HC-256 스트림 암호화를 이용한 범용성 및 확장성을 가진 DRM 기법 설계

종신회원 박 준 철*

Design of A Generic and Scalable DRM Scheme using HC-256 Stream Cipher

Jun-Cheol Park* *Lifelong Member*

요 약

DRM은 미디어 저작권자들을 보호하기 위해 디지털 콘텐츠의 사용을 제한하는 기술을 총칭하는 용어이다. 본 논문에서는 오디오, 비디오 등 미디어 유형에 상관없이 적용할 수 있는 범용의 복제 방지 기법을 제안한다. 제안 방법을 써서 공격자가 불법 복제한 콘텐츠를 타 플랫폼에서 재생하거나, 암호화에 사용된 비밀 키 값을 알아내는 것을 매우 어렵게 만들 수 있다. 기존의 DRM 기법들과 달리 제안 기법은 미디어 콘텐츠 암호화를 위해 스트림 암호화인 HC-256만을 이용하기 때문에, 클라이언트에서는 콘텐츠 파일의 스트림 복호화를 거쳐 재생이 즉시 시작될 수 있도록 한다. 이를 위해 저장하는 비밀 키의 수는 미디어 콘텐츠의 수에는 무관하고, 클라이언트(플레이어 카피)의 수에만 비례해 증가하도록 하여 뛰어난 확장성을 보인다. 또한, 제안한 방법은 단순화를 위해 널리 통용되는 다운로드-무한재생의 미디어 라이센스 정책만을 지원하기 때문에 라이센싱을 위한 별도의 서버나 과정이필요하지 않게 된다.

Key Words : Generic DRM, Stream Cipher, Copy Protection, Digital Content, Scalability

ABSTRACT

Digital Rights Management(DRM) is a term that refers technologies for imposing limitations on the use of digital content for protecting media copyright holders. This paper proposes a generic scheme for digital media copy protection that can be applied to any digital media such as audio, video, etc. The scheme will make it very hard to play a copied content on a foreign platform and to guess secret content encryption keys. Unlike other DRM techniques, the scheme uses the stream cipher HC-256 only for encrypting media content and it allows a client to start content playback immediately following its streamed decryption. As to the encryption, it requires to generate several secret keys for each new client(player copy), rather than for each media content, which makes it scalable in terms of managed keys. Also, for simplicity, the scheme supports the popular unlimited-play-after-download policy only, which would eliminate the necessity for deploying separate server or process for licensing.

※ 본 연구는 2008학년도 홍익대학교 학술연구진흥비에 의해 지원되었습니다.
* 홍익대학교 컴퓨터공학과(jcpark@hongik.ac.kr)
논문번호 : KICS2009-06-230, 접수일자 : 2009년 6월 2일, 최종논문접수일자 : 2009년 8월 26일

Ⅰ. 서 론

인터넷의 대중화와 미디어 기술의 발달로 디지털 콘텐츠를 활용하는 다양한 방법들이 계속 등장하고 있다. 디지털 콘텐츠는 아날로그 콘텐츠와 달리 매우 쉽게 실시간으로 복제할 수 있고, 무한대의 복사에도 복제된 콘텐츠가 원본과 동일한 품질 상태를 유지한다. 이러한 특징들로 인해 사용자는 저작권 보호를 받는 디지털 콘텐츠에 대해 불법 복제 및 배포의 유혹에 빠지기 쉬우며, 이에 대한 대응책이 필요하다. DRM(Digital Rights Management)은 디지털 콘텐츠의 불법적인 복제, 배포, 사용 등을 방지하기 위한 기술을 통칭한다[1],[2].

DRM의 기본 아이디어는 보호하려는 콘텐츠에 대해 허가 받은 방식의 제한된 접근만을 허용하는 것이다. 이러한 접근 제한을 위해 보통 암호화 기법이 사용된다. 즉, 보호하려는 콘텐츠를 암호화하여 사용자에게 제공하고, 허가받은 사용자만이 이 콘텐츠를 적절하게 복호화하여 재생하도록 하는 것이다. 그런데 콘텐츠 복호화를 위한 시간이 많이 걸린다든지, 지나치게 세분화된 라이센스 정책으로 콘텐츠를 얻는 과정이 복잡하다든지 하면 사용자의 입장에서는 불편을 느끼게 되고, 따라서 이런 것을 요구하는 DRM 기술은 외면당하게 될 것이다. 결국 강력한 불법 복제 방지 기능을 제공하면서도 사용자 입장에서는 콘텐츠 보호 기술이 적용된 것조차 느끼지 못할 만큼 불편을 주지 않는 DRM 기술 개발이 필요하다. 한편 암호화를 위해 키(key)가 사용되는데 콘텐츠 배포자는 보안성을 높이기 위해 각각의 콘텐츠 마다 고유의 키를 생성하며, 사용자 플랫폼이 달라지는 경우에는 동일한 콘텐츠라도 다른 키가 사용되도록 하는 것이 일반적이다. 따라서 사용자는 자신의 플랫폼에 다운로드된 수많은 콘텐츠 파일들 모두에 대해 각각 서로 다른 키를 유지해야 하는 부담을 가지게 된다. 또한 콘텐츠 배포자의 입장에서도 사용자가 다운로드한 콘텐츠 파일 각각에 대해 해당 키를 유지하고 있어야, 필요시 사용자에게 이를 전달할 수 있다. 이와 같이 키의 생성, 배포 및 관리는 확장성 있는 DRM 시스템의 설계를 위해 필수적으로 고려해야 할 문제이다.

