자바카드 기반 보안 메커니즘 설계 및 구현

전형득, 송유진

동국대학교, 정보산업학과

Design and Implementation of Security Mechanism based on JavaCard

Hyung Deuk Jeon, You Jin Song

Department of Management Information System, Dongguk University.

요약

최근 들어 일반인들을 대상으로 하는 인터넷 서비스가 보편화되면서 B2C 기반 전자상거래의 수요가 급속히 증가하고 있다. 뿐만 아니라 기업과 기업간(B2B) 혹은 기업과 정부간(B2G)의 전자상거래를 위한 포탈 사이트나 허브 사이트 등의 구축도 매우 활발히 이루어지고 있다. 이와 같은 인터넷 기반의 전자상거래 시스템은 사용자들에게 기존 상거래에서는 경험해 보지 못했던 다양하고 방대한 정보와 편리성 등을 제공하고 있다. 반면, 지불 정보나 구매 정보 혹은 개인 신상 정보 등의 유출과 같은 보안상의 문제점들을 해결해야 하는 어려움도 내재되어 있어 안전한 전자상거래를 보장하기 위한 일환으로 자바카드 기반의 사용자 인터페이스가 요구되고 있다. 여기서, 자바카드 상의 개방형 플랫폼 보안 서비스를 구현하는 것이 본 논문의 목적이다. 이러한 목적을 달성하기 위해 자바카드를 기반으로 하는 개방형 플랫폼상의 CEPS 보안 메커니즘을 분석하고, 개방형 플랫폼 구성요소인 보안 도메인(Security Domain)의 보안 요구사항에 근거한 자바카드 보안 서비스를 제공하는 기능을 설계 및 구현한다.

Ⅰ. 서론

최근 들어 일반인들을 대상으로 하는 인터넷 서비스가 보편화되면서 B2C 기반 전자상거래의 수요가 급속히 증가하고 있다. 뿐만 아니라 기업과 기업간(B2B) 혹은 기업과 정부간(B2G)의 전자상거래를 위한 포탈 사이트나 허브 사이트 등의 구축도 매우 활발히 이루어지고 있다. 이와 같은 인터넷 기반의 전자상거래 시스템은 사용자들에게 기존 상거래에서는 경험해 보지 못했던 다양하고 방대한 정보와 편리성 등을 제공하고 있다. 반면, 지불 정보나 구매 정보 혹은 개인 신상 정보 등의 유출과 같은 보안상의 문제점들을 해결해야 하는 어려움도 내재되어 있다.

최근 이러한 보안관리의 대상이 되는 정보들은 안전하게 관리하기 위한 방법으로서 휴대 가능한 스마트카드 시스템이 주목을 받고 있다. 현재 다양한 COS(Card Operating System)를 탑재한 스마트카드 시스템들이 존재하고 있으나 강력한 보안 기능 및 다중 응용프로그램 환경을 제공하는 MULTOS 기반 시스템과 JVM 기반의 자바카드 시스템으로 양분되어 가고 있는 추세이다.

자바카드 기반 개방형 보안 메커니즘은 다수의 애플릿이 하나의 카드 내에서 보안상의 충돌 없이 공존할 수 있도록 지원하고, 애플릿간의 상이한 보안 기능 지원이 요구된다. 응용 프로그램의 요청에 의해 보안 알고리즘 서비스를 제공하고, 발급자의 키와는 별도의 키를 생성하여 키 관리를 지원할 뿐만 아니라 카드 발급 후 응용 프로그램의 로딩, 인스톨, 삭제에 필요한 보안 절차와 관리 기능에 대한 요구사항을 필요로 한다.

본 논문에서는 이러한 보안 요구사항을 만족시킬 수 있는 자바카드 보안 메커니즘의 설계 및 구현에 대하여 논한다.

인터넷의 급속한 발달에 따라 다양한 비즈니스를 인터넷상에서 수행하는 전자상거래 움직임이 활발하게 전개되고 있으며 안전한 전자상거래를 보장하기 위한 일환으로 자바카드 기반의 사용자 인터페이스가 요구되고 있다. 여기서, 자바카드 상의 개방형 플랫폼 보안 서비스를 구현하는 것이 본 논문의 목적이다. 이러한 목적을 달성하기 위해 자바카드를 기반으로 하는 개방형 플랫폼상의 CEPS 보안 메커니즘을 분석하고, 개방형 플랫폼 구성요소인 보안 도메인(Security Domain)의 보안 요구사항에 근거한 자바카드 보안 서비스를 제공하는 기능을 설계 및 구현한다.

자바카드는 자바카드 2.1 사양과 CEPS의 설계 기준에 근간을 둔 개방형 플랫폼으로 설계한다. 개방형 플랫폼은 카드 발행자의 다양한 요구사항을 만족시키는 다기능 칩 카드시스템을 구축하는데 유연하면서도 강력한 표준으로 설계할 수 있는 장점이 있다. 또한, 플랫폼 및 하드웨어 공급업체와 상관없는 이러한 접근 방식은 시스템 운영업자의 투자, 특히 인프라 투자를 현실화시킬 수 있다. 그리고 개방형 플랫폼 아키텍처는 발급이후 애플릿서버에 저장된 새로운 응용프로그램의 구현뿐만 아니라 응용프로그램을 갱신 또는 제거할 수 있는 새로운 기능을 제시할 수 있다.[1][3]

본 논문의 구성은 다음과 같다. 2장에서 자바카드의 개요와 개방형 플랫폼에 대해 살펴보고, 3장에서 자바카드 기반 보안 메커니즘의 설계 및 구현에 대해 설명한다. 마지막으로 4장에서 결론을 내린다.

