USIM 기반 자바카드용 CAP 파일 분석기 설계 및 구현

이신혜°, 정민수, 권오형,
경남대학교 컴퓨터공학과

Design and Implementation of CAP File Analyzer for Java Card based on USIM

Shin-Hye Lee, Min-Soo Jung
Dept. of Computer Engineering, KyungNam Univ.
E-mail :shi-ny@hanmail.net

요 약

USIM 기반 자바카드가 요즘 많은 응용분야에서 활용되고 있다. 이에 자바카드에 대한 관심과 연구가 활발히 진행되는 가운데 본 논문에서는 자바카드 플랫폼에서 수행되는 애플릿 개발 과정 중 중간 코드인 CAP 파일의 구조에 대해 중점적으로 다루며 CAP 파일을 구성하고 있는 컴포넌트를 개별적으로 분석해주는 CAP 파일 분석기를 설계하고 구현했다. CAP 파일을 구체적으로 분석 함으로서 CAP 파일에 대한 전반적인 이해를 돕는다.

1. 서론

현재 자바카드 가상 머신 기반의 스마트 카드가 교통, 통신, 신분 확인, 전자 화폐 등의 여러 응용분야에 활용되고 있다. 스마트 카드는 계속적인 하드웨어의 발전으로 CPU는 기존의 8bit 위주에서 보다 처리 성능이 뛰어난 32 bit로 전환되었고, 데이터의 암.복호화를 빠르게 하기 위해 특정 암호 기법에 대한 전용보조프로세서의 사용이 늘어나고 있는 추세이다. 또한 현재에는 다양한 애플리케이션 및 통신, 네트워크까지의 응용에서 큰 발전을 보이고 있다.

하지만 소형단말기에 대한 인식이 여전히 부족하고 애플리케이션 개발자들에 대한 개발 도구가 부족한 것이 현실이다. 이에 자바카드에 대한 이해를 바탕으로 자바카드 플랫폼에서 스마트 카드의 적재와 카드 상에서 메모리 적재 및 링킹 등 운영의 편의를 위해 제공되어 지는 CAP파일을 소개한다. 그리고 CAP 파일을 쉽게 분석해볼 수 있도록 구현되어진 CAP 파일 분석기를 제시한다.

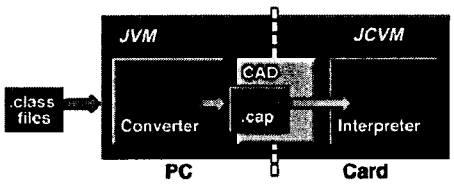
본 논문의 구성은 다음과 같다. 2장은 관련 연구분야로 자바 카드에 대한 전반적인 부분에 대해서 설명하며, 3장은 CAP 파일 분석기를 어떻게 설계하였

는지에 대해 설명한다. 4장에서는 CAP 파일 분석기가 어떻게 구현되었는지 예를 들어 살펴보고, 마지막으로 5장에서 결론을 내리며 본 논문을 마무리한다.

2. 관련연구

2.1 자바카드의 전반적인 이해

자바카드의 바이트 코드를 수행하는 수행 엔진인 JCVM은 스마트 카드에 적합하도록 최적화되어 애플릿간의 방화벽, 최적화된 명령어 등 subset의 개념이라기 보다는 지원 내용은 작지만 별도의 새로운 수행환경을 구성한다. JCVM은 Off-Card VM과 On-Card VM으로 나뉘는 분할 가상기계로 이루어진다. JCVM의 구성은 아래 [그림2-1]과 같고 각 기능을 간략히 기술하면 다음과 같다. [4][5][6]

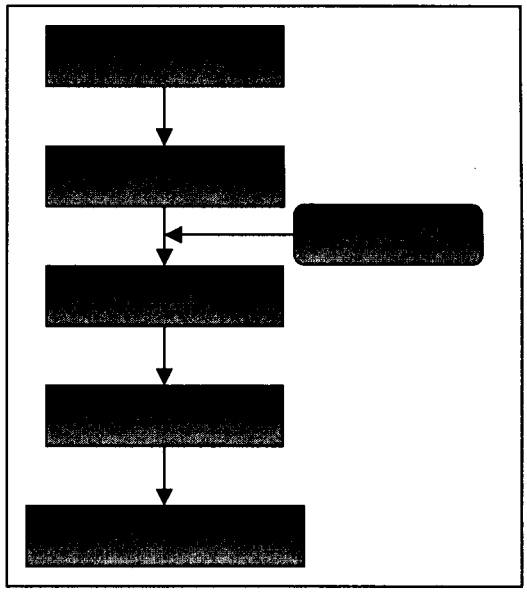


[그림2-1] Java Card Virtual Machine 구성

- Off-Card VM
  - 자바 클래스 로딩, 링킹, 네임 솔루션 및 패키지 변환
  - 검증
  - 컨버터를 이용한 CAP 파일, EXP 파일, JCA 파일 생성
  - 바이트 코드의 최적화
- On-Card VM
  - 바이트 코드 수행(Excute)
  - 수행시 메모리관리
  - 방화벽 루틴을 수행

