라이센스 기반 유통시스템을 위한 DRM 클라이언트의 설계 및 구현

남 도 원, 윤 기 송
한국전자통신연구원

Design & Implementation of DRM Client for License-based Digital Content Distribution System

Do-Won Nam, Ki-Song Yoon
Electronics and Telecommunications Research Institute
dwnam@etri.re.kr, ksyoon@etri.re.kr

요 약

무한 복제가 가능한 디지털 컨텐츠의 보호를 위해 사용한 암호화 기술은 최종 사용자의 환경에 컨텐츠의 복호화를 위한 모든 키가 존재해야 한다는 전제를 달고 있다는 점과, 사용자의 클라이언트 환경이 다양한 악의적인 공격에 노출되어 있다는 점은 디지털 컨텐츠의 유통에서 클라이언트 환경의 설계가 매우 중요하다는 것을 말해준다. 본 논문에서는 라이센스 기반의 디지털 컨텐츠 유통 시스템을 위한 DRM 클라이언트를 설계한다. 여기서 설계된 DRM 클라이언트는 사용자에게 노출되어서는 안되는 복호화 키를 비롯한 중요한 정보들을 저장하는 시큐어 데이터 영역, 컨텐츠를 활용하기 위한 뷰어 프로그램을 DRM 정책에 따라 제어하기 위한 플러그인 구조, 라이센스와 암호화된 컨텐츠에 담긴 정보를 바탕으로 DRM 정책을 결정하고 이에 따라 클라이언트를 구성하는 각 모듈을 제어하는 DRM 제어부, 암호화된 컨텐츠를 복호화하여 넘겨주는 필터 드라이버로 나뉘어 있으며 이를 구현하기 위한 기술과 함께 소개한다.

I. 서론

PC와 인터넷의 보급에 힘입어 기존에 아날로그 위주이던 컨텐츠는 디지털로 빠르게 전환되고 있다. 디지털 컨텐츠는 원본과 완벽히 동일한 품질의 복사본을 만드는 것이 가능하고, 빈번한 사용으로 인한 품질의 저하 현상이 발생하지 않기 때문에 새로 등장하는 컨텐츠는 물론, 기존의 아날로그 컨텐츠들도 디지털로 변환되고 있다.

디지털 컨텐츠의 유통으로 고품질의 컨텐츠를 손쉽게 이용할 수 있게 되었지만, 컨텐츠 저작권자의 입장에서는 자신의 컨텐츠를 품질의 저하 없이 무단으로 복제하여 사용하는 사람이 생기기 때문에 저작권자로서의 권리가 침해당하는 결과를 낳기도 한다. 이러한 문제를 해결하기 위한 시도로서 DRM이라 불리는 암호화 기반 컨텐츠 보호 기술이 연구/개발 되고 있다.

일반적으로 DRM 기술에서는 컨텐츠는 암호화되어 그대로는 이용할 수 없는 형태로 사용자에게 제공되고, 사용자는 이를 복호화 할 수 있는 키를 라이센스의 형태로 구매하게 된다. DRM 클라이언트라 불리는 소프트웨어는 라이센스로부터 컨텐츠를 복호화 하기 위한 키를 추출하고, 이를 이용하여 암호화된 컨텐츠를 복호화 한 후에 이를 뷰어 프로그램에 제공한다.

암호화된 컨텐츠가 사용자 환경에서 동작하기 위해서는 복호화를 위한 키가 모두 사용자 환경에 존재해야 한다. 이는 보안상 중요한 모든 정보가 사용자들의 공격에 노출되어 있다는 것을 의미한다. 따라서 DRM 기반의 컨텐츠 유통 시스템을 설계할 때 클라이언트의 보안문제가 가장 큰 이슈가 되는 것이다.

II. 기본 아키텍처

1. 플러그인 구조

DRM 클라이언트는 일반 사용자 뿐만 아니라 악의적인 사용자에게도 노출된 환경에서 동작하기 때문에 다양한 공격에 대한 강인성과 함께 사용자의 편의성을 고려한 유연한 구조도 가져야 한다. PDF, 워드, 엑셀 파일과 같이 널리 이용되나 특정 업체의 소프트웨어에 종속된 컨텐츠를 지원하는 경우, 사용자의 편의성을 위해서는 해당 업체의 전용 소프트웨어를 이용할 수 있게 해야 한다. 하지만 포맷이 공개되지 않은 파일의 경우 타 업체에서 이를 지원하기가 쉽지 않다는 문제가 있다.

