主題

J2ME와 보안

드림시큐리티 이 근 수

차 례

Ⅰ. 개 요

유선환경에서의 인터넷이 활성화됨으로써 보안이 중요한 영역으로 부각되고 많은 솔루션들이 나와있는 실정이며 또한, 최근 무선 인터넷이 새로운 사업영역으로 떠오름에 따라 무선쪽에서도 보안의 중요성이 증대하고 있다. 일본의 경우 I-mode라는 서비스로 무선 인터넷의 혁신적인 변화를 주도하면서 대중화에 많은 부분을 기여했다.

기존 무선 인터넷 플랫폼의 경우 WAP, ME 방식이 시장에서 대표적으로 쓰여졌었는데 Sun에서 새로운 플랫폼인 J2ME를 발표하면서 기존 플랫폼들의 정적인 서비스를 동적인 서비스로 바꾸는 계기가 마련되었다. 무선 플랫폼으로의 자바는 동적이고 그래픽이 지원되며 applet처럼 다운을 받아서 실행을 시키는 형태이다. 기존 mobile device처럼 업그레이드를 위해 하드웨어적으로 할 필요가 없으며 필요할 때 다운을 받아서 하는 형태이므로 기존 환경에 비해서 혁신적인 변화라 할 수 있다.

J2ME는 기존 자바의 플랫폼 독립성을 그대로 유지하지는 못했지만, 많은 부분을 포용하므로 유선에서의 자바의 위력을 그대로 무선으로 옮겨가는데 한 몫을 하리라 생각한다.

이번 글에서는 이러한 무선 인터넷의 증대와 J2ME의 등장으로 부각되고 있는 mobile java라는 주제로 보안을 적용하기 위한 기본 개념들과 자바의 기본 보안 모델 및 자바와 보안의 접목이라는 내용으로 알아보겠다.

Ⅱ. 공개키 기반 구조 (Public Key Infrastructure)

1. 공개키 기반 구조의 개요

통신과 컴퓨터 기술이 비약적으로 발전하면서 현재 우리가 일상적으로 행하던 은행업무, 쇼핑 등의 거래를 전자적으로 행하는 전자거래가 시행되고 있

고, 특히 인터넷의 급속한 확산에 의한 인터넷상의 전자상거래가 활발히 전개되고 있는 실정이다.

국가경쟁력 강화의 일환으로 전자상거래를 활성화하기 위해서는 먼저 전자상거래의 안전성 및 신뢰성을 확보해야 한다. 현재 국내에서는 전자상거래의 정보보호 문제로 인하여 전자상거래를 활성화하는 데 문제점들이 노출되고 있으며 이를 해결하기 위하여 관련 부처에서 현행 법/제도 정비 필요성을 인식하여 많은 논의가 이루어지고 있는 실정이다.

전자상거래의 안전성 및 신뢰성을 확보하는 시작은 전자인증제도의 정립이라고 말할 수 있다. 전자인증 제도란 가상공간상의 전자문서, 전자거래 등 관련 전자업무에서의 당사자의 신분확인 기능, 전자업무 내용의 정보보호 및 무결성 기능, 전자행위에 대한 부인봉쇄 기능 등 전자업무의 중요 인증과 관련하여 신뢰할 만한 제 3자(인증기관)가 확인 및 증명해 주는 제도이다. 전자인증제도의 핵심은 전자상거래의 안전성과 신뢰성을 확보하기 위한 핵심 기술인 전자서명기술의 안전한 운영을 의미한다.

전자서명기술은 공개키 암호 알고리즘으로 개인키와 공개키가 사용된다. 공개키 암호 알고리즘에서 사용되는 개인키가 전자서명을 생성하는 생성키가 되고 공개키가 전자서명을 검증하는 검증키 역할을 한다. 그러므로 전자서명기술의 안전한 운영은 서명키(공개키 암호 알고리즘의 개인키)와 검증키(공개키 암호 알고리즘의 공개키)의 안전한 운영에 달려있으며 서명키의 안전한 운영은 개인키의 안전한 보관을 말하며 검증키의 안전한 운영은 공개키의 안전한 관리를 의미한다. 공개키는 공개된 정보이므로 어떻게 공개키 위·변조 문제를 해결하는가 하는 공개키 인증문제로 귀착된다.

이러한 공개키의 인증문제를 해결하기 위해 나온 것이 바로 공개키 기반 구조(PKI: Public Key Infrastructure)이다.