Ⅱ. 자바카드와 개방형 플랫폼

1. 자바카드의 개요

(1) 자바카드의 정의 및 특징

자바 카드란 COS(Card Operating System)위에 JCVM(Java Card Virtual Machine)이 랩핑(Wrapping)되어 있는 구조의 스마트 카드를 말하며, 자바카드 애플릿이란 자바카드 플랫폼에 맞게 설계된 자바카드용 응용 프로그램을 말한다.[2] 그러므로 자바카드 애플릿의 구현에는 기존의 자바 환경에서 개발되는 자바 응용 프로그램 구현 조건과는 다른 카드가 가지는 제한적인 하드웨어 리소스 활용 방안과 CAD(Card Acceptance Device)와의 커뮤니케이션 인터페이스, JCVM, 그리고 JCRE(Java Card Runtime Environment)요구 조건과 같은 카드의 소프트웨어적인 특성, 그리고 이러한 특성을 효과적으로 검증하기 위한 검증 방안 등이 고려되어야 한다.

자바카드의 특징은 플랫폼 독립성, 보안성 및 객체지향성 등과 같은 자바의 특징을 대부분 그대로 이어받고 있다는 점이다. 또한 개발자의 측면에서 자바 컴파일러 및 상용 IDE(Integrated Development Environment)와 같은 기존의 개발 도구를 그대로 사용할 수 있으며 개방형 API(Application Programming Interface)인 자바카드 API를 제공하고 있는 장점이 있다. 특히 스마트 카드 기술적인 측면에서 볼 때 자바카드 시스템은 다중 응용 프로그램의 지원, 응용프로그램의 후 발행(Post-issuance) 기능, ISO 7816과 같은 국제 표준과의 호환성 제공 등의 장점을 가지고 있다.

(2) 자바카드의 구조

자바 카드 하드웨어의 전형적인 칩 크기는 $23mm^2$로서 ISO 7816으로 규정된 IC 카드의 구조 및 인터페이스 규약을 따른다. 칩 내부에는 마이크로 프로세서, 암호 보조 프로세서, 메모리, 메모리 관리 유니트 그리고 해킹 방지 모듈들이 내장되어 있으며, 전체적인 하드웨어 구조는 아래 [그림 1]에 나타낸 바와 같이 데이터 부와 메모리 부 그리고 인터페이스 부로 나뉘어 진다.[2]

데이터 부는 메인 프로세서와 암호 연산용 보조 프로세서를 포함하고 있어 메모리의 읽기 쓰기 및 외부와의 통신 기능을 담당하고, 메모리 부는 ROM, RAM, 그리고 EEPROM 같은 기억 장치들을 소유하고 있어 데이터의 기록 및 보존을 담당하며, 인터페이스 부는 주변 기기와의 통신을 위한 입출력단자, 전압 공급 단자, 외부 클럭, 리셋, 접지 같은 접점들을 포함하고 있어 카드의 인터페이스 기능을 담당한다.

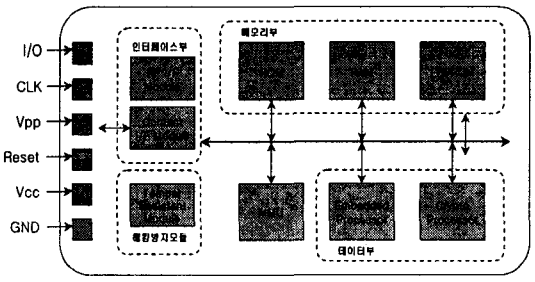

그림 1 : 자바카드의 하드웨어 구조

자바 카드의 소프트웨어 구조는 아래 [그림 2]에서 주어진 바와 같이 COS, JCVM, 자바카드 API, Industry Specific API, 그리고 애플릿으로 구성된다. COS 영역에는 메모리 액세스 및 I/O

핸들링을 위한 디바이스 드라이버와 암호 모듈 액세스 드라이버 코드가, JCVM 영역에는 자바 바이트코드 서브셋(Bytecode Subset) 지원을 위한 자바 인터프리터(Interpreter)코드와 서명과 로그인 같은 외부 접근 통제 코드가 적재된다. 자바 카드 API영역에는 자바카드 구현을 위한 기본 API코드, Industry specific API 영역에는 사용자에 의해 정의되는 암호 API 코드가 적재되며, 애플릿 영역에는 전자화폐, 로열티, 그리고 신분 증명 등과 같은 응용 프로그램이 적재된다.

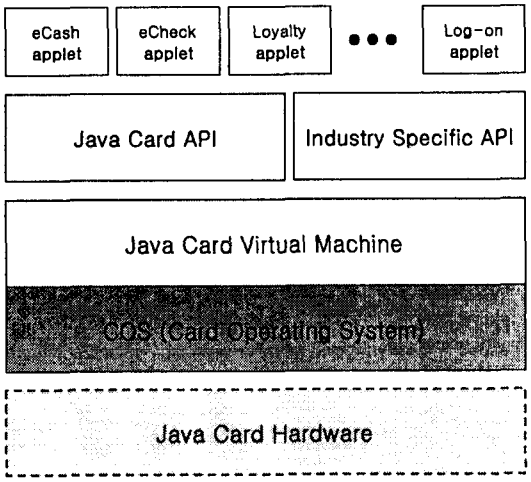

그림 2 : 자바카드 소프트웨어 구조

자바카드에 탑재되는 가상머신은 SUN Microsystems사와 자바카드 포럼에 의해 제안된 가상머신으로 기존의 JVM(Java Virtual Machine) 기능 중에 일부 데이터 타입 지원과 쓰레딩(Threading) 지원 그리고 쓰레기 수집(Garbage Collection) 지원 기능 등을 제외시킨 JCVM(Java Card Virtual Machine)이다.