2.2 애플릿 개발

애플릿 개발 순서를 도식화 하면 [그림2-2]와 같다. [3]



[그림2-2] 애플릿 개발 순서

Converter 는 카드리더기가 연결된 단말기 측에 탑재되어 Class 파일을 입력으로 받아 On-Card 상에서 실행될 바이너리 파일 형태인 CAP 파일과 EXP 파일을 생성한다. 그리고 해당하는 내용을 토큰 형식으로 가지고 있다가, ROM 에 Masking 될 때 사용되는 JCA 파일로 변환시키는 모듈이다.

2.3 CAP 파일의 정의와 구조

CAP 파일은 스마트 카드에서 메모리 제한 및 로드시의 안정성을 고려하여 자바 플랫폼에서 post-issuance를 위해 사용하는 파일의 형식으로 패키지 단위의 실행 가능한 바이너리 표현을 가지고 있다.

자바 카드 CAP 파일은 11개의 컴포넌트로 구성되고 각 컴포넌트는 자바 패키지에 정의된 요소들이나 CAP 파일의 용도를 설명하고 있다

CAP 파일의 각 컴포넌트들을 기능 별로 분류해 보면 아래와 같이 분류할 수 있다. [2]

- Information Components : Header, Diretory
- Core Components : Class, Method, StaticField
- Link Components : Import, Constant Pool, ReferencedLocation
- Other Components : Applet, Export, Descriptor

각 컴포넌트는 [표3-1]에서의 태그 값으로 CAP 파일 내에서 각 컴포넌트를 구별 할 수 있다.

Component Type	Value
COMPONENT_Header	1
COMPONENT_Directory	2
COMPONENT_Applet	3
COMPONENT_Import	4
COMPONENT_ConstantPool	5
COMPONENT_Class	6
COMPONENT_Method	7
COMPONENT_StaticField	8
COMPONENT_ReferenceLocation	9
COMPONENT_Export	10
COMPONENT_Descriptor	11

[표3-1] CAP 컴포넌트의 태그들

각 컴포넌트의 역할을 살펴보면 [4][5][6]

가. Header Component

Header Component는 CAP 파일의 일반적인 정보와 CAP 파일이 정의하고 있는 package의 정보-magic number, version, package의 길이, package AID-를 저장한다.

나. Directory Component

Directory Component는 CAP 파일에 정의되어있는 각 component의 크기 리스트를 만들고, 새로운 component를 위한 엔트리를 포함한다.

다. Applet Component

Applet Component는 package 안에 정의되어진 각 Applet의 엔트리-applet AID, install method offset-를 포함하고있다.

라. Import Component

Import Component는 클래스에 import한 외부 package 집합의 리스트를 만든다.

마. Constant Pool Component

Constant Pool Component는 Method Component의 내용 중 메소드의 instruction에서 참조되는 클래스, 필드 및 메소드에 대한 연결 정보를 포함한다.

바. Class Component

Class Component는 super class, instance size와 method component로의 offset을 각 원소로 하는 virtual method table 등을 포함 하고 있다.

사. Method Component

Method Componet는 현재 패키지 안에 선언되어진 메소드들과, 실행중인 <clinit> 메소드, 인터페이스 메소드 선언들을 기술한다. 클래스에 정의된 Abstract method 를 포함하고, 그리고 각 메소드와 결합되는 예외처리를 기술하고 있다.

아. Static Field Component

Static Field Component는 현재 패키지안에 정의되어있는 모든 static field의 이미지(image)의 초기화와 생성을 위해 요구되는 모든 정보를 포함한다. 그러나 Primitive type의 Final static field는 static field image에 표현되지 않는다.

자. Reference Location Component

Constant Pool Component의 배열 constant_pool내에 있는 인덱들을 포함하는 항목들에 대한 Method Component 내에 바이트 코드 중 opcode에 따르는 operand에 대한 offsets의 list를 나타낸다.

