情報保護學會論文誌
第16卷 第5號, 2006. 10

능동형 콘텐츠 지원을 위한 OMA DRM 프레임워크의 확장

김 후 종,[1† 2‡] 정 은 수,[1†] 임 재 봉,[2‡]
[1]SK 텔레콤, [2]국민대학교

Extending the OMA DRM Framework for Supporting an Active Content

Hoo-Jong Kim,[1† 2‡] Eun-Su Jung,[1†] Jae-Bong Lim[2‡]
[1]SK Telecom, [2]Kookmin University

요 약

무선 인터넷 통신의 빠른 성장으로 차세대 이동 단말들에서 이미지, 음악, 비디오 및 응용들과 같은 무선 디지털 콘텐츠들의 배포가 가능하게 되었다. 불법 복사 방지, 사용 권한 제어, 인증 기능이 없는 상태에서 이동 단말이 빠른 속도로 통신 채널을 확대하기 위한 주요 수단이 되면, 인증되지 않은 이동 단말을 통해 무선 디지털 콘텐츠는 불법적으로 복사, 편집, 배포된다.

본 논문은 일반적인 OMA DRM v2.0의 목적과 기능에 대해서 살펴본다. OMA는 모바일 DRM을 위한 공개 표준을 개발하고 있는 유일한 곳이다. 다음으로 능동형 콘텐츠의 특징에 대해서 소개하고, 능동형 콘텐츠와 수동형 콘텐츠의 차이에 대해서 설명한다. 능동형 콘텐츠를 빠르게 재생하기 위한 OMA 기반 DRM 프레임워크를 제안한다. 본 프레임워크에는 콘텐츠 부분 암호화를 위한 DCF 확장, 콘텐츠 암호화 키 관리, 능동형 콘텐츠를 위한 렌더링 API가 포함된다. 실험 결과는 제안한 방법을 통해 QoE를 만족할 수 있는 정도로 충분히 빠르게 능동형 콘텐츠를 렌더링할 수 있음을 보여 준다. 본 프레임워크는 이동 단말 환경을 위해 제안된 것이지만, 휴대용 재생기, 셋탑 박스, 개인 컴퓨터에서도 적용할 수 있다.

ABSTRACT

With the rapid growth of the wireless Internet communication, a new generation of mobile devices have made possible the broad distribution of mobile digital contents, such as image, music, video, games and applications over the wireless Internet. Mobile devices are rapidly becoming the major means to extend communication channels without copy protection, usage rule controlling and authentication. As a result, mobile digital contents may be illegally altered, copied and distributed among unauthorized mobile devices.

In this paper, we take a look at Open Mobile Alliance (OMA) DRM v2.0 in general, its purpose and function. The OMA is uniquely the focal point for development of an open standard for mobile DRM. Next, we introduces features for an active content and illustrates the difference between an active content and an inactive content. Enabling fast rendering of an active content, we propose an OMA-based DRM framework. This framework include the following: 1) Extending DCF Header for supporting an selective encryption, 2) Content encryption key management, 3) Rendering API for an active content. Experimental results show that the proposed framework is able to render an active content fast enough to satisfy Quality of Experience. This framework has been proposed for a mobile device environment, but it is also applicable to other devices, such as portable media players, set-top boxes, or personal computer.

Keywords : *Digital Rights Management, Active Contents, Open Mobile Alliance*

Ⅰ. 서 론

무선 인터넷 환경의 꾸준한 발전과 고성능 컨버전스형 단말기의 보편화로 인하여 무선 콘텐츠의 유통은 꾸준히 증가하고 있다. 이에 따라 이동 통신사, 단말 제조사, 플랫폼 제조사, DRM 솔루션 개발사들은 무선 콘텐츠를 안전하게 보호하고 저작권자의 합법적인 권리 보장 및 다양한 비즈니스 모델 적용을 위해 표준화된 디지털 저작권 관리 (Digital Rights Management, DRM) 시스템을 이동 단말에 적용하는 추세에 있다.

OMA (Open Mobile Alliance)는 무선 인터넷 환경에서 시장 요구에 따른 신속한 서비스 규격 제정 및 액세스 망에 독립적인 다양한 단말과 시스템간의 상호호환성을 확보한다는 목표하에 2002년 6월에 발족한 국제 무선 인터넷 표준 기구 중 하나이다. OMA는 무선 DRM 표준 규격을 서두르지 않을 경우 DRM 간의 상호호환이 어려워 콘텐츠 유통에 큰 장애가 될 것이라고 판단하고 OMA DRM 표준화 작업에 착수하여 지난 2004년에 OMA DRM v1.0 Approved Enabler를 발표하였다.[1] 하지만, OMA DRM v1.0은 시장의 요구사항을 빠르게 대응하기 위해 나온 규격이었기 때문에 많은 기술적 문제점을 안고 있었고 이를 해결하기 위해 곧바로 OMA DRM v2.0 표준 규격 작성을 진행하게 된다.[2] 결국 OMA DRM v2.0 Approved Enabler가 2006년 3월에 정식으로 발표되었다.

본 논문은 OMA DRM v2.0을 상용 서비스 특히 게임 콘텐츠 서비스에 적용할 때 생기는 문제들을 발견하고 이를 해결하기 위한 방안을 제안하는 것이 목적이다. 특히, 최근에 이동 통신사들이 빠르게 선보이고 있는 능동형 (Active) 콘텐츠 기반형 서비스들을 OMA DRM v2.0 규격에 기반하여 구축할 때 발생할 수 있는 클라이언트 입장에서의 문제들을 고려한다. 능동형 콘텐츠는 게임, 문서, 응용 프로그램 등과 같이 콘텐츠 자체가 사용자에 의해서 쉽게 변경이 되거나, 이동 단말의 플랫폼에 종속적인 콘텐츠들을 일컫는다.

OMA DRM v2.0을 능동형 콘텐츠 중 하나인 게임 콘텐츠에 적용할 경우, 콘텐츠 실행에 걸리는 지연 시간이 길어서 이용자의 불편이 크다. 이를 해결하기 위해 본 논문은 DCF (DRM Content Format)[3] 규격의 확장 헤더를 이용하여 부분 암호화를 위한 헤더 규격을 제안한다. 또한 OMA DRM v2.0 규격의 보안 요구 사항을 만족하는 범위 내에서 안전한 키 관리를 위한 구조를 모색하고, 콘텐츠 특성에 맞는 렌더링 API 구조를 제시하여 콘텐츠 실행의 지연 시간을 줄이도록 한다.

본 논문의 구조는 다음과 같다. 2장은 무선 DRM과 OMA DRM v2.0 표준 규격을 상세히 분석하도록 한다. 3장은 본 논문에서 정의한 능동형 콘텐츠와 수동형 콘텐츠에 대해서 기술하고 이동 단말에 적용 시 발생할 수 있는 문제들을 제기한다. 4장은 제기된 문제들을 해결하기 위한 개발 방안을 제안하고, 5장에서 제안된 개발 방안에 대한 실험 결과를 확인하도록 한다.

Ⅱ. OMA DRM v2.0 표준

OMA DRM v2.0 표준 규격은 이전 버전의 가장 큰 취약점 중에 하나로 평가된 보안성 강화에 많은 노력을 기울였다. 기존 OMA DRM v1.0은 무선 서비스 업체들의 콘텐츠 유통에 따르는 DRM 요구 사항을 빠르게 수용하기 위해 주로 DRM 콘텐츠 포맷 (DRM Content Format, DCF), ODRL[12]에 기반한 권리 표현 언어 (Rights Expression Language, REL) 정의, DCF와 Rights의 다운로드 방식 등에 주안점을 둔 규격이다. 이와 다르게 OMA DRM v2.0 표준 규격은 PKI에 기반하여 단말 인증 및 DCF, RO (Rights Object)의 안전한 전달 등과 같은 보안 체계를 구축하고, 다양한 비즈니스 모델을 수용할 수 있는 서비스 시나리오 지원에 초점이 맞춰진 규격이다.

2.1 기술 시나리오

OMA DRM v2.0 규격의 요구 사항 및 아키텍처 문서는 OMA DRM v2.0을 통해서 실현 가능한 기술 시나리오들을 기술하고 있다. 그림 1은 기술 시나리오를 적용할 수 있는 아키텍처와 액터들을 묘사한 그림이다. 기술 시나리오는 사용자가 콘텐츠와 Rights를 어떻게 접근하며, RI (Rights Issuer)가 RO를 어떻게 안전하게 전달할 수 있는지를 명시한다. CP (Content Provider)는 최종 사용자에게 전달될 원본 콘텐츠를 제공한다. 원본 콘텐츠는

접수일: 2006년 7월 7일; 채택일: 2006년 9월 22일
† 주저자, hjkim2@sktelecom.com
‡ 교신저자, ljb@kookmin.ac.kr

User에게 그대로 제공되지 않으며, CI를 통하여 전달된다. User는 CI를 통하여 콘텐츠 정보를 브라우징하고 필요한 콘텐츠를 내려 받는다.