다시 말해, 전자상거래의 안전성과 신뢰성을 확보하기 위해서는 전자인증제도가 요구되며 전자인증제도는 바로 전자서명기술의 안전한 운영을 의미하고 다시 전자서명기술에 사용되는 공개키 암호알고리즘의 개인키의 기밀성과 공개키의 무결성을 보장해야 하며 이를 해결하고자 하는 것이 바로 공개키 기반 구조이다. 즉, 공개키 기반구조 구축은 전자인증제도를 실체화하는 것이다.

그렇다면 PKI가 구현되기 위한 최소한의 개체들을 알아보도록 하겠다.

2. 공개키 기반 구조의 구성 요소

정책승인기관
(PAA : Policy Approving Authority)
- PKI 전반에 사용되는 정책을 생성하고 PKI 구축의 루트 CA로의 역할 수행- PKI 전반에 사용되는 정책과 절차를 생성, 수립하고 하위 인증기관들의 정책 준수상태 및 적성성을 감사
- PKI 상호인증을 위한 정책을 수립하고 승인하며 하위기관의 공개 키를 인증하고 인증서, 인증서 취소목록 등을 관리

정책인증기관(PCA : Policy Certification Authority) - PAA 하위 계층
- 자신의 Domain 내의 사용자와 인증기관(CA)이 따라야 할 정책을 수립
- 인증기관의 공개 키를 인증하며, 인증서, 인증서 취소목록 등을 관리.

인증기관 (CA : Certification Authority)
- 사용자의 공개 키 인증서 발행, 취소
- 사용자 자신의 공개 키와 상위기관의 공개 키 전달
- 등록기관의 요청에 따른 인증서 생성과 발급
- 인증서와 가입자 정보의 관리, 감사 파일 관리 등을 수행

등록기관(RA : Registration Authority)
- 인증기관과 사용자 사이에 위치하여 사용자들의 인증서 신청 시 신분과 소속을 확인하고 인증서 발급신청을 대행

디렉토리
- 인증서와 사용자 관련정보, 인증서 취소목록 등을 저장 및 검색하는 기능
- DAP(Directory Access Protocol)이나 LDAP(Lightweight DAP)을 이용하여 X.500 디렉토리 서비스를 제공.

사용자 or 가입자 시스템
- 일반적인 클라이언트로 자신의 비밀 키, 공개키 쌍 생성
- 공개 키 인증서 요청 및 획득, 전자서명 생성 및 검증
- 특정 사용자 인증서 획득 및 상태 결정 등의 기능 수행
- 인증경로 해석과 디렉토리를 이용한 자신의 인증서를 다른 사용자에게 제공

Ⅲ. 보안 서비스 유형

이제는 보안의 측면에서 제공될 수 있는 서비스의 형태를 알아보도록 하겠다.

일반적으로 인터넷에서 나타날 수 있는 취약한 점들을 알아보고 그러한 부분들을해결할 수 있는 방법들을 알아보도록 하겠다.

1. 인증(Authentication)

가. 사용자 인증(User Authentication)

분산 환경이 보급되면서 통신망에 접속한 사용자가 정당한가를 확인하는 과정이 필요하게 되었으며 중요한 정보나 자원을 보호하기 위해서는 컴퓨터 통신망에 불법 접속을 시도하는 것을 차단하는 방법이 필요한데 이러한 정당한 사용자를 확인하는 과정을 사용자 인증이라고 한다.

• 패스워드 기반 인증

현재 인터넷에서 가장 일반적으로 사용되고 있는 인증방식으로 ID와 Password쌍의 조합으로 사용자를 확인하는 방법이다. 구현하기 용이하기 때문에 많이 쓰이고 있기는 하지만 데인 서버에 부정 접속을 시도하는 사용자에 취약한 약점을 가지고 있다.

• 공개키 암호 인증

공개키 기반의 인증이라 함은 위에서 설명한 인증서 기반으로 상호 인증을 하는 것을 말한다. 기본적으로 자신의 공개키는 공개를 함으로써 수신자가 송신자를 인증하기 위해서는 송신자의 공개키가 있으면 된다. 송신자는 개인키는 본인만이 비밀리에 관리하며 자신이 공개한 공개키에 대응하는 개인키를 가지고 있다는 사실로 자신을 인증 받게 된다.

