

자바 카드 API를 이용한 RSA 알고리즘 구현

이직수*, 김상국**, 이명선**, 이원구*, 이재광*
한남대학교 컴퓨터공학과**
한국과학기술정보연구원*

Implementation of RSA Alogrithm Using Java Card API

Jik-Su Lee*, Sang-Kuk Kim**, Myung-Sun Lee**, Won-Goo Lee*,
Jae-Kwang Lee*

Dept of Computer Engineering, Hannam University*
Korea Institute of Science and Technology Information**

요 약

자바카드 API는 스마트 카드와 같은 작은 메모리를 가진 임베디드 장치에서 실행환경을 최적화하기 위해 구성되었다. 자바카드 API의 목적은 한정된 메모리를 가진 스마트카드 기반의 프로그램을 개발할 때 많은 이점을 제공한다. 본 논문에서는 자바 카드 API를 이용하여 공개키 암호 알고리즘 중에서 대표적인 RSA 알고리즘 구현하는 것에 목적을 둔다.

1. 서 론

효과적인 정보보호 서비스를 제공하기 위해서 스마트 카드의 사용이 급증하고 있으며, 이와 관련한 기술 개발이 활발히 이루어지고 있다[4]. 개인용 컴퓨터나 금융망, 행정망 및 의료망 등에서 정보보호에 스마트 카드를 사용하는 기술이 이미 일부 국가에서는 실용화 단계에 있으나 국내에서는 실용화를 위한 준비단계에 있다.

스마트 카드를 사용하는 가장 큰 목적은 카드 내에 저장된 데이터를 안전하게 보호하는 일이다. 지금까지 스마트 카드는 일반적인 컴퓨터 시스템에 대한 정보 보호 예방 기술만을 제공해 왔으나, 안전한 인터넷 언어라고 알려진 자바 언어를 스마트 카드에 적용한 자바 카드는 스마트 카드의 정보보호 특성을 그대로 보존할 뿐만 아니라, 여러 사람에게 스마트 카드를 위한 프로그래밍 기술을 공개해줌으로써 스마트 카드 자체를 인터넷을 위한 하나의 새로운 응용 플랫폼으로 활용할 수 있도록 하였다.

자바 카드를 비롯한 스마트 카드는 인터넷 프로그래머들에게 개인 암호 키와 같은 비밀 정보를 안전하게 생성하고 저장해줄 수 있는 공간을 제공해준다.

즉, 개인 프라이버시 보장을 위한 서명 및 인증 기능과 기존에 서로 만나서 행해야 했던 계약, 양방향 서명 기능 등을 원격으로 그리고 좀 더 안전하게 행할 수 있게 해준다. 뿐만 아니라 이동성 측면에서 볼 때 개인 암호 키와 같은 비밀 정보를 자신의 PC에 저장하는 것보다는 자바 카드나 스마트카드에 저장하는 것이 훨씬 유리하므로 전자상거래 및 암호 기술 발전과 더불어 이들 카드를 PC와 전자상거래로 연결시키려는 움직임이 활성화되고 있다.

본 논문에서는 자바 카드 상에서 공개키 암호 알고리즘 구현에 반드시 필요한 여러 가지 연산을 지원하는 클래스를 설계하고 구현하였다. 2장에서는 자바카드의 기본 구조와 동작 과정을 살펴보고, 3장에서는 클래스의 설계와 적용된 알고리즘을 설명하고, 그리고 4장에서는 클래스를 구현하고, 구현된 클래스의 동작을 검증하였다. 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1 자바 카드 구조

일반적인 자바 카드의 구조는 다음의 그림 1에 주어진 바와 같이 카드 운영체제(COS: Card Operating System), 자바카드 가상 머신(JCVM: Java Card Virtual Mahine), 자바 카드 API(Application Programming Interface), 사용자 확장

* 본 연구는 과학기술부 지역협력연구사업(R12-2003-004-02004-0)지원으로 수행되었음

(Industry-Specific Extension) API 그리고 다양한 애플릿(Applet) 프로그램들로 구성된다.

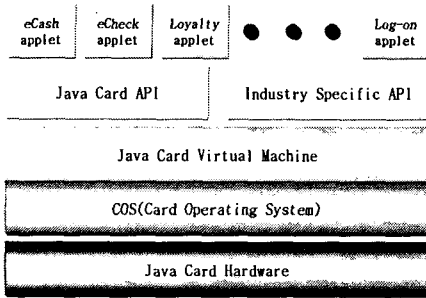


그림 1 자바 카드 구조

2.2. 자바 카드 특징

자바 카드 구현에 있어서 중요한 것은 자바 프로그래밍 기술을 제한된 스마트 카드 리소스에 활용하는 일이다. 자바 카드에 탑재되는 가상 머신은 기존의 자바 가상 머신 기능 중에 기본적으로 간단한 기능만을 집약시킨 것으로 자바 카드 가상 머신(JCVM: Java Card Virtual Machine)이라고도 불리우며 선과 자바 카드 포럼 측에 의해 리소스 활용 효율을 높이기 위한 방안으로 제안되었고 Float, Double, Long등의 데이터 타입 지원과 쓰레딩 지원 기능 등을 표준 자바 규격에서 제외시킨 가상 머신이다.

