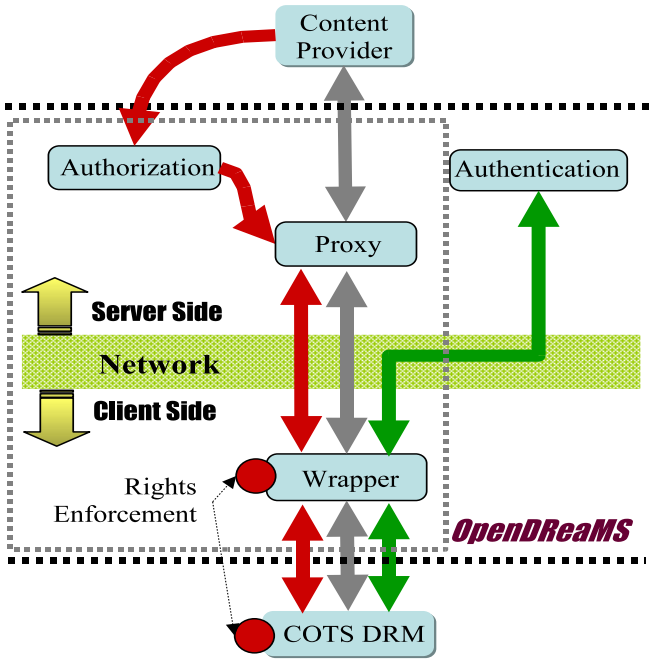


圖 1: **The architecture of OpenDReaMS integration with COTS DRM.**

一個數位內容檔案, 依據不同的權限, 會需要不同的權限檔案來描述所授予的權限。數位內容本身會加密保護, 無法單獨被開啓。必須配合權限檔案, 在權限受到控制的狀況下才能解密開啓數位內容。這樣的優點是: 一、被保護的數位內容可以透過各種不同的管道來分散、取得, 例如透過 P2P 網路來散佈。二、當兩位使用者擁有同一份數位內容而要相互交換權限時, 只要交換權限檔案即可, 不必交換數位內容本身。特別是當數位內容檔案很大時, 這樣的做法能提高交換的效率, 也減少網路流量的負荷。

因此在 Client 與 Server 端, 都必須處理權限檔案與數位內容兩部份。

3.1 Client Side Integration

在 Client 端, 任何有 DRM 機制的數位內容播放平台, 一定會進行權限控管的動作。這些動作包括**身份確認**, 確定使用者能否使用特定數位內容; **授權**, 根據不同的條件(例如付費的多寡) 給予不同的權限設定; **權限限制**, 播放平台根據權限, 允許特定動作的執行(例如儲存、列印、播放等)。

不同的數位內容格式其權限管理機制互異。以音樂及電子文件爲例: 許多線上音樂下載服務都採用 Windows Media DRM[8] 機制來做音樂檔案的權限管

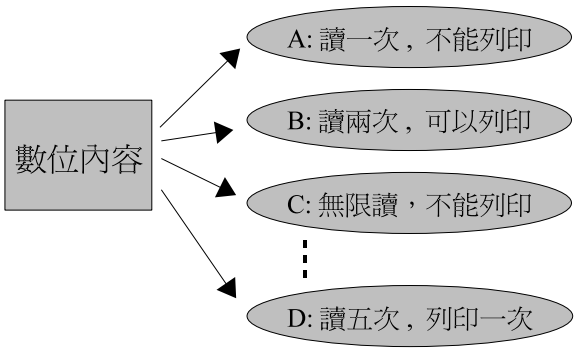


圖 2: One Digital Content maps to multiple License File.

理。在電子文件方面,Adobe 公司的 Content Server[2] 配合 Acrobat Reader 外掛 DRM Plug-in 來做權限控管。這兩種 COTS DRM 系統, 有不同的權限管理特色。

4. RIGHTS ENFORCEMENT

DRM 系統必須確保權限被正確的套用執行。因此Client 端的整合重點是將 OpenDReaMS 的權限檔案與數位內容, 及現有的 COTS DRM 有效的整合, 並增加控管能力。我們運用 Detours[9] 軟體包裹(Software Wrap) 技術, 來攔截存取系統資源相關的 WIN32 API。以此擴大數位內容播放平台對權限的管理能力。