나. 메시지 인증(Message Authentication)

또한 통신망에서 전달되는 정보가 위조되지 않고 상대방에게 전달이 되는지 확인하는 것 역시 중요한 일이다. 송신되는 정보의 변경은 송신자와 수신자 사이에 불신과 분쟁을 유발할 염려가 있어 송신자가 전송한 내용이 수신자에게 정보의 변경 없이 전송되었는가를 상호 확인 할 수 있어야 한다. 이러한 과정을 메시지 인증이라 한다.

2. 접근 통제(Access Control)

불법적인 제3자의 접근을 차단하고 사용자마다 정보 및 시스템에 다른 권한을 부여할 수 있어야 한다. 특히 다수의 사용자가 이용하는 시스템에서는 이러한 접근 통제는 더욱 중요하게 된다.

3. 비밀 보장(Confidentiality)

인터넷이 대중화 되면서 공개된 전송로상으로 정보를 주고 받는 경우가 빈번해지고 있으며 TCP/IP의 취약성으로 인해 제3자가 중간에서 정보를 가로채는 것이 가능해졌다.

이러한 도청을 방지하기 위해서 전송되는 데이터를 알아볼 수 없도록 암호화를 하는 것이 필요하게 된다. 이러한 암호화는 키의 종류에 따라서 대칭키 및 비대칭키(공개키) 방식으로 구분이 되는데 이러한 부분에 대해서 간단하게 알아보겠다.

가. 대칭키 암호

대칭키 암호 방식은 암호화 및 복호화에 쓰이는 키가 동일한 방식을 말한다.

따라서 송신자 및 수신자는 정보를 주고 받기 위해서 서로 키를 공유하고 있어야 하는데 대칭키 암호방식에서는 이러한 키의 분배의 문제점이 있다. 일반적으로 많이 쓰이는 방식으로 DES, RC계열, SEED 등이 있다.

나. 공개키 암호

위에서 설명했듯이 공개키는 사용자 인증에 쓰이기도 하지만 순수하게 데이터를 암호화하는데 쓰이기도 한다. 대칭키 암호 방식에서 키의 분배에 문제가 있다고 했는데 처음에 서로 키를 모르는 상태에서 서로의 키를 공유하기 위해서 송신자는 수신자의 공개키로 대칭키를 암호화해서 전송하게 되면 그 공개키에 해당하는 개인키는 수신자만이 알고 있으므로 수신자만이 복호화 할 수 있게 된다. 대표적인 공개키 방식으로는 RSA, DH, ECC 등이 있다.

4. 데이터 무결성(Integrity)

공개된 전송로상으로 데이터를 주고 받을 때 수신자는 송신자의 메시지가 중간에 위조가 되지 않았는지 확인을 하는 절차가 필요한데 이러게 데이터 중간에 변조가 되지 않게 하는 서비스를 무결성이라 한다. 일반적으로 이러한 무결성을 제공하기 위해서 Hash라고 하는 기법이 사용된다.

가. 해쉬(Hash)

일반적으로 일방향 함수로 알려져 있는데 수학적으로는 A=H(M)에서 M을 가지고 A를 만들어 낼 수는 있지만 A를 가지고 M을 알아낼 수는 없다는 점을 이용하는 것이다.

보내는 메시지의 Hash의 결과를 같이 보내서 수신자가 메시지의 변조 여부를 확인할 수 있게 하는 것이다.

5. 부인 봉쇄(Non-repudiation)

실세계에서는 서로를 만나서 계약서를 만들고 도장을 찍고 하기 때문에 나중에 계약에 대해서 부인을 할 수가 없는데 통신로상에서도 이러한 것들을 가능하게 해주는 서비스가 부인 봉쇄라 하며 대표적인 것으로 전자서명이 있다.

가. 전자 서명

위에서 설명한 공개키 방식이 전자서명의 방법으로 쓰이는 대표적인 것이다. 개인 키와 공개키가 있다고 했는데 송신자는 자신만이 알고 있는 개인키(서명키)로 암호화를 해서 그 결과를 보내주게 되는데 자신이 공개한 공개키(서명 검증키)로 수신자는 송신자의 서명을 검증 할 수 있게 된다.

Ⅳ. PKI 관련 서비스

1. SSL(Secure Sockets Layer)

SSL은 웹브라우저로 잘 알려져 있는 netscape

사에서 만들었으며 현재까지 웹에서의 보안의 대표적인 프로토콜이다. 현재는 버전3.0까지 나와 있으며 인증서 기반으로 클라이언트와 서버간의 인증 및 메시지의 기밀성, 무결성을 제공해준다.

