논문번호 12-37C-09-12

한국통신학회논문지 '12-09 Vol.37C No.09
http://dx.doi.org/10.7840/kics.2012.37C.9.851

안드로이드 플랫폼을 위한 자바 보안 프로바이더 설계 및 구현

손 미 경◆, 강 남 희°

Design and Implementation of Java Crypto Provider for Android Platform

Mikyung Son◆, Namhi Kang°

요 약

안드로이드 기반 스마트기기에서 보안 응용 서비스를 개발하기 위해 SUN JCA/JCE나 BC JCE와 같은 자바 기반 보안 라이브러리를 사용한다. 자바를 사용하여 제작된 응용은 자바 가상 머신(JVM: Java Virtual Machine)에서 기능이 수행되므로 시스템 하드웨어에 의존하는 기능을 사용하기 어렵고 실행 속도가 저하되는 문제가 있다. 보안 프리미티브(primitive)의 경우 계산양이 많고 연산 복잡도가 높아 JVM에서 수행되는 보안 프리미티브를 적용할 수 있는 응용은 제한적이다. 특히, 실시간 특성을 요하는 스트리밍 서비스나 계산양이 많은 공개키 기반 알고리즘을 빈번히 적용하는 보안 응용 서비스의 경우 성능의 문제로 자바 라이브러리 사용이 어렵다. 이를 해결하기 위해 본 논문에서는 JNI와 NDK 도구를 사용하여 C나 C++ 언어로 구현된 네이티브 기능을 사용하여 보안 기능을 수행할 수 있는 프로바이더를 설계하고 구현한다. 제안하는 자바 보안 프로바이더(DSCrypt)는 기존 자바 기반 보안 라이브러리와 동일한 방식으로 사용할 수 있지만 빠르게 보안 기능이 수행되는 장점을 제공한다.

Key Words : JCA/JCE, Android, Smart-Phone, JNI, NDK

ABSTRACT

Java crypto library such as SUN JCA/JCE or BC JCE is generally used to implement secure applications for smart devices using Android platform. Programming functions written by Java language are launched and executed inside Java Virtual Machine (JVM), thereby difficult to use system hardware specific functionalities and degrading performance as well. In case of crypto primitive, few secure applications can use crypto primitive executing in JVM because both amount of computing and complexity of such primitives are very high. From the aspect of performance, in particular, time sensitive real time applications such as streaming services or secure application frequently applying public key based crypto algorithm cannot use Java crypto library. To solve the problem, we design and implement crypto library which employ JNI and NDK methods to directly access functions that implemented by native language such as C or C++. The proposed Java Crypto provider supports faster execution. Also developer can use our provider in the same way by writing traditional Java crypto library.

※ 본 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 연구되었음 (No.2012-0003377)
※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2012-H301-12-4008)
◆ 주저자 : 덕성여자대학교 컴퓨터공학부, skysmk02@duksung.ac.kr, 준회원
° 교신저자 : 덕성여자대학교 컴퓨터공학부, kang@duksung.ac.kr, 정회원
논문번호 : KICS2012-06-298, 접수일자 : 2012년 6월 30일, 최종논문접수일자 : 2012년 8월 13일

Ⅰ. 서 론

애플사의 아이폰 판매 성공을 계기로 스마트폰이 대중화되기 시작하여 매년 사용자 수가 급성장하고 있다. 국내의 경우, 5000만 이상의 이동통신 전화 가입자 중 50%이상이 이미 스마트폰을 사용하고 있다. 가트너의 2011년 자료에 따르면 전 세계적으로 스마트폰 시장이 PC 시장을 추월했다고 보고된다. 스마트폰의 대중화와 더불어 오픈 마켓을 활용한 모바일 응용 및 서비스 역시 빠르게 배포/전파되고 있다. 이러한 패러다임 변화는 사용자에게는 편리성을 제공하고 서비스 제공자에게는 새로운 이윤 창출 시장을 제공한다.

상기 기술한 긍정적 측면에 반하여 이기종(heterogeneous) 기기들이 시간 장소에 제한받지 않고 연결되는 스마트 ICT(Information and Communication Technology) 환경에서는 알려지지 않은 다양한 보안 위협들이 존재한다. 보안의 관점에서 기기나 서비스의 다양성과 이종성은 보안 서비스의 정의나 취약성에 대응하는 시스템 설계 시 큰 난제 요소가 된다. 특히, 사용자의 개인 정보가 많이 저장되는 스마트폰이나 스마트패드의 경우 공격의 주 대상이 된다.