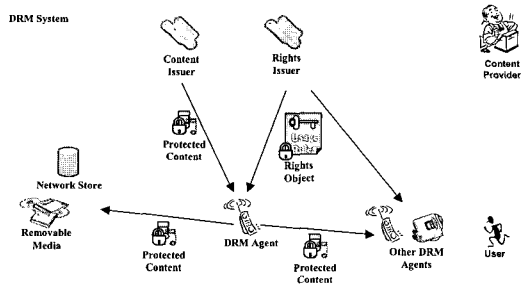

그림 1. OMA DRM v2.0 기능적 아키텍처

CI는 원본 콘텐츠를 암호화하며 비즈니스 모델과 과금 정책에 따라 암호화된 콘텐츠에 대한 Rights Offer를 정의한다. User는 암호화된 콘텐츠에 대한 RO를 구매한 후, RI를 통해서 내려 받는다. RI는 RO를 내려 주기 전에 User 단말에 설치된 DRM Agent를 인증한다. RI와 DRM Agent가 성공적으로 인증될 경우 RO를 내려 준다. User는 Removable Media에 암호화된 콘텐츠와 RO를 저장할 수도 있다. CI로부터 내려 받은 암호화된 콘텐츠를 다른 User에게 재배포할 경우, 다른 User는 다시 RI를 통하여 암호화된 콘텐츠를 사용할 수 있는 RO를 받을 수 있다.

2.2 OMA DRM v2.0 데이터 개체

2.2.1 DCF (DRM Content Format)

DCF는 OMA DRM에서 정의한 암호화된 콘텐츠의 파일 포맷이다. DCF는 불연속적인 미디어 (Discrete Media) 형태의 콘텐츠를 보호하고 콘텐츠 메타 정보를 기술할 수 있다. 연속적인 미디어 (Continuous Media) 형태의 경우에는 PDCF를 이용한다. DCF는 DCF의 메타 정보를 기술하는 헤더와 암호화된 콘텐츠가 저장되고 콘텐츠 메타 정보를 기술할 수 있는 컨테이너로 구성된다. 하나 이상의 컨테이너는 DCF에 저장이 가능하며, 각 컨테이너마다 별도의 콘텐츠 ID를 부여 할 수 있다.

2.2.2 RO (Rights Object)

RO는 사용자가 특정 DCF에 대해 합법적으로 허용된 사용 권한과 복호화 키 정보가 ODRL에 기반한 OMA DRM v2.0 REL[4]로 표현된 XML 형태의 데이터 파일이다. RI는 다양한 방법으로 RO를 발급할 수 있으나, 가장 안전한 방법은 DRM Agent가 RI에 접속하여 ROAP을 통해서 획득하는 것이다. RO에는 DCF를 복호화할 수 있는 콘텐츠 복호화 키 (Content Encryption Key, CEK)를 가지고 있다. CEK는 REK (Rights Encryption Key)로 암호화 되어 있는데, CEK를 REK로 암호화함으로써, DCF를 암호화한 CEK가 노출되지 않기 때문에 DCF를 재배포하더라도 위험이 없는 것이다.

2.2.3 ROAP PDU

RI와 DRM Agent는 ROAP 통신을 통해 필요한 정보를 주고 받는다. ROAP PDU는 ROAP 통신에서 사용되는 메시지 규격이다. ROAP PDU는 ROAP Trigger 메시지와 ROAP Request / Response 메시지로 구성된다. ROAP PDU는 XML로 표현되며, 단말 등록, RO 획득, 도메인 가입 및 탈퇴를 위해 RI와 DRM Agent가 알아야 할 필요한 정보들을 교환할 수 있도록 구성되어 있다.

2.3 ROAP

ROAP은 RI와 DRM Agent가 상호간에 필요한 정보를 안전하게 교환하기 위해 정의된 프로토콜이다. OMA DRM v2.0 규격은 목적과 성격에 따라 5개 프로토콜을 정의하고 있다. 각 프로토콜은 ROAP Trigger에 의해 구동되지만, DRM Agent의 상태에 따라 다른 프로토콜을 먼저 자동으로 실행시키도록 정의되 있다. 예를 들어, 단말 등록이 안된 상황에서 RO 획득을 요청하면, RO 획득 Trigger를 이용하여 먼저 단말 등록 프로토콜을 실행 시킨 후, RO 획득 프로토콜이 실행된다.

2.3.1 등록

4-pass Registration 프로토콜은 DRM Agent와 RI가 상호간에 필요한 보안 정보를 안전하게 교환하기 위해 정의되었으며, DRM Agent가 처음 접근하는 RI에 대해서 최초로 실행된다. 4-pass Registration 프로토콜이 실행된 후에는 DRM Agent와 RI에 RI Context와 Device Context라는 DRM 정보가 각각 생성된다. 4-pass Registration 프로토콜은 DRM Agent가 탑재된 단말의 클록이 잘못되거나 RI / Device Context가 만기되

기 전까지는 다시 실행되지 않는 프로토콜이다.

2.3.2 RO 획득

단말에 설치된 DCF를 복호화하여 사용하기 위해서는 RI가 발급한 RO가 있어야 한다. DRM Agent는 RI로부터 RO를 획득하기 위해서 2-pass/1-pass RO Acquisition 프로토콜을 사용한다. 2-pass RO Acquisition 프로토콜은 RI와 DRM Agent간에 상호 인증을 통하여 RO를 안전하게 요청하고 전달하는 프로토콜이다. 1-pass RO Acquisition 프로토콜은 이동 통신망의 푸시 기반 전달 기법을 이용한다. 1-pass RO Acquisition 프로토콜은 상호 인증을 지원하지 않기 때문에 Trigger나 Request 메시지 없이 바로 Response 메시지만 DRM Agent로 전달된다.

2.3.3 Domain 가입 및 탈퇴

2-pass Join/Leave Domain 프로토콜은 DRM Agent가 Domain 가입/탈퇴에 필요한 정보를 교환하기 위한 프로토콜이다. 2-pass Join Domain 프로토콜은 DRM Agent가 유효한 RI Context 정보를 가지고 있어야만 동작한다. 2-pass Join Domain 프로토콜을 통해 DRM Agent는 Domain Context가 생성된다. Domain Context는 DRM Agent가 2-pass RO Acquisition 프로토콜을 통해 Domain RO를 획득하여 설치하고 사용할 때 필요한 정보이다.

DRM Agent는 2-pass Leave Domain 프로토콜을 통해 안전하게 Domain을 탈퇴할 수 있으며, 탈퇴한 직후부터는 해당 Domain에 대한 새로운 Domain RO를 사용할 수 없게 된다.

2.4 키 관리 및 전송

OMA DRM v2.0은 DRM Agent의 공개키를 이용하여 DCF를 복호화할 수 있는 키 재료 (Key Material)와 RO의 무결성을 검증하는데 필요한 키 재료를 전달할 수 있는 키 전송 및 관리 규격을 정의하고 있다. DRM Agent는 RO에서 암호화된 키 재료 C (C는 C1과 C2를 연결시킨 데이터)로부터 C1을 추출하여 DRM Agent의 개인키를 이용하여 복호화한 후, Z를 얻는다. 이렇게 획득한 Z에 KDF (Key Derivation Function)를 적용하면 KEK (Key Encryption Key)를 다시 얻을 수 있다. KEK를 이용하여 C로부터 추출한 C2에 AES-WRAP[8] 함수를 적용하면 REK를 추출할 수 있다. 추출된 REK를 RO의 〈rights〉에 명시된 KeyInfo의 암호화된 키 데이터에서 AES-WRAP 함수를 적용하여 CEK를 추출할 수 있게 된다.

Ⅲ. 능동형 콘텐츠

OMA DRM 규격은 콘텐츠에서 추출한 공통적인 속성에 기반하여 필요로 하는 전반적인 기능을 제공해준다. 이는 DRM 규격이 콘텐츠 특성에 관계없이 적용 가능한 장점이 있지만, 각 콘텐츠에 대한 특성이 반영되지 않음으로 서비스에 적용할 때 큰 장애에 부딪치는 경우가 많다. 서비스 적용 입장에서 볼 때 무선 디지털 콘텐츠의 특성을 파악하여 DRM을 적용하는 것은 유용한 작업이다. 이를 위해 먼저 능동형 콘텐츠에 대해서 정의하고, 수동형 콘텐츠와의 차이점을 비교해 보고자 한다.