가. SSL의 구조

SSL은 하나의 프로토콜이 아니며 기존 프로토콜에서 사용되며 응용프로그램과 TCP/IP사이에서 보안서비스를 제공하는 형태이다

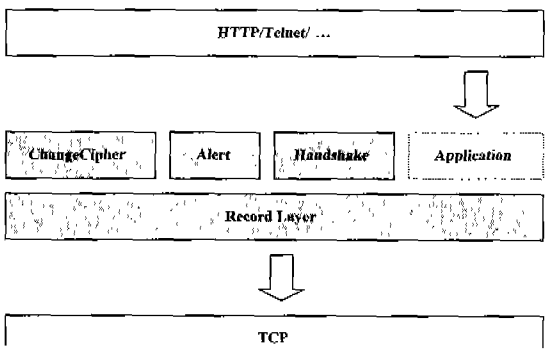

그림 1. SSL의 구성요소

그림 1에서 보듯이 SSL은 4부분(회색부분)으로 나누어져 있다. 각 부분에 대해서 간단하게 알아보겠다

나. SSL의 동작

• Handshake

클라이언트와 서버간 안전한 통신을 하기 위해서는 서로를 인증해야 하며 필요한 암호매개변수를 공유해야 하는데 이러한 절차를 하기 위한 부분이 Handshake이다. 간단하게 그림 2로 알아보겠다.

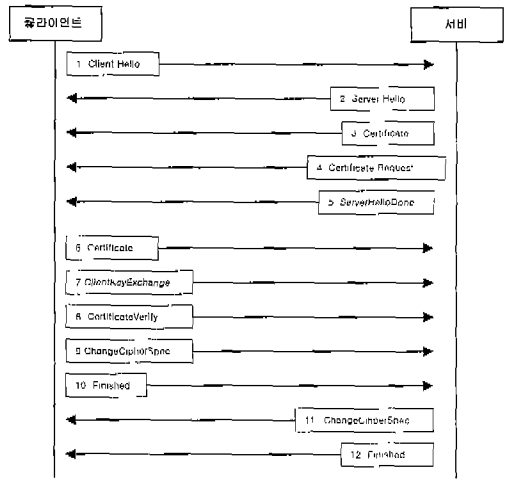

그림 2. SSL Handshake 과정

1. ClientHello
   클라이언트가 자신이 원하는 암호화 알고리즘 등 자신이 지원할 수 있는 타입들을 제안.
2. ServerHello
   클라이언트가 제안한 옵션들 중 서버가 하나씩 선택하여 응답을 전송.
3. Certificate
   서버는 클라이언트에게 자신의 공개키를 인증서의 형태로 전송.
4. CertificateRequest
   서버는 클라이언트에게 인증서를 요구.
5. ServerHelloDone
   서버는 자신의 메시지를 다 보냈다는 메시지를 전송.
6. Certificate
   클라이언트가 자신의 공개키를 인증서의 형태로 서버에게 전송.
7. ClientKeyExchage
   클라이언트는 3번에서 받은 서버의 공개키로 앞으로 통신할 때 쓸 대칭키에 관한 정보를 암호화해서 전송.
8. CertificateVerify
   6번에서 보낸 클라이언트의 공개키에 해당하는 개인키를 가지고 있다는 것을 증명하기 위해 자신의 개인키로 서명을 해서 전송.
9. ChangeCipherSpec
   클라이언트는 지금까지 서버와 협상한 사항들을 앞으로 통신할 메시지들에 대해서 활성화 시키겠다는 메시지를 전송.

10. Finished
클라이언트는 새롭게 협상된 내용들을 서버가 확인해 보도록 메시지 전송.
11. ChangeCipherSpec
서버는 지금까지 클라이언트와 협상한 사항들을 앞으로 통신할 메시지들에 대해서 활성화 시키겠다는 메시지를 전송.
12. Finished
서버는 새롭게 협상된 내용들을 클라이언트가 확인해 보도록 메시지 전송.

• Alert

오류를 알리기 위한 Record Layer의 메시지 타입중 하나.

• Change Cipher Spec

이 메시지 이후의 레코드에 대해 협상된 옵션을 이용하여 보호될 수 있도록 수신측에 알려줌.