스마트폰은 개방형 플랫폼 기반의 범용 운영체제가 주로 사용되고 있어 악성 코드 등 악의적 소프트웨어 제작이 용이하다. 스마트기기에 이를 대응할 수 있는 방법이 극히 열악한 현 상황도 큰 문제로 지적되고 있다[1]. 또한 전 세계적으로 사용이 확산되고 있는 오픈 마켓을 이용하면 배포 역시 용이하다[2]. 배포된 악의적 소프트웨어는 개인 정보 도용 및 유출, 불법 사용 등 2차 공격으로 발전될 수 있다. 따라서 기밀성, 무결성, 인증 등의 보안 서비스 정의와 안전한 시스템 설계 기술들이 요구된다.

본 논문에서는 스마트 기기 중심의 ICT 환경에서 보안 응용이 쉽게 개발될 수 있는 안드로이드 기반 보안 라이브러리를 설계하고 구현한다. 안드로이드는 애플의 iOS와 더불어 가장 많이 사용되는 스마트폰 플랫폼이다. 안드로이드 플랫폼에서는 자바 기반 보안 라이브러리를 제공하고 있다. 현재 구현되어 있지 않은 보안 프리미티브의 경우도 구현 후 확장 적용할 수 있는 방안 역시 제공한다. 그러나 자바 기반 응용 서비스는 달빅(Dalvik) 가상 머신에서 수행되므로 하드웨어 의존 작업 처리가 어렵고 실행 속도가 저하된다[3]. 특히, 실시간 멀티미디어 스트리밍 응용과 같이 실시간성이 보장되어야 할 서비스에 자바 보안 라이브러리를 채택하기에는 무리가 있다.

본 논문에서는 자바 네이티브 인터페이스 (JNI: Java Native Interface) 기술[4]을 사용하여 가상 머신 외부에 네이티브 언어로 구현된 보안 기능들을 사용하여 성능 저하를 줄이면서 보안 기능을 수행할 수 있는 방법을 제안한다. 또한 제안하는 보안 라이브러리는 개발자에게 기존 방식과 동일한 프로그래밍 방법을 제공한다. 즉, 보안 응용이나 서비스 개발자는 기존 자바 보안 라이브러리를 사용하는 것과 동일한 방식으로 개발하지만 실제 동작은 JNI를 통해 네이티브 기능이 사용되어 보안 서비스의 성능을 향상시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 보안 라이브러리를 설계 구현하기 위해 사용되는 라이브러리와 도구들을 간략하게 기술한다. 3장에서는 제안 시스템의 설계 방식을 설명하고 4장에서 개발 방법을 기술한다. 5장에서 해쉬 함수를 예로 구현된 결과물의 동작 및 성능을 평가한다. 마지막으로 6장에서 결론을 맺는다.

Ⅱ. 관련 연구

2.1. 자바 보안 라이브러리

프로그래밍 언어이자 플랫폼으로 많이 사용되는 자바에서는 보안 응용이나 서비스 개발을 위해 JCA(Java Cryptography Architecture)와 JCE(Java Cryptography Extension) 라이브러리를 제공한다. JCA는 보안 프리미티브 처리를 위한 인터페이스 및 팩토리를 정의하고 있다. 또한 신규 보안 프리미티브 개발, 디자인 패턴 및 알고리즘을 정의하기 위해 확장 가능한 구조로 설계되었다. JCE는 미국 정부의 정책 중 암호 소프트웨어 수출의 제한을 수용하기 위해 제한된 프리미티브를 포함한 라이브러리로 구분되어 출시되었다[5].

JCA/JCE는 기밀성, 무결성, 인증 등의 보안 서비스를 제공할 수 있는 다양한 보안 프리미티브를 포함한다. 자바 개발자는 구현되어 있는 모든 보안 프리미티브 자체에 대한 깊은 지식 없이도 제공되는 SPI(Service Provider Interface)를 통해 쉽게 보안 응용을 설계하고 구현할 수 있다. 즉, 이미 검증된 공개 라이브러리인 JCA/JCE를 사용하면 보안 프리미티브 구현 자체보다 응용 서비스의 보안 요구사항에 더 집중할 수 있다. 따라서 양질의 보안 응용 서비스를 빠르고 안전하게 구현할 수 있다. 이러한

장점은 자바를 사용하는 안드로이드 기반 모바일 플랫폼에서도 동일하게 적용된다.

2.2. 암호서비스 프로바이더

자바 보안 라이브러리는 프로바이더(CSP: Crypto Service Provider) 개념을 사용한다. SUN에서 개발하여 배포한 SunJCE를 포함하여 BouncyCastle의 BC, Graz 대학의 IAIK 등 다양한 CSP가 공개되어 있다. BC는 Sun사의 프로바이더에 비해 더 다양한 보안 알고리즘을 제공하며 국내 표준 블록 암호 알고리즘인 SEED 또한 포함하고 있다[6]. IAIK JCE는 자바 플랫폼을 위한 암호 응용을 위해 포괄적인 기능을 제공하는 CSP이다[7]. 자바 개발자는 자신이 선호하는 (혹은 신용하는) 기관에서 구현하여 배포하는 CSP를 선택할 수 있다. 그림 1은 보안 플랫폼이 포함하고 있는 다양한 CSP 중 개발자가 선택한 CSP 내의 보안 알고리즘이 구현된 클래스가 인스턴스화되어 제공되는 동작 흐름을 나타낸다[5].