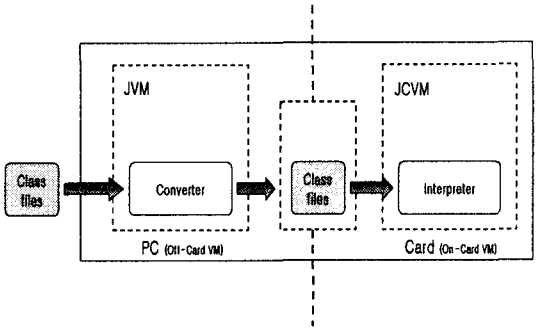

그림 3 : 분리 가상머신 구조

JCVM은 위 [그림 3] 에 나타난 바와 같이 온-카드 가상머신(On-Card VM)과 오프-카드 가상머신(Off-Card VM)으로 구성된 분리 가상머신(Split VM)이며, 온-카드 가상머신에는 바이트코드 수행을 위한 인터프리터가, 오프-카드 가상머신에는 클래스 로딩과 검증 그리고 바이트코드 최적화 및 변환을 위한 컨버터(Converter)가 구현되어 있다. JCVM을 이와 같이 분리 가상머신으로 구현하는 목적은 카드의 제한된 하드웨어 리소스를 효율적으로 활용하기 위함이다.

현재 대부분 스마트카드 솔루션 업체에서 자바카드 및 관련 S/W 패키지를 제공하고 있으며 대표적인 것으로는 SUN Microsystems 의 자바카드 Development Kit을 비롯하여 Gemplus의 GemXpresso, Schlumberger의 Cyperflex, De La Rue의 GalactIC, Bull의 Odyssey등이 있다.

2. 개방형 플랫폼 개요

(1) 개방형 플랫폼 구조

개방형 플랫폼의 구조는 [그림 4]와 같이 카드실행과 직접 관련된 COS(Chip Operating System)와 자바 가상머신(JVM : Java Virtual Machine)을 기반으로 한 Runtime Environment와 카드 매니저(Card Manager)가 카드의 동작을 위한 기반을 이루고 있으며, 이를 토대로 보안 도메인(Security Domain)이 보안에 관련된 기능을 제공한다. 또한 Open Platform API는 각종 서비스를 지원하기 위한 기능적 함수들을 제공하며 이러한 모든 기반을 바탕으로 응용 프로그램이 카드에 적재되어 수행된다. 본 논문에서는 카드의 실행환경이 자바 기반이기 때문에 이러한 응용 프로그램도 자바 언어를 이용하여 구현한다.

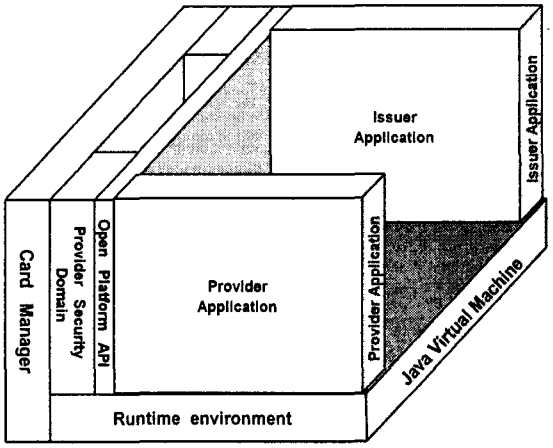

그림 4 : 개방형 플랫폼 구조

(2) 개방형 플랫폼 구성요소

가. 자바 가상머신(Virtual Machine)

자바 명령어 실행을 위한 코드 해석기로 카드의 카드 매니저와 밀접한 연계를 가지고 수행된다. 가상머신은 카드의 COS를 상위 언어인 자바 언어로 접근하기 용이하도록 도와주는 역할을 하는 동시에 카드의 동작을 수행하는 근간을 이룬다.

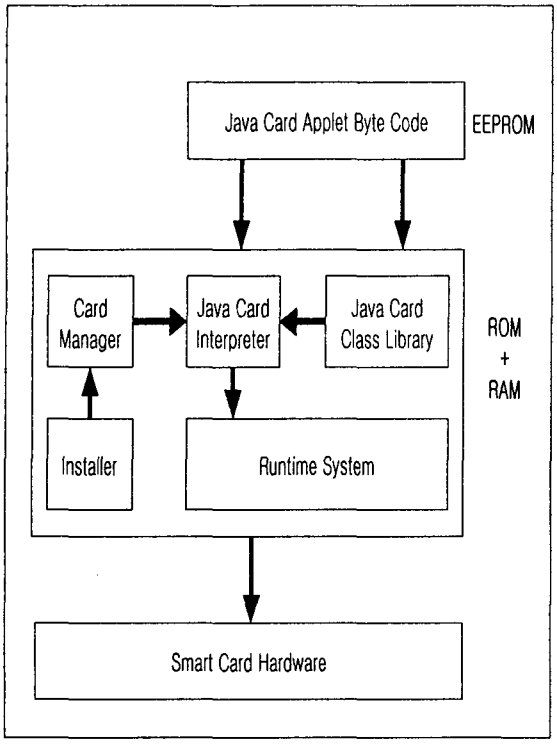

그림 5 : 자바카드 가상머신 구조도

나. 카드 관리자 개요

발행자로 하여금 카드의 동작제어 및 관리를 원활히 하도록 하는 카드상의 프로그램. 카드 발급 후 응용 프로그램이 카드 상에 로드(load) 되는 것을 관리한다.