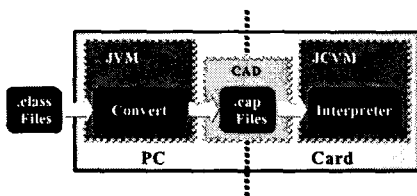


그림 2 분리 가상 머신 구조

자바 카드의 특징 중의 하나는 COS위에 랩핑되는 가상 머신의 구조로 그림 2 에 나타낸 바와 같이 온-카드 가상 머신(On-Card VM)과 오프-카드 가상 머신(Off-Card VM)으로 이루어진 분리 가상 머신(Split VM)이라는 점이다. 자바 카드에서 가상 머신을 이와 같이 분리하여 구현하는 목적은 자바 카드가 가지는 하드웨어의 리소스 제한을 보다 효율적으로 활용하기 위함이며 온-카드 가상 머신에는 바이트코드 수행(Bytecode Execution)기능을, 오프-카드 가상 머신에는 클래스 로딩 및 검증(Class Loading & Verification) 기능과 바이트코드 최적화 및 변환(Bytecode Optimization & Conversion)기능을 따로 두어서 이러한 목적을 달성하고자 하였다.

2.3 자바 카드 애플릿

자바 카드 애플릿은 자바 카드 상에서 실행될 수

있는 자바 프로그램이다[1]. 자바 응용 프로그램과 달리 자바 애플릿은 카드의 ROM에 설치될 필요가 없고, 단지 카드 상에 다운로드 함으로써 사용이 가능하게 된다. 자바 카드 애플릿의 특징은 다음과 같다.

- 자바 카드 런-타임 환경에서 수행된다.
- APDU(Application Program Data Unit)교환을 통해 JCRE와 통신한다.
- AID(Application Identifier)에 의해 식별된다.
- 카드 상에 동적으로 다운로드 될 수 있다.

애플릿과 호스트간의 통신은 그림 2에서 나타낸 바와 같이 명령어 APDU와 응답 APDU로 구성되는 APDU 교환을 통해서 이루어진다. APDU 교환은 애플릿과 호스트간에 직접 이루어지는 것이 아니라 JCRE를 매개로 하여 이루어지고, JCRE는 애플릿과 호스트간에 교환되는 APDU의 관리와 감독 역할을 수행한다. 따라서 애플릿과 CAD(Card Access Device) 또는 호스트간의 직접적인 통신은 불가능하며, JCRE를 통한 통신만이 가능하다[1].



그림 3 애플릿 통신

아래 그림 3과 그림 4는 APDU의 구조를 살펴본 것이다.

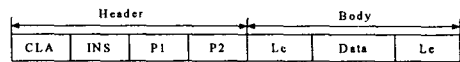


그림 4 명령(Command) APDU 구조

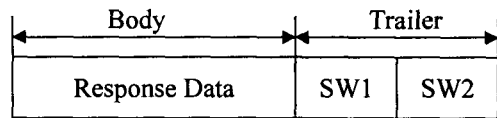


그림 5 응답(Response) APDU 구조

2.4 RSA 알고리즘

현재 공개키 암호기법들 중에서 가장 많이 사용되고 있는 알고리즘으로써, 1977년 Ron Rivest, Adi Shamir 그리고 Leonard Adleman 이라는 세명의 수학자들에 의해 제안된 방식이다. 큰 소수수의 Exponent Modulo 연산을 이용한 공개키 암호화 알고리즘 랜덤하게 구해진 파라미터인 P, Q로부터 e (encrypt exponent)와 d(decrypt exponent), n(modulus)를 구하고 $M \pmod n$ 과 $C \pmod n$ 연산으로 암호화와 복호화를 수행한다. [5]

- 송신 : 송신자는 A는 B의 공개키 e를 획득하여 $C = M^e \pmod n$ 를 계산하여 암호화된 메시지 C를 전송한다.
- 수신 : 수신자 B는 자신의 비밀키 d로 $M = C^d \pmod n$

n를 계산하여 원래의 메시지 M을 얻는다.