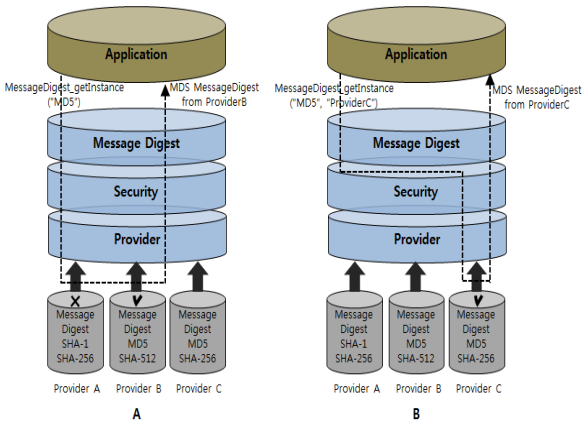


그림 1. 선택한 CSP의 MD5 기능 사용 예
Fig. 1. Example of MD5 implementation in Specific CSP

자바 보안 라이브러리의 Security 클래스는 프로바이더 객체의 목록을 유지 관리한다. Security 클래스는 개발자가 선택한 프로바이더가 목록 내에 존재할 경우 해당 클래스의 객체를 반환해준다. 그림 1의 경우 Provider A, B, C가 순차적인 우선순위로 설정되어 있는 프로바이더 목록에서 MD5 함수를 선택하는 예를 나타낸다. 선택하고자 하는 프로바이더 명을 명시하지 않을 경우 MD5를 구현한 높은 우선순위의 프로바이더가 선택된다(그림 1의 A). 이와 달리, 개발자가 구현 시 프로바이더 이름을 명시할 경우 해당 프로바이더의 MD5가 선택 된다 (그림1의 B, Provider C의 MD5가 인스턴스화 됨).

안드로이드 SDK(Software Development Kit)에는 Bouncy Castle에서 제작한 BC 프로바이더가 기본 CSP로 사용된다. 최근 버전인 안드로이드 4.0 ICS(Ice Cream Sandwich)는 BC 1.46 버전을 사용한다. 그러나 모바일 기기의 저장 공간을 고려하여 기존 자바 시스템을 위해 제공되었던 보안 알고리즘 전체가 제공되진 않는다. 또한 안드로이드 플랫폼 버전에 따라 제공되는 알고리즘은 다르다. 예로, 경량화 된 키 적용 특성으로 모바일 디바이스에 적용되기 용이하다고 평가되는 ECDH나 EC- DSA의 경우 안드로이드 2.3버전인 Gingerbread에서는 제공하지 않는다. 개발자가 제공되지 않는 보안 알고리즘을 안드로이드 플랫폼에 추가하고자 할 경우 기본 CSP의 이름인 BC와 충돌을 방지하기위해 프로바이더 명을 다르게 구분해야 한다. Bouncy castle에서는 SC(SpongyCastle)의 이름으로 BC보다 많은 보안 프리미티브가 구현된 통합 프로바이더를 제공한다[6].

2.3. OpenSSL

C/C++ 언어 기반으로 개발된 OpenSSL은 서버와 클라이언트간의 안전한 통신을 위해 배포된 개방형 보안 라이브러리이다[8]. OpenSSL은 정적 라이브러리 또는 동적 라이브러리 형태로 제공되고 높은 신뢰성과 호환성을 장점으로 보안 응용 서비스 개발에 널리 사용되어져 왔다.

표 1. OpenSSL.에서 제공되는 보안 프리미티브
Table 1. Crypto Primitive in OpenSSL

Crypto Classification	Crypto Primitive
Hash Function	MD5, MD2, SHA-1, SHA-256, MDC-2
Symmetric Key	AES, DES, Triple DES, RC2, RC4, RC5, IDEA
Public Key	RSA, DSA, Diffie-Hellman, ECC

현재 최신 버전은 2012년 5월 14일 발표된 1.0.1 버전이다. OpenSSL은 표 1에 기술된 해쉬함수, 대칭키, 공개키, 인증서 등 다양한 보안 프리미티브를 제공한다.