다. 보안 도메인

각 응용 프로그램들이 사용할 수 있는 암호화 알고리즘과 암호화키를 관리하고 애플릿의 로딩, 인스톨, 삭제를 하는 권한을 가진다.

카드가 발급되는 시점에서 발급 자에 의해 카드 관리자와 더불어 카드 내에 내장되며 애플릿 공급자 가 자신의 애플릿을 위한 별도의 보안 도메인을 만들어 제공할 수도 있다.

라. 개방형 API 및 자바 애플릿

개방형 플랫폼을 이용한 각종 애플릿을 개발하기 위한 Java 라이브러리이고, 전자 화폐 애플릿은 국제 전자 화폐 표준인 CEPS(Common Electronic Purse Specification)에 근거하여 개발된 전자 화폐이다. 신용/직불 애플릿은 국제 전자 금융 표준인 EMV'96(Euro Master Visa '96) 규격 V3.1.1 기술 규격에 근거하여 개발된 금융 카드이다.[4]

Ⅲ. 자바카드 기반 보안 메커니즘 설계 및 구현

1. 자바카드 보안 요구사항

자바카드 기반의 개방형 플랫폼 상에서 요구되는 보안 요구사항을 정리하면 다음과 같다.

㉠ 다수의 애플릿이 하나의 카드 내에서 보안상의 충돌 없이 공존할 수 있도록 지원해야 한다.

㉡ 애플릿간의 상이한 보안 요구사항을 지원해야 한다.

㉢ 응용 프로그램의 요청에 의해 보안 알고리즘 서비스(암호화, 복호화, 디지털 서명 생성 및 확인 등)를 제공해야 한다.

㉣ 키 관리를 지원해야 한다.(발급자의 키와는 별도의 키 생성)

㉤ 카드 관리자는 기본적으로 발급자의 보안 도메인을 내장하고 있어야 한다.

㉥ 카드 발급 후 응용 프로그램의 로딩, 인스톨, 삭제에 필요한 보안절차와 관리기능을 제공해야 한다.

㉦ DAP Verification(응용 프로그램 로딩 시 제공자에 대한 검증과 데이터 검증)을 제공해야 한다.

㉧ 실행할 수 있는 Load 파일의 무결성을 인증해야 한다.

㉨ 보안 위임 관리(응용프로그램 제공자의 실행할 수 있는 Load 파일의 로딩을 위임)를 제공해야 한다.

2. CEPS 보안 메커니즘

(1) 보안 메시징(Secure Messaging)

가. 개요

Command-APDU 통신 메시지 형식은 다음과 같이 분류될 수 있다.

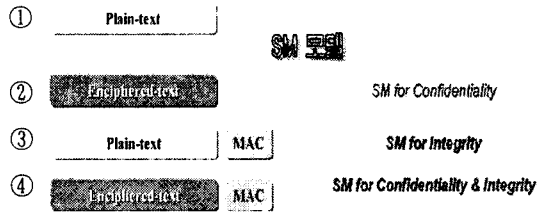

그림 6 : Command-APDU 통신 메시지 형식의 분류

여기서 ①~④가 보안 메시징(Secure Messaging)의 범주에 해당되는데, 카드에서는 ③및 ④를 지원한다.

데이터의 기밀성 보장은 데이터를 암호화(Encipherment)함으로써 달성되고, 송신측의 정당성 보장 및 데이터의 무결성 보장은 통신 메시지에 MAC(Message Authentication Code를 부가함으로써 달성된다.

나. MAC 기반 무결성

MAC 계산 절차는 다음과 같다.

① 명령 헤더 CLA, INS, P1, P2, Lc 및 Data field로 구성된 Data Block을 만든다.

② ▪ Padding rule에 따라 Pad pattern을 부가한다.
  ▪ 마지막 블록의 길이가 1~7 bytes이면 80, 80 00, 80 00…00 등의 Pad Pattern을 부가한다.

③ 16 bytes Session Key로 Triple DES 알고리듬을 이용해서 CBC 모드로 데이터 블록을 암호화한 다음 그 결과의 LSB (하위 4 Bytes)를 취한다.

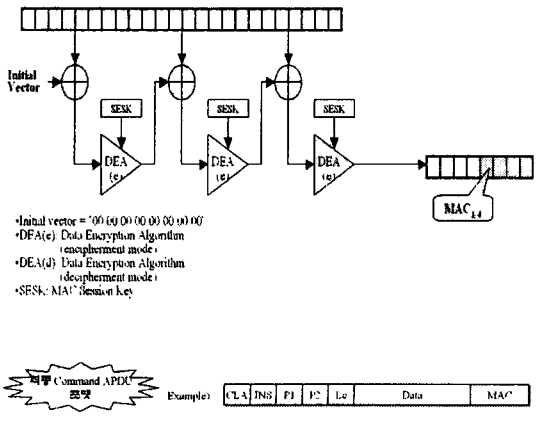

그림 7 : MAC 생성 방법

다. 데이터 암호화

카드 데이터의 암호화 절차는 다음과 같다.