본 논문에서는 능동형 콘텐츠를 "콘텐츠가 사용되는 시점에 따라 상태 정보를 가지며, 이러한 상태 정보는 추후 콘텐츠를 다시 사용할 때 영향을 줄 수 있고, 콘텐츠의 접근 형태가 불규칙한 콘텐츠"로 정의한다. 상태 정보를 가진다는 의미는 사용자에 의해 콘텐츠가 한 번 사용된 이후 콘텐츠의 변경된 사실이나 변경된 값을 콘텐츠 자체에 포함한다는 것이다. 예를 들어, 콘텐츠에 따른 일반적인 서비스 형태를 가정할 때, 문서, 실행 프로그램, 게임 등은 능동형 콘텐츠, 음악, 이미지, 동영상 등은 수동형 콘텐츠로 분류할 수 있다. 이와 같은 특성을 고려할 때, 능동형 콘텐츠는 대화형 콘텐츠 (Interactive Content), 수동형 콘텐츠는 비대화형 (Non- Interactive Content)라고 부를 수 있다. 표 1은 능동형 콘텐츠의 특성을 설명하기 위해 수동형 콘텐츠와의 차이점을 비교하였다.

표 1. 능동형 콘텐츠와 수동형 콘텐츠의 비교

	능동형 콘텐츠	수동형 콘텐츠
대표적인 콘텐츠	문서, 실행 프로그램, 게임	음악, 이미지, 동영상
플랫폼 종속성	플랫폼이 콘텐츠 핸들링의 주체로써, 종속성이 큼	별도의 플레이어가 핸들링의 주체로써, 종속성이 적음
데이터 접근 형태	주로 임의 접근	주로 순차적 접근
콘텐츠 변경 여부	사용자나 응용에 의한 잦은 변경	거의 발생하지 않음

3.1 플랫폼 종속성

수동형 콘텐츠로 분류할 수 있는 음악이나 이미지, 동영상 콘텐츠의 경우에는 일반적으로 플레이어가 콘텐츠를 다룬다. 대부분의 수동형 콘텐츠는 표준 포맷을 따르므로 플레이어 자체만을 수정해서 DRM 지원이 가능하므로, 플랫폼의 추가적인 수정사항이 적다. 하지만 실행 프로그램이나 게임과 같은 능동형 콘텐츠의 경우에는 콘텐츠를 사용할 수 있는 환경에 종속적인 경우가 많다. 실행 프로그램은 운영 체제에 종속되며, 게임은 플레이어나 운영체제에 종속적이다. 그러므로 DRM을 능동형 콘텐츠에 적용하기 위해서는 종속되는 플랫폼 자체에 DRM이 포함되어야 한다. 다양한 형태의 주체들이 공유하는 운영체제와 같은 플랫폼은 특정 콘텐츠만을 위한 기능을 추가하기 위해서는 어려움이 많으며, 적용되는 DRM 기능 또한 플랫폼에 특화된 형태로 설계되어야 한다.

3.2 데이터 접근 형태

수동형 콘텐츠를 사용할 경우에는 이미지처럼 전체 데이터 모두를 필요로 하거나, 음악과 동영상처럼 순차적인 데이터 접근 방식이 사용된다. 음악과 동영상의 경우 특정 부분에 대한 재생도 가능하지만 이 역시 데이터 접근 방식 측면에서는 연속성이 존재한다. 따라서 암호화된 형태의 수동형 콘텐츠를 렌더링할 경우 처리 응답 지연 시간을 줄이기 위해기 위해 버퍼링과 같은 기능들을 적용할 수 있다. 하지만 능동형 콘텐츠의 경우에는 사용자의 사용 방식, 실행 형태에 따라서 데이터 접근 위치가 임의로 달라질 수 있으며, 결과적으로 버퍼링 등과 같은 기능을 제공할 수 없다. 이는 요청하는 시점에 암호화된 콘텐츠를 실시간으로 복호화 처리해야 하는 것을 의미한다. 하지만 데이터 처리 능력이 상대적으로 떨어지는 이동통신 환경에서 콘텐츠 전체에 대한 실시간 복호화 기능을 제공할 경우 심각한 처리 응답 시간 문제가 발생할 수 있다. 더욱이 빠른 데이터 처리 시간을 요구하는 애플리케이션이나 게임과 같은 능동형 콘텐츠를 사용하는 경우에는 문제가 더욱 심각하다.

3.3 콘텐츠의 변경 여부

일반적으로 DRM 시스템은 콘텐츠를 암호화 (패키징)하여 기밀성을 유지하면서, 신뢰할 수 없는 네트워크를 통해 유통되는 콘텐츠의 불법 복제를 막고 합법적인 사용을 제어하는 것이 목적이다. 따라서 콘텐츠의 전체를 암호화하는 것이 보편적이다. 하지만, 콘텐츠의 특성과 사용 권한에 따라 콘텐츠의 일부 내용이 지속적으로 변경하는 경우가 발생할 수 있다.

예를 들어, 이동 통신사들이 서비스하는 일부 게임 콘텐츠들의 포맷을 보면, 파일 하나가 그 자체로 게임 콘텐츠인 것처럼 보이지만, 콘텐츠의 내용을 보면, 여러 개의 다른 파일들로 구성된 파일의 모음임을 할 수 있다. 즉, 게임 실행 파일 자체 뿐 만 아니라, 환경 설정 파일, 게임 이미지 파일, 게임 사운드 파일들과 같은 게임 리소스 파일들이 함께 포함되어 패키징된 형태의 파일임을 알 수 있다. 이와 같은 게임 콘텐츠는 사용자가 게임 콘텐츠를 실행시켜 사용하는 과정에서 환경 설정 파일의 변경이 발생할 수 있다. 이와 같인 콘텐츠가 이용자에 의해 사용되는 과정에서 변경되는 정보를 콘텐츠의 상태 정보라고 말할 수 있다.

능동형 콘텐츠의 불법 복제를 막고 합법적인 사용을 제어하고, 재배포를 지원하기 위해서는 바로 앞 구절에서 설명한 일반적인 DRM 시스템으로는 한계가 있다. 본 논문에서 확장하려는 OMA DRM v2.0 역시 기존 DRM 시스템의 구조와 크게 다르지 않으나 표준 규격에 위배되지 않는 범위에서 콘텐츠의 변경을 지원할 수 있는 방안을 제시한다. OMA DRM v2.0 DCF 규격이 앞서 3장에서 언급한 콘텐츠 특성과 상관없이 전체 콘텐츠를 암호화하는 방식과 전혀 암호화하지 않는 방식만을 지원하는 가장 큰 이유는 OMA DRM v2.0 DCF 규격이 보편적인 콘텐츠의 공통적인 요구 사항만을 반영했기 때문이다. 본 논문에서 제안하는 부분 암호화 방식은 콘텐츠의 개별적인 특성이 반영되지 않을 경우, 콘텐츠의 기밀성이 노출될 우려가 있다.

Ⅳ. 능동형 콘텐츠를 위한 DRM Agent 설계

능동형 콘텐츠가 3장에서 나열된 기준에 따라 유사한 특성들을 가지고 있다 하더라도, 각 콘텐츠의 특성에 따라 다소간의 다른 특성들도 가지고 있다. 따라서, 본 논문은 여러 가지 형태의 능동형 콘텐츠 중에서 게임 콘텐츠를 선택하여 제안한 설계 방안을 적용하도록 한다.

본 논문에서 가정하는 게임 콘텐츠의 특성은 다음과 같다. 게임 콘텐츠는 게임 실행 파일, 환경 설정 파일, 이미지 파일, 사운드 파일, 등으로 구성된 패키징된 형태이며, 환경 설정 파일은 콘텐츠의 상태 정보를 담고 있기 때문에 게임이 실행될 때마다 변경된다. 게임 콘텐츠는 게임 실행 파일의 일부가 없으면 실행이 불가능한 구조이다. 또한, 게임이 실행될 때 게임 콘텐츠에 포함된 대부분의 파일들이 필요하기 때문에 게임 콘텐츠에 임의 접근이 빈번히 발생한다. 특히 게임 실행 파일은 전체가 있어야 실행이 가능하다. 게임 콘텐츠는 별도의 플레이어가 플랫폼에 내장되어 실행할 수도 있으며, 플랫폼 자체의 프로그램 로더가 실행할 수도 있다.