2.4. JNI와 NDK

자바로 구현된 응용들은 달빅(Dalvik) 가상 머신

상에서 동작되는데 이는 네이티브 수행에 비해 성능이 저하된다[3]. 이를 보완하기 위해 자바에서는 C나 C++과 같은 이종 언어로 작성된 코드와 연동할 수 있는 인터페이스인 JNI(Java Native Interface)를 제공한다. 즉, 자바 개발자는 JNI를 사용하여 플랫폼 내의 네이티브 코드를 직접 사용할 수 있고, 응용의 동작 성능을 개선할 수 있게 된다. 안드로이드 환경에서 JNI로 최적화하는 경우 다른 모바일 운영체제에 비해 향상된 성능을 가진다[9,10].

안드로이드 NDK(Native Development Kit)는 C나 C++로 작성된 코드를 사용하여 응용을 개발할 수 있도록 하는 개발 도구이다. 자바 언어로 사용자 인터페이스를 작성하고 연산 복잡도가 높아 성능 저하 요인이되는 기능은 C/C++로 구현된다. 구현된 코드는 NDK 도구를 통해 확장명이 so인 라이브러리 파일을 생성하게 된다. 이렇게 생성된 라이브러리를 JNI를 통해 호출하여 사용하는 방식을 기본으로 한다. 그림2는 JNI와 NDK를 통해 공유 라이브러리를 생성하는 과정을 나타낸다.

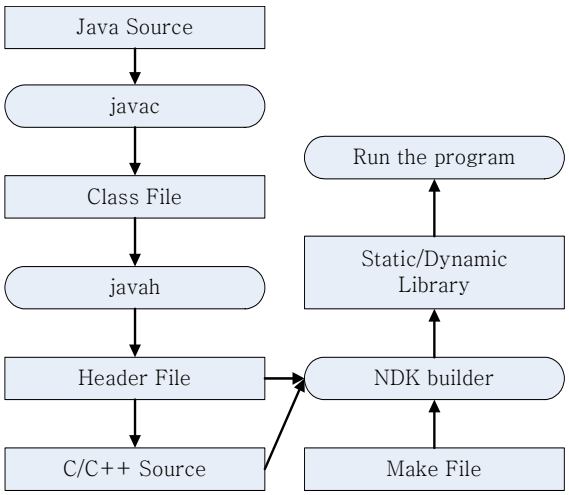


그림 2. JNI와 NDK를 이용한 라이브러리 생성 과정
Fig. 2. Library Building using JNI and NDK

안드로이드 프로젝트 내부에 구현되는 자바 소스코드와의 연동을 위해 javac와 javah 유틸리티를 사용한다. 전술 했듯, C나 C++기반 코드는 NDK 도구를 이용하여 빌드된 네이티브 코드가 라이브러리 형태로 생성된다. 생성된 라이브러리를 사용하여 네이티브 메소드에 접근하고 기능을 이용하게 된다.

Ⅲ. 제안 시스템 설계

안드로이드는 리눅스 커널을 기반으로 한 모바일 기기 운영 플랫폼이다. 이 플랫폼에서 실행되는 응용은 주로 자바를 기반으로 개발된다. 따라서 안드로이드 개발 환경은 자바 가상 머신 적용의 장점으로 인해 모든 OS에서 구현되고 실행될 수 있다. 그림 3는 안드로이드 플랫폼 구조를 나타낸다. 안드로이드 플랫폼은 리눅스 커널, 라이브러리, 안드로이드 런타임, 어플리케이션 프레임워크, 어플리케이션과 같이 5계층으로 구성되어있다[11].

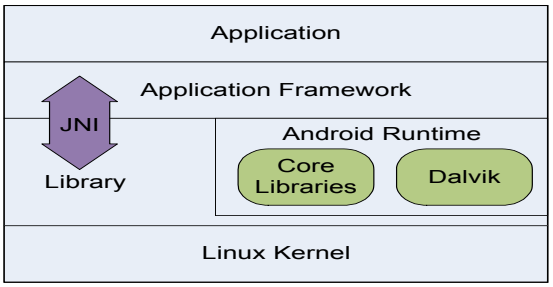


그림 3. 안드로이드 플랫폼 구조
Fig. 3. Android platform structure

안드로이드 플랫폼 최하단에 위치한 리눅스 커널은 메모리, 디바이스 드라이버 하드웨어 등을 관리하며 C언어로 작성되어 있다. 라이브러리 계층은 SQLite나 SSL과 같이 C와 C++로 구현된 다양한 라이브러리를 제공한다. 안드로이드 런타임 계층은 Java 코아 라이브러리와 달빅으로 구성되며 안드로이드의 특별한 가상 머신인 달빅 가상 머신에 의해 달빅 코드로 변환하여 구동 된다. 어플리케이션 프레임워크는 개발자의 응용 개발을 원활하게 하기 위해 지원되는 패키지 컴포넌트로 자바로 구현되었으며 최상위 어플리케이션 계층에서의 자바로 개발된 프로그램인 응용을 지원한다.