4.1 Architecture

圖3(a) 爲整合的架構圖。當 Client 端取得授權檔案 (License) 與加密內容 (Encrypted Content) 時,License/Content Dispatcher 首先把內容解密。再根據內容的格式啓動相對應的 Wrapper, 並同時傳送授權檔案中的 Rights 與解密後的內容, 再由 Wrapper 與 COTS DRM 處理權限控制的部份。

由於 COTS DRM 的實際作法都不同, 因此必須針對每個 COTS DRM 設計可與之整合的 Wrapper, 並以數位內容檔案的副檔名來命名, 如 PDF Wrapper、WMA Wrapper 等。

圖3(b) 所示即爲 Wrapper 子系統。Rights Enforcer 會根據 Rights 所描述的權限, 攔截對應的 WIN32 API。Content Regeneator 則整合解密後的內容與 COTS DRM 所支援的權限控制項目 (Specific Rights), 重新製作出 COTS DRM 能接受的數位內容。最後, Wrapper 會啓動內容播放/閱覽器 (Content Player) 讀取新產生的內容。如此即使 COTS DRM 被破解, 仍有 Wrapper 維持權限管理的角色。

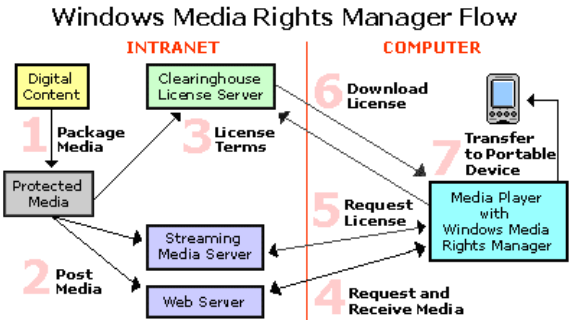


圖 4: Windows Media Rights Manager Flow.

5. CLIENT SIDE INTEGRATION

Client 端目前的實作重點以整合 Windows Media Player 及 Acrobat Reader 爲主。其分別爲主要音樂及電子文件播放/閱覽平台, 因此 OpenDReaMS Client 以此爲整合對象。以下分別說明兩者的權限管理架構及整合細節。其他數位內容播放/閱覽平台也在 OpenDReaMS 的整合計畫中。

5.1 Windows Media Player

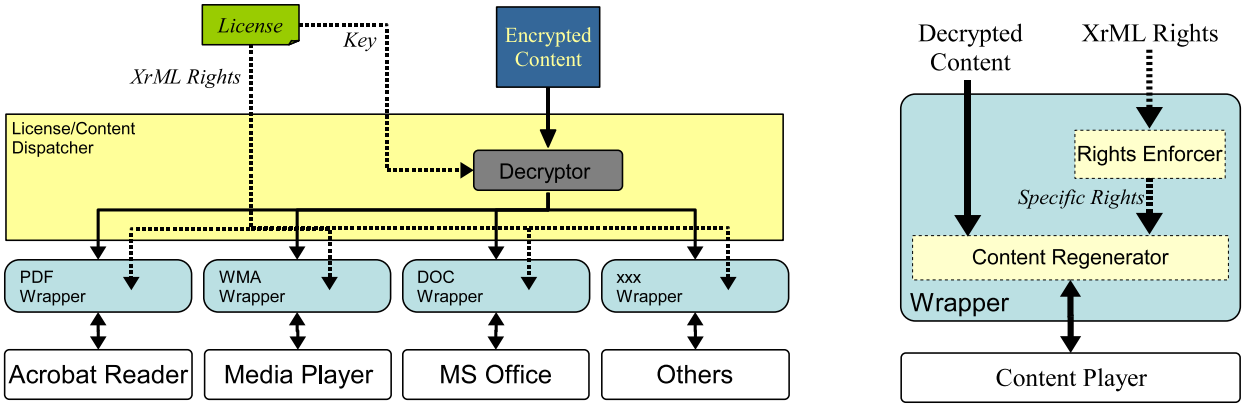
Windows Media DRM 爲 Windows Media 系列的權限管理機制, 目前最新的版本爲 Windows Media DRM 10, 其權限管理的流程如圖4[1]。整個系統又可以分爲三個角色, 如圖5, Content Provider、License Issuer 及 Consumer's Computer。由 Content Provider 製作 Media File。當 Consumer 要播放此 Media File 時, 必須到 License Issuer 取得 License 才能播放。

