情 報 保 護 學 會 誌
第23卷 第1號, 2013. 2

개방형 스마트 폰 환경에서 안전한 금융 어플리케이션 실행을 위한 보안 시스템

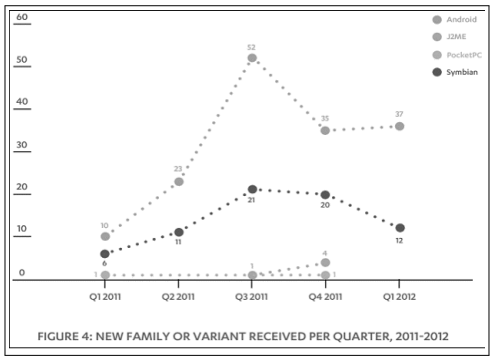
김 진 형*, 김 태 호**

요 약

SNS, Game, Media Play, DMB, 모바일 결제 등 다양한 기능을 손쉽게 설치하여 이용할 수 있는 스마트 폰의 장점으로 인해 스마트 폰 이용률이 급증하고 있다. 특히 안드로이드 운영체제 기반의 스마트 폰 환경에서는 오픈소스로 인하여 더욱 손쉽게 어플리케이션을 개발 및 배포가 가능하여 수많은 사용자들이 다양한 어플리케이션을 이용하고 있다. 하지만 이런 장점의 이면에는 악의적인 어플리케이션의 개발 및 배포, 또한 용이하여 보안사고의 위험성도 존재한다. 앞으로도 스마트 폰의 취약점을 이용한 개인정보 유출 및 위·변조 공격들이 더욱 정교해지고 다양화 될 것으로 예상되어 이를 대응할 수 있는 보안기술의 개발이 요구되고 있다. 특히 민감한 정보를 다루는 금융 어플리케이션을 실행하는 데에는 높은 보안을 제공하는 기술 도입이 더욱더 필요하다. 기존에 제안된 방식들은 소프트웨어로 구현되어 있어 악의적인 공격에 대응하는 데에는 한계가 있다. 높은 보안성으로 주목받는 기술로, 하드웨어 기반의 보안 기술이 있지만 아직 하드웨어적인 자원의 부족 등으로 활성화에 한계가 있다. 본 논문에서는 자원 제약이 있는 하드웨어 보안 기술을 효과적으로 활용하여 보다 안전하게 금융 어플리케이션을 실행 및 관리를 할 수 있는 보안 시스템을 제안하고자 한다.

Ⅰ. 서 론

현재 이동통신 시장은 종래 피쳐폰 중심에서 스마트 폰 중심으로 빠르게 전환되고 있다[1]. 스마트 폰의 대부분이 개방형 플랫폼을 채택하고 있고, 특히 안드로이드 운영체제는 오픈소스 정책으로 소스가 공개되어있어서 iOS, Windows8과 같은 스마트 폰 운영체제보다 악의적 공격에 노출될 위험이 높다[2]. 또한 스마트 폰의 어플리케이션들은 On-line으로 공개된 앱스토어(App-Store)를 통하여 자유롭게 접속하여 스마트 폰에 다운받아 설치가 가능하여 이를 이용한 악성코드의 배포도 용이하여 보안 사고의 위험이 증가되는 단점도 존재한다. 미국의 한 보안업체의 조사에 따르면 2011년 10개였던 악성코드가 2012년 1분기에는 37개로 약 4배가 증가되었다([그림 1] 참조). 또한 스마트 폰 사용자의 금융정보를 탈취하는 악성코드의 공격형태가 증가하고 있다고 보고했다. 앞으로도 스마트 폰의 이용률이 증가할 것으로 예상되어 악의적인 공격에 의한 피해 규모는 더욱더 증가될 것으로 예상된다[3]. 특히 고객의 민감한 정보를 다루는 모바일 금융 서비스에는 치명적인 피해를 초래 할 수 있다.