OMA DRM v2.0 규격은 능동형 콘텐츠의 특성을 반영하기 힘든 구조이다. OMA DRM v2.0 규격서는 콘텐츠의 특성에 따라 세부적인 기능을 규격화하는 것이 목적이 아니라, 이동 통신 환경에서 안전하게 콘텐츠와 사용 권한을 보호하고 정당한 기기에 전달하는 것이 목적이다. 또한, 아직까지는 이동 통신 환경에서 유통되는 콘텐츠의 대부분이 음악, 동영상, 벨소리, 배경 이미지와 같은 수동형 콘텐츠가 주류를 이루고 있다는 측면도 있다. 예를 들어, 방송 콘텐츠에 OMA DRM v2.0 규격을 적용하려고 하자, 콘텐츠 특성에 따른 여러 가지 이슈가 발생하여 OMA BCAST에서는 OMA DRM v2.0 규격을 확장한 OMA DRM v2.0 Extensions for Broadcast Support 규격을 개발 중인 것은 대표적인 사례라 할 수 있다.

4.1 콘텐츠 부분 암호화

애플리케이션이 복호화된 원본 콘텐츠를 요구할 때 DRM Agent는 일정한 응답 시간내에 원본 콘텐츠를 복호화하여 응답해야 한다. 만약, 허용된 시간내에 응답하지 않으면 사용자는 콘텐츠를 사용하는데 불편을 느끼게 된다. 응답 속도에는 알고리즘 연산 복잡도, 데이터의 크기, 단말의 처리 속도 등의 요소들이 영향을 준다. 알고리즘 연산 복잡도나 단말의 처리 속도에 대한 내용은 본 논문의 주제에서 벗어나므로 제외하고 데이터의 크기를 통한 응답 속도 향상 방안을 제안한다.

OMA DRM v2.0 DCF 규격은 AES_128_CBC와 No-Encryption을 DCF 콘텐츠의 암호화 방식으로 정의하고 있다. 전자는 콘텐츠 전체를 암호화한다는 의미이며, 후자는 암호화하지 않는다는 의미이다. 전체 콘텐츠의 암호화 방식은 어떤 암호 알고리즘을 사용하던 효율적이지 못하다. 콘텐츠 부분 암호화를 능동형 콘텐츠에 적용하면 콘텐츠의 일부만 없더라도 재생이 불가능하기 때문에 콘텐츠 기밀성을 유지하는 것이 가능하다.

능동형 콘텐츠에서 콘텐츠 상태 정보가 기술된 부분은 암호화하지 않을 수 있기 때문에 이용자에 의해 빈번하게 변경되는 콘텐츠 상태 정보를 지원하는 것이 가능하다. 예를 들어, 게임 콘텐츠의 경우 환경 설정 파일은 암호화하지 않음으로써 이용자에 의해 콘텐츠 상태 정보가 변경되도록 허용할 수 있다.

콘텐츠 부분 암호화는 원본 콘텐츠의 전부를 암호화하지 않고, 의미 있는 일부만을 암호화하는 방식이다. OMA DRM v2.0 PDCF의 Selective Encryption 개념을 확장하여 DCF 규격에 응용한 것이라 할 수 있다. 본 논문은 부분 암호화를 위해 DCF 헤더 확장 규격으로 Regular Selective Encryption Header와 Irregular Selective Encryption Header를 제안한다. 전자는 보호해야 할 콘텐츠 오프셋이 규칙적으로 발생할 경우를 위한 헤더이다. Selective Encryption Header의 길이가 고정되어 있으므로, 최소 정보로 암호화된 오프셋 표현이 가능하다. 후자는 오프셋이 불규칙적으로 나타날 경우를 위한 헤더로써, 암호화된 오프셋마다 그 위치를 헤더내에 명시해야 하므로, 오프셋의 수만큼 헤더의 길이가 늘어난다. 이와 같이 두 가지 헤더 규격을 제안하는 이유는 DCF 헤더의 길이가 불필요하게 늘어나는 것을 막기 위함이다.

본 논문에서 제안하는 부분 암호화 방식은 콘텐츠의 개별적인 특성이 고려하지 않을 경우, 콘텐츠의 기밀성이 노출될 우려가 있다. 어떤 콘텐츠는 콘텐츠의 일부분만으로도 의미 있는 데이터의 추출이 가능하나, 어떤 콘텐츠는 의미 있는 데이터의 추출이 불가능할 것이다. 따라서 부분 암호화 방식을 적용할 경우에는 각 콘텐츠의 특성을 감안하여 암호화할 부분과 그렇지 않은 부분을 분리한 후, 다음 절에 제시된 방안에 따라 DCF 헤더를 생성하고 패키징을 수행해야 한다. OMA DRM v2.0 DCF 규격서는 모든 콘텐츠의 공통적인 특성을 수용할 수 있는 구조를 설계하는 것이 목적이므로, 이와 같이 특수한 경우에 대해서는 규격화하지 않았다.

4.1.1 DCF 규격 확장

본 논문은 콘텐츠 부분 암호화 정보를 DCF에 명시하기 위한 DCF Selective Encryption Header를 제안한다. DCF Selective Encryption Header가 삽입될 위치는 DCF에 정의된 여러 헤더 중 Textual Header의 Custom Header이다. Custom Header는 OMA DRM v2.0 DCF 규격에 영향을 주지 않으면서 확장에 필요한 추가적인 정보를 표현할 수 있는 헤더이다. DCF Selective Encryption Header를 인식하지 못하는 DRM Agent는 Custom Header에서 DCF Selective Encryption Header를 무시하므로, DRM Agent 동작에 영향을 미치지 않는다.

Custom Header의 구문은 OMA DRM v2.0 DCF 규격에서 다음과 같이 정의하고 있다.

```
OtherHeader = Header-name ":" Header-value
Header-name = token
Header-value = token
```

Custom Header는 Name-Value 쌍의 형식을 취하며 Name과 Value는 콜론으로 구분되며 각 쌍은 반드시 널 문자로 종료되어야 한다. 인코딩 방식은 UTF-8 Encoding을 지원해야 한다.

4.1.2 Regular Selective Encryption Header

본 논문의 Encryption Mode는 능동형 콘텐츠 속성에 따라 암호화해야 할 블록이 콘텐츠에서 규칙적으로 나타날 때 적용할 수 있는 Regular Selective Encryption Mode와 불규칙적으로 나타날 때 적용할 수 있는 Irregular Selective Encryption Mode를 제안한다. Regular Selective Encryption Mode는 보호되어야 할 콘텐츠 블록이 규칙적일 때 적용되는 헤더이다. Custom Header에 표현되는 구문은 다음과 같다.

```
RegularEncHdr = "RegularEncHdr" ":"
"start-encryption-offset" "=" parameter "&"
"encryption-length" "=" parameter "&"
"plain-data-length" "=" parameter
parameter = *digit
```

이 름	의 미
RegularEncHdr	Regular Selective Encryption Mode 헤더임을 표현
start-encryption-offset	원본 콘텐츠에서 가장 최초로 암호화된 영역을 지시하는 오프셋을 표현
encryption-length	원본 콘텐츠에서 암호화된 영역의 길이를 표현
plain-data-length	원본 콘텐츠에서 암호화되지 않은 영역의 길이를 표현

그림 2는 Regular Selective Encryption Header에서 사용되는 파라미터를 도식화한 것이다.

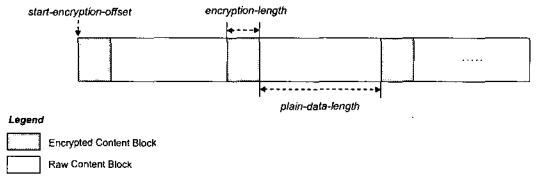

그림 2. Regular Selective Encryption

start-encryption-offset이 최초 암호화된 콘텐츠 블록 (이하 ECB, Encrypted Content Block)의 오프셋을 가리키기 때문에 start-encryption-offset부터 encryption-length가 최초 ECB를 표현한다. start-encryption-offset + encryption-length부터 start-encryption-offset + encryption-length + plain-data -length까지는 최초 원본 콘텐츠 블록 (이하 RCB, Raw Content Block)을 표현한 것이다. 이와 같은 패턴으로 ECB를 규칙적으로 표현할 수 있다. 파라미터는 unsigned int (64)형으로 표현된다.

4.1.3 Irregular Selective Encryption Header

Irregular Selective Encryption Mode는 암호화해야 할 콘텐츠의 내용이 불규칙적으로 나타날 때 적용되는 헤더이다. Custom Header에 표현되는 구문은 다음과 같다.

```
IrregularEncHdr = "IrregularEncHdr" ":"
encrypted-content ( ";" encrypted-content )
encrypted-content: "encryption-offset" "="
parameter "&" "encryption-length" "="
parameter
parameter = *digit
```

이 름	의 미
IrregularEncHdr	Irregular Selective Encryption Mode 헤더임을 표현
encryption-offset	암호화된 영역의 오프셋을 표현
encryption-length	암호화된 영역의 길이를 표현

그림 3은 Irregular Encryption Header에서 사용되는 파라미터를 도식화한 것이다.

