자바 카드 API를 이용한 SEED 알고리즘 구현

Implementation of SEED Using Java Card API

채철주, 이성현, 이재광*
한남대학교 컴퓨터공학과*

Chae Cheol-Joo, Lee Seoung-Hyun,
Lee Jae-Kwang*
Dept. of Computer Engineering, Hannam University*

요약

인증 및 전자 서명 기능은 기존의 계약 행위를 원격으로 안전하게 행할 수 있게 해준다. 그러나 이를 위해서는 개인 암호 키와 같은 비밀 정보를 안전하게 저장하고 관리해줄 수 있는 방안이 필요하다. 이러한 방안으로 자바 카드와 같은 스마트카드를 인터넷에 활용하여 보안 문제를 해결하고자 하는 노력이 점차 활성화되고 있다. 본 논문에서는 자바 카드를 기반으로 SEED 알고리즘을 구현하였다.

Abstract

Authentication and digital signature make secured existing contract in remote spot. But, It required method of storing and managing secret, such as private key password. For this method, we make efforts solution of security with smart cart, such as java card. This paper implement SEED algorithm based on Java Card

I. 서 론

스마트 카드는 오늘날 사용되는 가장 작은 컴퓨팅 플랫폼중의 하나이다. 스마트 카드의 메모리 구성은 RAM 1K, EEPROM 16K, 그리고 ROM 24K 정도를 가진다. 자바 카드 기술 디자인의 가장 큰 도전과제는 응용프로그램들을 위한 충분한 공간을 유지하면서 스마트카드에 자바 시스템 소프트웨어를 적재하는 것이다. 해결책은 자바 언어 특징들 중의 일부분만을 지원하는 것이다.

자바 카드(Java Card)는 1996년 3월 스마트 카드(Smart Card) 제조 업체인 Schlumberger사에 의해 처음 소개된 것으로 자바 기술을 스마트 카드 기술에 접목시킨 것이다. 처음에 출시된 자바 카드는 축소된 기능의 자바 바이트코드 인터프리터(Java Bytecode Interpreter)를 OS로 탑재하고 변형된 자바 클래스 파일을 다운로드하여 사용하던 것이었으나, 몇 가지 단점으로 인해 실용화되지 못하였다. 그러나 1996년 10월 Sun Microsystems사가 Schlumberger사의 개발 경험을 바탕으로 자바 카드의 구현 목표 및 구조와 자바 언어 서브셋 그리고 암호 API 규격 등을 기록한 초기 자바 카드 규격을 공표함에 따라, 1997년 2월에는 Gemplus사와 Schlumberger사가 자바 카드 공동 포럼을 결성하였다. 여기에 De La Rue, Bull, 그리고 G&D(Gieseke & Devrient)같은 카드 제조업체들이 자바 포럼에 참여하기 시작하면서부터 실질적인 자바 카드 구현이 가능해졌으며, 1997년 말에는 보다 구체적이고 실질적인 자바 카드 규격으로

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

Java Card 2.0 규격이 공표되었다.

자바 카드를 비롯한 스마트 카드는 인터넷 프로그래머들에게 개인 암호 키와 같은 비밀 정보를 가장 안전하게 생성하고 저장해줄 수 있는 공간을 제공해 준다. 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 카드와 같은 스마트카드에 저장하는 것이 훨씬 유리하므로 전자상거래 및 암호 기술 발전과 더불어 스마트카드를 PC와 전자상거래로 연결시키려는 움직임이 활성화되고 있다.

스마트 카드를 사용하는 가장 큰 목적은 카드 내에 저장된 데이터를 안전하게 보호하는 일이다. 지금까지 스마트 카드는 일반적인 컴퓨터 시스템에 대한 정보보호 예방 기술만을 제공해 왔으나, 안전한 인터넷 언어라고 알려진 자바 언어를 스마트 카드에 적용한 자바 카드는 스마트 카드가 가지는 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.