〔그림 1〕 모바일 악성코드 증가 추이(출처=지디넷)

* 삼성전자 무선사업부 책임연구원 (mailforjhk@gmail.com)
** 삼성전자 무선사업부 연구원 (azzurri3@gmail.com)

악의적 공격에 의한 민감한 정보 유출, 단말 장애 유발, 불법 과금 등과 같은 보안 위협으로부터 스마트 폰 사용 환경의 안전성, 가용성, 신뢰성을 제공하는 보안 기술이 요구된다.

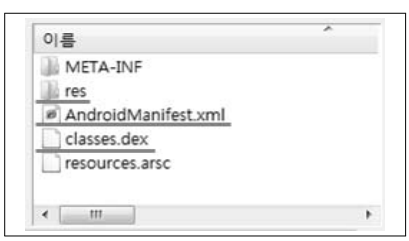
본 논문에는 다양한 공격 형태 중에 금융 서비스를 포함한 어플리케이션의 실행코드를 역공학(Reverse-Engineering)기술을 통하여 코드를 위·변조하여 개인의 금융 정보를 유출하거나 거래 내역 변경 등을 유발하는 공격과 정당한 절차로 어플리케이션이 설치된 스마트 폰에서 허가받지 않은 다른 스마트 폰으로의 불법 복제하는 공격에 안전할 수 있도록 어플리케이션을 보호하는 보안 시스템을 제안하고자 한다.

2장에서는 현재 안드로이드 스마트 폰에 적용된 어플리케이션의 보호기술에 대해 살펴보고, DRM 시스템을 이용한 어플리케이션 보호 기술, 하드웨어 기반 보안 기술 등 기 제안된 보안 기술을 소개하고 각 기술에서 발생할 수 있는 취약점 및 제약 사항을 살펴보겠다. 3장에서는 기존 제안된 기술보다 안전하고 효율적인 어플리케이션 보호를 위해 제안된 보안 시스템을 상세히 기술한다. 마지막으로 4장에서 마무리 한다.

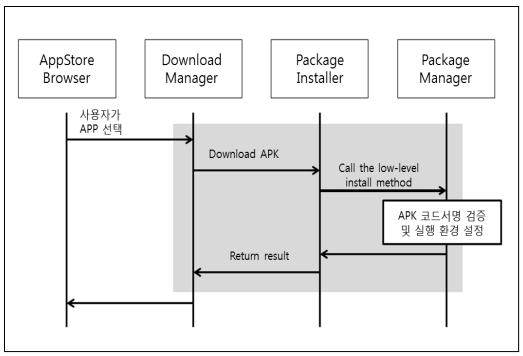
Ⅱ. 관련기술

2.1. 안드로이드 어플리케이션 보안

안드로이드의 어플리케이션은 APK(Application PacKage)라는 형태의 압축파일로 존재한다. APK는 일반적으로 이미지나 UI(User Interface)를 정의하는 XML 파일로 구성된 res 폴더, 컴파일 된 자바코드를 Dalvik bytecode 형태로 저장하는 classes.dex 파일, 어플리케이션에 대한 전반적인 정보를 가지고 있는 AndroidManifest.xml 파일 등으로 구성된다([그림] 2 참조).



〔그림 2〕 일반적인 APK 파일 구성



〔그림 3〕 안드로이드 어플리케이션 설치 과정

먼저 안드로이드 운영체제에서 어플리케이션의 설치 과정을 살펴보겠다. 아래 [그림 3]과 같이 어플리케이션 설치에 관련된 모듈은 Download Manager, Package Installer, Package Manager이다.

사용자가 안드로이드 앱스토어인 안드로이드 마켓에 접속하여 원하는 어플리케이션을 선택하면 Download Manager는 해당 APK를 다운받고, MIME type과 저장 위치 URI을 intent를 통해서 Package Installer에 전달한다. Package Installer는 해당 APK에 포함된 AndroidManifest.xml을 기반으로 Permission Requirement을 사용자에게 보여주며 install에 대한 진행여부 입력을 요구한다. 만약 사용자가 설치에 동의를 했다면, Package Manager는 APK의 코드서명 검증을 수행하고 실행환경을 설정하고 결과를 사용자에게 보여주고 설치 과정을 종료한다. 어플리케이션 설치 시 코드서명 검증을 수행하지만 이것은 어플리케이션의 무결성 및 제어를 위한 것이 아니라 추후 발생할 수 있는 업데이트 및 데이터 교환하기 위한 목적이다. 더욱이 어플케이션을 실행시킬 때에는 실행코드의 무결성 등을 확인하지 않는다[4][5].

