자바카드 애플리케이션 활용을 위한 APDU 자동 분석기 구현

조증보°,김도우, 정민수
경남대학교 컴퓨터공학과

Implementation of APDU Auto-Analyzer for using applications on Java Card

Jeung-Bo Cho, Do-Woo Kim, Min-Soo Jung
Dept. of Computer Engineering, KyungNam Univ.
E-mail :godofworld@hanmail.net

요 약

전자 화폐에 대한 관심이 많이 증가한 요즘, 스마트 카드 기반의 자바 카드에 대한 연구가 활발히 진행되고 있다. 이에, 자바 카드 플랫폼으로 수행 가능한 다양한 애플리케이션을 실제로 생성해서 실행 가능한 파일로 변환하는 과정과 사용되는 파일들의 분석을 본 논문에서 중점으로 다룬다. 그리고 생성된 파일의 구조을 자동으로 분석해내는 APDU 자동 분석기에 대한 내용도 다룬다.

1. 서론

자바 카드 가상 기계(Java Card Virtual Machine) 기반의 스마트 카드가 현재 교통, 통신, 신분 확인, 전자 화폐 등의 여러 응용 분야에 활용되고 있다. 이는 머지 않아 구현될 One-Card System 을 예견하고 있다. 이에, 스마트 카드 중에서도 높은 점유율을 보이는 자바 카드 가상 기계에 대해 알아보기로 한다. 스마트 카드의 경우 최초 출시 되었던 1970년대 후반의 1 bit 체제에서 현재 32 bit 까지 많은 하드웨어적인 발전을 이뤘다. 뿐만 아니라, 최초 1회용 공중전화카드의 목적에서 현재의 다양한 애플리케이션 및 통신, 네트워크까지 분야까지의 응용에서 보듯이 기능 또한 무한한 발전을 이루어 왔다.

하지만 소형 단말기에 대한 인식의 부족과 구현의 어려움으로 애플리케이션 개발이 부족한 것이 현실이다. 이에 하나의 애플리케이션을 생성해서 자바 카드 플랫폼으로 변환하여 실행 가능하도록 하는 과정을 서술하며, 애플릿 실행 테스트 보조 도구로 활용될 수 있는 APDU 자동 분석기를 제시한다.

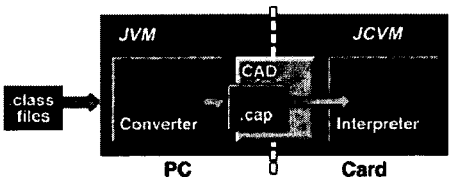
본 논문의 구성은 다음과 같다. 2장은 자바 카드에 대한 전반적인 이해 및 자바 카드 기술에 대해 언급하고, 3장에서는 애플릿을 실제로 개발할 때 있어서 어떤 절차로 만들어야 하는지를 설명하며, 4장에서는 실제 생성되는 APDU 파일의 포맷에 대한 연구와 분석을 다루며, 5장에는 이를 자동으로 분석해주는

본 연구는 학국과학재단 산학협력연구(2001-30300-002-1) 지원으로 수행되었음

APDU 자동 분석기에 대해 언급한다. 마지막으로 6 장에서는 결론과 향후 연구방향을 제시하며 본 논문을 마무리한다.

2. 관련연구

2.1 자바 카드의 구조



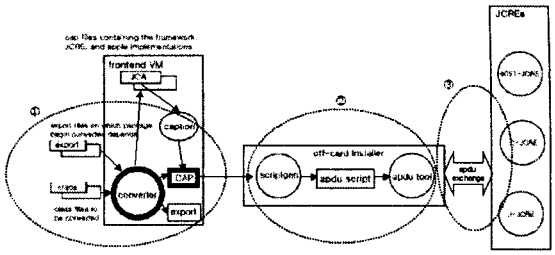
[그림 1] 자바 카드 가상 기계의 구조

위의 그림에서 보는 바와 같이 자바 카드 가상 기계는 크게 Off-Card 부와 On-Card 부로 구성되어 있고, 그에 해당하는 기능은 다음과 같다.