除了 Consumer 在電腦上所執行的 Media Player 外, 其他兩個角色必須由 OpenDReaMS 擔任。亦即對 Media Player 而言, License Issuer 和 Content Provider 的功用仍存在, 而與它溝通的對象卻是 OpenDReaMS 如圖3(b)。

5.1.1 Windows Media DRM Overview

圖5 中, Content Provider 在製作 Media File 時, 會指定一個 KeyID, 並亂數產生 seed。以 KeyID 與 seed 爲輸入參數, 使用特定演算法產生 Key(步驟1), 並用此 Key 加密 Media File(步驟2)。將加密過的 Media File 與 Header 包裝成一個新的 Media File 後傳送 (步驟3)。其中的 Header 包括專

[1]資料來源:http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx, 此處有各步驟詳細說明。



(a) Rights Enforcement Block Diagram (b) Content Player Wrapper

圖 3: Rights Enforcement Architecture

輯名稱、作者、年份等資訊。其中主要的欄位爲 LicenseAcqURL(License Acquisition URL)。

當 Consumer 用 Media Player 播放此檔案時, 若含有權限管理資料, Media Player 識別其爲受保護的檔案, 即啓動 DRM 機制 (步驟4)。Rights Manager 會以 KeyID 爲識別碼, 到本機的 License Store 查看是否有對應的 License。如果沒有, 就發出 License Request 到 LicenseAcqURL 所指定的網頁取得 License(步驟6)。License Issuer 根據訊息中的 KeyID 與從 Content Provider 所得到的 seed, 以同樣的演算法產生 Key。根據消費者所持有的憑證 (例如付費憑證。但付費機制並不在此系統內, 須透過額外方式, 如線上小額付費機制), 給予相對應的 Rights。再把 Key、Rights 等資訊放入 License 中。最後 License Issuer 會把加簽 (Sign) 過 (步驟8) 的 License 回覆給 Rights Manager(步驟9)。

當 Rights Manager 拿到 License 後, 會先存到 License Store(步驟10), 以供下次播放相同檔案之用。最後 Media Player 會從 Rights Manager 取回 License, 依據其中所設定的 Rights 限制播放此 Media File 的動作。

5.1.2 *Integration with Media Player*

若要取代 Content Provider 與 License Issuer 的角色, 必須找出 Wrapper 與其對應關係。參考圖3(b)與圖5, 涵蓋 Wrapper 必須完成與實作的**功能**與攔截**訊息**。

在功能方面, Content Provider 所產生的 Media File 是 Dispatcher 輸入 Wrapper 的 Decrypted

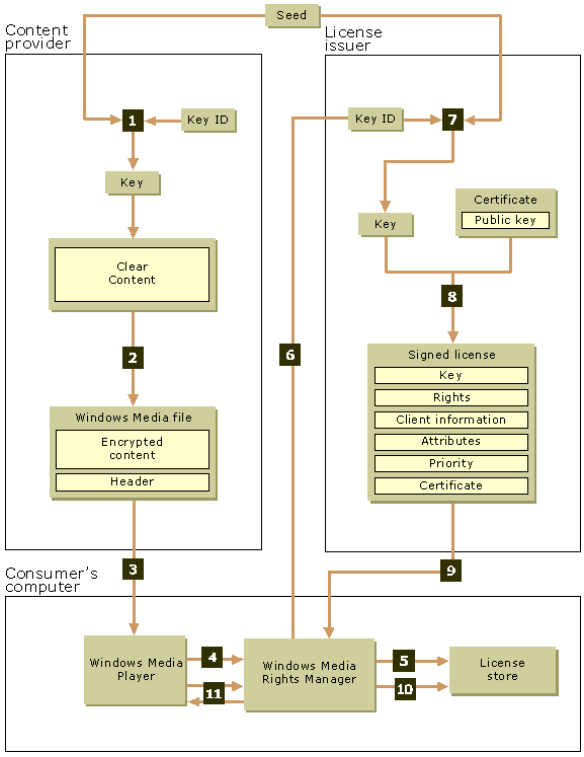


圖 5: Windows Media Rights Manager Overview.