안드로이드에서 기본적으로 제공되는(pre-load) 어플리케이션은 /system/app 폴더에 저장되며, 사용자가 마켓에서 다운로드 받은 어플리케이션은 /data/app 폴더에 APK 파일이 원본 그대로 저장된다. 보안상의 이유로 상용화된 스마트 폰 단말에서는 기본적으로 해당 경로에 접근이 불가능하도록 되어있지만 디바이스의 모든 제어가 가능한 루트 권한을 얻는 루팅(rooting)이 되면 ADB Tool 등을 이용하여 설치된 폴더에 존재하는 APK 파일을 복사하여 타 스마트 폰에 설치가 가능하다. 즉 실행파일의 불법 복제를 방지하기 위한 보안 기

술이 제공되지 않음을 의미한다.

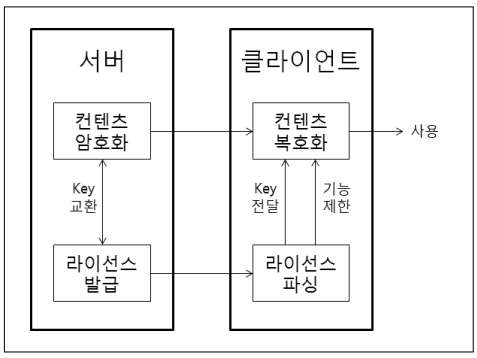
어플리케이션 개발자는 어플리케이션 코드의 추가 없이 Market protection 옵션을 이용하면 설치 된 APK를 /data/app-private 폴더에 저장 및 접근제어를 함으로써 APK 파일을 보호 할 수 있다. 이는 특히 유료 어플리케이션 개발자에게 유용한 옵션이다. 하지만 안드로이드 단말은 어떤 단말도 루팅이 될 수 있기 때문에 어플리케이션 보호에 있어서 강력한 보안수준을 제공할 수 없다.

이처럼 현재의 안드로이드 운영환경에서는 어플리케이션을 보호하기 위한 보안기술이 미흡하여 해킹에 쉽게 노출될 수 있다.

2.2. 어플리케이션 보호를 위한 DRM 기술

스마트 폰에는 이미지, MP3, 동영상 등 다양한 형태의 멀티미디어 파일을 보호하기 위해 OMA(Open Mobile Alliance), PlayReady(Microsoft), Widevine (Google) DRM(Digital Rights Management) 기술 등이 이용되고 있다. 현재의 스마트 폰 생태계에서는 DRM을 유료 컨텐츠의 무단복제 및 도용방지, 저작권자의 보호, 적절한 과금 체계 유지 등의 용도로 사용한다. 실제로 다수의 스마트 폰 단말에서 유통되는 음악, 동영상 등의 유료 컨텐츠는 DRM을 이용하여 재전송 금지, 캡쳐 금지, 재생횟수 제한, 이용가능 시간제한 등의 기능을 하고 있다.

모바일 환경에서 사용하는 DRM 시스템은 일반적으로 서버와 클라이언트로 구성된다. 먼저 DRM 서버는 컨텐츠를 암호화하는 모듈과 이에 맞는 라이선스를 발급하는 모듈로 구성된다. 컨텐츠를 암호화 하여 컨텐츠

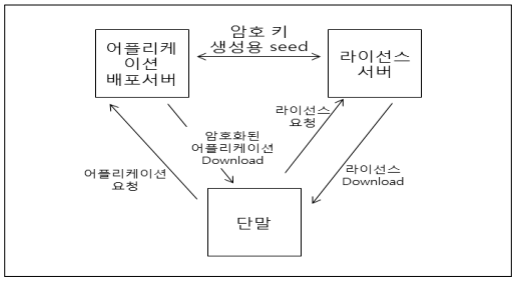


〔그림 4〕 일반적인 DRM 시스템

만 가지고는 사용이 불가능하게 제한하며, 해당 컨텐츠에 대응하는 라이선스가 있어야만 사용이 가능하도록 구성된다. 또한 여기서 발급하는 라이선스를 이용하여 재전송 금지, 재생횟수 제한 등의 기능을 추가할 수 있다([그림 4] 참조).