- Off-Card 가상 기계
  - 클래스 로딩(Class Loading), 링킹(Linking) (메모리 할당(Preparation), 해석(resolution))
  - Off-Card 검증(Verification)
  - 컨버터(Converter)를 이용한 CAP(Converted Applet) File, EXP(Export) File, JCA(Java Card Assembly) File 생성.
  - 그 외 많은 자원을 요구하는 업무 처리

- On-Card 가상 기계
  - 바이트 코드 수행(Excute)
  - 실행시(runtime) 메모리에 대한 전반적인 관리

하지만 [그림 2]에서 보듯이 Off-Card 부에서 생성된 실제 실행파일인 CAP 파일은 복잡한 변환 과정을 통해서 APDU 라는 패킷을 이용해서 On-Card 상으로 다운로딩(Downloading)된다.



[그림 2] Java Card 내의 CAP 파일의 흐름

2.2 APDU 관련 파일들에 대한 정의

2.2.1 APDU

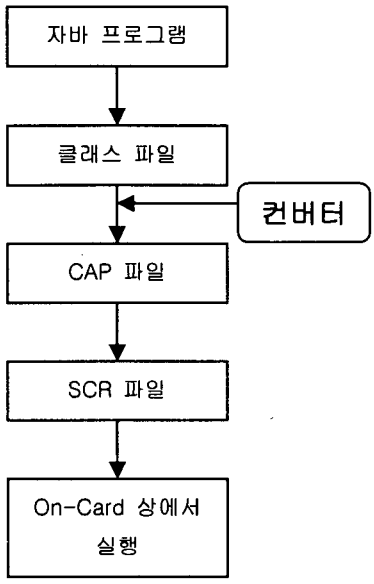
APDU 란 Application Protocol Data Unit 의 약어로 Java Card 기반의 스마트 카드에서 사용되는 메시지 구조이다.

2.2.2 SCR

SCR 파일이란 Script 의 약어로, Sun 홈페이지에 있는 자바 카드 개발 도구(Java Card Development Kit)에 있는 명령어에 의해서도 생성되는데, APDU 들로 이루어진 명령어의 집합이다.

3. 애플릿 개발 순서

애플릿 개발 순서를 도식화 하면 다음과 같다.



3.1 컨버터를 이용한 변환

컨버터는 카드리더기가 연결된 단말기 측에 탑재되어 클래스 파일을 입력으로 받아 On-Card 상에서 실행될 바이너리 파일 형태인 CAP 파일과 EXP 파일, 그리고 이에 해당하는 내용을 토큰 형식으로 표현하고 있다가, ROM 에 마스킹(Masking) 될 때 사용되는 JCA 파일로 변환시키는 모듈이다.

3.2 APDU 생성 및 포맷 분석

자바 카드에서 사용되는 APDU 는 크게 두가지로 나누는데, 명령(command) APDU 와 응답(response) APDU 가 그것이다. 구조는 다음과 같다.

[표3-1] APDU 형식

Command APDU						
Mandontary Header				Conditional Header		
CLA	INS	P1	P2	Lc	DATA	Le

Response APDU		
Mandontary Header	Conditional Header	
DATA Field	SW1	SW2

APDU 포맷에서 CLA 는 해당 클래스를 나타내고, INS 경우 그 클래스내의 메소드(명령어)를 나타내며, P1, P2 는 매개변수, Lc 는 데이터의 길이, Le 는 응답 Data 의 길이를 나타낸다.

4. 실제 애플릿 개발 예

자바 카드 개발 도구에서 제공하는 예제와 유사한 기능을 하는 TrafficCard 라는 예를 만들어 보았다.

4.1 자바 파일 및 클래스 파일 생성

자바 파일에서 클래스 파일로 변환 시(Compile) 자바 카드에서 제공하는 API 들을 패스에 추가시켜 줘야 한다. 그리고 자바 카드에서 구동되는 애플릿의 경우 프레임워크(framework) 패키지내의 애플릿이라는 클래스를 확장해서 사용한다.