① 평문(Plaintext) 데이터 블록을 만든다.

② Padding rule에 따라 Pad pattern을 부가한다.
  ▪ 메시지를 8 바이트 블록으로 나눈다.
  ▪ 마지막 블록의 길이가 1~7 바이트이면 80, 80 00, 80 00 … 00 등의 Pad Pattern을 부가한다.

③ 16 바이트 Session Key로 3-DES 알고리듬을 이용해서 ECB 모드로 데이터 블록을 암호화한다.

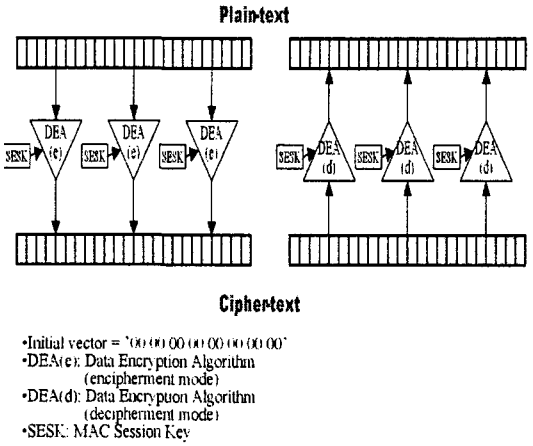

그림 8 : 데이터 암호화 과정

(2) 동적 서명 검증(Dynamic Signature Verification)

동적 서명 검증은 POS 장비와 CEPS 카드간 구매 트랜잭션 시에 사용되며 POS 장비가 전자서명(Digital signature)을 생성하고 CEPS 카드가 이를 인증하는 것을 주요 내용으로 한다. 이때 사용되는 서명을 PS2라고 한다.

동적 서명 검증의 특징은 다음과 같다.

- PSAM과 CEPS 카드 사이에 수행된다.
- 디지털서명이 사용되며, 이것은 CEPS 카드의 공개키로 암호화되어 전달된다.
- CA(Certification Authority)의 존재를 필요로 한다.
- PSAM 및 CEPS 카드는 CA의 공개키(PK_{CA})를 갖고 있어야 한다.
- PSAM은 PSAM 비밀키(SK_{PSAM})를 갖고 있

어야 한다.

- CEPS 카드는 ICC 비밀키(SK_{ICC})를 갖고 있어야 한다.
- RSA Signing/Recovery Function 및 SHA-1 Hash 알고리듬이 사용된다.

동적 서명 검증의 개요는 다음 그림과 같다.

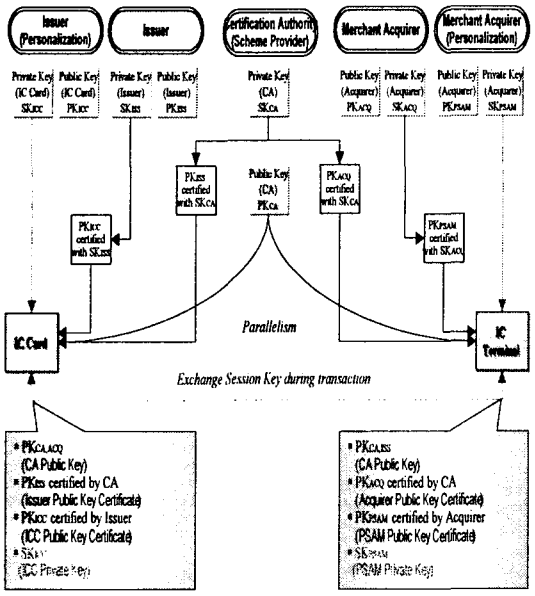

그림 9 : 동적 서명 검증

동적 서명 검증의 수행 및 인증 절차는 다음과 같다.

- PSAM : PS2 생성 후 Debit for Purchase명령을 통해 CEPS 카드로 보낸다.
  - PSAM 비밀키를 이용하여 DS(Digital Signature)를 생성한다.
  - CEPS카드의 공개키를 이용하여 PS2를 생성한다.
- CEPS카드 : PSAM으로부터 받은 PS2를 인증하고 데이터를 복원한다
  - CEPS 카드의 비밀키를 이용하여 PS2를 복호화 한다.
  - PSAM의 공개키를 이용하여 DS를 인증하고 데이터를 복원한다.
  - 16bytes DES Key($SESSKey_{PSAM}$)를 저장한다.

3. 보안 메커니즘 설계 및 구현

(1) 개발환경

보안 메커니즘을 구현하기 위한 개발환경은 VisualCafe 4.0과 GemXpresso 2.4PK버전을 사용한다.

VisualCafe 4.0은 개발을 단순화 해주고, 검증, 디버깅기능 등을 제공한다. 또한 GemXpresso의 플러그 인을 실행하기 위한 환경이다.

GemXpresso 개발 킷은 개발자가 효율적으로 소스코드를 디자인하고 쉽게 템플릿과 프로젝트를 사용할 수 있게 해 주며, 개발자를 위해 다양한 개발 환경을 제공하기 위해 카드의 제한된 자원을 극복 할 수 있도록 GSE(GemXpresso Simulation Environment)를 제공한다. GemXpresso 2.4 PK는 개발된 애플릿의 기능적인 테스트를 위해 디버깅 기능을 갖추고 있다. 디버깅은 Symantec Visual Cafe 내에서 플러그인 형태로 GemXpresso를 실행하거나, JDK1.2와 호환되는 개발환경에서 가능하다.