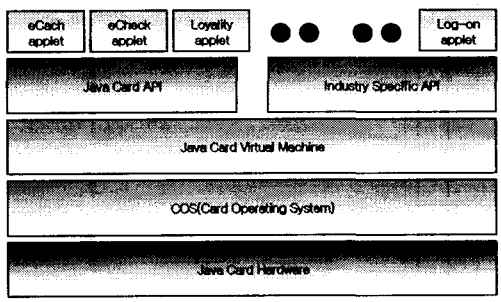
자바 카드의 출현은 그 동안 해결하지 못했던 스마트 카드의 응용 제한성을 완화시켰는데, 이것은 자바 언어로 개발되는 응용 프로그램이 갖는 플랫폼 독립(Platform Independence)적인 특성과 오픈 플랫폼(Open Platform)적인 특성으로 인한 것이다. 플랫폼 독립적인 특성은 응용 프로그램 개발 시 프로그램 개발자가 모토로라나 인텔 칩 같은 하드웨어 의존 어셈블리 코드로 프로그래밍할 필요 없이 자바 언어로 직접 프로그래밍할 수 있는 특성을 말하며, 프로그램 개발 기간 단축 및 비용 절감의 효과를 가져올 수 있는 수단이 된다. 오픈 플랫폼 특성은 카드가 발급된 이후에도 기존의 응용 프로그램을 삭제하고 인터넷을 통해 새로운 응용 프로그램들을 안전하게 다운로드하여 사용함으로써 카드의 용도 변경 및 활용 효율성을 강화시켰을 뿐만 아니라 자바 카드의 새로운 응용 처들을 창출하는 계기가 된다.[1]

본 논문의 구성은 2장에서는 자바카드의 구조 및 특징에 대해서 기술하고, 3장에서는 SEED 알고리즘에 대해서 기술하였다. 그리고 4장에서는 SEED 알고리즘을 자바 카드 기반으로 구현하는 방법에 대하여 기술하였다.

Ⅱ. Java Card의 구조 및 특징

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



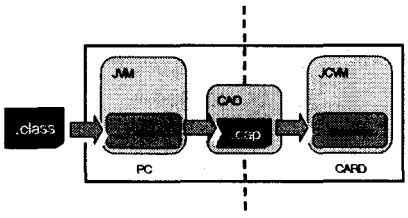
▶▶ 그림 1. 자바 카드 구조

카드 운영 체제에는 메모리 액세스 및 I/O 핸들링을 위한 디바이스 드라이버와 암호 모듈 액세스를 위한 드라이버 코드가 적재되며, 자바 카드 가상 머신에는 자바 바이트 코드 서브 셋 수행을 위한 자바 인터프리터 코드와 서명 및 로그-인 같은 외부 접근 통

제 코드가 적재된다. 자바 API에는 JDK(Java Development Kit)로 제공되는 코어 API와 사용자에 의해 정의되는 Industry Specific API가 적재되며, 각종 응용 애플릿 프로그램으로는 신용카드, 전자화폐, 그리고 신분증명을 위한 코드 들이 적재되게 된다.

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 기본적이고 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 2에 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 둠으로써 이러한 목적을 달성하고자 하였다.

자바 카드의 또 다른 특징은 자바 카드를 위한 특별 API 셋 콜과 자바 바이트 코드의 적용 환경이다. 자바 카드를 위한 플랫폼에는 다이나믹 클래스 다운로드 기능 대신에 클로즈드-패키지(Closed-package) 개념이 도입되었는데, 이것은 자바 카드용으로 개발된 애플릿 코드가 컴파일된 후 컨버터를 거치게 될 때 런 타임 다이나믹 다운로딩에 필요한 모든 데이터의 삭제와 카드릿패키지(CAP: CardletPackage) 형태(라이브러리와 클래스 파일들이 패키지화되어 동시에 다운로드되는 형태)의 파일 변환이 이루어지게 한 것으로 이러한 특성 역시 카드의 리소스 활용 능력을 높이기 위해 제안된 것이다.

Ⅲ. SEED 알고리즘

SEED는 대칭키 암호 알고리즘으로써, 블록 단위로 메시지를 처리하는 국내표준 블록 암호 알고리즘이다. n비트 블록 암호 알고리즘이란 고정된 n비트 평문을 같은 길이의 n비트 암호문으로 바꾸는 함수를 말한다(n비트 : 블록 크기). 이러한 변형 과정에 암호키가 작용하여 암호화와 복호화를 수행한다[2].