차. Description Component

Description Component는 CAP file의 모든 element를 parse, verify하기에 충분한 정보를 제공한다. Description component는 Constant Pool Component, Class Component, Method Component Static Field Component의 element를 참조한다.

3. CAP File 분석기 설계

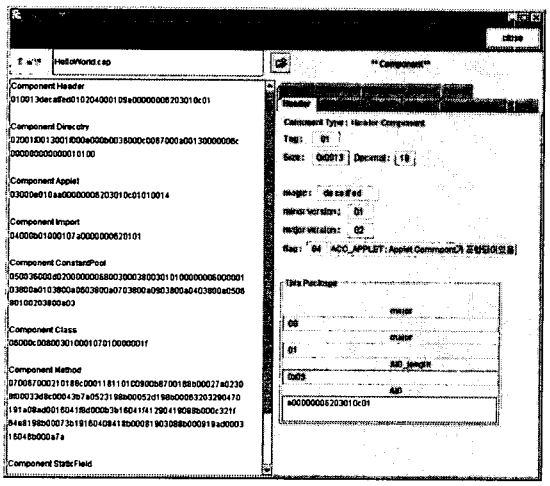
CAP 파일 분석기의 구성은 파일을 읽어 들일 수 있는 파일 입력부와 입력한 파일을 *.dump 파일 형식으로 뿌려주는 컴포넌트 선택부가 있다. 그리고 각 컴포넌트의 구조를 설명하고 있는 Tab으로 구성되어 있는 화면이 있다.

이 CAP 파일 분석기는 원하는 디렉토리에서 파일을 선택하거나 직접 입력으로 분석을 원하는 파일을 선택할 수 있다. 선택되어진 파일은 각 컴포넌트별로 클래스 정보, 링킹 정보, 검증 정보 등의 불필요한 정보를 제외한 실 데이터만 보여지게 된다. 각 컴포넌트별로 분류되어진 데이터의 내용을 알고자 하면 해당하는 컴포넌트를 클릭하거나 오른쪽의 해당 컴포넌트의 탭을 선택하여 주면 된다.

각 컴포넌트를 설명하기 위한 탭 화면은 각각의 자료구조에 맞게 구성되어있고 특히 visual 하게 보여줄 수 있어 애플릿 개발자가 쉽게 CAP 파일을 분석할 수 있도록 한다.

4. CAP File 분석기 구현

클래스 파일을 입력으로 받은 Converter의 출력물인 *.cap 파일을 개발자가 쉽게 분석하기 위해 CAP 파일 분석기를 사용할 수 있다.

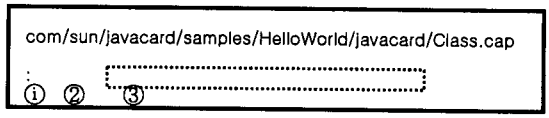


[그림 4-1] CAP 파일 분석기 화면

CAP 파일의 구조 분석의 예로서 HelloWorld.cap 파일의 각 컴포넌트 중 핵심 컴포넌트인 Class, Methodd, StaticField 컴포넌트의 구조를 분석하도록 한다.

4.1 클래스 컴포넌트 (Class Component)

클래스 컴포넌트는 상위 클래스, 인스턴스 크기와 method 컴포넌트로의 오프셋을 각 원소로 하는 가상 메소드 테이블을 포함하고 있는 컴포넌트이다.



[그림4-2]Class Component

클래스 컴포넌트의 값을 분석해 보면

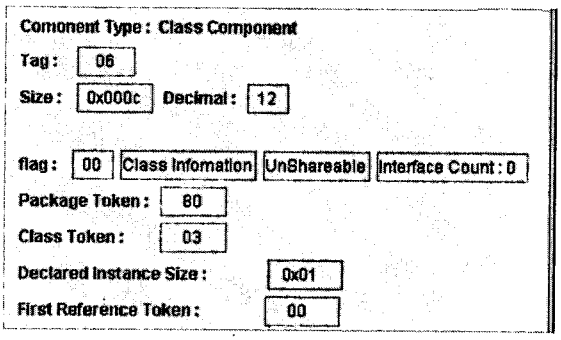
① 은 현재의 컴포넌트 값을 구별해 주는 태그 (tag)

② 는 ①의 태그 값과 현재의 2바이트 값을 제외한 컴포넌트의 총 크기를 나타내는 값이다.