RSA 알고리즘의 암호화와 복호화 단계를 간단히 표현하면 다음과 같은 수식으로 표현할 수 있다.

암호화	$C=M^e \text{ mod } n$
복호화	$M=C^d \text{ mod } n$

3. 구현

본 장에서는 RSA 알고리즘을 자바 카드 명세서에 따라 구현하는 방법에 대하여 설명한다. 자바 카드 API에는 모듈러 지수 연산, 최대공약수 계산, 승산역원의 계산등 공개키 암호 알고리즘 구현에 반드시 필요한 연산 메소드를 지원하지 않는다. 따라서, 이미 개발한 자바 카드용 BigInteger 클래스를 이용하여 구현하였다.

3.1 개발 환경

RSA 알고리즘 구현 환경은 다음의 표 1과 같다.

표 1 구현 환경

운영환경	Windwos 2000 Professional	
하드웨어	CPU	Pentium III 800 MHz
	RAM	256MB
개발도구	Java Card 2.1.2 Development Kit	

3.2 BigInteger 클래스

BigInteger 클래스는 JDK1.3에 포함되어 있는 클래스이며, 정수의 범위를 넘어가는 수들에 대한 모듈러 지수 연산, 최대 공약수 연산, 승산역원 그리고 소수 판정 및 생성등을 지원한다. 이 클래스는 공개키 암호 알고리즘에 반드시 필요하며 자바 카드 API 는 제공 하지 않기 때문에 구현을 한 후, 이를 이용하여 RSA 를 구현하였다.

3.3 RSA 클래스 구현

구현된 RSA 에서 실제 암호화와 복호화 부분만을 표현하면 다음과 같다. 연산에 필요한 파라메터와 키 를 처리하는 부분은 생략하였다.

```
//암호화 메소드
private byte[] Encryp (byte input[]) {
//공개키 모듈러 nB 값을 읽음
short mod_len = pubKey.getModulus(mod_byte,0);
//읽어온 mod_byte를 BigInteger 형으로 변환
BigInteger nB = new BigInteger(1, mod_byte);
//공개키에서 공개키 exponent n 값을 읽음
short exp_len = pubKey.getExponent(exp_byte,0);
//읽어온 exp_byte를 BigInteger 형으로 변환
BigInteger e = new BigInteger(1, exp_byte);
//암호화할 메시지를 BigInteger형으로 변환
BigInteger M = new BigInteger(1, input);
//암호화 C=M^e mod n을 계산
BigInteger C = M.modPow(e,nB);
//암호화된 C의 바이트 배열을 반환
return C.toByteArray();
}
```

```
//복호화 메소드
private byte[] Decrypt(byte input[]) {
//개인키에서 모듈러 n 값을 읽음
short mod_len = priKey.getModulus(mod_byte,0);
//읽어온 mod_byte 를 BigInteger 형으로 변환
BigInteger n = new BigInteger(1, mod_byte);
//개인키에서 비밀키 exponent d 값을 읽음
short exp_len = priKey.getExponent(exp_byte,0);
//읽어온 exp_byte를 BigInteger 형으로 변환
BigInteger d = new BigInteger(1, exp_byte);
//복호화할 메시지를 BigInteger 형으로 변환
BigInteger C = new BigInteger(1, input);
// 복호화 M = C^d mod n을 계산
BigInteger M = C.modPow(d,n);
//복호화된 M의 바이트 배열을 반환
return M.toByteArray();
}
```

4. 시뮬레이션

원칙적으로 개발한 RSA 알고리즘을 검증하기 위해서는 오프 카드 설치 프로그램과 온 카드 설치 프로그램 을 이용하여 실제 자바 카드 상에 설치한 후에 검증을 해야 하지만, Sun에서 제공하는 Java Card Development kit을 이용하여 검증하였다. 이때 생성된 CAP 파일의 정당성을 검증해주는 verifycap 명령을 이용하여 파일의 정당성을 검증받고, 실행은 작성한 API 와 검증용 애플릿을 메모리에 함께 로드한다. 그 후 검증용 애플릿과 APDU 명령으로 서로 통신을 수행하는 애플릿 호출 파일을 작성해서 그 통신 내용을 확인하였다. 검증용 애플릿에서 암호화/복호화를 요청하면 자바카드 API 가 이를 받아서 RSA 클래스를 호출한다. 이때 RSA 클래스가 BigInteger 클래스를 호출해서 연산을 수행하고, 결과를 반환하게 된다. 이 단계들은 모두 애플릿 호출 파일에서 APDU 명령