Content。因此主要的功能包含於 License Issuer 子系統。此系統具下列功能: (1) 從 License Request 訊息中擷取所有資訊。(2) 產生 Key。(3) 設定 Rights 資訊。(4) 製作 License。(5)Sign。而必須攔截的訊息包含於圖5 的步驟6、9中, 分別爲 License Request 及 Response。

若要產生 Key 及設定 Rights 等功能, 可利用 Microsoft 所提供 Media Rights Manager 的 SDK, 能夠與 Media Player 完全相容。

訊息攔截實作方面, 步驟6包括:(1) 先連線。連線成功之後才是 (2) 傳送訊息。連線到某個網址的 WIN32 API 爲 `connect`, 訊息的傳送是 `send`。步驟9爲資料接收, 其 WIN32 API 爲 `WSARecv`。

當 Media Player 被 wrap 時, Wrapper 所攔截的 WIN32 API 會完整的處理所有的訊息。我們以 LicenseAcqURL 來識別所開啓的 Media File, 決定由 OpenDReaMS 或經由原來的 Media Rights DRM 程序處理。假設 U 爲此一特定 URL(這個網址不存在), 當 `connect` 連線到 U 時所 return 的 socket 爲 S 。則在 `sned` 及 `WSARecv` 中, 我們只要判斷連線的 socket 是否爲 S 即可。

而當 Media Player 單獨執行, 且開啓此 Media File (假設能從 Dispatcher 與 Wrapper 中間將 Decypted Content 攔截下來)。在沒有被 wrap 的情況下, Media Player 會連線到 U , 而因爲 U 不存在, 所以無法取得 License, 也就無法開啓此 Media File。

5.2 Acrobat Reader

Acrobat Reader 已具有權限控制的能力。如圖6 所示爲 Acrobat Reader 中的 Security Properities。其中有9項限制項目, 控制此文件的存取動作。此針對單一文件所做的列印、修改、儲存等控制, 具有基本 DRM 能力, 屬於 Acrobat Security 的部份功能。

同時 Adobe 也提供完整的 DRM 系統。在 Server 端 Adobe Content Server 3 提供完整的 數位內容製作、發行及權限管理系統; 而 Client 端則是 Acrobat Reader 並具有 DRM Plug-in。

5.2.1 Integration with Acrobat Reader

圖 **6: Acrobat Security Control.**

如5.1 所提, 爲了整合 Adobe Content Server 的 DRM 機制, 我們嘗試以 Acrobat Security 加上 Plug-in 的方式來實作權限管理。FileOpen System 公司[4] 就是以此方式, 爲 Acrobat Reader 提供 DRM 的解決方案。

我們仍使用 Software Wrapper 的方式來實作。當 Acrobat Reader 讀取檔案時, 將被 Wrapper 攔截, 判定是否經由 OpenDReaMS 進行保護。Wrapper 中的 Content Regenerator 會 根據 Rights 的設定加入所支援的 Rights 項目, 再傳回 Acrobat Reader。由於 Acrobat Security 所支援的權限項目較少, 因此 Wrapper 必須處理其他項目。例如: 起始與截止日期、使用人、讀取次數、列印次數等。

與存取 PDF 檔案有關是 File 相關的 WIN32 API。其中以 `CreateFile` 、`ReadFile` 、`ReadFile` 最爲直接相關。所有的限制條件判斷都經由上述 API 處理。

6. SERVER SIDE IMPLEMENTATION

OpenDReaMS 伺服器端的實作分成：網頁代理伺服器與 IE 瀏覽器的 Plug-In, 其架構圖如下7。

(1) 首先, 使用者依照正常程序瀏覽網頁, 使用者瀏覽網頁的行爲不需改變。(2) 在一個 URL request 發出後, 透過 OpenDReaMS Proxy 轉交給後端之原有網頁伺服器, 網頁伺服器依照所收到的 URL request 回覆適當的網頁內容, 將其送回 OpenDReaMS Proxy。(3)OpenDReaMS Proxy 將此網頁轉換成封裝文件格式, 如 mht 或 pdf 等文件格式, 並將此