안드로이드 응용은 어플리케이션 프레임워크를 통해 필요에 따라 리눅스 커널위에서 동작하는 C/C++ 라이브러리를 호출하는 코어 라이브러리 기능을 사용 할 수 있다[12]. 어플리케이션 프레임워크와 라이브러리 사이의 연결고리 역할이 JNI이며 이 인터페이스를 통해 네이티브 코드로 작성된 라이브러리를 호출하여 원하는 기능을 자바가 아닌 C/C++로 구현된 모듈에서 수행할 수 있다[3].

안드로이드 보안 응용의 경우 기본적으로 JCA나 JCE를 사용하여 자바 언어로 구현한다. 그러나 속도 저하와 배터리의 큰 소모량 등이 단점으로 지적된다. 본 논문은 이와 같은 문제점을 해결하기 위해 자바 가상 머신 내부에 포함되어 있는 JNI를 사용

한다.

그림 4는 본 논문이 제안하는 보안 라이브러리 개발을 위한 설계 구조를 나타낸다. 그림에서 자바 언어로 구현된 부분은 A와 B로 표시했다(B의 범위는 그림내에 박스로 표기했음). 반면, 네이티브 코드를 직접 사용하는 (즉, C/C++로 개발된) 부분은 C로 표시했다. A 부분의 보안 응용은 안드로이드 플랫폼에서 동작하는 응용으로 자바가 제공하는 보안 라이브러리인 JCA/JCE를 통해 필요한 보안 프리미티브를 사용한다.

상기 기술한 것처럼 본 논문에서는 자바 보안 라이브러리의 속도 저하 문제를 해결하기 위해 네이티브 기능을 사용한다. 즉, 계산량이 많이 요구되어 성능에 영향을 주는 보안 프리미티브 연산이 자바 라이브러리 내부가 아닌 네이티브 코드 영역에서 수행된다. 이를 위해 C언어 기반의 OpenSSL을 안드로이드 플랫폼에 이식한다. 제안하는 암호 프로바이더인 DSCrypt는 응용에서 요청한 보안 기능을 JNI를 통해 OpenSSL에 구현된 네이티브 함수에서 수행할 수 있도록 해준다.

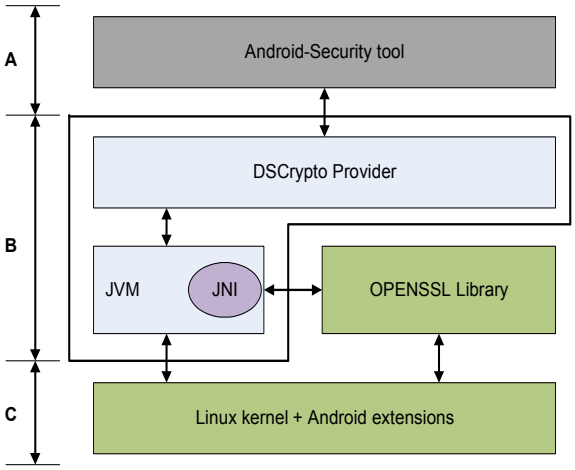


그림 4. DSCrypt 프로바이더 설계 구조
Fig. 4. Design Architecture of DSCrypt Provider

Ⅳ. DSCrypt 프로바이더 구현

그림 5는 안드로이드 응용에서 무결성을 제공하기 위해 DSCrypt 프로바이더의 해쉬 함수를 이용하는 예를 나타낸다. 자바 보안 라이브러리 내의 해쉬 함수를 이용하던 방식과 다르게 동작될 수 있도록 DSCrpty를 프로바이더 이름에 해당하는 매개변수에 명시한다. 즉, 기존의 안드로이드 기반 보안 응용에서 개발하는 방식과 동일하게 인스턴스를 생성하고 사용할 수 있도록 한다.