어를 메모리 상에 로드되어 있는 애플릿에 전달되면서 실행된다. 아래에 검증용 애플릿에서 암호화와 복호화를 호출하는 부분이 있다.

```
private void proc(APDU apdu) {
//주어진 파라미터로 공개키와 개인키를 생성
Cipher test = Cipher.getInstance(Cipher,ALG_RSA_PKCS1,
true);
//암호화를 위한 초기화
test.init(pubKey,Cipher.MODE_ENCRYPT);
//암호화
short number = test.doFinal(M,0,M.length,C,0);
//복호화를 위한 초기화
test.init(priKey,Cipher.MODE_DECRYPT);
//복호화
number = test.doFinal(C,0,C.length,M,0);
}
```

다음은 검증용 애플릿과 통신을 위한 APDU 명령어로 구성된 파일이다.

```
... (생략) ...
// 평문 메시지 입력
0xC0, 0x10 0x00 0x00 0x14 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x02;
//Call RSA 호출 - 암호화와 복호화 수행
0xC0, 0x70 0x00 0x00 0x00 0x02;
//Call 암호문 블록 1 요청
0xC0, 0x21 0x00 0x00 0x00 0x7F;
... (생략) ...
//Call 복호화 결과 요청
0xC0, 0x25 0x00 0x00 0x00 0x7F;
... (생략) ...
```

앞에서 작성한 검증용 애플릿과 통신을 위한 APDU 명령어 파일을 이용하여 실행하여 얻은 결과가 다음에 있다.

```
... (생략) ...
CLA:c0, INS: 10, P1:00, P2:00, Lc:14, 01, 01, 01, 01, 01,
01, 01, 01, 01, 01, 01, 01, 01, 01, 01, 01, 01, 01,
00, SW1:90, SW2:00
CLA : c0, INS:70, P1:00, P2:00, Lc:00, Le:00, SW1:90,
SW2:00
CLA: c0, INS:21, P1:00, P2:00, Lc:00, Le:20, 1c, 27, f1,
53, 86, 09, 88, 68, 5f, de, 1c, 6e, 75, 9e, b4, 3b, 2c, cf, 7f
b1, 9e, 5a, 89, cb, 4a, a7, 7f, fa, d2, b9, e5, 62, SW1:90,
SW2:00
... (생략) ...
CLA: c0, INS:25, P1:00, P2:00, Lc:00, Le:14, 01, 01,
01, 01, 01, 01, 01, 01, 01, 01, 01, 01, 01,
01, 01, SW:90, SW2:00
... (생략) ...
```

위의 실행 결과에서 CLA 는 명령어를 수행할 애플릿을, INS는 명령어의 종류를, Lc는 데이터의 길이를 의미하고, Le는 응답 데이터의 길이를 의미한다. 즉, 첫줄에서 Lc 14는 평문 20바이트가 입력된다는 것을 의미하며, 여섯 번째 줄의 Le 20은 암호화된 결과 중 에서 32바이트만을 요청한 것이다. 이 블록은 총 4이며, 마지막 부분에서 다시 복호화를 수행해서 원래의 평문 20바이트가 반환된 것을 볼 수 있다. 상태 코드 SW1 이 모두 90으로 성공적으로 응답이 완료되었다는 것을 확인할 수 있다.

5. 결론

본 논문에서는 자바카드 API에서 지원하지 않는 BigInteger 클래스를 이용하여 공개키 암호 알고리즘인 RSA 알고리즘을 구현하였다. 일반적으로 자바 카드와 같은 IC 카드에서는 별도의 수치 연산기를 하드웨어적으로 구현하여 공개키 암호화를 수행하지만, PC 환경에서 이를 이용하기란 불가능하다. 만약 실제 카드 상으로 이식을 하더라도, 연산 부분을 호출하는 부분이 별도의 메소드로 구현되어 있기 때문에, 연산 부분만을 대치하면 사용이 가능하다. 새로운 공개키 암호 알고리즘으로 대두되고 있는 타원 곡선 알고리즘 구현은 향후 연구 과제로 남긴다.

[참고문헌]

- [1] Chen, Zhiquan "Java Card Technology for Smart Cards", pp.42-77. ADDISON-WESLEY Company, 2000
- [2] "Java Card TM 2.1.2 Development Kit User's Guide", pp.66, Sun MicroSystems, Inc. 2001
- [3] 문상재 외 "차세대 IC 카드를 사용한 정보보호 신기술 시스템 개발", pp17, 정보통신부 1997.
- [4] 김연선, 이창욱, "자바 카드 애플릿 설계 및 검증에 관한 연구", 한국통신정보보호학회 종합학술발표회 논문집 Vol.10, No.1 pp805, 2000
- [5] "PKCS#1: RSA Encryption Standard", pp.10-11, RSA laboratories, 1992
- [6] <http://java.sun.com/products/javacard/datasheet.html>