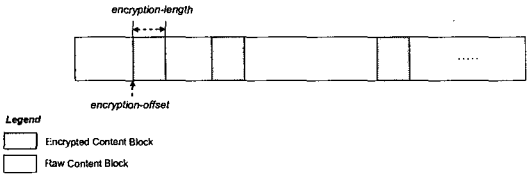

그림 3. Irregular Selective Encryption

처음으로 파싱된 encryption-offset + encryption-length는 콘텐츠에서 최초 ECB의 오프셋을 가리킨다. 만약, 첫 번째 encryption -offset이 0이 아닌 값이라면 처음 콘텐츠 오프셋 (0) + encryption-offset까지가 첫 번째 RCB를 가리킨다. 위와 같은 값이 반복해서 헤더에 표현된다.

4.1.4 Content Block Encryption

모든 ECB들은 모아져 하나의 세션을 통해서 암호화된다. 따라서 콘텐츠 전체 암호화와 콘텐츠 부분 암호화모두 암호화된 DRM 콘텐츠의 길이는 원본 콘텐츠에 비하여 동일하게 늘어난다. 콘텐츠 부분 암호화에 비하여 부분 복호화는 복잡한 경우의 수를 고려해야 하는 작업이다. 만약, 암호화될 블록의 길이를 암호 알고리즘의 블록 사이즈로 고정하지 않을 경우, 경우의 수는 더욱 늘어나게 된다. 따라서 암호화될 블록의 길이는 암호 알고리즘의 블록 사이즈의 배수로 제약한다.

4.2 콘텐츠 복호화를 위한 키 추출

OMA DRM v2.0 규격서는 v1.0의 키 관리 및 전송의 보안 취약점을 개선하기 위해 PKI 기반의 키 전송 및 관리 메커니즘을 정의하고 있다. CEK의 노출을 막고, RI에 의해서 인증된 디바이스의 DRM Agent에서만 RO가 설치, 사용될 수 있도록 하기 위해 2.4절에서 설명된 바와 같이 여러 단계의 키 재료의 보호 및 추출 과정을 거친다.

만약 DRM Agent가 콘텐츠 복호화하기 위해 CEK를 별도로 관리하지 않는다면 원본 콘텐츠를 재생하기 위해서는 매번 2.4절에 기술한 보안 연산을 수행한 후 CEK를 획득할 수 있다. 표 2는 CEK를 추출할 때 필요한 각 단계의 키 재료를 획득하기 위해 소요되는 처리 시간을 측정한 것이다. 표 2를 통해서 다음과 같은 사실을 확인할 수 있다.

RO는 CEK를 안전하게 DRM Agent까지 전달하기 위한 개체이다. RO에 적용된 키 전송 메커니즘은 신뢰할 수 없는 네트워크를 통해 전달되는 과정에서 CEK를 안전하게 보호하기 위한 것이라고 이해할 수 있다. DRM Agent가 RO를 획득하여 안전하게 설치한 후, 각 단말 환경에 따라 CEK를 안전하게 관리할 수 있다면 굳이 CEK를 추출 RO에서 획득하는 방식으로 설계할 필요가 없다. 즉, RO와 CEK (혹은 CEK를 추출할 수 있는 키 재료)를 분리하여 관리하는 것이 가능하다. 일반적으로 DRM Agent가 탑재된 단말이 신뢰할 수 없는 네트워크보다는 안전하며 사용자의 접근이 쉽지 않아서 RO에 적용된 수준의 보안 메커니즘은 불필요하다. 표 2의 측정 결과는 RO가 인증된 DRM Agent에 전달되어 설치된 후에 Z값 추출을 위해 필요한 연산을 제거함으로써 얻을 수 있는 처리 시간을 얼마나 단축시킬 수 있는지 보여 준다.

참조 문헌 [6]은 하드웨어와 소프트웨어로 구현된 보안 라이브러리를 바탕으로 속도 차이가 얼마나 나타나는지에 대한 실험 결과를 자세히 보여 준다. 이 실험 결과는 표 2와 유사한 결과가 보여 주고 있다. 결론적으로 본 논문은 OMA DRM Agent를 설계할 때 콘텐츠를 복호화할 때마다 RO에서 CEK를 추출하는 방법 대신, 단말의 고유 정보를 바탕으로 REK를 암호화하여 DRM Agent가 RO와 CEK를 분리하여 관리하는 형태로 설계되어야 한다고 제안한다. REK를 어떻게 암호화할 것이며, RO와 CEK를 어떻게 관리할 것인가는 DRM Agent 개발사나 사업자의 환경에 따라 다르기 때문에 본 논문에서 정의하지는 않는다. CEK를 RO에서 직접 획득하지 않는 구조는 플랫폼의 프로그램 로더 (Program Loader)나 렌더링 애플리케이션 (Rendering Application)이 능동형 콘텐츠를 로딩하는 초기 시간을 단축시켜 사용자의 QOE (Quality of Experience)를 지원하는데 밑거름이 된다.

표 2. CEK 획득을 위한 각 단계별 처리 시간 (단, 시험 환경은 5.1.2절과 동일함)

키 추출 절차 및 보안 연산	처리 시간 (ms)
Z값 추출 (RSA 복호화[10])	289
KEK 추출 (KDF)	1
REK 추출 (AESWRAP)	2
CEK 추출 (AESWRAP)	3
MAC 검증 (HMAC SHA-1[7])	56
콘텐츠 복호화 (1,024bytes)	19

4.3 콘텐츠 임의 접근

암호화된 DRM 콘텐츠를 처리한 후 렌더링하기 위해서 DRM Agent는 일반적인 File I/O 함수들과 다소 다른 요소들을 고려해야 한다. 하나는 DRM 콘텐츠를 파싱하여 DRM 콘텐츠 헤더를 처리해야 한다는 것이다. 다른 하나는 렌더링 애플리케이션이 요청한 DRM 콘텐츠의 오프셋에 해당하는 콘텐츠를 찾기 위한 처리가 필요하다는 것이다. 렌더링 애플리케이션이 요청한 DRM 콘텐츠의 오프셋을 올바로 계산하여 콘텐츠를 렌더링하기 위해 제안 시스템은 1) 전체가 암호화된 경우, 2) 전혀 암호화되지 않은 경우, 3) 4.1절에서 제안한 부분 암호화된 경우를 고려해야 한다. 3)을 이용하여 암호화된 경우 두 가지 고려 사항 중 후자는 1), 2)에서 발생되지 않은 그림 4와 같은 상황을 처리해야 한다.

렌더링 애플리케이션의 요청에 의해 부분 암호화된 능동형 콘텐츠를 렌더링하는 DRM Agent는 빈번하게 발생하는 임의 접근을 효과적으로 처리할 수 있어야 한다. 그림 4는 흔히 발생하는 몇 가지 사례를 도식화한 것이다. 그림 4.a는 렌더링 애플리케이션이 요청한 콘텐츠 영역이 최초 ECB (ECB 1)가 포함된 경우이다. DRM Agent는 렌더링할 때 콘텐츠의 복호화시 사용되는 초기 벡터 (Initial Vector, IV)의 길이만큼을 더 렌더링해야 한다. 그림 4.b는 그림 4.a와 반대로 마지막 ECB (ECB n)가 포함된 경우로 마찬가지로 RCB에서 초기 벡터의 길이만큼을 더 렌더링해야 하고, 패딩을 고려해야 한다. 그림 4.c의 경우는 중간 ECB (ECB n-m)가 포함된 경우로 이전 ECB를 찾는 작업이 필요하다. 이전 ECB에서 암호 알고리즘이 필요한 만큼 블록을 찾아야 한다. 오프셋 및 길이 계산은 그림 4.a와 마찬가지로 초기 벡터의 길이를 고려해야 한다. 그림 4.d의 경우는 그림 4.c와 동일하나 RCB로 시작한다는 차이점이 있다. 그림 4.e의 경우는 다수 개의 ECB에 걸쳐서 렌더링하는 경우이다.