본 논문에서는 오디오, 비디오 등 미디어의 유형에 상관없이 어떤 파일이든 원래의 종속된 플랫폼의 플레이어에서만 재생이 됨을 보장하는 소프트웨어 기반의 DRM 기법을 제안한다. 제안 기법은 콘텐츠의 암, 복호화를 위해 스트림 암호화인 HC-256[3]만을 사용하여, 암호화된 콘텐츠를 복호화하면서 연속해서 스트리밍 방식으로 재생할 수 있다. 또한 모든 콘텐츠가 고유의 키를 사용하게 하면서도 관리해야 하는 키의 개수는 전체 콘텐츠의 개수에 상관없이 참여하는 사용자 플랫폼의 수에만 비례하도록 설계하였다. 그리고, 가장 일반적이고 이해하기 쉬운 다운로드-무한재생의 정책만을 고려하여 제안 기법이 다양한 인프라에서 적용될 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서 기존 DRM 기술들의 암호화, 키 생성 및 관리 기법들을 설명한다. 3장은 키 생성 및 관리, 콘텐츠 암호화 및 복호화를 중심으로 제안 기법을 설명한다. 4장은 제안 방법이 다양한 무단 복제 공격 시나리오에 어떻게 대비되어 있는지 보인다. 5장은 결론과 향후 연구 과제를 제시한다.

Ⅱ. 관련 연구

DRM 기술의 대표적인 예가 마이크로소프트사의 Windows Media DRM(이하 WMDRM)[4]으로, 이 기술은 마이크로소프트사의 Windows Media Player 및 관련 컴포넌트에 포함되어 있다. 이 기술은 타원곡선 암호화를 통해 키를 교환하고, 콘텐츠를 보호하기 위해서는 DES 블록 암호화 및 RC4 스트림 암호화를 사용하고 있다. 또한 별도의 라이센스 취득 프로토콜을 가지고 있다. 라이센스 취득 프로토콜을 통해 사용자는 특정 콘텐츠에 대해 라이센스를 구매하고, 구매한 라이센스로부터 콘텐츠 복호화에 필요한 키를 획득하게 된다. WMDRM은 본 논문의 제안 방법과 달리 콘텐츠를 이용하기 위해 별도로 라이센스 취득이 필요하다는 부담이 있으며, 암호화에 있어서도 복수의 방법을 사용해야 한다. WMDRM은 2005년의 Version 10 등 여러 차례 깨진 사례가 보고되고 있다[5].

또 다른 대표적 DRM 기술이 Apple의 FairPlay[6] 기술로 iPhone, iTunes, iPod에서 사용된다. 라이센스 정책은 다운로드-당-지불(pay-per-download) 방식인데, 사용자는 콘텐츠를 구매하면서 암호화된 콘텐츠와 이를 복호화할 수 있는 마스터 키를 전송받는다. 콘텐츠 암호화를 위해 AES 블록 암호화를 MD5 해쉬와 함께 사용한다. 이 때 콘텐츠 자체는 마스터 키에 의해 암호화되지만, 이 마스터 키를 안전하게 전달하기 위해 별도의 키를 사용한다. 이 별

도의 키로 마스터 키를 암호화하는데, 이 별도 키는 사용자와 콘텐츠마다 각각 다른 값을 사용한다. 이 키들은 사용자와 콘텐츠 배포 서버에 모두 저장된다. 하나의 마스터 키로 여러 콘텐츠를 암호화하는 경우, 만약 어떤 콘텐츠에 대해 마스터 키가 노출되면 다른 여러 콘텐츠들은 별도의 노력 없이 노출된 마스터 키에 의해 복호화될 수 있다. 또한 이 기술은 본 논문의 제안 방법과 달리 사용자 플랫폼 및 개별 콘텐츠 모두에 대해 별도의 키를 유지해야 하는 부담을 가진다. FairPlay를 채택한 iTunes도 공격당해 깨진 사례가 보고되고 있다[7].