클라이언트에서는 DRM이 적용 된 컨텐츠와 라이선스를 모두 서버에서 전달받아 이를 이용한다. 발급받은 라이선스를 파싱하여 암호화 된 컨텐츠를 복호화 해서 사용하는데 이때 발급받은 라이선스의 상태에 따라 컨텐츠의 사용에 제약을 받을 수 있다.

기 제안된 DRM 기술은 멀티미디어 파일 뿐만 아니라 실행 가능한 컨텐츠인 어플리케이션을 보호할 수 있도록 DRM 기술을 확장한 기술이 제안되었다[6][7]. 이 기술은 기존의 DRM 시스템과 마찬가지로 [그림 5]에서와 같이 어플리케이션을 암호화하여 배포하는 배포서버가 존재하고, 복호화를 위한 암호키와 어플리케이션 권한 제어를 위해 라이선스 생성하고 배포하는 라이선스 서버로 구성되기 때문에 어플리케이션 보호를 위해서 기존 DRM 시스템 구조를 최소한의 변경으로 이용할 수 있는 장점이 있다[8][9].



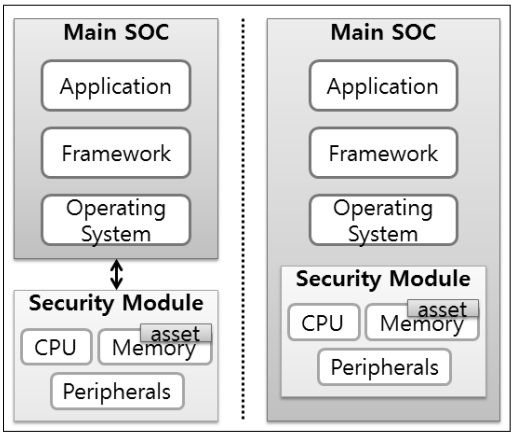
〔그림 5〕 어플리케이션 보호를 위한 DRM 시스템 구조

하지만 위 시스템에서 라이선스를 검증하는 라이선스 서버 인증서 관리 및 단말의 개인키와 인증서의 관리에 대한 취약점이 존재한다. 공격자가 자신이 생성한 서버 인증서를 추가하거나 단말의 개인 키를 획득하는 공격을 통해 DRM을 무력화 시킬 수 있다[10].

2.3. H/W 방식 보안 기술

H/W를 이용한 보안 기술에는 스마트카드, TPM, TrustZone 등의 방법이 있다.

빠르게 성장하는 현재의 모바일 환경에서 하나의 단



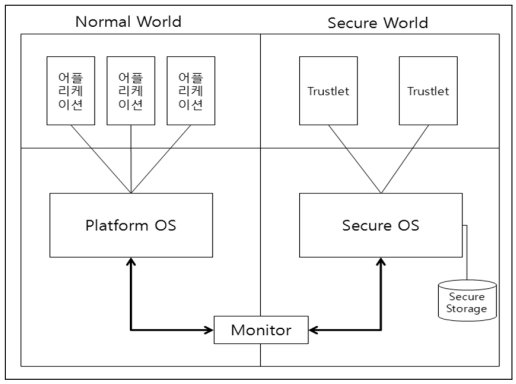
〔그림 6〕 일반적인 H/W 보안과 TrustZone 비교

말에 저장되는 보안이 필요한 데이터의 양이 증대함에 따라 H/W 기반의 보안 기술은 그 필요성이 점차 증대되는 추세이다.

강력한 보안수준을 유지하기 위해서는 일반적으로 추가적인 하드웨어 장치와 통신하는 방식으로 H/W 보안을 구현하는데 비해 ARM의 TrustZone 기술은 추가적인 H/W 장치가 필요 없이 구현이 가능하기 때문에 최근의 모바일 환경에서 그 사용성이 높아지고 있는 추세이다([그림 6] 참조).