③ 은 class_info에 해당되는 값들로 첫 바이트 (00)는 bitfield 로서 상위 4비트는 현재의 값들이 인터페이스 또는 클래스를 설명하기 위해서 사용될지를 마크한다. 인터페이스를 표현할 경우에는 1의 값을, 클래스를 표현할 경우에는 0의 값을 가질 것이다.

그 다음의 2바이트는 (80 03) 클래스 또는 인터페이스에 대한 참조를 나타내는 자료 구조로서 현 현재 클래스의 슈퍼 클래스를 표현하는 구조이고 만약 현재 클래스의 수퍼 클래스가 없다면 0xffff로 표현되어진다.

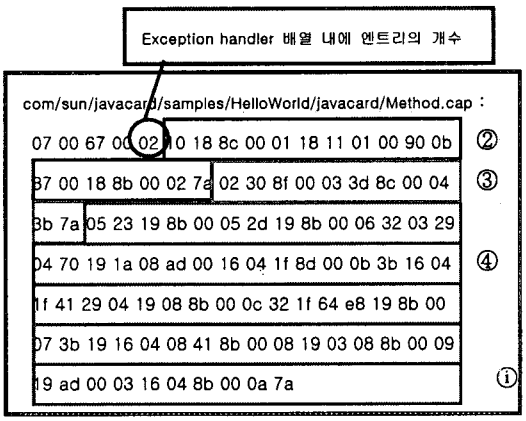
그림[4-3]은 CAP 파일 분석기에서 클래스 컴포넌트를 보여주는 화면이다.



[그림 4-3] 클래스 컴포넌트 화면중 일부

4.2 메소드 컴포넌트(Method Component)

Method Componet는 현재 패키지 안에 선언되어진 메소드들과, 실행중인 <clinit> 메소드, 인터페이스 메소드 선언들을 기술한다. 클래스에 정의된 Abstract method 를 포함하고 ,그리고 각 메소드와 결합되는 예외처리를 기술하고 있는 컴포넌트 이다.



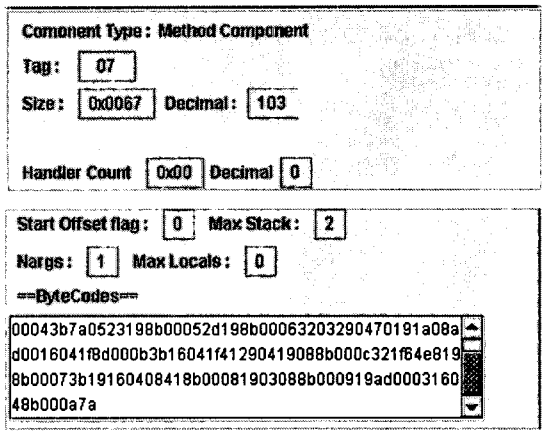
[그림 4-4] Method Component

클래스 컴포넌트의 내용을 분석해보면

[그림 4-4]에서 보이는 바와 같이 첫 3 바이트 값들은 어느 컴포넌트에서나 동일하게 태그와 크기를 나타낸다.

①은 메소드 컴포넌트 자료구조 중 method_info의 배열을 나타내는 값으로 배열의 각 엔트리(entri)들은 예로든 HelloWorld.cap 파일 내에서 정의된 각 메소드들□ 생성자, install(), process() - 의 모든 오브젝트코드(opcode)를 표현하고 있다 ①(HelloWorld의 생성자)의 오브젝트 코드에 대해서 설명하겠다. 첫번 4비트는 현재 메소드의 접근 제한을 나타내며 다음 4비트는 현재의 메소드를 수행하기 위해서 사용되는 최대 스택의 개수를 가르키고 있다. 3번째 4비트의 값은 현재 메소드가 가지고 있는 인자를 전달하기 위해 요청되는 16 비트 셀의 개수를 나타내며 4번째의 4비트 값은 현재 메소드에 의해 선언되어진 지역 변수를 표현하기 위해 요구되는 16 비트 셀의 개수를 가리키고 있다. 그 아래의 값들은 이 메소드를 구현하고 있는 자바 바이트 코드의 값들이다.

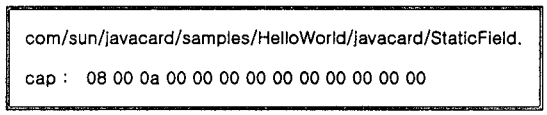
그림[4-5]은 CAP 파일 분석기에서 클래스 컴포넌트를 보여주는 화면이다.