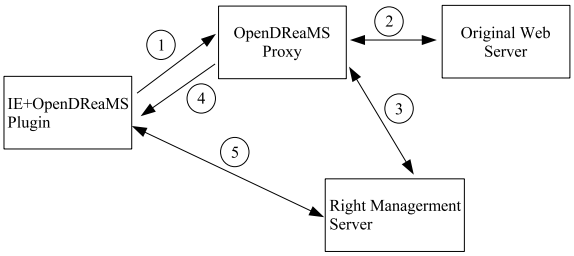


圖 **7: OpenDReaMS Proxy** 流程圖

封裝文件加密, 同時於 Rights Management Server 取得相對於此使用者之「基本數位產權描述」, 將此使用者對應此網頁之「基本產權描述」附加在以加密之封裝文件格式之外。(4)OpenDReaMS Proxy 隨即將此文件回傳給使用者。(5) 使用者收到此文件後透過 OpenDReaMS Plug-In 解譯「基本產權描述」, 利用此「基本產權描述」回傳 Rights Management Server 驗證使用者身份與權限, 以取得最終之「使用者產權描述」。

OpenDReaMS Plug-In 會依據「使用者產權描述」中所記錄的數位產權限制使用者可以對此文件所進行的動作, 包括列印、修改、轉寄等。

由於 OpenDReaMS 伺服器端架構並未影響原有使用者之行爲與網頁伺服器之架構, 因此, 在 (步驟1) 與 (步驟2) 時依舊可以使用原有網頁伺服器所提供之身分認證方法, 來初步驗證使用者是否可以瀏覽此網頁。

在目前 OpenDReaMS Proxy 的實作中使用 MHTML 格式作爲封裝文件格式。MHTML 代表 MIME Encapsulation of Aggregate HTML, 此標準定義於 RFC 2557[6], 原本用於電子郵件訊息本文, 以傳送 HTML 內容, 並使用MIME 結構、將 HTML 內容中包含的所有部分儲存成單一檔案。利用此標準就可以在電子郵件本文中使用 HTML 格式傳輸。OpenDReaMS Proxy 利用此標準, 將使用者端所發出的 URL request 轉交後端網頁伺服器後, 將網頁伺服器所回傳的所有網頁內容包裝成一個單獨的 MHTML 檔案。以此單一的 MHTML 檔案爲基準, 向 Rights Management Server 要求相對應與目前使用者之「基本產權描述」。對於使用者來說每一個 URL link 都如同對應到書本的每一頁,OpenDReaMS Proxy 會將每一個 URL link 包裝成 MHTML 再送到使用者端。

在目前的實作中 OpenDReaMS Proxy 雖然是以 MHTML 格式作爲封裝格式, 但是也可以使用其他的封裝格式, 例如 PDF。透過將網頁轉換成 PDF[7] 格式, 再送到使用者端, 就可以跟PDF 所能提供的 Rights Management 服務結合, 使用現有之 PDF Reader 來限制使用者端的行爲。

「基本產權描述」與「使用者產權描述」在實體上都是用 XrML[3] 格式來描述,XrML 起源於1996年在 Xerox PARC 實驗室所設計的 DPRL 語言 (Digital Property Rights Language),XrML 也是標準的 XML 語法。XrML 希望在數位的世界中, 能夠提供一個標準的描述語言, 用來描述數位化資料的權限限制, 並且也能夠規範在數位世界中的交易行爲。

「基本產權描述」是由網頁發行者所產生的, 裡面紀錄了網頁發行者的資料, 與網頁發行者希望此網頁的使用權限, 並記錄了 Rights Management Server 的資訊。在 OpenDReaMS Proxy 上是動態去詢問 Rights Management Server 所產生的「基本產權描述」, 也就是數位內容本身與數位產權的描述是分開存在的。其它如在網頁電子書出版的應用上, 也可以將事先產生的「基本產權描述」與數位內容包裝成單一的檔案, 存放在 URL 所指的位址中, 讓使用者透過 URL 取得有產權描述的整合檔案。

OpenDReaMS Proxy 會將加密後之網頁封裝格式與「基本產權描述」, 結合成爲單一檔案後, 回傳給使用者。使用者在獲得「基本產權描述」與數位內容後, 藉著「基本產權描述」中所記錄的 Rights Management Server 資訊, 將「基本產權描述」送回給 Rights Management Server 作爲第二次身分驗證之用, 以防止使用者僞造攻擊。Rights Management Server 驗證無誤後, 根據此「基本產權描述」, 對照儲存在 Rights Management Serve 上所記錄的此使用者的權限, 依此兩者之交集產生「使用者產權描述」。「使用者產權描述」敍述了讓 OpenDReaMS IE Plug-In 所使用的數位權限描述與解密鑰,OpenDReaMS IE Plug-In[1] 依此解密鑰解開編碼保護之數位內容, 並依照數位權限描述, 限制使用者在瀏覽此網頁時之權限, 例如可否轉寄、列印、複製等。