TZ(TrustZone)는 하나의 CPU를 물리적으로 독립시켜 안전한 실행환경을 제공하는 H/W 보안 기술이다. 이는 아래 [그림 7]과 같이 범용 OS가 설치되어 운영되는 NWd(Normal World)와 외부로부터의 접근이 차단되어 안전한 실행환경을 보장하는 SWd(Secure World)로 분리되어 실행된다. 기존 기술들에 비해 TZ는 다양한 서비스를 제공하는 Trustlet을 실행할 수 환경을 제공하고 있어 확장성을 높였다. SWd 상에서 실행되는 Trustlet은 안전하게 관리되며 하드웨어적으로 안전한 Secure Storage를 이용 할 수 있고 peripheral 장치들을 직접 접근하는 등 안전성도 제공한다.

TZ는 NWd와 SWd를 시분할 형태의 인터럽트를 이용하여 NWd의 어플리케이션이 동작하면서 SWd가 필요 할 경우 인터럽트를 발생시켜 SWd에서 동작을 하도록 한다. 이는 TZ가 별도의 H/W를 추가하지 않고 AP 안에 포함되어 있는 자원을 사용하기 때문이다. 이로 인해 메인 코어를 NWd에서와 같이 동일하게 사용하므로 성능도 타 H/W 솔루션에 비해 우수하며 개발 및 생산적인 측면에서 비용절감의 효과가 있다.



〔그림 7〕 TrustZone 구성도

이러한 TZ 기술로 현재까지는 일부 단말기 제조사들에 의해 DRM Contents 재생 등에 제한적으로 활용되고 있다. TZ를 이용함으로써 DRM Contents가 Platform OS를 통하지 않고 직접 비디오 재생 드라이버에 접근하게 할 수 있어 악성코드가 중요한 메모리에 접근하는 것을 방지할 수 있다.

하지만 이런 장점에도 불구하고 ARM 코어에 구현된 TrustZone은 가용할 수 있는 하드웨어 자원에 제약이 존재하여 모든 금융 어플리케이션 전체를 SWd에서 실행 시키는 것에는 어려움이 있으며 Main Core를 시분할 방식으로 사용하기 때문에 SWd를 수동적으로 사용할 수밖에 없는 한계가 따른다.

Ⅲ.금융 어플리케이션 보호를 위한 보안 시스템

본 장에서는 하드웨어 기반의 안전한 실행 환경(Secure Execution Environment)을 제공하는 TZ 기술에 존재하는 제약 사항을 고려한 보안 시스템을 제안한다. 이 시스템은 스마트 폰에 설치되는 금융 어플리케이션 뿐만 아니라 일반적인 어플리케이션을 보호할 수 있는 효과적인 보안 시스템이다. 이를 위해서 특정 어플리케이션의 설치 및 실행을 TZ에서 수행하는 것 보다는 실행코드[1]를 암호화 하여 TZ보다 저장공간이 큰 파일 시스템에 안전하게 저장하고 앱 런처가 해당 APK을 실행할 때 복호화 및 검증 기능만을 TZ에서 실행하도록 하여 자원의 효율성 및 보안성을 제공하는 보안 시스템을 제안하였다. 이번 장에서는 이 시스템에 도입된 요소

1) Native의 경우는 실행 가능한 .text section을 의미, APK의 경우는 classes.dex을 의미

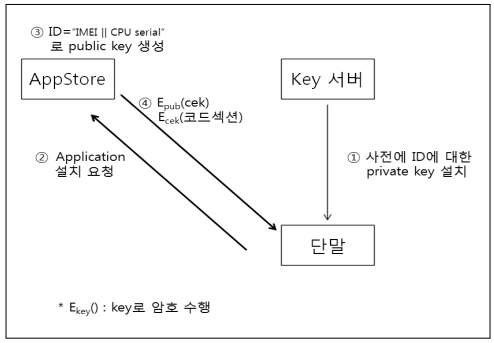
기술에 대해서 설명한다.