설치 과정은 먼저 VisualCafe 4. 0 을 설치하고, GemXpresso 2.4 PK버전을 설치한다. 자동으로 VisualCafe를 찾아서 Plug-in이 설치된다.

GempXpresso 2.4 Disk에서 V2 path 디렉토리를, 설치된 디렉토리로 복사한 후(custom과 lib폴더를 C:\Gemplus\GemXpresso.rad2 하위디렉토리로 복사), C:\Gemplus\GemXpresso.rad2\custom\ targets\ card.properties.v2 파일을 card.properties파일로 대체한다.

(2) 자바카드상의 wallet SEED의 기능 구성

가. 클래스 구성

자바카드 상의 wallet SEED의 클래스 구성은 다음과 같다.

- Wallet : 자바카드와 관련된 main부분으로 SEEDCipher를 호출한다.
- SEED : SEEDCipher부분과 결합된 실질적인 Encryption하는 부분
- SEEDCipher : CBC/ECB/Padding을 선택하고 선택한 알고리즘을 통한 실제적인 Encryption 하는 부분
- BigIntNumber : 바이트 연산 처리하는 클래스 (+, - , compare)
- SEEDKeyGenerator : SEEDKey를 만든다.

(Random Number로 채움)

- SEEDSecretKey : SEED 비밀키에 대한 정보를 가지고 있음.
- SecreteKey : javacard.security의 SecretKey class를 오버로딩한 Class
- SecureRandom : Random Number를 생성하는 Class

나. 주요 내용

wallet SEED를 구성하는 클래스들의 주요 기능들은 다음과 같다.

- SEEDCipher 객체 생성

```
SEEDCipher cipher = new SEEDCipher();
```

- Algorithm Instance생성

```
//ALG_SEED_ECB_NOPadding,ALG_SEED_
  ECB_Zeroes,
//ALG_SEED_ECB_PKCS5Padding,ALG_ SEED_
  CBC_NOPadding,
//ALG_SEED_CBC_Zeroes,ALG_SEED_CBC
  _PKCS5Padding
  cipher.SetAlgorithm(cipher.ALG_SEED_
  ECB_PKCS5Padding);
  System.out.println("Generating the random
  number...");
  SecureRandom rand = new SecureRandom();
```

- Key Initialize : randomize

```
System.out.println("Initializing the KeyGener
ator ...");
SEEDKeyGenerator kg = new SEEDKey
Generator();
kg.Init((short)256,rand);
```

- SecreteKey Generate

```
System.out.println("Generating SecretKey...");
SEEDSecretKey key = kg.GenerateKey();
```

- Input message

```
System.out.print("\nEnter the message to
encrypt : ");
byte plainText[] = in.readLine().getBytes();
    cipher.init(cipher.ENCRYPT_MODE,key);
    debug("The inputed Message(hexadecimal):
    ",plainText);
    System.out.println("The inputed Message
    (string): " + new String(plainText));
```

- Encrypt the message

```
byte[] encryptText = cipher.doFinal(plainText);
debug("The Encrypted message: ", encryptText);
```

- initialize the Cipher in decryption mode

```
cipher.init(cipher.DECRYPT_MODE,key);
plainText = cipher.doFinal(encryptText);
debug("The Decrypted message: ", plainText);
```

ECB/ CBC가 구현되여 있고, 틀린점은 초기벡터를 설정하는 부분만 다르다.

(3) 자바카드상의 HMACwithHAS160 / HMACwithMD5 / HMACwithSHA1 기능 구성

가. 클래스 구성

자바카드상의 HMACwithHAS160, HMACwithMD5, HMACwithSHA1의 클래스 구성은 다음과 같다.

- HMACSecretKey : MAC 비밀키에 대한 정보를 가지고 있다.
- HMACwithHAS160 : 실질적으로 HAS160과 결합된 Main Class
- HMACwithMD5 : 실질적으로 MD5과 결합된 Main Class
- HMACwithSHA1 : 실질적으로 SHA1과 결합된 Main Class
- HAS160 : HAS160 알고리즘
- MD5 : MD5 알고리즘
- SHA1 : SHA1 알고리즘

나. 주요부분

HAS160, MD5, SHA1을 구성하는 클래스들의 주요 기능들은 다음과 같다.

- HAS160, MD5, SHA1

```
HMACwithHAS160 mac = new HMACwithHAS160();
// HMACwithMD5 mac = new HMACwith
  MD5 ();
```

```
// HMACwithSHA1 mac = new HMACwith
   SHA1 ();

System.out.print("\nEnter HMAC Key(variable
bytes:default 128bits) : ");

byte[] hmacKeyData = in.readLine().getBytes();

HMACSecretKey hmacKey = new HMACSec
retKey();

hmacKey.setKey(hmacKeyData,(short)0);

mac.init(hmacKey);

System.out.print("\nEnter the message to
keyed-hash : ");

byte plainText[] = in.readLine().getBytes();

debug("The inputed Message(hexadecimal):
",plainText);

System.out.println("The inputed Message
(string): " + new String(plainText));

mac.update(plainText);

byte[] hashText = mac.doFinal();

debug("The Hashed message code: ",
hashText); [6][7][8][9]
```

(4) 자바카드상의 wallet SEED 구현 화면

㉠ 자바카드에 맞게 작성된 SEED소스를 컴파일 한다.