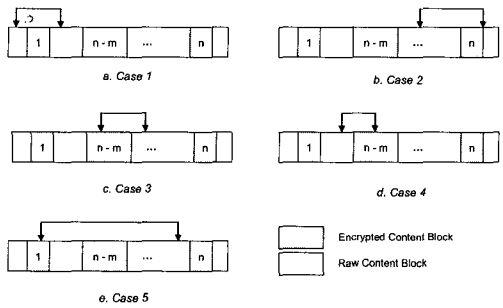

그림 4. 콘텐츠 임의 접근 사례 (n: ECB 전체 개수)

4.4 Rendering API 정의

소프트웨어 형태의 DRM Agent를 설계할 때 고려해야 할 중요한 사항 중 하나는 DRM Agent와 단말의 플랫폼이나 렌더링 애플리케이션과의 인터페이스를 어떻게 정의할 것이냐이다. 첫 번째 방안은 DRM Agent가 별도의 렌더링 API를 제공하는 것이다 (그림 5.a). 플랫폼의 프로그램 로더나 렌더링 애플리케이션이 DRM 콘텐츠를 렌더링해야 할 경우에는 DRM Agent의 렌더링 API를 이용하고, 그렇지 않은 경우에는 표준 File I/O API를 사용하는 것이다. DRM Agent의 렌더링 API도 File I/O와 동일한 구조와 기능을 갖기 때문에 개발자가 별 다른 어려움 없이 DRM 콘텐츠의 이용이 가능하다.

두 번째 방안은 기존 플랫폼이 제공하는 File I/O API를 DRM 렌더링 API로 대체하는 것이다 (그림 5.b). 렌더링 애플리케이션이나 플랫폼의 프로그램 로더는 DRM 콘텐츠나 일반 콘텐츠의 구분 없이 플랫폼이 제공하는 표준 File I/O API를 사용하면, 플랫폼이 DRM 콘텐츠인지 일반 콘텐츠인지를 식별한 후 DRM 렌더링 API의 사용 여부를 판단한다.

만약, 플랫폼의 수정이 불가능한 경우에는 앞서 설명한 방안을 일부 변경한 세 번째 방안을 적용할 수도 있다 (그림 5.c). 렌더링 애플리케이션이나 프로그램 로더에게 표준 File I/O API만 제공된다는 점은 동일하나 플랫폼이 DRM 콘텐츠 여부를 식별하지 않고 DRM Agent가 DRM 콘텐츠 여부를 판단한 후, 적합한 API를 호출하는 방식이다.

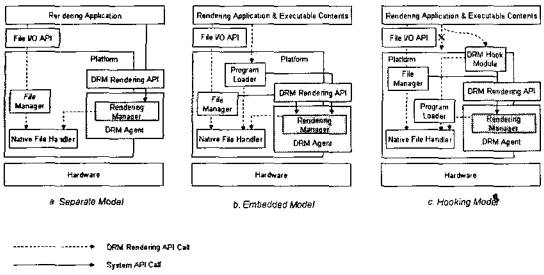

그림 5. DRM Agent가 플랫폼에 설치된 방식에 따라 달라지는 API 호출 관계

본 논문은 두 번째 방안에 기반하여 능동형 콘텐츠의 빠른 렌더링을 지원할 수 있는 DRM 렌더링 API의 구조를 제안한다. 능동형 콘텐츠를 처리하는 렌더링 애플리케이션은 DRM 콘텐츠를 열고, 닫는 횟수가 빈번하도록 콘텐츠 구조가 개발된 경우가 자주 발생한다. 4.1절에서 제시한 바와 같이 이러한 콘텐츠는 보호해야할 부분과 그렇지 않은 부분들로 구분할 수 있으며, 3장에서 언급된 것처럼 능동형 콘텐츠의 경우 특정 부분만 암호화하더라도 전체를 암호화한 효과를 얻을 수 있다. 하지만, DRM 콘텐츠의 전체가 암호화되었는지, 일부가 되었는지, 전혀 암호화되지 않았는지 여부는 오직 DRM Agent만 알 수 있기 때문에 그림 5.b처럼 플랫폼의 파일 매니저(File Manager)나 프로그램 로더는 DRM 콘텐츠일 경우 DRM Agent가 처리하도록 전달한다. DRM Agent는 DRM 콘텐츠를 해석한다. 만약, 복호화가 필요하면 복호화를 수행한 후에 요청한 원본 콘텐츠를 넘겨준다. 하지만, 이러한 처리 절차는 DRM Agent의 복호화와 상관없이 매번 불필요한 처리(e.g. DRM Agent 초기화, 환경 설정, 사용 권한 확인, 등)를 해야 하기 때문에 실행 가능한 콘텐츠의 실행 응답 속도가 늦어지는 현상이 발생한다.

이러한 현상을 줄이기 위해 DRM 렌더링 API를 이원화하고, DRM Agent와 프로그램 로더간에 약간의 정보 교환을 통해 프로그램 로더가 DRM 렌더링 API를 선별적으로 호출하는 구조는 선택적 암호화 기법을 이용한 능동형 콘텐츠의 처리 응답 속도를 줄일 수 있는 방안이다. DRM Agent는 두 종류의 DRM 렌더링 API를 제공한다. 하나는 콘텐츠 복호화를 처리하는 일반적인 절차에 따르는 DRM 렌더링 API이다. 다른 하나는 DRM Agent와 프로그램 로더간에 Stateful DRM 세션을 맺도록 하여 사용 권한 확인 및 콘텐츠 복호화가 필요 없는 특수한 종류의 DRM 렌더링 API이다. DRM Agent는 후자의 API를 통해 렌더링 요청이 들어올 경우 DRM 콘텐츠를 파싱한다는 것만 제외하면 일반적인 File I/O 처리하는 것과 동일하게 처리되므로 현저하게 빨라진 응답 속도를 확인할 수 있다.

V. 실험 결과

5.1 실험 환경

제안된 시스템의 성능 측정을 위해 WIPI용 단말과 WIPI 에뮬레이터가 설치된 PC에 DRM Agent를 개발하여 포팅하였다. 이와 같이 동일한 DRM Agent에 대해 각기 다른 두 가지 환경에서 실험을 수행한 것은 제안 내용이 DRM Agent가 실행되는 특정 플랫폼 종속적인 성능 향상이 아닌 범용적인 성능 향상을 가져올 수 있음을 입증하기 위한 것이다. 각 환경에서 실험을 위해서 사용되는 DRM Agent 버전, 성능 측정 코드, 데이터는 모두 동일하다.

5.1.1 WIPI(Wireless Internet Platform of Interoperability)

WIPI[13]는 한국 무선 인터넷 표준화 포럼의 모바일 플랫폼 특별 분과에서 만든 모바일 플랫폼 표준 규격으로서 무선 인터넷을 통해 다운로드된 응용 프로그램을 이동 통신 단말기에 탑재시켜 실행 시기키 위한 환경을 제공하는데 필요한 표준 규격이다. 현재 새롭게 국내 이동 통신사를 통해 출시되는 신규 단말은 모두 WIPI 플랫폼을 탑재하도록 법제화 되어 있으며, WIPI 플랫폼 기반에서 다양한 서비스가 제공되고 있다. 이렇듯 WIPI 플랫폼을 적용함으로써 이동 통신사 혹은 단말 제조사에 독립적인 어플리케이션 및 콘텐츠 제작이 이루어질 수 있는 장점이 있다.

5.1.2 단말

단말 환경에서 성능 평가를 하기 위해 사용된 단말은 S전자에서 개발한 ARM 9 CPU를 탑재한 B모델을 사용하였으며, 단말 바이너리 버전은 ZD29.1220 버전이다. DRM Agent는 단말 바이너리의 WIPI 플랫폼에 내장되는 방식으로 탑재되었으며, 기존 WIPI 파일 시스템에 연동되어 작동하는 방식으로 구현되었다. DRM Agent가 콘텐츠 렌더링을 위해서 필요로 되는 암호화 및 PKI 관련 라이브

러리는 WIPI용으로 개발되어 현재 출시되는 폰에 탑재되고 있는 상용 라이브러리가 사용되었으며, 단말 및 PC에뮬레이터에는 동일한 버전이 탑재되었다.

5.1.3 PC 에뮬레이터

PC 에뮬레이터는 단말 개발을 위해서 제공되는 WIPI SDK의 1.10.00 버전을 기반으로 한다. 1.10.00버전의 PC 에뮬레이터에 DRM 탑재를 위해 수정된 단말 바이너리가 포함하고 있는 동일한의 WIPI플랫폼이 포팅되었다. 그럼으로 실험에 사용된 PC에뮬레이터는 기능적인 측면에서 단말 시험시 사용한 단말기와 동일하게 작동한다. PC 에뮬레이터는 실행한 PC의 성능에 종속되며, 에뮬레이터를 실행한 PC환경은 표 3을 참조한다.