이 외에도 다수의 소프트웨어 기반의 DRM 기법들[8]-[10]이 발표된 바 있으나, 여러 방법으로 비밀키를 설정하고, 대칭 키 암호화 기법을 채택하여 콘텐츠를 보호하고, 다양한 수준의 라이센스 정책을 통해 콘텐츠의 접근을 규정하는 등의 공통점을 가진다. 하지만 제안 기법을 포함한 소프트웨어 기반의 이러한 DRM 방식들은 소프트웨어 역 공학(reverse engineering)에 의한 공격 및 아날로그 채널을 이용한 공격(예: MP3 파일 재생 중 녹음기로 녹음)에는 취약하다. 기존 방식들이 깨진 것도 암호화 알고리즘 자체에 대한 공격이 아니라 소프트웨어 역 공학에 의해 키가 평문으로 사용되는 시점을 노리거나 콘텐츠가 평문으로 복호화 되는 것을 가로채는 방식으로 진행되었다.

Ⅲ. 제안 기법

제안 기법의 플레이어 등록, 콘텐츠 암호화 및 복호화, 키 생성과 관리 방법을 중심으로 서술한다.

3.1 제안 구조 및 전체 상세 흐름도

제안하는 기법은 미디어 서버와 미디어 콘텐츠의 재생을 원하는 다수의 클라이언트가 참여하는 간단한 구조상에서 실현될 수 있다(그림 1). 미디어 서버는 키 관리, 파일 서버의 역할을 담당하며 클라이언트의 요청에 따라 플레이어 카피 및 미디어 콘텐츠를 제공한다. 클라이언트는 미디어 플레이어의 다운로드 및 등록 과정을 거친 후, 원하는 미디어 콘텐츠를 서버로부터 구입하고 다운로드 후 재생한다. 플레이어 다운로드 요청에서 콘텐츠 파일의 재생까지 전체 과정의 흐름도는 그림 2와 같은데, 핵심인 콘텐츠 파일 암호화 및 콘텐츠 파일 복호화/재생은 각기 별도의 그림을 통해 추후 상세 설명한다.

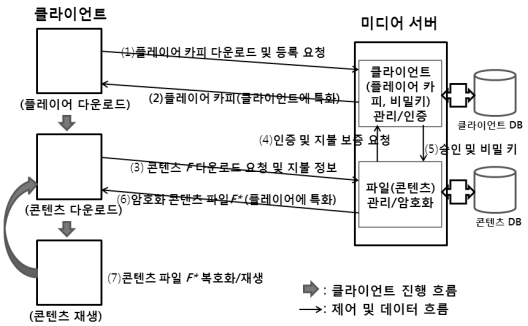

그림 1. 제안 구조상의 클라이언트-서버 상호 작용

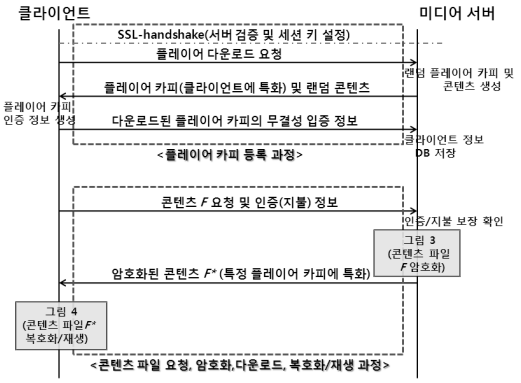

그림 2. 클라이언트-서버 사이의 전체 과정 흐름도

3.2 플레이어 등록

어떤 클라이언트가 최초로 서버에 접속하여 플레이어 소프트웨어 카피를 다운로드받고 자신을 등록하는 과정은 다음과 같다.

(i) SSL handshake 프로토콜[11]을 이용하여, 클라이언트는 서버의 인증서로 서버를 인증한 후 서버와의 보안 통신에 쓸 공유 세션 키 A를 생성한다.

(ii) 클라이언트는 서버로부터 자신의 플랫폼 환경에 맞는 플레이어 소프트웨어의 다운로드를 신청한다.

(iii) 서버는 해당 플레이어 소프트웨어 카피를 다른 플레이어 카피와 다르게 하기 위해 두 부분의 랜덤한 고유 코드를 삽입한 실행 파일을 만든다. 이 때 별도로 임의의 콘텐츠 파일(제대로 재생되지 않는 것이어도 상관 없음) F가 클라이언트에 같이 다운로드 되도록 한다. 두 부분의 코드는 각각 서로 다른 목적을 가지고 있으며 다음과 같이 만들어진다.