由於目前的OpenDReaMS Proxy 是將網頁轉換成爲 MHTML 格式, 並且加上權限描述,IE 本身並無法支援這樣的格式, 因此我們也必須提供 IE Plug-In

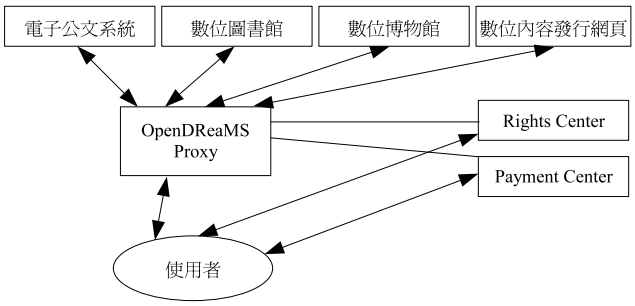


圖 8: **OpenDReaMS** 系統應用

來讓使用者可方便地存取此類網頁。如之前所提, 也可以將網頁內容轉換成其他已經現有之 Digital Rights Management 架構之文件格式, 例如 PDF 等。透過現有之 Digital Rights Management 架構所提供的服務, 與相對應的閱讀器, 也可以提供網頁提供者限制使用者相關的權限, 並可以透過 OpenDReaMS 客戶端包裹器, 提供現有之 Digital Rights Management 架構所無法限制之權限。

7. CONCLUSION

在 OpenDReaMS 伺服器端的部份可以應用於一般以 Client-Server 爲基礎的網頁架構上, 並在不影響原有架構下, 提供需要嚴謹數位產權保護之數位內容發佈系統。尤其對於數位內容的產權控管需要嚴密保護的公司或政府組織內部的公文發佈系統, 可以提供數位公文發佈於網頁型態, 並且可控制使用者端對此發布的公文所具的權限, 同時可控制使用者能否讀取、修改、列印、轉寄等動作。

OpenDReaMS也可以應用於數位圖書館、數位博物館等系統, 提供數位內容提供者一個安全的架構, 讓具有價值的數位內容可以安全的以網頁的型態發佈, 讓眞正有被授權的使用者可以使用此數位內容, 而不用擔心發布之數位內容可能被盜用的情況。在此應用下, 也可以提供數位電子書的網頁型態發佈一個安全的架構保護其數位產權。另外,OpenDReaMS 伺服器端也可以簡化成保護網頁連結被無授權連結的情況。

OpenDReaMS伺服器端的延伸架構也可以提供數位內容提供者獨立產權授權伺服器, 並且與獨立之付費機制結合。提供數位內容提供者單純的環境、以專注於數位內容的製作與發佈。在此架構下, 數位內容提供者、產權管理中心、付費中心分別爲獨立的管理單位, 提供了數位內容提供者、發佈者、使用者更多彈性的選擇, 並可適用於現有網頁架構。

8. REFERENCES

[1] Activex controls. `http://msdn.microsoft.com/library/default.asp?url=/workshop/components/activex/activex_node_entry.asp`.

[2] Adobe content server 3. `http://www.adobe.com/products/contentserver/main.html`.

[3] extensible rights markup language. `http://www.xrml.org/`.

[4] Fileopen system. `http://www.fileopen.com/`.

[5] Http authentication: Basic and digest access authentication. `http://www.faqs.org/rfcs/rfc2617.html`.

[6] Mime encapsulation of aggregate documents, such as html (mhtml). `http://www.faqs.org/rfcs/rfc2557.html`.

[7] Pdf reference, fourth edition, version 1.5. `http://partners.adobe.com/asn/tech/pdf/specifications.jsp`.

[8] Windows media drm 10. `http://www.microsoft.com/windows/windowsmedia/drm/default.aspx`.

[9] G. Hunt and D. Brubacher. Detours: Binary interception of win32 functions. In *Proceedings of the 3rd USENIX Windows NT Symposium*, pages 135–143, July 1999.