이를 위해 클라이언트 환경에서 DRM 정책을 결정하는 제어부와 뷰어 프로그램을 분리하고, 그 사이에 플러그인 계층을 두어 DRM 제어부의 결정에 따라 뷰어 프로그램을 제어하도록 한다. DRM을 염두에 두지 않고 만들어진 뷰어 프로그램을 DRM 제어부의 정책에 따라 제어하기 위해서는 몇 가지 구현 기술이 필요하다.

먼저, Acrobat Reader와 같이 플러그인을 허용하는 소프트웨어의 경우는 이를 통해 DRM 정책을 적용시킬 수 있으며, 두 번째로 소프트웨어가 Active X 컨트롤로 제공되는 경우에는 OLE 컨테이너의 형태로 플러그인 계층을 구성하여 제어할 수 있다. 세 번째로 외부에서 개입될 여지를 남기지 않고 완전히 독립적으로 동작하는 소프트웨어의 경우는 운영체제와 소프트웨어 사이에 오고 가는 메시지를 가로채어 DRM 정책을 적용시킬 수 있다.

2. 시큐어 데이터 영역

앞서 언급했듯이 암호화 기술은 최종 사용자 환경에서 모든 복호화 키가 존재해야만 하기 때문에 라이센스의 형태로 발급받은 복호화 키는 사용자의 PC에 안전하게 보관되어야 한다.

실제로 이를 가장 안전하게 구현하기 위해서는 스마트카드와 같은 안전한 물리적인 저장 장치를 사용해야 하나, 현실적인 어려움으로 인해 하드디스크 내에 저장을 하되 악의적인 사용자의 공격으로부터 복호화 키를 보호할 수 있는 장치를 마련해야 한다.

본 논문에서는 커널 레벨의 드라이버를 이용해 가상 드라이브와 같은 데이터 영역을 만들고, 이 안에 복호화 키를 비롯한 중요 정보들을 암호화하여 저장하는 방법을 이용하였다. 암호화는 사용자 PC에 종속된 하드웨어 정보와 DRM 클라이언트의 설치시에 생성된 난수를 조합하여 이용한다.

저장영역 내에는 이 공간에 접근이 허용된 프로세스의 목록이 포함되어 있다. 이 목록에는 기본적으로 DRM 제어부와 필터 드라이버만이 포함되어 있으며, 유효한 라이센스를 가진 프로세스가 컨텐츠를 이용하기 위해 로딩될 때 DRM 제어부는 이 프로세스를 접근 허용 목록에 추가하고, 프로세스가 종료하거나 라이센스가 만료되면 목록에서 삭제한다.

3. 필터 드라이버

필터 드라이버는 커널 레벨의 드라이버로, 어떤 프로세스가 암호화된 컨텐츠에 접근할 경우에 해당 프로세스가 컨텐츠에 접근할 수 있는 권한이 있는지의 여부를 확인한 후 복호화하여 넘겨주는 역할을 한다. 접근하는 프로세스는 컨텐츠가 암호화 되어 있는지의 여부를 알지 못하고 일반적인 파일 입출력 함수를 이용하며, 필터 드라이버는 이를 가로채어 동작한다.

필터 드라이버가 복호화를 수행할 때 필요한 키는 시큐어 데이터 영역에 저장되어 있고, 매번 입출력이 발생할 때마다 해당 프로세스의 접근 허용 여부를 시큐어 데이터 영역의 접근 허용 목록에서 확인한 후에 복호화를 수행한다.

4. DRM 제어부

DRM 제어부는 클라이언트 프로그램의 가장 중심이 되는 모듈로 라이센스 서버와 통신하여 라이센스를 전송받고, 이를 시큐어 데이터 영역에 저장하며, 컨텐츠에 대한 이용 요청이 발생했을 경우 라이센스를 확인하여 플러그인을 통해 뷰어를 실행해 준다. 유효한 라이센스를 가진 프로세스가 실행될 때 이를 접근 허용 목록에 추가해주며, 프로세스의 종료나 라이센스의 만료시에 이를 다시 삭제해주는 역할도 한다.