4.2 CAP, EXP, JCA 파일로의 변환

컨버터라는 명령어가 자바 카드 개발 도구에서 제공되므로 이를 이용해서 쉽게 변환(convert)을 할 수 있다. 명령어 형식은 다음과 같다.

```
Converter [options] <package_name>
        <package_aid> <package_version>
```

<package_name>

변환될 패키지의 완전한 이름(fully-qualified name)

<package_aid>

콜론(:)에 의해 분리되는 5~16 바이트 크기의 8,10,16진수

<major_version> <minor_version>

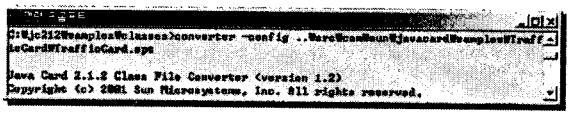
패키지의 사용자 정의 버전

다음은 실제 패키지 이름, AID, 버전의 예다.

```
com.sun.javacard.samples.HelloWorld
0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x1 1.0
```

```
com.sun.javacard.samples.JavaPurse
0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x2 1.0
```

다음과 같이 □config 옵션을 주고, *.opt 파일을 추가함으로써 간단히 변환 할 수도 있다.



[그림 4-1] 성공적인 변환 모습

```
-out EXP JCA CAP
-exportpath .
-applet 0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x7:0x1
com.sun.javacard.samples.TrafficCard.TrafficCard
com.sun.javacard.samples.TrafficCard
0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x7 1.0
```

[그림4-2] *.opt 파일의 내용

자바 카드 개발 도구에서 제공하는 scriptgen 이라는 명령어를 사용하면, CAP 파일을 APDU 포맷의 열거 형태인 SCR 파일로 쉽게 변환 할 수 있다. 하지

만, 이 SCR 파일은 변형 단계만 거쳤을 뿐 실제 실행 가능한 파일은 아니다. 헤더(Header) 부분과 풋터(Footer) 그리고 실제 실행하는 순서는 빠져있기 때문이다.

4.3.1 실행 가능한 스크립트 파일(SCR) 만들기

Powerup // 헤더 부분
인스톨러 애플릿 선택 // 자바 카드 API 에서 제공하는 다른 애플릿을 인스톨 시키는 인스톨러 애플릿의 활성화
SCR 파일 // 위에서 변형한 SCR 파일
실행 명령어들 // 위에서 변형한 CAP 파일 내에 존재하는 메소드들을 매개변수를 이용하여 호출
Powerdown // 풋터 부분

[그림4-3] 실행가능 한 SCR 파일 구조

```
powerup;

// Select the installer applet
0x00 0xA4 0x04 0x00 0x09 0xa0 0x00 0x00 0x00 0x62 0x03 0x01
// 90 00 = SW_NO_ERROR

0x80 0xB0 0x00 0x00 0x00 0x7F;

// com/sun/javacard/samples/TrafficCard/javacard/Header.cap
0x80 0xB2 0x01 0x00 0x00 0x7F;
0x80 0xB4 0x01 0x00 0x16 0x01 0x00 0x13 0xDE 0xCA 0xFF 0xED
0x80 0xBC 0x01 0x00 0x00 0x7F;

                    :

0x80 0xBA 0x00 0x00 0x00 0x7F;

// create TrafficCard
0x80 0xB8 0 0 0xOc 0x0a 0xa0 0 0 0 0x62 0x03 0x01 0xOc

//Select Wallet
0x00 0xA4 0x04 0x00 0x0a 0xa0 0x0 0x0 0x0 0x62 0x3 0x1

//Get wallet balance
0xa0 0x50 0x00 0x00 0x00 0x02;


powerdown;
```

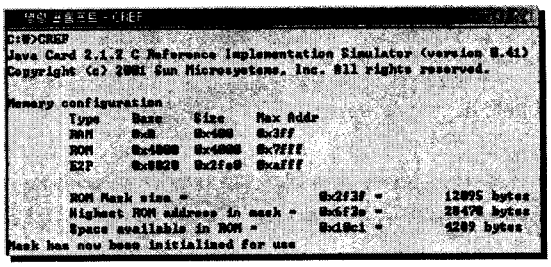
[그림4-4] 실제 SCR 파일의 예

4.4 SCR 파일을 이용한 APDU 통신

완성된 SCR 을 이용해서 실제 자바 카드와 카드리더기 간의 통신과 같은 APDU 통신을 PC 상으로 실행하기 위해서는 자바 카드 개발 도구에서 제공하는 APDUTool 이라는 명령어가 필요하다.