표 3. 실험 환경

	테스트 환경	Crypto&PKI Library	기타
단말	ARM 9 CPU 탑재 단말	WIPI 상용 라이브러리	-
PC 에뮬레이터	WIPI Emulator : 1.10.00 버전	상동	PC 환경 CPU : Pentium 4 3.00GHz Memory : 512MB OS : Windows 2000

5.2 실험 데이터

실험을 위해 2 종류의 데이터가 사용되었다. 첫 번째 데이터는 리소스 API 지원과 관련된 실험을 진행하기 위해서 사용된 WIPI-C 콘텐츠인 'O' 게임이다. 'O'게임은 Irregular Selective Encryption방식이 적용 되었으며, 상위 1,024 바이트에 대해 암호화가 적용된 콘텐츠이다. 두 번째 데이터는 일반 바이너리 데이터를 콘텐츠 패키징 방식으로 생성한 바이너리 콘텐츠이며, 부분 암호화와 전체 암호화에 대한 성능 평가를 위해 사용되었다. 각기 다른 크기를 가진 총 10개의 바이너리 데이터를 준비하였으며, 각각의 바이너리 데이터는 전체 암호화, 부분 암호화가 적용되어 콘텐츠로 패키징 되었다. 부분 암호화가 적용된 콘텐츠의 경우 ECB의 오프셋에 대한 정보는 표 4를 참조한다. 실험 항목에 따라 위에서 명시한 콘텐츠들이 선택적으로 사용되었으며, 선택된 콘텐츠에 대해서는 각 실험 결과에서 명시하기로 한다.

표 4. 실험 데이터

콘텐츠 명	콘텐츠 종류	암호화 형태	기 타
'O'게임	WIPI-C	Irregular Selective	오프셋 : 0, 길이 : 1024
바이너리 콘텐츠	바이너리 데이터	Irregular Selective, 전체	{오프셋 : 0, 길이 : 1024} {오프셋 : 256, 길이 : 1024, 오프셋: 4096, 길이 : 1024, 오프셋: 16384, 길이 : 1024, 오프셋: 20488, 길이 : 1024, 오프셋: 36494, 길이 : 1024}, 전체

5.3 실험 결과

5.3.1 콘텐츠 부분 암호화

본 논문의 실험 콘텐츠인 게임 콘텐츠는 사용자의 QoE가 매우 중요한 성능의 척도가 된다. 본 실험에서는 단말 상에서 부분 암호화가 되어 있는 콘텐츠와 전체 암호화가 되어 있는 콘텐츠에 대한 전체 복호화 시간을 측정하며, 제안한 부분 암호화가 적용된 경우 전체 암호화와 비교하여 향상되는 QoE를 증명한다. 게임 콘텐츠에서 복호화 시간이 단축되었음은 사용자가 게임을 실행하거나 진행할 때 소요되는 시간이 단축되었음을 의미함으로 QoE의 척도가 될 수 있다. 향상되는 QoE를 정량화하여 비교하기 위해서는 실험 데이터로 사용될 콘텐츠가 일정 크기별로 존재해야 하지만 실제 콘텐츠를 사용할 경우에는 적합한 실험 데이터 그룹을 구비할 수 없다. 따라서 본 실험에서는 일정 크기가 차이가 나는 바이너리 데이터를 패키징하여 실험 데이터 콘텐츠를 구비한 후 실험했다.

바이너리 데이터는 사이즈에 따라 원본 바이너리 데이터, 상위 1k만을 암호화한 콘텐츠, 임의의 5개 영역을 1k씩 암호화하여 총 5k를 암호화한 콘텐츠, 전체 데이터를 암호화하는 방식으로 패키징을 하였다. 준비된 테스트 콘텐츠를 사용하여 단말 상에서 각 암호화된 콘텐츠 별로 데이터 요청을 하여 전체 원본 데이타가 저장되기까지 소요되는 시간을 측정하였다. 암호화된 콘텐츠의 경우 데이터를 요청할 때 DRM Agent에서 제공하는 API가 사용되었으며, 일반 파일의 요청 시에는 일반 WIPI 파일 I/O API

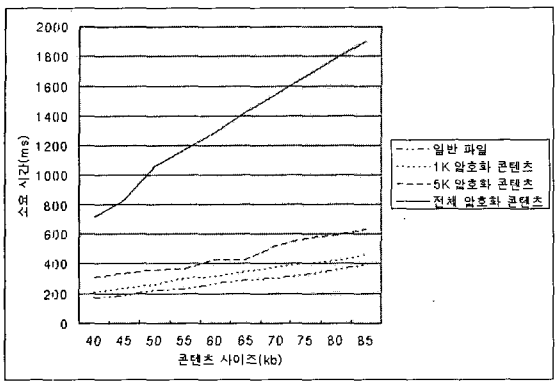

그림 6. 콘텐츠별 전체 복호화시 소요 시간

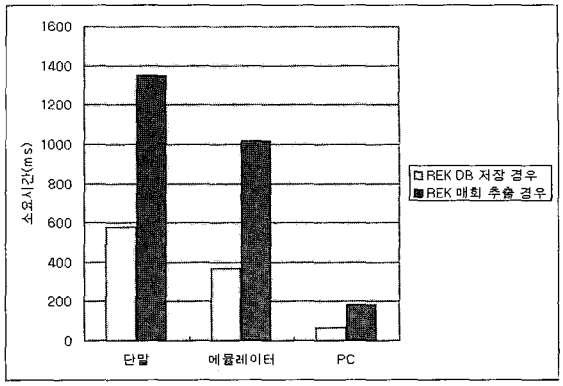

그림 7. 콘텐츠 로딩 시작 소요 시간

를 사용하였다. 부분 암호화 방식을 적용할 경우 부분 암호화 정보의 해석 및 콘텐츠 복호화시 암호화된 영역에 대한 위치를 계산하기 위한 오버헤드가 존재한다. 하지만 복호화 처리 시간에 비해서는 부분 암호화 헤더를 처리하는데 소요되는 처리 시간이 미미하여 별도의 실험 결과는 정리하지 않는다.

그림 6은 각 종류별 콘텐츠의 크기를 일정하게 증가 시키며 전체 복호화에 소요되는 시간을 측정한 결과이다. Irregular Selective Encryption 방식이 적용된 콘텐츠의 경우 일반 바이너리 데이터를 읽고 저장하는데 소요되는 시간과 유사한 증가율을 보여준다. 하지만 전체 암호화가 적용된 콘텐츠의 경우 소요 시간의 증가율이 콘텐츠 크기에 따라 가파르게 상승하는 것을 볼 수 있다. 이는 데이터의 크기가 늘어나도 부분 암호화 콘텐츠의 복호화 연산 시간은 완만한 상승률을 보이는 반면에 전체 암호화 콘텐츠에 대한 복호화 시간은 콘텐츠 크기 증가에 따른 증가폭이 매우 가파른 것을 알 수 있다. 이런 결과는 콘텐츠를 보호하면서 게임 콘텐츠와 같이 빠른 응답 시간을 필요로 하는 경우에 대해서는 부분 복호화가 효과적임을 보여준다.

5.3.2 CEK 추출

본 실험에서는 설치된 RO에서 직접 CEK를 추출하여 복호화를 수행하는 WIPI 코어 버전, 데이터베이스 상에 REK를 저장한 상황에서 복호화를 수행하는 WIPI 코어 버전을 나누어 탑재한 단말과 에뮬레이터 및 OpesnSSL[11]을 기반으로 하여 개발된 DRM Agent를 사용하여 실험을 진행한다. 전자의 실험 환경은 콘텐츠에 대한 사용 요청이 들어올 경우 규격에 명시된 방법에 따라 RO로부터 CEK를 도출하는 과정이 수행되며, 후자의 실험 환경은 4.2에서 명시한 방법이 적용된 형태로 CEK를 도출하는 과정이 수행된다. 실험 내용은 두 가지 경우에 대해 키를 복호화하기 위해서 사용되는 CEK를 추출하여 게임이 로딩되는 시점까지 소요되는 시간을 측정하는 것이다. 실험을 통해 시간을 측정하는 범위는 실제 사용자가 게임 콘텐츠를 실행할 때 게임 화면이 처음 나타나는 시점을 의미하며, 이 영역의 시간 단축이 이루어질 경우 사용자가 체감하는 QoE가 향상되었음을 증명할 수 있다.

그림 7은 실험 결과를 그래프 형태로 표현한 것이다. 단말의 경우 REK를 DB에 저장하는 경우 10회 실행 시 평균 578ms의 소요시간이 측정되었으며, REK를 매회 추출하는 경우에는 1350.2ms의 소요시간이 측정되었다. 에뮬레이터의 경우에는 각 소요시간이 365.8ms, 1017.8으로 측정되었으며, PC 버전의 경우에는 각 소요 시간이 66.4ms, 184.4ms으로 측정되었다. 측정된 데이터를 정리해보면 단말의 경우 REK를 RO에서 직접 추출하여 콘텐츠를 실행하는 속도가 DB에 REK를 저장한 상태에서 실행하는 속도보다 약 2.3배가 더 소요되며, 에뮬레이터와 PC 버전의 경우에는 각각 2.8배, 2.7배의 속도차가 나는 것을 알 수 있다. 이러한 실행 속도의 차이는 REK를 추출하기 위해서 실행되는 RSA 복호화 연산의 부하에 따른 것으로 판단된다. 본 실험의 결과를 통해 RSA 연산의 부하 감소를 위해 제안한 REK의 DB 저장 방법이 속도 향상에 기여하는 것을 확인할 수 있다.

