제33회 한국정보처리학회 춘계학술발표대회 논문집 제17권 제1호 (2010. 4)

문서 DRM의 제어 방법에 대한 연구

김승환*, 박선호*, 정태명**
*성균관대학교 전자전기컴퓨터공학과
**성균관대학교 정보통신공학부
e-mail:{shkim, shpark}@imtl.skku.ac.kr, tmchung@ece.skku.ac.kr

A Study on the Mechanism for Controlling Document DRM

Seung-Hwan Kim*, Seon-Ho Park*, Tai-Myoung Chung**
*Dept of Electrical and Computer Engineering, Sungkyunkwan University
**School of Information and Communication Engineering, Sungkyunkwan

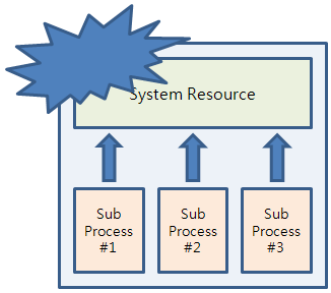
요 약

IT 기술의 발달로 문서의 제작, 관리, 유통, 복사 등이 점차 간단하며 쉬워지고 있다. 또한 네트워크의 발달, 저장매체의 발달로 인해 문서의 불법 복제, 무단 유출, 불법 유통 등도 함께 쉬워지고 있다. 따라서 각종 문서에 대한 보안의 중요성이 부각되기 시작하였다. 이 중 DRM(Digital Right Management)이란 기술이 가장 널리 알려졌으며, 가장 많이 사용되게 되었다. 따라서 많은 DRM 제품이 출시 되었지만 각각 다른 문서 DRM의 경우 어떤 기준없이 개발되어 하나의 PC에서 동작할 수 없는 문제점이 발생하게 되었다. 본 논문에서는 디지털 문서의 대표적인 형태인 Compound Document File Format의 Header를 수정하고, 각 문서 DRM에 고유 식별번호를 부여하여 이와 같은 문제점을 해결하는 방안에 대해 제안하고자 한다.

1. 서론

IT 기술의 발달로 문서가 디지털화 되어감에 따라 관리, 보관 등이 쉬워졌다. 또한 저장매체, 네트워크도 함께 발달함으로써 필요에 따라 유통이 쉬워진 반면 중요 문서, 아이디어, 소스코드 등의 유출도 증가하고 있다[1]. 이에 따라 문서를 보호하기 위한 기술이 개발되기 시작하였으며, 그 중 문서 DRM(Digital Right Management)이란 기술이 가장 널리 사용되고 있다.

문서 DRM이란 문서에 대한 저작권자의 권리나 이익을 보호하거나, 중요한 문서가 유출되었을 경우 이에 대해 보호하기 위한 기술이다. 각종 문서뿐만 아니라 다양한 디지털 매체에도 많이 적용되고 있는 기술이다. 문서 DRM을 통해 문서의 열람, 출력, 캡쳐 등 다양하게 권한을 부여하여 제어할 수 있다[1].



(그림 1) 문서 DRM의 호환성에 대한 문제점

이렇게 문서 DRM의 중요성이 부각되면서 많은 문서 DRM이 출시되었다. 하지만 Windows에서 어떠한 기준 없이 개발된 서로 다른 문서 DRM은 서로 경쟁관계를 가지게 되어 결국 제대로 동작하지 않는 (그림 1)과 같은 문제점이 발생하게 되었다. (그림 1)에서 나타난 문제는 여러 문서 DRM이 운영됨으로써 서로 하나의 자원을 점유하기 위해 경쟁하는 것이다. 이런 경쟁관계로 인해 어떤 시스템이 동작하지 않거나 모두 동작하지 않는 현상이 발생하게 된다[1].

일반적으로 현재 제작되고 있는 대부분의 문서는 Compound Document File Format을 따른다. 따라서 본 논문에서는 각 문서 DRM에 고유 번호를 부여하며, Compound Document File Format의 Header에 필드를 수정함으로써 위와 같은 문제를 해결하고자 한다. 문서 DRM에 고유 번호를 부여함으로써 각 문서 DRM을 식별할 수 있으며, Header를 수정함으로써 해당 고유번호를 삽입할 수 있다. 또한 Header에 추가적인 필드를 삽입하는 것이 아닌 예약된 필드를 사용하였다. 따라서 문서 DRM이 동작하는 과정이나 문서를 제작하는 중에 큰 오버헤드가 발생하지 않는다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로써 문서의 기본적인 포맷에 대해 살펴보고, 3장에서는 본 논문에서 제안하는 방식에 대해 살펴본다. 그리고 4장에서는 제안하는 방안에 대한 평가를 할 것이며, 5장에서 결론 및 향후 연구계획에 대해서 설명한다.