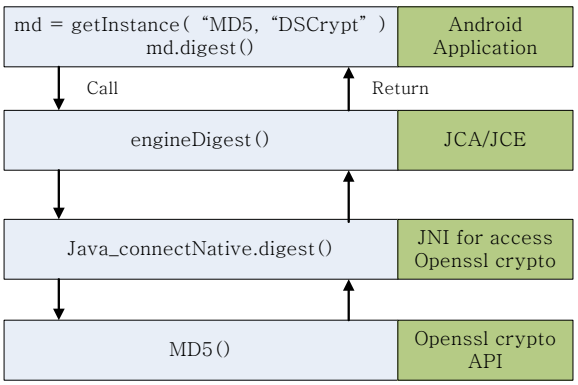


그림 5. DSCrypt 프로바이더의 MD5 인스턴스
Fig. 5. Instance of MD5 in DSCrypt Provider

DSCrypt 보안 라이브러리의 engineDigest() 함수는 해쉬 함수 기능이 수행되지 않는다. 대신 JNI의 접근 함수(즉, Java_connectNative.digest())를 통해 OpenSSL 라이브러리 내부에 구현된 MD5()함수를 호출한다. 이러한 동작 흐름에 따라 안드로이드 응용에서 필요한 보안 프리미티브 연산이 네이티브 영역에서 수행될 수 있다. 이를 통해 안드로이드 응용은 성능 저하 없이 보안 서비스를 제공하게 된다. 그림 6은 상기 동작 흐름을 코드 단위로 자세히 나타낸다.

그림 6은 C로 작성된 (즉, 네이티브 영역) OpenSSL 라이브러리의 MD5() 함수를 성능 평가용으로 제작한 응용(SecurityTool)에서 접근하여 이용하는 동작 흐름을 나타낸다. 안드로이드 자바 응용에서 입력받은 텍스트 데이터가 네이티브 라이브러리에 입력되어 해쉬 되는 처리 과정은 다음과 같다.

- 그림 6의 ①은 안드로이드 응용인 SecurityTool 클래스에서 DSCrypt 프로바이더 내부의 MD5 해쉬함수의 인스턴스를 생성하는 것을 나타낸다.
- ②의 과정을 통해 네이티브 메소드를 선언하고 라이브러리를 로드하는 자바코드를 작성하고 javah를 사용하여 헤더 파일을 생성한다.
- 생성된 헤더파일을 기반으로 C/C++로 구현 네이티브 메소드를 구현하는 과정을 ③에서 수행한다.
- ④에 나타낸 것처럼 libcrypto.so와 같이 필요한 외부 라이브러리를 참조해야 할 시 해당 모듈을 로드하는 함수를 호출한다.

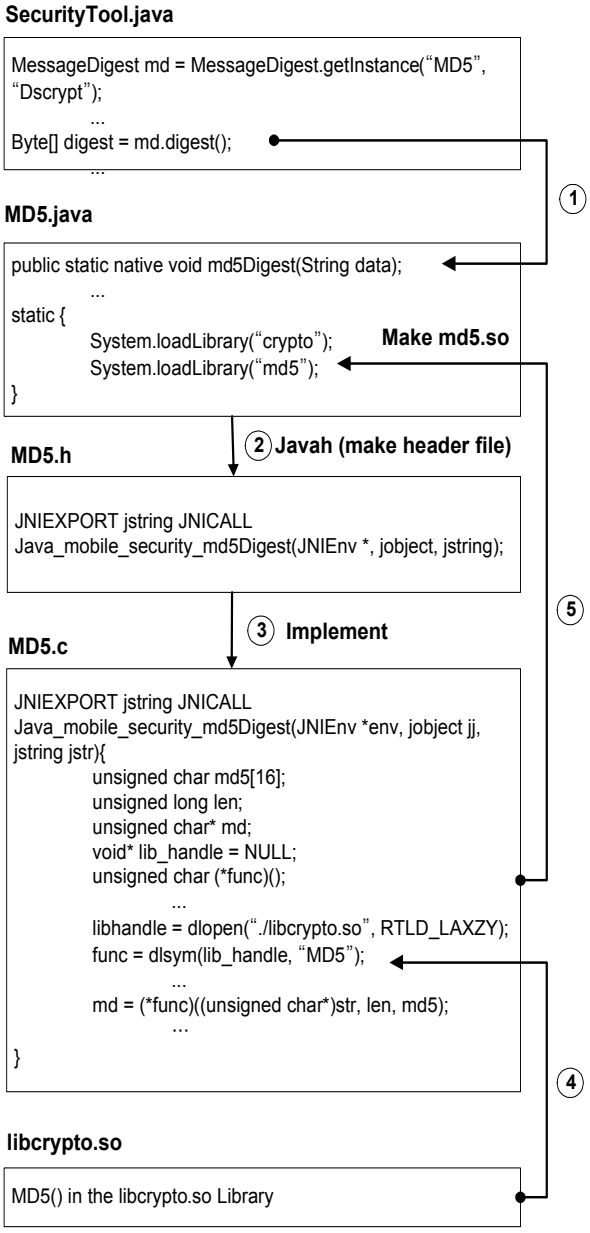


그림 6. 동작 흐름
Fig. 6. Functional Flow

- 위와 같은 일련의 과정들로 구현한 C/C++로 구현된 네이티브 라이브러리를 컴파일하여 so 확장명을 가진 동적 라이브러리를을 생성하여 적재하는 모습을 ⑤의 과정으로 나타내었다.
- 결과적으로 SecurityTool.java에서 DSCrypt 프로바이더의 MD5 해쉬함수 사용 시 기존의 메시지 다이제스트를 호출하는 방식 그대로 digest()함수를 사용한다.
- 그러나 프로바이더 내부적으로는 C/C++로 네이티브 메소드를 선언 및 구현한 MD5.h와 MD5.c 및 외부 라이브러리인 libcrypto.so를 가지고 생성한 md5.so 동적 라이브러리를 통해 자바로 수행되는 기능이 아닌 네이티브 기능을 제공 받을 수 있다.