(a) 첫 번째 랜덤한 고유 코드는 클라이언트 플랫폼에 고유한 정보(예: Ethernet 카드의 48-bit MAC 주소)를 읽어오는 부분과 그 값을 변형시키는 부분으로 구성된다. 고유한 정보로 MAC 주소를 사용한다고 할 때, 하나의 예를 들면 다음과 같다.

```
X = read(플랫폼의 MAC 주소);
    // 프로그램이 실행되면서 읽어 옴, 숫자 취급.
if ((Y = X * 0x3bbc98f2)의 <47번째 bit, 12번째의 bit> == <01> 또는 <10>)
    then Y = Y * 7;    else Y = Y * 9;
Z = (Y || 3*Y || 5*Y || 7*Y || ...);
    // 512 bits 채울 때까지 반복, 나머지는 버림.
```

위와 같은 코드는 플레이어 카피마다 다르게 설정되며, 서버는 이를 위해 `if-then-else`, 사칙연산, 연결연산(||), 비트 관련연산(비교, and, or, xor, complement), 랜덤한 숫자(예: 0x3bbc98f2) 생성 등을 무작위로 조합하여 짧은 코드를 미리 만들어 둔다. 코드의 목적은 플랫폼에 고유한 정보를 즉석에서 읽고, 이를 이용하여 특정한 크기(512-bit)의 고유 값 Z를 생성하는 것이다. 이 Z 값이 콘텐츠 복호화에 사용되게 함으로써 타 플랫폼으로의 콘텐츠 복제 시 복호화가 불가능하도록 만들 수 있다.

(b) 두 번째 랜덤한 고유 코드는 각 콘텐츠마다 서로 다른 키를 사용하도록 하기 위한 것이다. 콘텐츠 파일을 F라고 할 때, 예를 들어 다음과 같은 코드가 만들어질 수 있다. $F(i,j)$는 파일 F의 i-번째 비트부터 j-번째 비트까지의 블록($0 \le i \le j < length(F)$)을 나타낸다.

```
W = [F(24,47)||F(33,50)||...||F(4228,4229)]에서 처음 512 bits를 취함;
    // 처음 512 bits의 prefix 취하거나, 필요한
    // 만큼 동일한 연결연산을 반복함. 위치 값에
    // mod length(F)의 연산 생략됨.
```

매우 높은 확률로 이 값은 콘텐츠마다 다를 것이기 때문에 콘텐츠마다 고유한 키를 생성하는데 사용한다. 그리고 계산에 사용된 값(위의 예에서 $\langle(24,47),(33,50),...,(4228,4229)\rangle$)은 플레이어 카피마다 서로 다르게 서버에서 할당을 한다. 그런데 콘텐츠 크기를 미리 알 수 없기 때문에 W 계산에서 파일의 크기를 넘어가는 비트 위치는 mod 연산(%로 표기)을 통해 해당 파일 내의 위치로 바꿔주어야 한다. 예를 들어, $F(4228,4229)$는 F의 크기가 2000 bits이라면, $F(4228\%2000,4229\%2000) = F(228,229)$로 계산되도록 코드를 만든다.

(c) 클라이언트는 손상되지 않은 플레이어 소프트웨어 카피를 받았다는 것을 서버에 확인시키기 위해 (i)의 공유 키 A를 이용하여 $E_A(id,pw,X,Z,W)$를 서버에 보낸다. 이 값은 평문 id,pw,X,Z 및 W를 키 A를 이용하여 대칭 키 암호화시킨 값이다. 서버는 스스로 생성한 공유 키 A를 이용하여 이를 복호화하고, 그 결과 X를 가지고 자신이 플레이어에 삽입했던 코드를 실행시킨 결과가 Z와 맞는지 확인한다. 또한 파일 F에 대해 W를 계산하는 코드를 실행한 결과가 수신한 W와 일치하는지 확인한다. 모두 맞으면, 이후 서버는 코드는 기억할 필요가 없으며, 이 클라이언트 플레이어 카피에 대한 정보로 $\langle id,pw,Z,L\rangle$를 저장한다. 여기서 L은 (b)에서 설명한 코드에 삽입되었던 블록들의 위치를 나타내는 리스트로, 위의 예에서는 $L = (\langle 24,47\rangle, \langle 33,50\rangle,...,\langle 4228,4229\rangle)$이다. 향후 이 클라이언트는 id 및 pw를 이용하여 자신이 누구인지(어떤 플레이어 카피를 사용하고 있는지) 서버에 알린다.

이제 클라이언트는 서버에 콘텐츠 구매와 다운로드를 요청하고, 자신의 플레이어 카피에 바인딩된 암호화된 콘텐츠를 받아서 재생할 수 있다.