III. 보완 기능

1. 모듈간 인증

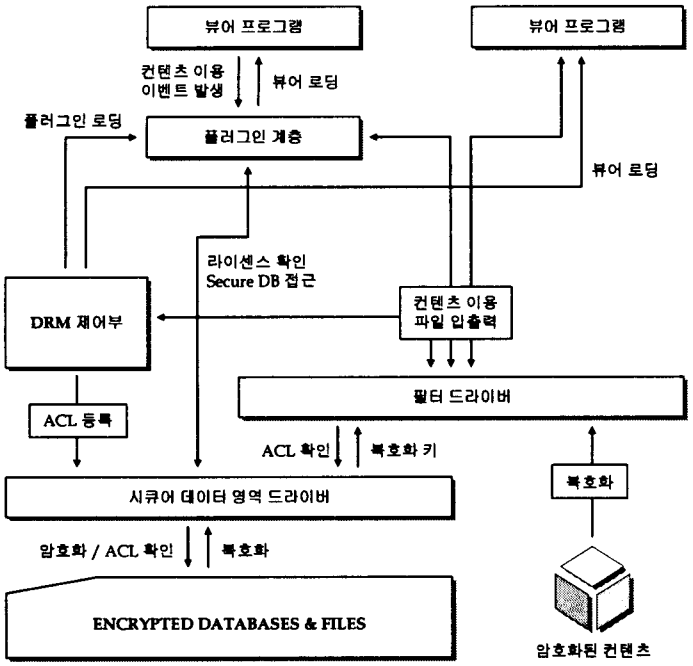

그림 20. DRM 클라이언트 구성도

클라이언트를 구성하는 모든 모듈들은 해쉬값이 계산되어 시큐어 데이터 영역내에 그 정보가 기록된다. 이 정보는 매번 모듈이 로딩되거나 서로 통신하는 경우에 재계산되어 모듈이 악의적인 공격에 의해 수정되지 않았는지 검사하는데 이용된다.

웹을 통해 업데이트 받거나 추가로 설치되는 모듈 패키지에는 신뢰할 수 있는 기관이 서명한 모듈 정보 파일이 첨부되어 있어 인증받지 않은 모듈이 설치되는 것을 막는다.

2. 표준 API

새로운 컨텐츠 타입을 지원하기 위해서는 적절한 뷰어 프로그램이 있어야 하고, DRM 정책을 기반으로 뷰어를 제어하기 위한 플러그인이 있어야 한다. DRM 업체마다 DRM 정책의 기술방법이 다르고, 뷰어 프로그램마다 동작 방식이 서로 다르기 때문에 현재는 DRM 업체에서 새로운 컨텐츠 타입의 추가시마다 직접 뷰어나 플러그인을 개발하고 있다. 또한, 클라이언트-플러그인-뷰어의 명확한 계층구조를 가지지 않고, 뷰어에 클라이언트의 기능이 통합되어 있기도 하며 플러그인 계층 없이 클라이언트에서 바로 제어하기도 하므로 표준화가 쉽지 않다.

본 연구에서는 클라이언트와 뷰어를 완전히 분리하고, 그 사이에서 DRM 정책에 따라 뷰어를 제어하는 역할로 플러그인을 이용하기 때문에 클라이언트와 플러그인이 통신하기 위한 표준 API 집합을 개발하였다. 이를 이용하면 서드파티 업체가 특정 컨텐츠 타입과 뷰어 프로그램을 지원하는 플러그인을 제작하여 클라이언트와 연동시킬 수 있다.

3. 블록 암호화

컨텐츠는 디스크에 암호화된 상태로 저장되어 있다가 복호화 요청이 들어오면 요청된 블록만이 복호화되어 뷰어 프로그램에 전달된다. 전체 데이터 블록을 한꺼번에 복호화하는 것은 메모리 공간의 효율성 측면에서도 부적합하고, 복호화된 전체 컨텐츠가 메모리 어딘가에 존재하는 상황이 발생하기 때문에 안전성의 측면에서도 좋지 않다.