3.1. 어플리케이션의 코드섹션 암호화

사용자가 앱스토어에서 취약성을 내포하는 어플리케이션을 다운받아 설치할 경우 사용자의 의도와 무관하게 버퍼 오버플로, 코드 포인터 취약성 공격 등에 노출될 수 있다. 기존 DRM 시스템을 이용할 경우는 전체 어플리케이션 파일을 암호화하므로써 위·변조 및 불법 복제 등과 같은 공격에 대해 안전성을 제공한다. 하지만 실행 파일의 크기가 작다면 문제가 없지만 3D 게임, 오피스 어플, 소프트웨어 코덱을 가진 멀티미디어 플레이어 등 어플리케이션 파일의 크기가 크다면 성능 이슈가 발생할 수 있다. 이는 어플리케이션을 실행시마다 복호화가 발생하여 로딩속도가 떨어질 수 있기 때문이다.

물론 복호화 시간을 단축하고자 부분 암호화를 적용한 DRM 시스템도 존재하지만 전체 실행코드를 암호화하는 것이 아니기 때문에 여전히 위·변조 공격에 대한 취약점이 존재한다. 본 제안된 기술에서도 부분 암호화를 지원한다. 하지만 전체 어플리케이션 파일을 균등하게 부분 암호화하는 것이 아닌 위·변조 공격의 대상인 실행 코드만을 암호화 하기 때문에 복호화시 부하가 감소하고 또한 공격자가 실행코드를 복호화 하는 것이 어렵기 때문에 실행코드 위·변조 공격에도 안전하다.

어플리케이션 보호를 위한 암호화에 사용하는 키를 사용하는데 있어서의 제약조건이 있다. 첫째는 모든 어플리케이션을 동일한 암호키로 암호하는 것에는 위험이 존재한다. 암호키가 노출시 설치된 모든 어플리케이션을 복호화 할 수 있기 때문이다. 두 번째는 암호키의 복제가 불가능하여 저장된 스마트 폰에서만 복호화가 수행되어야 한다. 이를 만족하지 않을 경우 어플리케이션 불법복제를 통해서 타 스마트 폰에 복사하여 실행시킬 수 있기 때문이다. 암호키 관리를 위해서 제안된 방법은 각 어플리케이션마다 새로운 암호키를 생성하여 암호화를 하며 사용된 암호키는 IMEI, CPU Serial number, 전화번호 등 스마트 폰마다 고유한 값을 사용하여 다시 암호화되어 사용자가 어플리케이션 다운로드 받을 때 같이 배포되어 안전한 곳에 저장되어진다. 위에서 나열된 고유 값들은 해당 값을 보유한 스마트 폰에서만 이용 가능하지만 해킹을 통해서 쉽게 노출되기 때문에 고유값을 직접 암호화 키로 사용하는 데는 문제가 있다. 본 시스템에서는 고유 값들을 이용하여 식별코드로 사용하는 용도와 암호화 키를 분배하는 용도로만 사용된다. 이를 위해서 ID를 공개키(Public Key)로 사용하는 IBE(ID-Based Encryption)기법을 사용하였다[11]. 이 기술의 장점은 고유 값들이 노출된다 하더라도 공개키에 대응하는 개인키(Private Key)를 알지 못하는 이상 암호키를 복호화 할 수 없다([그림 8] 참조).



〔그림 8〕 IBE 기반의 암호기법 적용 과정

사용자가 앱스토어(AppStore)에 어플리케이션 설치 요청을 하면 앱스토어는 실행코드를 암호화 키(cek)로 암호 후, 요청한 단말의 고유 ID로부터 공개 키를 생성하여 암호화 key인 cek를 암호화 값을 같이 전송한다. 단말은 사전에 설치된 개인키(Private Key)로 복호화를 수행 할 수 있다. 이 방식의 또 다른 장점은 각 스마트 폰마다 Device 인증서의 설치 및 관리가 필요 없어지므로 중요 정보 관리에 이점이 있다는 것이다.