• Record Layer

모든 메시지를 캡슐화하기 위해서 쓰이며 Alert, ChangeCipherSpec, Handshake에 공통된 포멧을 제공한다.

V. J2ME와 보안의 접목

1. J2ME의 개요

자바는 플랫폼이나 개발 성격에 따라서 3가지 Edition으로 나누어 지고 있다.

Java 1.1 버전까지는 일반적인 자바 Applet 이나 Application 의 개발이 목적 이였지만 Java 1.2 버전 이상에서는 자바의 용도가 좀 더 다양해지고 자바가올라갈 수 있는 플랫폼이 다양해져 그 특성에 맞는 자바가 등장하게 되었다.

그림 3에서 보는 것과 같이 세가지로 나뉘어져 있다.

Java 2 Enterprise Edition, Java 2 Standard Edition, Java 2 Micro Edition이 그것이다.

J2EE와 J2SE는 일반 PC나 Workstation에서 사용되는 플랫폼이나 OS에 맞는Edition으로 구성되어 있고, J2ME는 Handheld Device나 ScreenPhone, PDA, Set-top Box 등과 같은 네트워크로 연결되고 메모리나 CPU가 제한적인 Device에맞는 플랫폼 Architecture이다.

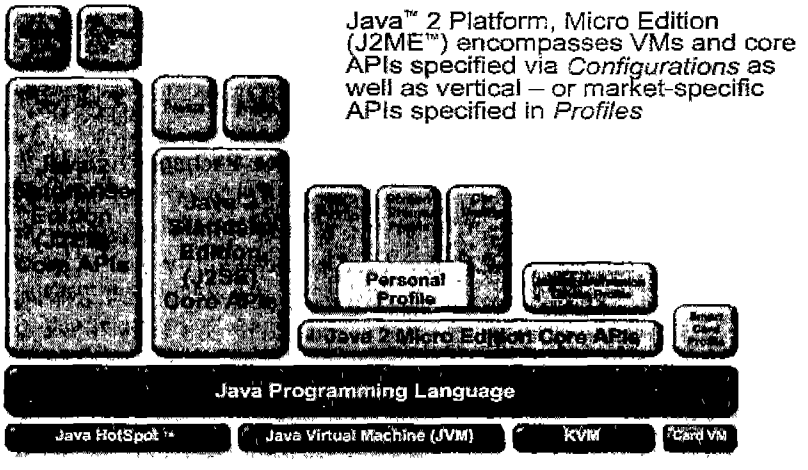

그림 3. 자바2 플랫폼과 그 목적 시장들

가. Configuration/Profile

Configuration이라 함은 하드웨어적으로 사양이 비슷한 기기들에 공통적으로 필요한 최소한의 가상 머신 기능과 핵심적은 API들을 묶어서 부른다.

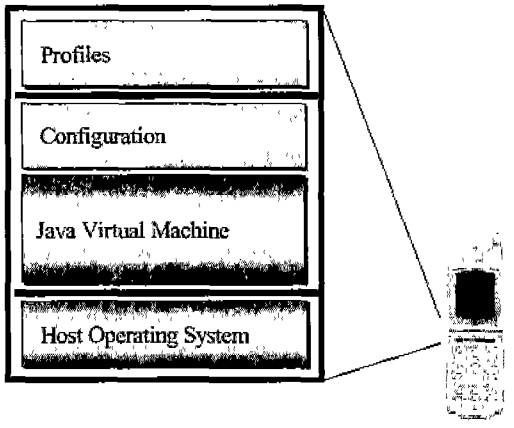

그림 4. J2ME software layer stack

Configuration기반에서 Target Device의 Type에 맞게 UI부분이던가 User input Method, 영구 데이터 저장공간, 메시지, 보안 그리고 무선 환경에서의 네트워크 처리 부분에 대해서 명세하게 되는데 이러한 부분이 Profile을 이루게 된다.

나. KVM

Kilobyte Virtual Machine의 약어로써 기존 자바 가상 머신의 서브셋이다. 적은 메모리의 용량과 CPU 처리 능력, 느린 무선 네트워크에 맞게 설계되었고, 크기는 40~80KB 정도이다. KVM은 128 KB정도의 메모리를 갖춘 16/32bit RISC/CISC 마이크로 프로세서에 목적을 맞추었다. J2ME의 라이브러리는 실제로 필요한 JDK의 축소된 classes로 되어 있으며 작은 기기에 맞는 간단한 UI Frame Work을 제공하고, 필요할 경우에만 플랫폼 Native function을 제공한다. 이들 라이브러리들은 CLDC와 관련된 spec의 일부인 Java Community Process에 의해서 표준화되어있다.