① 두 개의 명령 프롬프트 실행

② 첫 번째 명령 프롬프트에는 APDU 패킷을 받기 위한 대기 상태를 위해 CREF (자바 카드 개발 도구 제공) 명령어 실행



[그림4-5] APDU 패킷을 받기 위한 대기상태 화면

[그림4-6] APDU 패킷을 전송하는 모습

위의 그림은 APDUTool 이라는 명령어를 이용하여 사전에 완전한 형식으로 갖추어진 TrafficCard.scr 이라는 파일을 입력으로 APDU 통신을 하는 모습이다. 전송이 완료되면, 대기 상태에 있는 명령 프롬프트는 "C-JCRE was powerdown" 이라는 메시지와 함께 종료되며, TrafficCard.out 이라는 파일이 생긴다. 이 파일에는 APDU 형식에 맞추어진 여러 APDU 들이 나열되어 있으며, 전송에 있어서 성공, 실패 여부를 나타내는 SW(Status Word)도 함께 포함하고 있다. 다음 그림이 TrafficCard.out 파일의 내용이다.

```
Java Card ApduTool (version 0.15)
Copyright (c) 2001 Sun Microsystems, Inc. All rights reserved.
Opening connection to localhost on port 9025.
Connected.
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01,
CLA: 80, INS: b0, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b2, P1: 01, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
                                  :

CLA: 80, INS: b4, P1: 05, P2: 00, Lc: 20, 05, 00, 42, 00, 10, 02, 00,
CLA: 80, INS: b4, P1: 05, P2: 00, Lc: 20, 01, 06, 00, 00, fd, 06, 00,
CLA: 80, INS: b4, P1: 05, P2: 00, Lc: 05, 06, 03, 80, 0a, 04, Le: 00,
CLA: 80, INS: bc, P1: 05, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b2, P1: 09, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 09, P2: 00, Lc: 20, 09, 00, 25, 00, 07, 99, 11,
CLA: 80, INS: b4, P1: 09, P2: 00, Lc: 08, 12, 15, 0d, 0f, 05, 06, 07,
CLA: 80, INS: bc, P1: 09, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: ba, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 05, a0, 00, 00, 00, 62, Le: 00,
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0a, a0, 00, 00, 00, 62, 03, 01,
```

[그림4-7] TrafficCard.out 의 내용

4.5 APDU 분석

4.5.1 명령 APDU 분석

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xA4	0x04	0x00	0x09	↓	0x7F

0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01

[그림 4-7] APDU 분석

CLA 0x00 : Applet 클래스

INS 0xA4 : Select 메소드

P1 P2 : 매개변수들 (0x04 0x00)

Lc 0x09 : Data 의 길이 9 바이트

Data 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 : Installer Applet 의 AID

4.5.2 응답 APDU 분석

CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le: 00, SW1: 90, SW2: 00

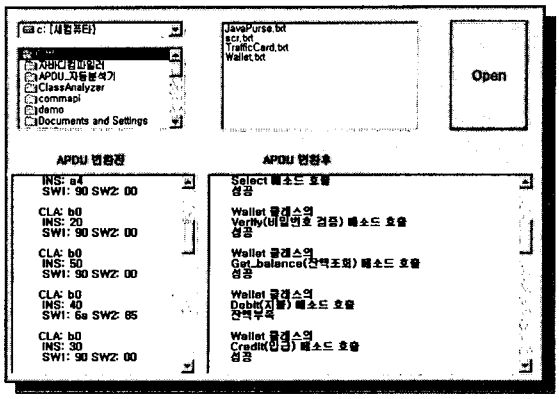
CLA~Data 부분은 명령 APDU 와 동일.