2. 관련연구

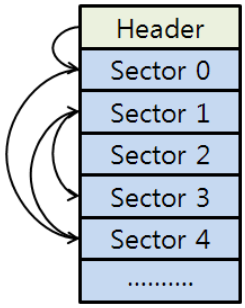
2.1 Compound Document File Format

Compound Document File Format은 마이크로소프트사에서 디스크에 데이터를 저장하기 위한 목적으로 개발되었다. Compound Document File Format은 여러 개의 가상 스트림으로 이루어져 있으며 가상 스트림의 최소 단위를 섹터(Sector)라고 한다. 이 섹터의 크기는 일반적으로 512바이트이며, Compound Document File Format이 생성될 당시에 유동적으로 변경된다. Compound Document File Format의 섹터는 Header, FAT, Directory, Mini-FAT, DIF, Storage와 같이 총 6개의 타입을 가지며 각 역할은 <표 1>과 같다.

<표 1> 각 섹터의 특징

타입	특징
Header	파일의 전체 구조 파악
FAT	섹터 체인의 위치 정보를 유지
Directory	계층적 구조를 표현하기 위해 사용되는 컨트롤 섹터
Mini-FAT	FAT와 비슷하지만 64 바이트의 섹터 크기를 가짐
DIF	FAT 섹터의 섹터 아이디를 가지는 섹터 아이디의 배열
Storage	문서의 내용이 되는 텍스트, 그림 저장

이러한 섹터는 파일에 독립적으로 위치하고 있으므로 스트림을 구성하기 위해 섹터 체인이란 것을 이용하여 스트림으로 구성한다. (그림 1)은 섹터 체인을 보여주는 것이다.



(그림 2) 섹터 체인

섹터체인을 구성하기 위해 각 섹터는 다음에 위치할 섹터의 식별자를 가지고 있다. Compound Document File Format에서 이용되는 섹터 체인은 FAT 섹터 체인, 디렉토리 섹터 체인, Mini-FAT 섹터 체인, DIF 섹터 체인이 있다.

Header 섹터는 <표 2>과 같이 구성되어 있다. <표 2>의 필드는 마이크로소프트사에서 지정한 필드명이며 오픈오피스 프로젝트 등과 같은 다른 문서 구조에서는 변수명은 다르지만 각 의미는 동일하게 사용된다.

<표 2> Header 타입의 구조체 정의

```
struct StructuredStorageHeader {
    BYTE        __abSig[8];             // 8 바이트
    CLSID       _clid;                  // 16 바이트
    USHORT      _uMinorVersion;         // 2 바이트
    USHORT      _uDllVersion;           // 2 바이트
    USHORT      _uByteOrder;            // 2 바이트
    USHORT      _uSectorShift;          // 2 바이트
    USHORT      _uMiniSectorShift;      // 2 바이트
    USHORT      _usReserved;            // 2 바이트
    ULONG       _ulReserved1            // 4 바이트
    ULONG       _ulReserved2;           // 4 바이트
    FSINDEX     _csectFat;              // 4 바이트
    SECT        _sectDirStart;          // 4 바이트
    DFSIGNATURE _signature;             // 4 바이트
    ULONG       _ulMiniSectorCutoff;    // 4 바이트
    SECT        _sectMiniFatStart;      // 4 바이트
    FSINDEX     _csectMiniFat;          // 4 바이트
    SECT        _sectDifStart;          // 4 바이트
    FSINDEX     _csectDif;              // 4 바이트
    SECT        _sectFat[109];          // 436 바이트
};
```

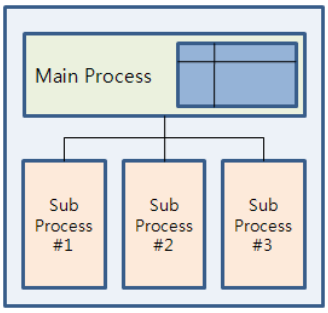
각 필드는 다음과 같은 의미로 사용된다.

- _abSig[8] : 문서 파일의 버전을 나타냄
- _clid : 파일에 대한 식별자
- _uMinorVersion : Minor Version 혹은 Revision Number
- _uDllVersion : Compound Document Format의 버전
- _uByteOrder : 현재 파일의 바이트 오더
- _uSectorShift : 섹터의 크기를 2의 제곱 승으로 나타냄
- _uMiniSectorShift
  : Mini-Sector의 크기를 2의 제곱승으로 나타냄
- _usReserved, usReserved1, usReserved2
  : 예약된 필드
- _csecFat : FAT 체인에 포함된 섹터의 개수
- _sectDirStart : Directory 체인의 첫 번째 섹터 ID
- _signature : 트랜잭션에 사용됨
- _ulMiniSectorCutoff
  : Mini-Sector로 이루어진 스트림의 최대 크기
- _sectMiniFatStart : Mini-FAT 체인의 첫 번째 섹터
- _csecMiniFat : Mini-FAT 체인의 섹터 개수
- _sectDifStart : DIF 체인의 첫 번째 섹터 ID
- _csecDif : DIF 체인의 섹터 개수
- _sectFat[109] : 처음 109개의 FAT 섹터

3. 제안 메커니즘

3.1 제안하는 구조

본 논문에서 제안하는 구조는 하나의 Control Process와 여러 개의 Sub Process를 가지며 (그림 2)와 같다.