위의 과정을 통해 필요한 네이티브 메소드를 제공받을 수 있는 JNI 모듈을 생성하고 안드로이드 응용에서 네이티브 영역에 구현된 기능을 수행할 수 있게 된다.

Ⅴ. 시험 및 성능 평가

DSCrypt 프로바이더의 동작 시험 및 평가를 위해 표 2에 기술된 시스템을 사용했다.

표 2. 시험 기기 사양
Table 2. specification of test device

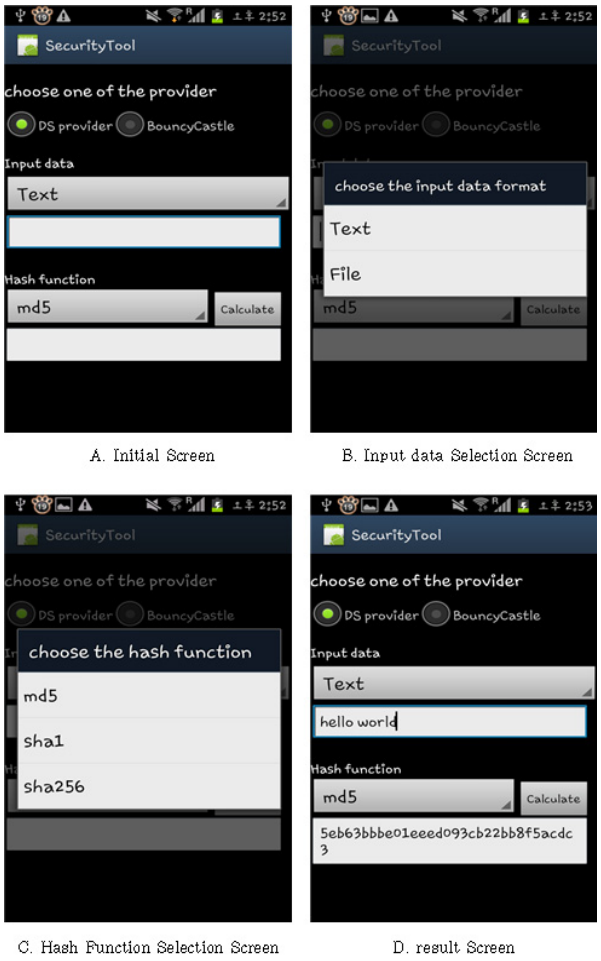
Processor	ARM Cortex-A9 1.2GHz Dual Core
Memory	1GB DDR2 RAM
OS	Android Platform 4.0.3 (icecream sandwich version)
Battery	1650mAh

그림 7은 안드로이드 기반 모바일 기기 상에서 시험용 응용이 구동된 화면의 스냅샷이다.

그림 7의 A 상단의 프로바이더 설정 기능은 사용자가 신뢰하는 프로바이더를 선택 할 수 있는 기능이다. 본 시험에서는 제안하는 보안 프로바이더와 이와 비교하기 위해 사용한 BouncyCastle 프로바이더 중 한 가지를 선택할 수 있다. B에 표시된 것처럼, 해쉬 함수에 입력되는 데이터는 직접 입력하는 텍스트나 파일 중 한 가지를 선택 할 수 있다. C에 나타낸 것처럼 사용자는 제공받고자 하는 해쉬 함수를 설정하고 난 뒤 연산을 수행하면 D와 같은 해쉬 함수의 출력 결과를 확인할 수 있다.

표 3은 DSCrypt 프로바이더의 성능을 비교한 결과이다. 현재 JNI와 NDK를 통해 네이티브 기능을 사용하여 성능을 향상시키는 제안 기술들은 많다. 그러나 보안 기능을 수행할 수 있는 프로바이더를 설계하고 구현한 기술은 발표되고 있지 않다. 따라서 본 논문에서 제시한 DSCrypt 프로바이더와 기본적으로 보안 응용 서비스를 개발할 때 사용하는 자바 기반 프로바이더와의 비교를 통해 보안 프리

미티브 성능을 평가했다.