3.3 키 생성 및 관리 방법

등록 과정에서 클라이언트의 플레이어 카피는 서버에 $\langle id,pw,Z,L\rangle$로 저장된다. 플랫폼에 다운로드된 다수의 콘텐츠에 대하여 서버는 정보를 기억할 필요가 없다. 이것은 제안 기법에서 콘텐츠는 다운로드 후 무한히 재생할 수 있다는 가정을 하고 있기 때문이고, 다양한 라이센스 정책(예: 3회 또는 3일 재생 가능)을 수용한다면 콘텐츠마다 저장해야 할 정보가 생기게 될 것이다. 하지만 오프라인 상의 일반 물건 구매와 마찬가지로 구매 후 영구히 콘텐츠 사용 권리를 가지게 하는 것이 사용자의 입장에서는 가장 이해하기 쉽고, 저작권자의 입장에서도 판매 모델을 단순화시켜 관리가 쉬워질 것이다. 결과적으로 보통 플랫폼의 수보다 사용 가능한 콘텐츠의 수가 훨씬 많다는 것을 고려하면, 제안 방법은

키 관리의 측면에서 뛰어난 확장성을 제공한다.

3.4 콘텐츠 암호화 및 복호화 방법

콘텐츠는 이를 다운로드한 플랫폼의 플레이어 카피에 아래와 같은 방법으로 바인딩된다.

3.4.1 다운로드 파일의 구조

다운로드될 파일은 헤더와 암호화된 페이로드로 구성된다. 헤더는 고정된 크기를 가지고 있어, 플레이어가 페이로드의 시작 위치를 쉽게 파악할 수 있도록 한다. 헤더에는 512-bit의 R 값이 포함되어야 하고, 또한 페이로드의 인코딩 방식(예: MP3)이 나타나야 한다. 이외의 헤더 정보로 페이로드의 이름, 저자, 연주자 등의 정보가 포함될 수 있다. 암, 복호화에 필요한 정보는 R로서 이것의 의미 및 계산 방법은 바로 아래에서 설명한다. 헤더에 바로 이어서 암호화된 미디어 콘텐츠 내용이 덧붙여진다.

3.4.2 서버에서의 콘텐츠 파일 암호화

콘텐츠 파일 F에 대해 클라이언트 플레이어 $\langle id,pw,Z,L\rangle$로부터 다운로드 요청을 받았을 때 서버가 수행하는 암호화 과정은 아래와 같다(그림 3). 이 때 사용하는 HC-256[3]는 유럽의 eSTREAM가 선택한 매우 빠른 스트림 암호화 알고리즘들 중 하나로 각각 256-bit의 비밀 키 K와 초기 벡터 IV를 이용하여 뛰어난 보안성을 가지는 랜덤 키스트림(keystream)을 생성한다. HC-256은 특허가 걸려있지 않아 자유로이 사용 가능하다.

(i) 적절한 스트림 암호화 알고리즘을 사용하여 랜덤 키스트림을 생성한다. 이 키스트림에서 512 bits 를 추출하여 N이라고 한다.

(ii) $Z\oplus N$을 계산하여, 결과 값을 256-bit 크기의 두 부분으로 나누어 처음 값을 K, 나중 값을 IV로 놓는다. 이제 HC-256 스트림 암호화를 비밀 키 K와 초기 벡터 IV를 가지고 실행하여 랜덤 키스트림을 생성하고, 이 키스트림과 콘텐츠 파일 F를 비트 단위 XOR 연산하여 암호화된 콘텐츠 파일 F^*를 생성한다. 결과 파일 F^*의 크기는 F와 동일하다.

(iii) $L=(<a_0,b_0>,<a_1,b_1>,...,<a_n,b_n>)$을 이용하여 파일 F^*에 대한 W 값을 계산한다. $W=[F^*(a_0,b_0)\|F^*(a_1,b_1)\|...\|F^*(a_n,b_n)]$. 단, 각 a_i,b_i 는 $\bmod length(F^*)$를 추가하여 계산하고, 연결연산(‖)은 필요한 만큼 반복한 후 512-bit 크기의 prefix를 취한다.

(iv) $R=N\oplus W$를 계산하여, 이 R 값을 헤더에 포함시킨다. 기타의 헤더 정보는 적절한 값을 가지도록 한다. 헤더 및 암호화된 콘텐츠 F^*를 결합하여 다운로드할 파일을 완성한다.

3.4.3 클라이언트에서의 콘텐츠 파일 복호화와 재생

암호화된 콘텐츠 F^*를 재생하기 위해 클라이언트의 플레이어는 다음과 같이 동작한다(그림 4).