(그림 3) 제안하는 구조

- Control Process

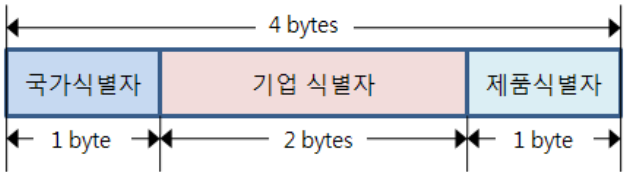
Control Process는 한 PC에 단 하나만 설치되어야 하며, Header 섹터에 있는 DRM Identifier로 Sub Process를 식별하여 선택된 Sub Process가 실행될 수 있도록 DLL을 로드시키는 역할을 한다. 따라서 Control Process는 DIT(DRM Install Table)에 설치된 문서 DRM에 대한 정보를 가지고 있다. Control Process는 Sub Process가 종료되면 사용하였던 자원을 해제하며, 운영체제에서 사용하는 테이블을 원상태로 복구한다.

- Sub Process

각 벤더가 제공하는 DRM 시스템이며, 한 PC에 여러 개의 Sub Process가 존재할 수 있다. 또한 각 Sub Process는 고유한 식별자 DRM Identifier를 가진다.

3.2 DRM Identifier

DRM Identifier를 저장하기 위해 Compound Document File Format에 Header 섹션의 4바이트 길이를 가지는 _ulReserved1 필드를 사용한다. DRM Identifier는 IP (Internet Protocol)와 같이 각 문서 DRM 시스템마다 고유한 값이 할당되어야 한다.



(그림 4) DRM Identifier

고유한 값을 할당하기 위해 4바이트는 (그림 3)과 같이 국가 식별자, 기업 식별자, 제품 식별자 세 가지를 사용한다. 세 가지 식별자는 모두 고유하게 할당된다.

국가 식별자는 1바이트의 길이를 가지므로 2^8 =256개를 할당할 수 있으며, 각 국가의 고유번호를 뜻하는 것이다. 또한 기업 식별자는 총 2바이트의 길이이므로 2^{16} = 65536개를 할당 할 수 있으며, 해당 국가에 존재하는 기업의 고유번호이다. 마지막으로 제품 식별자는 국가 식별자와 같이 1바이트이므로 2^8 =256개를 할당할 수 있으며, 해당 기업에서 출시하는 제품에 대한 고유번호이다.

따라서 DRM Identifier는 고유한 값을 가지게 될 수 있으며, 어떤 시스템에서 동작하더라도 DRM Identifier를 통해 각 벤더에서 제공되는 문서 DRM 시스템을 식별할 수 있다.

3.3 DIT(DRM Install Table)

DIT는 Control Process가 참조하는 테이블이며, 설치되어 있는 문서 DRM 시스템의 정보를 저장하고 있는 테이블이다. 테이블은 (그림 4)와 같은 요소를 포함하고 있다.

DRM Identifier	실행파일	디렉토리	로드 될 DLL
0x1f23de01	drm	c:₩drm	KERNEL32.DLL
0x39fc3w01	drmsystem	C:₩drmsystem	KERNEL32.DLL USER32.DLL
.....	.....	.....	.....

(그림 5) DIT의 구조

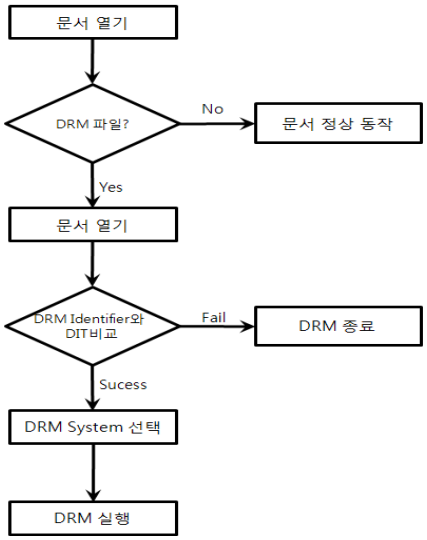
DRM Identifier는 설치된 문서 DRM 시스템의 식별자이다. DIT에 저장된 DRM Identifier를 참조하여 로드할 DLL을 식별하기 때문에 DIT에 저장하고 있어야 한다. 또한 문서 DRM이 동작하기 위해서 로드되어야 할 DLL 정보를 담고 있음으로써 먼저 로드해줄 수 있다. 디렉토리 필드를 참조하여 해당 문서 DRM 시스템이 설치된 위치를 식별할 수 있다. 또한 실행파일 필드를 통해 실행해야 될 파일을 식별할 수 있다.