[그림4-5] 메소드 컴포넌트 화면중 일부

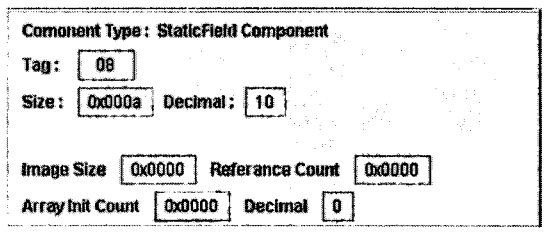
4.3 스태틱 필드 컴포넌트 (StaticField Component)

Static Field Component는 현재 패키지 안에 정의되어 있는 모든 static field의 이미지(image)의 초기화와 생성을 위해 요구되는 모든 정보를 포함하는 컴포넌트이다.



[그림4-6] StaticField

[그림4-7]은 스태틱 컴포넌트를 분석해놓은 화면이다.



[그림4-7] 스태틱 컴포넌트 화면중 일부

4.4 CAP 파일 분석기의 주요 구현 코드부분

CAP 파일 분석기는 메인 파일인 CapStruct.java와 CAP 파일을 덤프파일로 제공하는 CapDump.java, 각 컴포넌트의 구조를 나타내는 TabHeader.java 세 파일로 구성되어져 있다.

```
CapDump cd = new CapDump();
 this.fileString = "";
 this.fileString = listString;
 data = cd.getDumpData(listString);
 component_list.setListData(data);
 if(data.size() > 1 ){
  p_detail.removeAll();
  p_detail.addTab("Header",new TabHeader());
  p_detail.addTab("Direcotry", new TabDirecotry());
  p_detail.addTab("Applet",new TabApplet());
      ...
```

[CapStruct.java 소스 중 일부]

위 소스는 CAP 파일을 입력으로 받아서 CapDump 에 파라메타로 넘겨주어 덤프파일의 내용을 리턴받는다. 그리고 해당하는 파일의 내용이 있을 경우 탭들을 활성화 시켜준다.

```
//-------------Component_Header-----------------
if(douString.equals(Component_Header)){
// 벡터에 Component Header의 스트링을 넣음
 data.add("Component Header");
// Component Header 의 크기를 구함
 size = Hex_Change(tempString.substring(i+2,i+6));
 size = size *2;
 if(size > 50){
  div = size / 50;
  namegi = size % 50;
  start = i;
  for(int j = 0 ; j < div ; j++){
   data.add(tempString.substring(start,start=start
    +50));
  }
  data.add(tempString.substring(start,start+namegi
   +6));
 }else{
 data.add(tempString.substring(i,i+6+size));
}
```

[CapDump.java 소스 중 일부]

위 소스는 각 해당 컴포넌트를 분류하는 부분으로 해더 컴포넌트를 분류하는 방법을 보여주고 있다

6. 결론

본 논문에서는 USIM 기반 자바카드 플랫폼에서 스마트 카드의 적재와 카드상에서 메모리 적재 및 링킹 등 운영의 편의를 위해 제공되어 지는 CAP 파일을 분석 하기 위한 툴에 대해 다루었다. CAP 파일 분석기는 16진수로 표현되어진 CAP 파일의 정보를 개발자가 쉽게 분석할 수 있도록 각 컴포넌트 별로 자세히 표현하고 있다. 또 본 논문에서는 Java Card에 대한 전반 적인 이해와 CAP 파일의 구조에 대해 자세히 설명 되어지고 있어 Java Card을 처음 접하는 개발자들에게 CAP 파일에 대한 좀 더 구체적인 부분에 대해 이해를 도울 수 있을 것이다.

CAP 파일은 클래스 정보, 바이트코드, 링킹 정보, 검증 정보 등 많은 요소들로 구성되어있다. 향후 연구 계획은 이러한 CAP 파일의 불필요한 정보를 제거 할 수 있는 CAP 파일 최적화를 통해 스마트 카드 환경의 메모리를 효율적으로 사용 할 수 있도록 하는 것이다.

[참고 문헌]

[1] http://www.opencard.org

[2] http://java.sun.com/product/javacard

[3] 조증보 'Java Card Application 활용을 위한 APDU 자동 분석기 구현' 멀티미디어 학회 논문집

[4] Sun Microsystems, Inc., The Java Card™ 2.1.2Virtual Machine Specification, SUN, 2001

[5] Sun Microsystems, Inc., The Java Card™ 2.2Runtimel Environmen(JCRE)t Specification, SUN, 2002.

[6] Sun Microsystem. Java Card 2.1.1 Virtual Machine Specification, 2000