IV. 동작 시나리오

- 사용자는 암호화된 컨텐츠를 더블클릭한다
- DRM 제어부는 해당 컨텐츠에 대한 라이센스가 있는지 시큐어 데이터 영역의 라이센스 목록을 참조한다
- DRM 제어부는 라이센스가 확인되면 컨텐츠를 재생할 뷰어와 제어를 위한 플러그인을 로딩한다
- DRM 제어부는 로딩된 플러그인과 뷰어를 시큐어 데이터 영역의 접근 허용 목록에 등록한다
- 플러그인은 시큐어 데이터 영역에 접근하여 라이센스 보유 상태를 확인한다
- 플러그인은 라이센스의 허용범위 내에서 뷰어의 동작을 허용한다
- 뷰어는 컨텐츠 파일에 대한 입출력을 시작한다
- 필터 드라이버는 뷰어의 파일 입출력에 개입하여 시큐어 데이터 영역의 접근 허용 목록을 확인하고, 암호화된 컨텐츠를 복호화하여 뷰어에 넘겨준다

- 뷰어의 동작이 종료되거나 사용중에 라이센스가 만료되면 필터 드라이버는 복호화된 컨텐츠를 뷰어에 제공하는 것을 중단한다
- 뷰어와 플러그인이 종료되면 DRM 제어부는 시큐어 데이터 영역의 접근 허용 목록에서 해당 뷰어와 플러그인의 정보를 삭제한다

V. 문제점 및 한계

어떠한 방식의 암호화를 사용하더라도 소프트웨어적인 접근 방법만으로는 보호기술에 한계가 존재한다. 예를 들어 음악파일은 암호화 기법을 이용해 허가받지 않은 사용으로부터 보호받고 있지만 이 파일이 재생되기 위해서는 어느 순간 복호화된 원본 컨텐츠의 상태로 존재해야만 한다. 악의적인 사용자의 공격이 암호화 알고리즘이나 사용자 환경의 어딘가에 저장되어 있을 복호화 키를 추출하는데 대해 이루어진다면 이를 보완할만한 적절한 방법을 강구할 수 있을 것이다. 하지만 복호화가 이루어진 이후의 컨텐츠 데이터의 흐름 경로에 대한 공격이나 스피커를 통해 재생되는 음악을 다시 녹음하는 식의 공격에는 적절히 대응하기 어려운 것이 사실이다.

또한 보안과 사용자의 편의성은 반대의 성질을 띠는 것이기 때문에 보안성을 높이기 위해서는 사용자의 불편을 감수해야 하고, 사용자의 편의성을 고려하기 위해서는 보안성에 있어서 어느 정도의 희생을 생각할 수밖에 없는 것이다. 양자 사이의 적절한 타협점을 설정하고, 그것에 맞추어 적절한 보안 정책을 세우고 구현하는 것이 필요하다.

VI. 결론

우리는 DRM 클라이언트의 설계에 있어서 몇가지 기본 정책 - 1)기존에 개발되어 있는 뷰어 프로그램을 그대로 이용 가능하게 한다, 2)새로운 컨텐츠 타입과 뷰어를 지원하는게 용이하도록 한다. 3)클라이언트를 구성하는 모듈의 역할 구분을 분명하게 하여 투명한 상호 연동이 가능하도록 한다 4)모듈간의 상호 인증과 권한 설정을 통해 악의적 공격에 대한 기본적인 대응을 할 수 있도록 한다 - 을 설정했고 이를 만족시키는 범위 내에서 설계와 구현이 이루어 지도록 했다.

현재 워드 문서(HWP, DOC)를 비롯하여 음악 파일(MP3), 동영상 파일, PDF 문서 포맷 등에 대해 개발이 완료되어 있고, 추가의 컨텐츠 타입 지원과 보안 구조 개선에 관한 연구 및 리버스 엔지니어링에 의한 공격에 대한 대응방안에 대한 연구가 진행 중에 있으며, MPEG-21의 표준안 제정이 진행됨에 따라 이를 수용하는 연구도 진행 중이다.

VII. 참조자료

[1] Kenneth Louis Milsted, Automated Method and Appara-tus to Package Digital Content for Electronic Distribution using the Identity of the Source Content, United States Patent 6,345,256.

[2] Olin Sibert, DigiBox: A Self-Protecting Container for Information Commerce, 1st USENIX Workshop on Electronic Commerce, 1995.

[3] IMPRIMATUR business model, MPEG-21 st andard