3.4 동작 과정

문서의 소유권자는 문서 DRM을 사용하여 보호하고자 하는 문서를 암호화한다. 이 때 Control Process는 소유권자가 사용한 문서 DRM의 DRM Identifier의 값을 해당 필드에 삽입함으로써 제품에 대한 기록을 남긴다.

문서 DRM이 정상적으로 동작하기 위해서는 Control Process가 해당 문서 DRM 시스템을 식별할 수 있어야 한다. 따라서 문서 DRM 시스템이 설치 될 때는 해당 DRM Identifier를 Control Process의 DIT에 업데이트해야 한다.

(그림 6)은 제안하는 DRM의 동작을 순서도로 표현한 것이다.



(그림 6) 제안한 DRM의 동작 과정

문서를 열었을 경우 먼저 해당 파일이 문서 DRM에 의해 작성된 파일인지 확인해야한다. 확인 후 문서 DRM에 의해 작성된 파일이 아닐 경우 다른 문서 DRM 프로세스를 동작시키지 않고 열 수 있다. 하지만 문서 DRM에 의해 작성된 파일일 경우 해당 파일을 열기 전에는 필요한 과정을 수행해야한다. 먼저 Control Process는 문서의 Header 섹션에 본 논문에서 추가한 DRM Identifier를 참조하여 로드해야할 문서 DRM 시스템을 식별한다. 식별할 때는 역시 미리 작성된 DIT와 비교하여 동일한 DRM Identifier가 있는지 살펴본다. 동일한 DRM Identifier가 있다면 식별된 문서 DRM 시스템이 동작 할 수 있게 DLL을 로드한 후 선택된 문서 DRM 시스템의 실행파일을 실행시킨다. 동일한 DRM Identifier가 없을 경우 문서를 열 수 없으므로 종료한다.

4. 제안하는 DRM에 대한 평가

이번 장에서는 본 논문에서 제안하는 문서 DRM의 현재 사용되고 있는 문서 DRM은 어떠한 기준없이 개발되어 서로 다른 문서 DRM간에 상호 연동이 불가능 했다. 본 논문에서는 이러한 점을 해결하기 위해 문서의 Header의 예약된 필드를 사용하여 각 문서 DRM 시스템 간에 고유한 ID를 부여함으로써 이를 각 문서 DRM 시스템을 식별하여 실행함으로써 해결하였다. 또한 기존에 제안된 방식인 서버를 사용한 방식은 파일을 열 때 마다 서버로의 요청이 필요하였다. 하지만 본 논문에서 제안하는 방식은 로컬에서 동작함으로써 서버로 인증 및 필요한 데이터를 전송받을 필요가 없다. 또한 Compound Document File Format의 예약된 필드를 사용하였기 때문에 현재 동작하고 있는 시스템의 큰 수정이 불필요하다.

5. 결론 및 향후 연구 계획

본 논문에서는 문서 DRM의 상호 연동이 불가능한 점과 기존에 제안된 서버를 사용한 문서 DRM의 문제점을 해결하였다. 또한 제안한 방안이 기존의 문제점을 해결하며, 기존에 제안된 서버를 사용한 방안에 비해 우수할 것임을 예측할 수 있었다.

향후 연구 계획으로는 해당 시스템을 구현함으로써 발생가능한 문제점에 대해 살펴보고자 한다. 구현 과정에서 발생가능한 문제점을 분석하고, 이를 해결하기 위한 방안에 대해 연구하고자 한다.

참고문헌

[1] 김승환, 박선호, 정태명, “문서 DRM의 상호 운영 방안에 대한 연구”, 한국통신학회 동계 종합학술발표회 논문집 pp.285, Feb. 2010.

[2] Bill Rosenblatt, Bill Trippe, Stephen Mooney, "Digital Rights Management, Business and Technology", Published by M&T Books, pp79-102.

[3] Microsoft Open Specification Promise, “Windows Compound Binary File Format Specification”, Microsoft Corporation, Feb., 2008.

[4] Microsoft Open Specification Promise, “Word 97-2007 Binary File Format (.doc) Specification”, Microsoft Corporation, Feb., 2008.