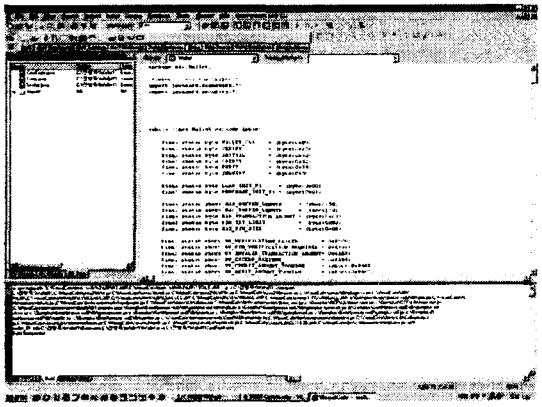

그림 10 : 소스 컴파일

㉡ VisualCafe에서 Define GemXpresso Project를 실행한다. Define Project : AID, PID 설정 (PID 5byte와 AID 5byte는 일치해야 한다)

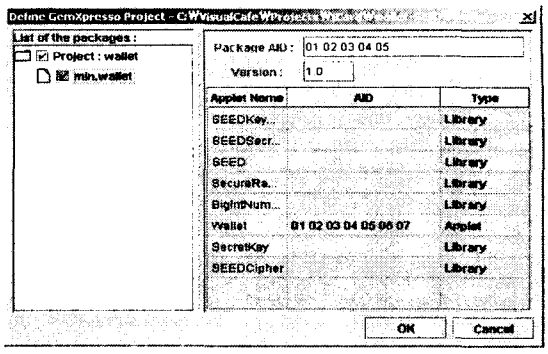

그림 11 : Define Project

㉢ VisualCafe에서 Verify를 선택한다.

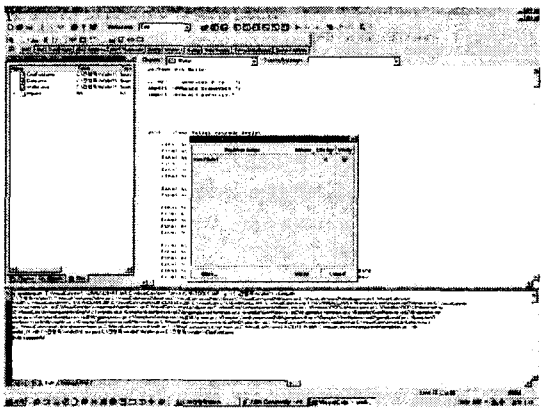

그림 12 : Verify

㉣ VisualCafe에서 Convert를 선택한다.

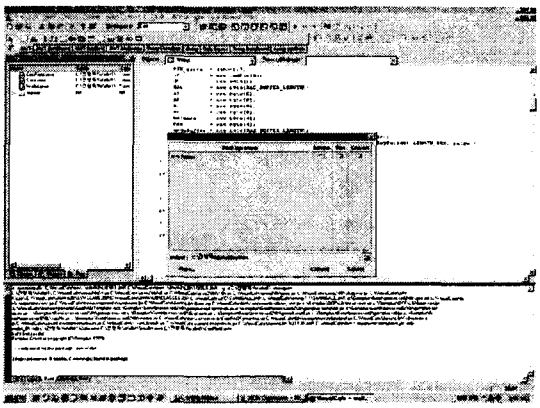

그림 13 : Convert

㉤ 자바카드매니저를 실행을 한다.

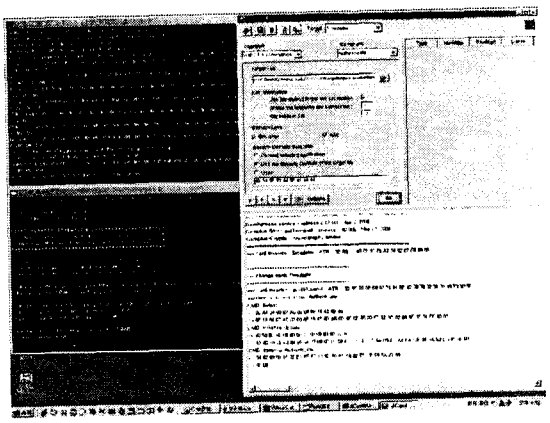

그림 14 : 카드 매니저

㉥ TargetFile설정 GemXpresso.rad2\custom\targets\card.properties를 지정하고 Authenticate를 실행한다.

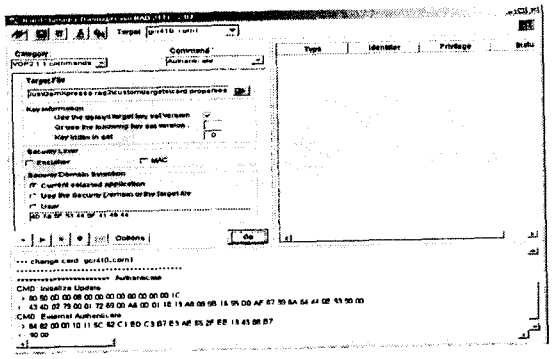

그림 15 : Authenticate

㉦ Command 에서 Upload file into a card를 선택하고, Target File에 wallet-SEED디렉토리 안에 있는 \GxpRad\min\wallet\javacard\wallet.jar 파일을 선택하고, VisualCafe에서 설정한 Package AID를 설정한 후 Go를 누른다. (위의 디렉토리에 wallet.jar파일이 없다면 Compile, Verify, Converting 과정이 잘못된 것임)

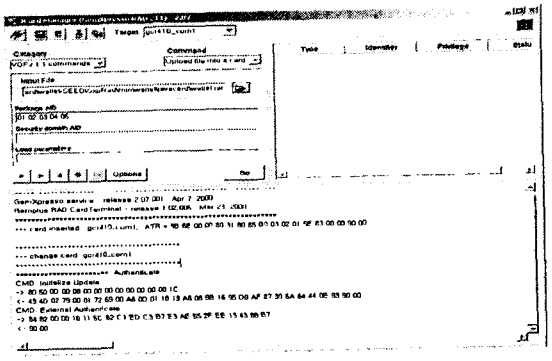

그림 16 : Upload file into the card

㉧ Command에서 Install를 선택하고, PID, AID를 설정한후.. Go를 누르면 다음그림과 같이 package 01 02 03 04 05와 applet 01 02 03 04 05 06 07이 생긴다.