(i) 암호화된 콘텐츠 F^*에 대하여 플레이어는 고유의 코드를 실행시켜서 512-bit의 W 값을 구한다.

(ii) 파일의 헤더에서 R 값을 읽고 $R\oplus W$를 계산한다. 파일의 훼손이 없다면 $R\oplus W=(N\oplus W)\oplus W=N\oplus(W\oplus W)=N$ 이 성립될 것이다.

(iii) 클라이언트 플랫폼에 고유한 정보를 즉석에

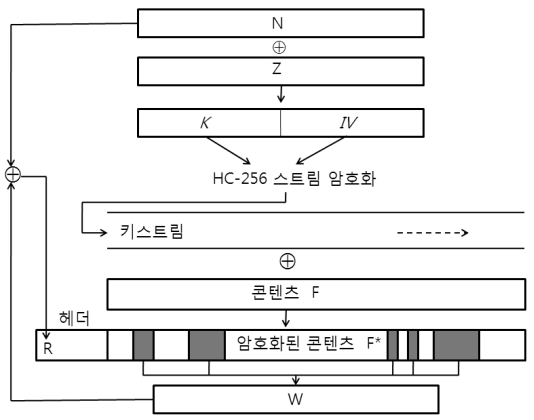

그림 3. 플레이어 카피 $\langle id,pw,Z,L\rangle$를 위해 서버에서 콘텐츠 파일 F를 암호화 하는 과정

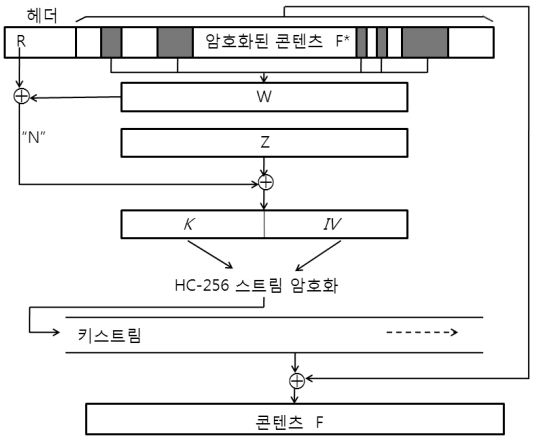

그림 4. 플레이어 카피 $\langle id,pw,Z,L\rangle$에서 암호화된 콘텐츠 F^*를 복호화 하는 과정

서 읽고 이로부터 512-bit 크기의 고유 값 Z를 계산하는 코드를 실행하여 Z 값을 얻는다.

(iv) Z⊕((ii)*의 결과*) 를 계산하여, 결과 값을 256-bit 크기의 두 부분으로 나누고 처음 값을 K, 나중 값을 IV로 놓는다. 이제 HC-256 스트림 암호화를 비밀 키 K와 초기 벡터 IV를 가지고 실행하여 랜덤 키스트림을 생성하고, 이 키스트림과 암호화된 콘텐츠 F^*를 비트 단위 XOR 연산하면서 콘텐츠 파일 F를 복원한다.

(v) 플레이어는 콘텐츠 F를 앞부분부터 연속해서 복원을 하면서, 복원된 내용에 대해 즉시 재생을 시작한다. 재생을 위해 콘텐츠 전체를 작은 단위로 나누며, 어떤 한 단위 a의 재생이 진행되는 동안 다음 단위 $a+1$의 복호화를 진행하고, 단위 a의 재생이 완료되어 단위 $a+1$의 재생이 시작되면 바로 단위 a의 공간에 단위 $a+2$의 복호화된 내용이 중첩되어 써지도록 한다. 이런 과정을 콘텐츠 끝까지 반복하여, 복호화가 완료된 콘텐츠가 노출되는 분량 및 시기를 대폭 줄인다.

Ⅳ. 보안성 분석

제안 기법이 다양한 공격 시나리오에 대해 어떻게 대비되어 있는지 서술한다. 단, HC-256 알고리즘 자체에 대한 암호해독 공격은 고려하지 않는다.

4.1 콘텐츠 및 플레이어 카피 파일의 불법 복제

콘텐츠 F가 어떤 플랫폼의 플레이어 카피 P에 바인딩되어 암호화된 콘텐츠 F^*를 생성한다고 하자. 콘텐츠 F의 암호화에 사용된 HC-256 알고리즘의 (K,IV) 쌍은 플레이어 카피에 따라서 다름(Z 값을 계산하는 고유 코드가 다르고, 이 값이 (K,IV) 계산에 사용되기 때문에)을 이용한다. (경우1) F^*를 다른 플랫폼으로 복사하여 플레이어 카피 Q에 의한 재생 시도: P와 Q가 다르기 때문에 Q에서 F^*를 복호화 하려고 하면 F가 아닌 F'가 만들어져 재생에 실패한다. (경우2) F^*와 플레이어 카피 P를 모두 다른 플랫폼으로 복사하여 새로운 플랫폼에서 재생 시도: P는 플랫폼의 MAC 주소 같은 고유 정보를 읽어서 Z를 계산하기 때문에 새로운 플랫폼에서 P를 실행하는 경우 원래 P가 사용되던 플랫폼에서와 다른 Z 값을 계산한다. 서버에서는 원래의 플랫폼에서 계산된 Z 값을 사용하여 F^*를 만들었기 때문에, 복호화는 실패한다.