다. CLDC

CLDC(Connected, Limited Device Configuration) KVM 기반의 네트워크 연결능력이 있고 적은 리소스를 가지는 디바이스을 위한 Configuration이다. 기본적으로 160~512kb 정도의 메모리를 요구하며 프로세스는 16Bit/32Bit에서 동작하고, Battery의 소모량을 고려하여 전력소모를 최소화하였다. 또한 적은 대역폭(9600 bps)에 맞게 네트워크 기능이 설계되었다.

CLDC는 KVM과 J2ME Core API로 이루어져 있고, 입출력 기능, 네트워크, 보안, 국제화 등을 지원한다, 그러나 어플리케이션의 라이프사이클 관리나, 사용자 인터페이스, 이벤트 핸들링, 상위 레벨 어플리케이션 모델은 지원하지 않는다. 이러한 기능들은 뒤에 나오는 MIDP 부분에서 지원하게 된다.

그림 5는 CLDC로 작성된 자바파일을 compile한 후 사전 검증을 한 후에 target device에 download 되어 실행되는 모습을 나타낸 그림이다.

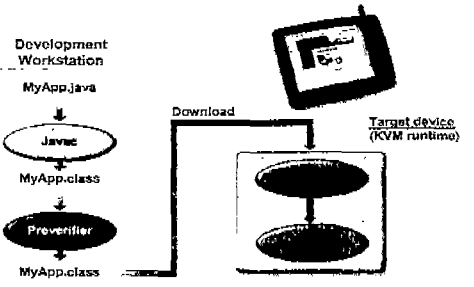

그림 5. CLDC/KVM에서 클래스 파일 검증

라. MIDP

Mobile Information Device Profile의 약자로써 KVM 기반의 Mobile Device을 위한 Profile 이다. MIDP는 Display toolkit과 User interface methods, Persistent data storage,

Messaging과 보안, Wireless Telephony Connection을 지원한다. 네트워크 기능의 확장으로 HTTP, TCP/IP 뿐만 아니라 Gateway도 지원하며 이는 WAP 과 I-Mode와의 연동을 뜻한다.

MIDP를 위한 하드웨어의 최소사양은 96x54의 스크린 사이즈와 1비트의 Display Depth, MIDP 구성을 위한 128KB의 비휘발성 메모리, 영구적인 데이터 공간을 위한 8KB, 자바 실행환경을 위한 32 KB정도의 메모리를 요구한다.

MIDP의 구조는 그림 6과 같다.

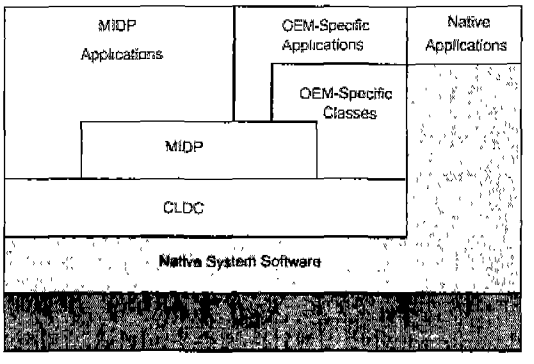

그림 6. MIDP 구조

2. J2ME에서의 보안

지금까지 보안에서 자주 나오는 내용과 J2ME 전반에 걸쳐서 알아보았다.

이제부터는 J2ME 환경에서 보안이라는 주제를 어떻게 접목을 시킬 수 있는지에대해서 알아보겠다.

가. J2EE/J2SE 보안

자바 2에서 암호 클래스의 전체적인 설계는 JCA (Java Cryptography Architecture)로 이루어져 있으며 그 구조를 그림으로 보면 그림 7과 같다.

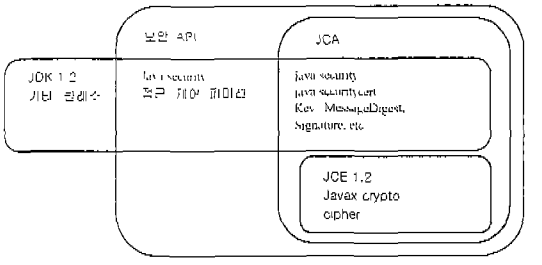

그림 7. 자바 보안 API 소프트웨어

그림 상에서 JCE(Java Crypto Extension)라고 되어 있는 곳이 암호화를 포함해서 critical한 부분들이 들어 있으며 현재 이 부분은 공개된 소스가 많이 있으며 자바로 암호 라이브러리가 구현되어 있으며 다음의 사이트에서 찾아 볼 수 있다.