5.3.3 Rendering API

본 실험에서는 리소스 접근 속도 향상을 위해 제공

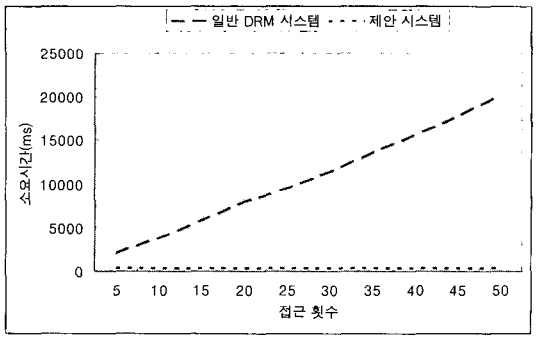

그림 8. 단말의 렌더링 API

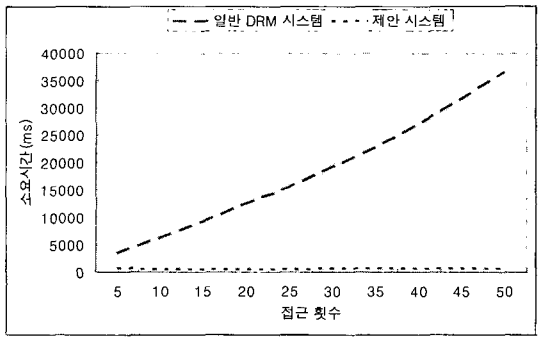

그림 9. PC 에뮬레이터의 렌더링 API

된 API가 사용된 WIPI 코어 버전과 사용하지 않은 WIPI 코어 버전을 나누어 단말과 에뮬레이터에서 실험을 진행하였다. 사용된 콘텐츠는 리소스 접근을 빈번하게 하는 WIPI-C 콘텐츠인 'O'게임이다. 실험 방법은 단말과 PC 에뮬레이터 상에서 리소스 접근 횟수에 따라 전체 리소스를 로딩하는데 소요되는 시간 값을 측정하였으며, 실험을 위해 'O'게임의 특정 리소스에 대해서는 중복 로딩하였다. 동일한 연산에 대해서 단말과 PC에뮬레이터에서 실험을 진행하는 것은 WIPI 코어가 실제 사용하는 하위 시스템 기능이 단말에서 제공하는 기능과 PC에서 제공하는 기능으로 각기 다르기 때문에 이에 따라 결과값이 달라질 경우 이를 확인하기 위해서이다.

그림 8과 그림 9에서 일반 DRM 시스템은 빠른 렌더링을 위해 제안된 API가 적용되지 않은 환경에서 리소스를 로딩하는데 소요되는 시간을 나타내며, 제안 시스템은 렌더링 API가 사용되는 환경에서 리소스 로딩을 수행한 결과를 의미한다. 실험 결과 제안한 렌더링 API가 적용된 단말의 경우 소요 시간의 변이폭이 평균 20ms내외로 측정되었으나, 적용이 되지 않은 단말의 경우에는 리소스 개수가 늘어남에 따라 약 50%의 변화율로 소요 시간이 증가됨을 알 수 있었다. 이는 리소스 접근시 파일 I/O 과정을 반복하는 기존 WIPI-C 콘텐츠 구조를 DRM이 적용된 환경에서도 동일하게 사용되는 문제로 파악되었으며, 이를 위해 제안된 DRM 렌더링 API의 별도 제공이 효과적인 것으로 판단된다. 또한 실제 단말 상에서 측정한 결과 값과 PC 에뮬레이터 상에서 측정한 결과 값의 분포가 유사함으로 제안한 Rendering API가 실행 환경에 종속적이지 않으며 동일한 수준의 유효한 성능을 제공함을 보여준다.

Ⅵ. 결 론

무선 인터넷 환경의 발전과 고성능 컨버전스형 단말기의 보편화로 무선 콘텐츠의 유통은 꾸준히 증가하고 있다. 이에 따라 이동 통신사, 단말 제조사, 플랫폼 제조사, DRM 솔루션 개발사들은 무선 콘텐츠를 안전하게 보호하고 저작권자의 합법적인 권리 보장 및 다양한 비즈니스 모델 적용을 위해 표준화된 DRM 시스템을 이동 단말에 적용하는 추세에 있다.

본 논문은 능동형 콘텐츠의 특징에 대해서 소개하고, 능동형 콘텐츠와 수동형 콘텐츠의 차이에 대해서 비교 설명하였다. 이를 바탕으로 모바일 DRM Agent를 설계하기 위한 새로운 요구 사항을 정의할 수 있었다. 이동 단말에서 능동형 콘텐츠를 지원하기 위한 OMA DRM 프레임워크를 제시하기 위해 게임 콘텐츠를 선택하였다. 게임 콘텐츠는 실행 후 빠른 응답 속도를 필요로 하는 대표적인 능동형 콘텐츠로써 OMA DRM v2.0을 그대로 적용할 경우 콘텐츠 실행에 걸리는 지연 시간이 길어서 이용자의 불편이 크다. 이를 해결하기 위해 본 논문은 DCF 규격의 확장 헤더를 이용하여 부분 암호화를 위한 헤더 규격을 제안하였다. 또한 OMA DRM v2.0 규격의 보안 요구 사항을 만족하는 범위내에서 안전한 키 관리 방안을 제시하였다. 콘텐츠 특성에 맞는 렌더링 API 구조는 특정 게임 콘텐츠에 대해서 콘텐츠 실행의 지연 시간을 줄이는데 큰 역할을 한다.

제안 시스템은 이동 단말 환경을 위해 제안된 것이지만, 다양한 휴대용 개인 디바이스에 적용이 가능하다. 사용자들이 좀 더 싸고 쉽게 콘텐츠를 유통할 수 있는 네트워크 채널이 휴대용 개인 디바이스의 기본 기능으로 탑재되고 있기 때문이다.

참 고 문 헌

[1] Digital Rights Management, Open Mobile Alliance, OMA-Download-DRM-v1_0, 2006.

[2] DRM Specification V2.0, Open Mobile Alliance, OMA-DRM-DRM-v2_0, 2006.

[3] DRM Content Format V2, Open Mobile Alliance, OMA-DRM-DCF-v2_0, 2006.

[4] DRM Rights Expression Language V2.0, Open Mobile Alliance, OMA-DRM-REL -v2_0, 2006.

[5] NIST FIPS 197: Advanced Encryption Standard (AES). November 2001.

[6] Daniel Thull, Roberto Sannino, "Performance Considerations For an Embedded Implementation of OMA DRM 2", *Design, Automation and Test in Europe*, 2005.

[7] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[8] J. Schaad and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002

[9] Myers, M., Ankney, R., Malpani, A., Galperin, S. and C. Adams, "Internet X.509 Public Key Infrastructure: Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[10] RSA Laboratories, "PKCS #1 v2.1: RSA Cryptography Standard", RSA Security, 2002.

[11] OpenSSL Project, http://www.openssl.org

[12] Open Digital Rights Language Initiative, http://odrl.net

[13] Wireless Internet Platform of Interoperability, http://wipi.or.kr

〈著者紹介〉

김 후 종 (Hoo-Jong Kim) 정회원

1988년 2월: 서강대학교 전자공학과 졸업
1995년 8월: 서강대학교 전자공학과 석사
2003년 2월: 국민대학교 전자공학과 박사과정 수료
1988년 1월 : LG전자 안양연구소 입사
1995년 9월~ 현재 : SK Telecom Terminal개발1팀장
〈관심분야〉 DRM,단말 플랫폼, 통신공학

정 은 수 (Eun Su Jung) 정회원

1995년 2월: 한양대학교 전자공학과 학사
2002년 8월: 세종대학교 정보통신학과 석사
2006년 3월~: 고려대학교 정보보호대학원 박사과정
1995년 2월~ 현재 : SK Telecom Terminal개발1팀
〈관심분야〉 DRM,WPKI,단말플랫폼

임 재 봉 (林 在 鳳) 정회원

서울대학교 전자공학과 학사
서울대학교 전자공학과 석사
서울대학교 전자공학과 박사
경력 : 충남대 전자공학과 조교수
미국 텍사스주립대 전기공학과 조교수
한국통신학회 이사
국민대학교 전자공학과 교수
〈관심분야〉 통신, GPS application, 셀룰러, 무선인터넷통신