Le 0x00 : 응답 Data 의 길이

SW1: 90, SW2: 00 : 미리 정의되어있는 상태워드로 0x90 0x00 은 에러가 발생하지 않았다는 의미

5. APDU 자동 분석기의 구현

APDU 자동 분석기는 APDUTool 을 이용한 APDU 통신에서 결과물로 생성되는 *.out 파일을 자동으로 분석하는 모듈이다. 그 실행의 결과물은 CLA, INS 등의 스트링과 16 진수로 구성되는데, 해당 입력 파일인 scr 파일의 영향을 많이 받는다. 본 논문에서 구현한 자동 분석기의 경우, 자바 카드 개발 도구에서 제공하는 예제들을 자동으로 분석하는 프로그램으로, JavaPurse, HelloWorld, Wallet 그리고 직접 작성한 애플릿인 TrafficCard 등을 자동으로 분석한다. 하지만 APDU 자동 분석기의 경우 애플릿이 추가 될 때 마다, 해당 클래스와 메소드들을 추가해 줘야 하는 번거로움이 있다.



[그림5-1] APDU 자동 분석기

APDU 자동 분석기의 구성은 다음과 같다.

① 파일을 읽어 들일 수 있는 파일 입력부

② 입력한 파일의 변환 전 모습을 보여주는 APDU 변환 전

③ 변환을 한 후의 모습을 보여주는 APDU 변환 후

아직 여러 예외 상황이나 입력 파일의 오류를 검증 해내지는 못하는 단계이지만, 개발자가 만든 애플릿을 SCR 파일로 변환하고 그 파일의 진행순서를 분석하는데 충분한 자료를 제공한다. 본 프로그램은 Microsoft Visual Basic 6.0 으로 작성되었으며, MS Window 2000 Professional 에서 구동된 프로그램이다.

6. 결론

본 논문에서는 자바 카드 플랫폼 상에서 카드와 카드 리더기(CAD) 간의 통신 시 이용되는 APDU 패킷에 대해 자세한 분석과 이를 자동으로 분석해내는 분석기에 대해 다루었다. 개발자들이 직접 작성한 애플릿이 실제 자바 카드 플랫폼상에서 어떤 오류를 발생시키는지, 아니면 성공적으로 구동 되었는지 등을 사용자에게 익숙한 언어로 표현하여 이를 쉽게 인지하고 보다 빠르고 다양한 애플리케이션을 개발할 수 있게 돕는다. 뿐만 아니라 자바 카드에 대한 전반적인 이해와 실제 애플릿을 개발하는 단계를 상세히 묘사함으로써 처음으로 자바 카드를 접하는 개

발자들에게도 많은 도움을 제공한다.

향후 연구 계획으로는 APDU 패킷의 최적화와 자바 카드에 RMI 를 접목해서 보다 간단하고 빠른 통신에 대한 연구 하며, 실행 시 On-Card 상에 검증 단계를 추가함으로써 애플릿의 보안을 강화할 계획이다.

[참고문헌]

[1] B. Venners, *Inside Java Virtual Machine*, McGraw-Hill, (1997)

[2] Zhiqun Chen, *Java Card™ Technology for Smart Cards : Architecture and Programming's Guide*, ADDISON-WESLEY, 2000

[3] 한국전자통신 연구원, *스마트 카드 기술 시장 보고서*, 1999,12.

[4] Sun Microsystems, Inc., *The Java Card™. 2.1.2 Development Kit User'sGuide*, SUN 2002.2

[5] Sun Microsystems, Inc., *The Java Card™ 2..2 Virtual Machine Specification*, SUN 2002.2

[6] Sun Microsystems, Inc. *The Java Card™ 2.2 Application Programming Interface*, SUN 2002.2

[7] Sun Microsystems, Inc. *The Java Card™. 2.2 Runtime Environment (JCRE) Specification*, SUN 2002.2

[8] 황욱철, *스마트카드 기반 자바카드 가상기계의 최적화 연구 (2002)*

[9] http://java.sun.com/, Sun Microsystems, Java Home Page