3.2. 안전한 개인 정보 저장

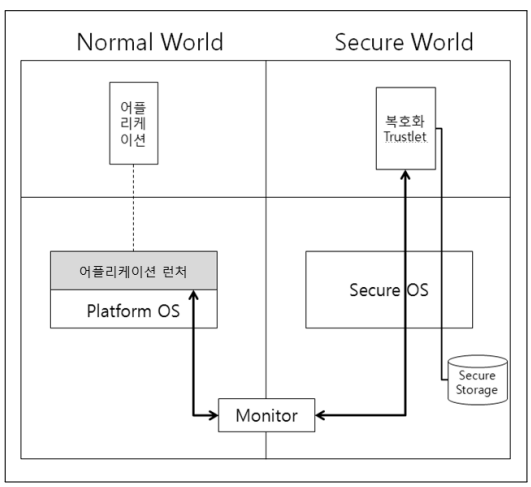
실행파일 보호를 위해 DRM 기술 등을 이용할 경우는 발급된 라이선스에 단말의 인증서로 암호화된 암호키가 존재하며 라이선스의 무결성을 보장하기 위해 라이선스 서버의 개인 키로 전자 서명되어 있다. 라이선스의 무결성을 검증하고 암호키를 획득하기 위해서 단말의 개인 키와 라이선스 서버의 인증서가 필요하다. 이는 중요 정보로써 스마트 폰에 안전하게 저장 및 관리되어야 한다. 만약 저장된 개인키가 노출된다면 모든 어플리케이션의 암호를 복호화할 수 있기 때문에 관리에 주의를 기울여야 한다. 또한 공격자가 만든 가짜 라이선스 서버 인증서를 생성하여 이를 단말에 저장시킨다면 위조된 라이선스를 생성하여 배포할 수 있기 때문에 신뢰된 라이선스 서버의 인증서만이 사용되도록 관리되거나 혹은 라이선스 서버 인증서의 유효성 검증을 위해 온라

인으로 상태검증이 필요하다. 이는 불필요한 네트워크 트래픽을 유발할 뿐만 아니라 코드 회피 기법을 통해서 우회가 가능하다. 제안된 보안 시스템에서는 관리되어야 하는 개인 정보가 적다. 단지 IBE 복호화에 사용되는 개인키만이 관리되며, 추가적인 검증을 위한 시스템이 불필요하다. 또한 하드웨어적으로 안전한 저장공간인 TZ의 Secure Storage에 저장하기 때문에 해킹에 노출되거나 복제가 불가능하여 안전하다.

3.3. 하드웨어 기반의 어플리케이션 런처

TZ의 SWd에는 Trustlet이라는 서비스 프로세스가 실행되며 NWd의 악성 코드의 접근이 불가능하다. 하지만 하드웨어의 자원적 제약으로 동시에 다수의 Trustlet를 실행할 수 없는 문제가 있다[12]. 이런 제약조건을 고려하여 모든 어플리케이션은 NWd에서 실행하며 이때 안전하게 어플리케이션을 실행시키는 어플리케이션 런처를 하드웨어 기반의 안전한 환경에서 실행한다면 모든 어플리케이션의 보안성은 높아질 수 있다. 제안된 기술에서는 어플리케이션 런처를 구성하는 기능 중에 복호화 기능만을 Trustlet을 이용하여 TZ에 구현하게 함으로써 효율을 기하였다([그림 9] 참조).

암호화된 어플리케이션의 복호화를 TZ에서 실행하므로써 복호화에 사용된 어떤 정보도 노출없이 어플리케이션을 복호화를 수행하여 실행시킬 수 있다. 또한 코드 우회 기법을 통해 TZ의 복호화 기능을 우회한다면 정확한 복호화가 이루어지지 않기 때문에 어플리케이션 실행에 실패하게 된다. 이로써 모든 어플리케이션을 TZ에서 실행하지 않고서도 안전하게 실행 시킬 수 있다.



〔그림 9〕 TZ를 이용한 어플리케이션 런처 구성도

Ⅳ. 결 론

개방형 플랫폼인 스마트 폰에서 소프트웨어 기반의 보안 기술을 이용하는 것에는 보안성의 한계가 있다. 이를 극복하고자 소프트웨어 기반의 보안 기술보다 높은 보안성을 제공하는 하드웨어 기반의 TZ를 이용하고자 하였다. 최상의 솔루션은 모든 어플리케이션을 TZ에서 실행하는 방법이 있다. 하지만 보안성은 높아질 수는 있으나 반면에 효율이 낮아지고 대용량의 어플리케이션을 다수 실행하는데 어려움이 존재한다.