A. Initial Screen B. Input data Selection Screen

C. Hash Function Selection Screen D. result Screen

그림 7. 시험 응용의 구동 화면
Fig. 7. Screen Shot of Test Application

본 논문에서는 성능 비교를 위해 자바 기반 프로바이더를 사용한 경우와 제안한 DSCrypt를 사용한 경우에서 해쉬 함수가 수행되는 시간을 측정했다. 시간 측정은 코드 레벨인 라이브러리 내부에서 실제 시간을 측정할 수 있는 함수를 구현하여 해쉬값 연산 시간만을 계산하였다.

성능 시험에 사용한 해쉬 함수로는 MD5, SHA1, SHA256을 사용했다. 네이티브 코드로 작성된 외부 라이브러리를 참조하여 보안 기능을 수행하는 DSCrypt Provider가 자바로 작성된 프로바이더에 비해 좀 더 빠르게 연산이 수행됨을 알 수 있다. 단일 연산일 경우 작은 차이일 수 있으나 스트리밍 서비스나 좀 더 복잡한 연산이 요구되는 보안 프리미티브일 경우 큰 성능차이를 보일 것이다.

표 3. 성능 측정 결과 비교
Table 3. Performance comparison

Hash Function	Input length(bit)	DSCrypt Provider (unit of measure μs)	Bouncy Castle (unit of measure μs)
MD5	512	12	119
	1024	20	139
SHA1	512	15	88
	1024	22	101
SHA256	512	16	150
	1024	28	211

Ⅵ. 결 론

본 논문에서는 안드로이드 기반 플랫폼에서 보안 응용 서비스를 개발할 수 있도록 해주는 보안 라이브러리를 설계 구현했다. 구현된 DSCrypt 프로바이더는 자바 기반으로 작성되었지만 네이티브 영역에 구현된 보안 프리미티브를 직접 호출할 수 있는 방식을 적용하여 기존 자바 기반 보안 라이브러리의 성능 문제를 개선했다. 또한 안드로이드 개발자가 기존 방식과 동일하게 보안 라이브러리를 사용할 수 있도록 하여 쉽게 사용할 수 있다.

참 고 문 헌

[1] P. Pocatilu, "Android Applications Security," *Informatica Economica*, vol. 15, no.3, 2011.

[2] K.Y.Kim, D.H.Kang, "Smart Phone Security Technology in Open Mobile Environment," *Korea Institute of Information Security & Cryptology*, vol. 19, no. 5, 2009.
김기영, 강동호, "개방형 모바일 환경에서 스마트폰 보안 기술," 정보보호학회지, 제19권 5호, 2009.

[3] Sang-Hoon Kim, "Integrated Development Environment for Java Native Methods," *The Korea Contents Association*, vol. 10, no. 7, pp. 122-132, 2010.

[4] R. Gordon, "Essential JNI: Java Native Interface," prentice Hall, 1998.

[5] JCA/JCE Documentation, Retrieved Aug., 8,

2012, from http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html
[6] Bouncy Castle Crypto API, Retrieved Aug., 8, 2012, from http://www. bouncycastle.org/
[7] Garz University, Crpto Toolkit, IAIK JCE, Retrieved Aug., 8, 2012, from http://jce.iaik.tugraz.at/
[8] OpenSSL Project, Retrieved Aug., 8, 2012, from http://www.openssl.org
[9] Won-Ki Jung, Beom-Jun Kim, Soo-Mook Moon, "A Study for Ensuring Compatability on Android of Low-Level Virtual Machine," *The Korean Institute of Information Scientists and Engineers*, vol. 38, no. 2, 2011.
[10] H. Lee, D. Shin, H. Jung, "Implementations of Block Cipher SEED on Smartphone Operating Systems," *Int. Conf. Proceeding of Conference on Emerging Security Information, Systems and Technologies*, 2011.
[11] C. Maia, L. M. Nogueira, L. M. Pinho, "Evaluating Android OS for Embedded Real-Time Systems," Technical Report, 2010.
[12] MinKoo Choi, Nakyoon Choi, Younglim Chool, Jong-Wook Kim, "implementation of uDEAS on the Android platform," Korean Institute of Information Technology, vol. 9, no. 6, 2011.

손 미 경 (Mikyung Son)

2011년 2월 덕성여자대학교 컴퓨터공학부 졸업
2011년 3월~현재 덕성여자대학원 전산정보통신학과 석사과정
<관심분야> 네트워크 보안, 안드로이드 플랫폼

강 남 희 (Namhi Kang)

2001년 2월 숭실대학교 정보통신대학원 공학석사
2004년 12월 University of Siegen 컴퓨터공학과 공학박사
2009년 3월~현재 덕성여자대학교 디지털미디어학과 조교수
<관심분야: 유무선 인터넷통신, 통신보안, 시스템보안>