4.2 콘텐츠 파일을 통한 키 획득 시도

만약 어떤 콘텐츠의 원본(암호화되기 전) 파일 F를 획득할 수 있다면, 이것과 암호화된 콘텐츠 F^*를 가지고 암호화에 사용된 키 값들을 복원하려 시도할 수도 있다. 특히 콘텐츠가 인기 있는 내용이라면 여러 경로를 통해 암호화되기 전에 노출될 가능성이 있다. 이제 공격자가 $\langle F,F^* \rangle$의 쌍을 가지고 플레이어 카피 P를 위해 F^*를 만들 때 사용한 비밀 값을 찾으려 시도한다고 하자. 일단 스트림 암호화의 특성 상 F와 F^*의 비트 단위 XOR를 통해 HC-256의 키스트림을 알아낼 수 있다. 하지만 키스트림만 가지고 이것을 만들기 위해 사용했던 (K,IV)는 알아낼 수 없다. 만약 그것이 가능하다면 이후에 나타날 키스트림 내용을 미리 알 수 있으므로 HC-256이 깨지는 것이나 마찬가지이기 때문이다. 한편, 공격자는 동일한 콘텐츠 F에 대해 두 가지 암호화된 콘텐츠 F^*와 F^{**}를 가지고 있을 수도 있다. 이 경우에도 공격자는 두 가지 암호 콘텐츠를 얻기 위해 사용했던 키스트림의 XOR 값만을 알 수 있을 뿐, 어떤 비밀 값도 얻을 수 없다.

4.3 특정 콘텐츠 키의 다른 콘텐츠에의 적용 시도

가능성이 매우 낮은 상황이지만, 플레이어 카피 P를 위해 암호화된 콘텐츠 F^*를 만들 때 사용했던 HC-256의 (K,IV) 값이 공격자에게 알려졌다고 하자. 이 (K,IV) 쌍은 동일 플랫폼의 F가 아닌 어떤 콘텐츠 F'에 대해서도 무의미하다. 그 이유는 콘텐츠가 달라지면 플레이어에서 (K,IV)를 만들기 위해 계산하는 W 값이 달라지기 때문이다. 만약 이 (K,IV)를 다른 플레이어 카피 Q의 플랫폼에서 사용한다면, 그 공격 대상이 콘텐츠 F를 암호화한 F^{**}인 경우에도 성공하지 못한다. 그 이유는 Q의 실행 중 계산하는 Z 및 W 값이 P에서 (K,IV)를 만들기 위해 사용한 Z 및 W 값과 다르기 때문이다. 따라서 F^{**}를 HC-256의 (K,IV)로 키스트림을 만들어서 복호화시킨 내용은 F와 다르게 된다. Q의 플랫폼에서 이 (K,IV)로 F가 아닌 F'을 암호화한 콘텐츠 F'^*를 복호화 시키려는 경우 대상 콘텐츠가 다르기 때문에 더욱 성공할 가능성이 없다.

4.4 소프트웨어 역 공학으로 플레이어 카피 분석

각 플레이어 카피마다 다르게 탑재된 고유 코드로부터 실제 콘텐츠 복호화에 필요한 키를 만들기 위한 비밀 값들을 얻을 수 있고, 이 비밀 값들은 플

레이어가 동작하는 플랫폼 및 각 콘텐츠 마다 다르게 지정된다. 따라서 만약 공격자가 Z 및 W를 계산하는 코드를 알아낼 수 있다면 사실상 그 플레이어 카피에 바인딩된 어떤 콘텐츠도 다 풀어서 재생시킬 수 있다. 각 콘텐츠의 헤더에 포함된 R 값은 암호화되어 있지 않으며, 플레이어 프로그램이 즉석에서 읽어오는 MAC 주소 등의 고유 값은 공격자가 어렵지 않게 얻을 수 있기 때문이다. 또한 플레이어 프로그램을 공격하여 (K,IV)가 만들어지는 시점 직후에 이를 가로채거나, 콘텐츠가 복호화되어 평문으로 바뀌는 시점에 이 평문 내용을 가로챈다면 제안 기법은 깨질 수 있다. 단, 제안 기법에서는 콘텐츠가 복호화되어 평문으로 노출되는 시기 및 그 분량을 줄여서 그런 공격이 어렵도록 만들었다.