본 논문에서는 어플리케이션을 실행시키는 어플리케이션 런처를 TZ를 이용하여 안전하게 수행 할 수 있게 함으로써 이런 어플리케이션 런처를 통해서 실행되는 어플리케이션도 안전성을 보장을 수 있다는 점에 주목하였다.

어플리케이션의 코드 위·변조 공격 및 불법 복제 방지를 위해 전체 APK 파일 중 코드영역인 classes.dex 만을 암호화하여 복호화로 인한 부하를 줄였고, 스마트폰에 안전하게 저장해야 하는 필수 정보를 최소화 하며 제한적인 TZ의 저장공간의 제약사항을 대처하였다. 암호화를 위해서 ID 기반의 암호 시스템을 도입하여 인증서를 사용하지 않아 관리의 효율을 높였다. 또한 어플리케이션 런처의 기능 중 핵심 부분인 복호화 기능만을 TZ에서 실행하도록 하여 전체 어플리케이션을 TZ에서 실행하는 것 보다 효율화를 이루었다.

본 논문에서는 다루지 않았지만 어플리케이션이 복호화된 형태로 메모리 상에 존재할 때 발생할 수 있는 문제에 대해서는 추후 더 연구할 부분으로 남겨둔다.

참고문헌

[1] http://mobizenpekr.files.wordpress.com/2012/06/vs.gif

[2] Android Open Source Project, http://source.android.com/source/downloading.html

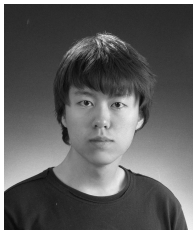
[3] http://www.ddaily.co.kr/news/news_view.php?uid=87801

[4] Chen-Yuan Chuang, Yu-Chun Wang, Yi-Bing Lin,"Digital Right Management and Software

Protection on Android Phones", IEEE, 2010
[5] Signing Applications, http://developer.android.com /tools/publishing/app-signing.html
[6] 김후종, 정은수, 임재봉, “능동형 콘텐츠 지원을 위한 OMA DRM 프레임워크의 확장”, *한국정보보호학회 논문지*, 2006.10
[7] 김영설, “모바일 플랫폼 환경의 애플리케이션 및 라이선스 보호를 위한 DRM 시스템”, *한국정보기술학회 논문지*, 2011.12
[8] 조경옥, “Mobile 환경에서 DRM 콘텐츠 보호를 위한 패키징 메커니즘 설계 및 구현”, *한국정보기술학회 논문지*, 2010.09
[9] 하태진, 김정호, 양원일, 김상언, 한승조, “무선 DRM 기반의 모바일 콘텐츠 보안 시스템 설계”, *전자정보통신학회 논문지*, 2006.03
[10] 김경태, 권성호, 표창우, “안드로이드 플랫폼의 코드 포인터 취약성”, *한국정보과학회 논문지*, 2010
[11] Identity-Based Cryptography Standard, http:// www.rfc-editor.org/rfc/rfc5091.txt
[12] Tiago Alves and Don Felton, "TrustZone: Integrated Hardware and Software Security", *TECHNOLOGY IN-DEPTH Information Quarterly [18] Volume 3, Number 4, 2004*

〈著者紹介〉

김 진 형 (Kim Jin Hyung)
2001년 2월 : 국립 인천대학교 컴퓨터공학과 졸업
2003년 2월 : 홍익대학교 컴퓨터공학과 석사
2003년 4월~2009년 4월 : 소프트포럼 연구원
2010년 4월~ 2011년 3월 : 금융보안연구원 선임연구원
2011년 4월~ 현재 : 삼성전자 책임연구원
관심분야 : 모바일 지급결제, DRM, 정보보호



김 태 호 (Kim Tae Ho)
2011년 2월 : 홍익대학교 컴퓨터공학과 졸업
2011년 2월 ~ 현재 : 삼성전자 연구원
관심분야 : TrustZone, 모바일 결제, DRM, 정보보호