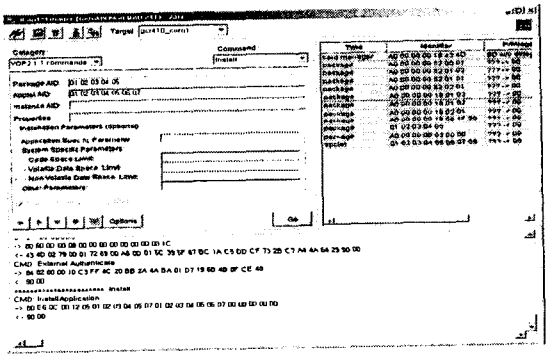

그림 17 : Install

㉨ Command에서 Select한후 AID를 선택하면 Upload된 applet이 선택된다.

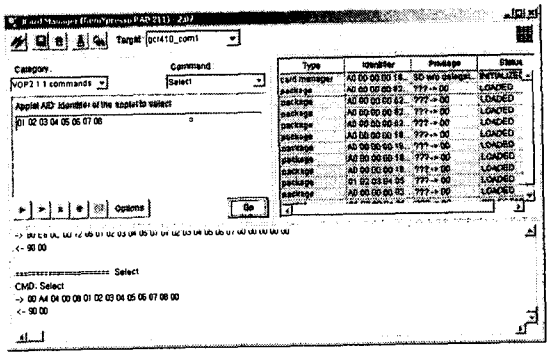

그림 18 : Select

㉩ Command에서 Send APDU를 선택한후.. 그림과 같이 CLA, INS를 선택한후 APDU 통신을

한다. 질의에 대한 응답 메시지가 return된다.

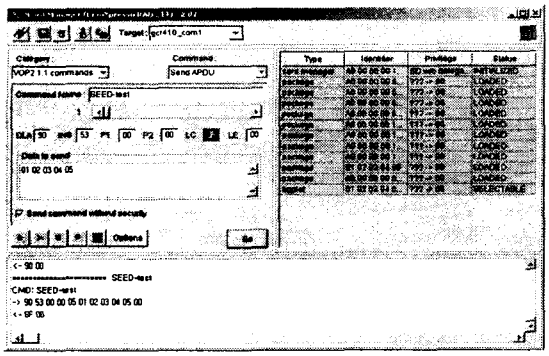

그림 19 : Send APDU

Ⅳ. 결론

본 논문에서는 자바카드 기반의 개방형 플랫폼 보안 요구사항과 자바카드를 기반으로 하는 개방형 플랫폼상의 CEPS 보안 메커니즘을 분석하고, 개방형 플랫폼 구성요소인 보안 도메인(Security Domain)의 보안 요구사항에 근거해서 자바카드 보안 서비스를 제공하기 위한 기능의 설계 및 구현에 대해 언급하였다. 이러한 설계와 구현 방안의 제시는 카드 발행자의 다양한 요구사항을 만족시키는 전자화폐 시스템을 구축하는데 활용되어질 수 있고, 플랫폼 및 하드웨어 공급업체와 상관없는 이러한 접근 방식은 시스템 운영업자의 인프라 투자를 현실화시킬 수 있다. 개방형 플랫폼 아키텍처는 발급이후 애플릿서버에 저장된 새로운 응용프로그램의 구현뿐만 아니라 응용프로그램을 갱신 또는 제거할 수 있는 새로운 기능을 제시할 수 있다.

향후 과제로서는 본 논문에서 구현한 보안 요구사항에 근거한 보안 기능의 평가를 들 수 있다.

참고문헌

[1] 하영국, 임신영, 함호상, "Card Applet 서명을 지원하는 Java Card 응용라이브러리," 한국통신정보보호학회 종합학술발표회 논문집 Vol.10, No.1, p. 296, 2001.

[2] 김영선, 이창욱, "자바카드 애플릿 설계 및 검증에 관한 연구," 한국통신정보보호학회 종합학술발표회 논문집 Vol.10, No.1, pp. 806-807, 2001.

[3] 하영국, "안전한 전자상거래를 위한 JavaCard Toolkit의 설계 및 구현," 한국정보처리학회 추계 학술발표 논문집 7권 2호, p773, 2000.

[4] (주)마니네트웍, "개방형 플랫폼 개발 중간보고서," 2장, 2000, 8.

[5] 백상수, 송상헌, 류재철, "CEPS 보안 구조분석,"

[6] Patrice Peyret, "JavaCard Technology for Smart Cards Architecture and Programmer's Guide," Apri 2000.

[7] Javaland, "jSec with javaland User's Reference," 2000, 9.

[8] Sun microsystems, "Java Card™ 2.1.1 Application Programming Interface," May 2000.

[9] Gemplus, "GemXpresso 2.4 PK User Guide, Getting Started," October 1999.