V. 결 론

강력한 복제 방지 기능과 사용자 불편 최소화를 목표로 개발된 범용 DRM 기법을 제시하였다. 제안 기법은 가장 보편적이고 널리 사용하는 다운로드-무한재생 라이센스 정책만을 허용하여 적용 구조를 단순화시킴으로써 효율성과 적용 가능성을 높였다. 범용성 측면에서 제안 기법은 전처리 단계의 키 생성 관련 코드 부분 삽입을 통해 기존의 어떤 미디어 플레이어에도 미디어 유형에 상관없이 사용할 수 있다. 효율적 재생을 위해서 스트림 암호화를 사용함으로써 콘텐츠가 파일의 크기에 상관없이 복호화 이후에 연속 재생이 가능하도록 하였다. 키 생성 및 관리 확장성 측면에서는 모든 콘텐츠가 고유의 키를 사용하면서도 관리하는 키의 개수는 참여하는 사용자 플랫폼의 수에만 비례하도록 설계하였다. 제안 기법은 보안성 제공의 측면에서도 여러 강점을 가진다. 어떤 식으로 콘텐츠 및 플레이어 카피를 복사하더라도 각 파일은 원래의 바인딩된 플랫폼의 플레이어에서만 재생이 됨을 보장한다. 평문의 콘텐츠 파일을 획득하더라도 암호화에 사용된 HC-256 입력 값을 알아낼 수 없으며, 설사 특정 콘텐츠의 HC-256 입력 값이 노출되었다 하더라도 이것을 다른 콘텐츠에 적용하는 것은 효과가 없음을 보였다. 소프트웨어 역 공학을 이용한 공격에 절대적으로 안전한 것은 아니지만, 제안 기법은 복호화가 진행된 콘텐츠 내용이 프로그램 상에서 그대로 노출되는 분량 및 노출 시기를 줄이는 방법을 제시했다.

고유 값 Z를 계산할 때 고정된 코드 로직에서 사용하는 숫자만을 플레이어 카피 별로 바꾸는 것 대신에, 코드 로직 자체를 랜덤하게 자동 생성하는 것이 보안성 향상을 위해 필요하리라 판단한다.

참 고 문 헌

〔1〕 D. Sohn, “Understanding DRM,” *ACM Queue*, pp. 32-37, Nov./Dec., 2007.

〔2〕 B.A. LaMacchia, “Key Challenges in DRM: An Industry Perspective,” in Proc. Digital Rights Management Workshop, pp. 51-60, 2002.

〔3〕 H. Wu, “A New Stream Cipher HC-256,” in Proc. of 11th Int’l Workshop on Fast Software Encryption, *LNCS 3017*, pp. 226-244, 2004.

〔4〕 Microsoft, Microsoft Windows Media Digital Rights Management (WMDRM), http://www.microsoft.com/windows/windowsmedia/drm/default.aspx

〔5〕 S. Gilbertson, Windows Media DRM Hacked Yet Again, http://blog.wired.com/monkeybites/2007/07/windows-media-d.html, 2007.

〔6〕 AppleForum, How FairPlay Works, http://www.appleforum.com/, 2007.

〔7〕 Digg Inc., How to break iTunes DRM, http://digg.com/software/How_To_Break_iTunes_DRM_2

〔8〕 S. Bhatt, R. Sion, B. Carbunar, “A Personal Mobile DRM Manager for Smartphones,” *Computers & Security*, 28(6), pp. 327-340, 2009.

〔9〕 N.P. Sheppard, R. Safavi-Naini, “Sharing Digital Rights with Domain Licensing,” in Proc. 4th Int’l Workshop on Contents Protection and Security, pp. 3-12, 2006.

〔10〕 Q. Liu, R. Safavi-Naini, N.P. Sheppard, “Digital Rights Management for Content Distribution,” in Proc. Australasian Information Security Workshop Conf. on ACSW Frontiers 2003, pp. 49-58, 2003.

〔11〕 C. Kaufman, R. Perlman, M. Speciner, *Network Security: Private Communication in a Public World (2nd Edition)*, Chapter 19 SSL/TLS pp. 477-482, Prentice Hall, 2002.

박 준 철 (Jun-Cheol Park) 종신회원

1986년 서울대 계산통계학과
1988년 KAIST 전산학과 석사
1998년 U. of Maryland, College Park, 전산학 박사
현재 홍익대 컴퓨터공학과 부교수
<관심분야> 네트워크/시스템 보안