Package	URL
jSec	http://osg.javaland.co.kr/servlet/OSG
JCE	http://java.sun.com/products/jce
Cryptix	http://www.cryptix.org
IAIK	http://jcewww.iaik.at/
lcrypto	http://www.bouncycastle.org

나. J2ME 보안

실제로 J2SE/J2EE 환경에서의 보안은 일반 PC 및 Workstation을 target으로 하고 있기 때문에 자바가 일반적으로 조금 느린 면에도 불구하고 요즘 컴퓨터의 사양에 비하면 신경을 쓰지도 않아도 될 만큼 속도의 차이가 없다. 하지만 J2ME환경은 개론에서 보았듯이 낮은 사양의 Device를 Target으로 하고 있기 때문에 쓰기에는 다소 무리기 따르는 것은 사실이다.

또한 J2ME의 환경은 J2SE/J2EE에서의 핵심 클래스들의 부분집합이기 때문에 JCE를 구현한 경우를 그대로 쓸 수도 없는 실정이다. 현재 J2ME환경에서 암호 라이브러리를 구현한 경우는 위의 표에서 마지막에 있는 lcrypto라는 패키지이다.

예를 들면 RSA 1024 Bit의 키를 생성하는 경우 일반 PC에서도 몇 초가 걸리는 작업인데 실제로 Palm Vx PDA에서 lcrypto라이브러리를 써서 작업한 결과 RSA 비트에 따라서 몇 십분에서 몇 시간까지 걸린다. 물론 알고리즘의 최적화 문제도 있겠지만, 현재 소형기기들의 사양과 자바의 특성상 아직까지는 순수 자바로 암호 라이브러리를 쓰는 것은 무리라 생각한다.

다른 방법으로 생각 할 수 있는 것이 J2ME spec에 나와 있는 Native Code를 이용하는 것이다. KVM의 경우 소스가 공개되어 있으며 C로 구현이 되어 있다. Native Code의 경우 KVM에 라이브러리를 구현하며 필요한 인터페이스만 자바로 정의를 하는데 VM을 통해서 호출되는 자바 라이브러리보다는 속도가 개선 될 것이다.

Ⅵ. 결 론

지금까지 무선 인터넷의 대중화 및 J2ME의 등장과 보안의 중요성이라는 주제로 알아보았다.

현재의 무선 환경은 많은 발전을 해왔으며 앞으로도 계속 발전을 해 나갈 것이다.

현재의 전화의 개념에서 컬러 액정, 동화상 및 사운드 지원 등 멀티미디어 기능이 강화되고 있으며 또한 PDA 같은 이동 단말기들도 무선 환경을 지원하고 있다. 이러한 하드웨어적인 발전으로 인해 자바 같은 정적인 플랫폼이 아닌 동적인 플랫폼의 등장은 어찌 보면 당연한 결과라 할 수 있다.

현재는 이러한 기술의 구현 및 발전이라는 주제로 시장이 흘러가고 있지만 무선 또한 유선처럼 발전을 해나가면 보안의 중요성은 더욱더 증대해 나갈 것이다. 이러한 무선 환경에서의 보안은 국가적으로도 기반기술에 속하며 많은 노력과 관심을 기울이는 것이 IT에서 경쟁력을 갖추는 길이라고 본다.

Ⅶ. 참 조

1. 참고 문헌

김 철, 암호학의 이해
생능 출판사, 전자 상거래 보안 기술
소프트 뱅크, 무선 인터넷 백서 2001
Stephen Thomas, SSL and TLS Essentials
Eric Giguere, Java 2 Micro Edition
Jonathan Knudsen, Java Cryptography

2. 참고 사이트

http://java.sun.com/j2me/
http://java.sun.com/products/cldc/
http://java.sun.com/products/midp/
http://www.rsasecurity.com/

이 근 수
1999년 광운대학교 벤쳐센터 보안 연구부, 1999년 성은 정보(주) ERP 업무, 2001년 드림 시큐리티 정보 보안 연구소