128비트 블록 암호 알고리즘 SEED는 128비트 암호키를 이용하여 메시지를 128비트 블록 단위로 암·복호화하는 알고리즘으로 데이터의 비밀성 등과 같은 기능을 제공하기 위해 사용될 수 있다.

대부분의 블록 암호 알고리즘은 Feistel 구조로 설계된다. 보통 Feistel 구조는 3라운드 이상이며, 짝수 라운드로 구성된다.

이러한 Feistel 구조의 장점은 다음과 같다.

- 라운드 함수에 관계없이 역변환이 가능하다(즉, 암·복호화 과정이 같다).
- 두 번의 수행으로 블록간의 완전한 diffusion이

이루어진다.

- 알고리즘의 수행속도가 빠르다.
- H/W 및 S/W로 구현이 용이하다.
- 아직 구조상의 문제점이 발견되고 있지 않다.

Ⅳ. SEED 알고리즘 구현

본 장에서는 현재 국내 128비트 블록 암호 알고리즘 표준으로 사용되고 있는 SEED를 자바 카드 기반으로 구현하는 방법에 대해 설명한다.

1. 개발 환경

SEED 알고리즘의 구현 환경은 다음의 표 1과 같다.

[표 1] 구현 환경

운영체제	Windows 2000 Professional
하드웨어	Pentium III 800MHz
개 발 도 구	JDK 1.3
	Java Card 2.1 Development Kit

적은 메모리의 문제로 인해 자바 카드 플랫폼은 자바 언어의 모든 특징들 중의 일부분만을 지원하고 있다. 이 부분집합은 자바의 객체 지향 능력을 보존하면서 스마트카드나 다른 작은 장치들을 위한 프로그램을 작성할 수 있도록 잘 구성된 특징들을 포함하고 있다.

2. SEED 클래스 구현

다음은 구현된 SEED 알고리즘에 대한 소스와 설명이다.

```
// 암호화나 복호화를 위해 매개변수로서 데이터 "pbData"
// 키"pdwRoundKey"를 입력 값으로 받는다.
private static int[] seed(int pbData[], int pdwRoundKey[])
{
int L0, L1, R0, R1, T0, T1, TEMP;
int[] K= pdwRoundKey;
TEMP=0;
// 입력받은 데이터를 4바이트씩 4부분으로 분할.
L0=pbData[0];     L1=pbData[1];     R0=pbData[2];
R1=pbData[3];
//실제적인 암호화나 복호화를 수행하는 부분
// "SeedRound1"과 "SeedRound2" 함수를 호출하여 필요한
연산을 수행
T0 = R0 ^ K[0];
T1 = R1 ^ K[1];
T1 ^ = T0;
TEMP = SeedRound1(L0, L1, R0, R1, K[0],K[1]);
L1 = SeedRound2(L0, L1, R0, R1, K[0],K[1]);
L0 = TEMP;
TEMP = SeedRound1(R0, R1, L0, L1, K[2],K[3]);
R1 = SeedRound2(R0, R1, L0, L1, K[2],K[3]);
R0 = TEMP;
   ----------  중략  ----------
TEMP = SeedRound1(R0, R1, L0, L1, K[30],K[31]);
R1 = SeedRound2(R0, R1, L0, L1, K[30],K[31]);
R0 = TEMP;
// 암호화나 복호화가 수행된 데이터를 반환하기 위한 "retval
" 변수를 선언하고 변수에 각각의 데이터를 할당한 다음 값을 반
환한다.
int retval[] = new int[4];
retval[0] = R0; retval[1] = R1; retval[2] = L0;
retval[3] = L1;
return retval;
}
```

```
private void seedkey(byte[] keyBlock)
{
int A, B, C, D, T0, T1;
int[] key = new int[4];
key = bytesToInts(keyBlock, 0, 16);
// 128비트 입력키를 32비트 4개의 부분으로 분할
A = key[0]; B = key[1];C = key[2]; D = key[3];
// 첫 번째 입력으로 들어갈 1 라운드 키를 생성
T0 = A + C - (int)KC0;
T1 = B + (int)KC0 - D;

K[0] = SS0[(short)(T0 & 0x000000ff)] ^ SS1[(short)((T0>>>8)
& 0x000000ff)] ^SS2[(short)((T0>>>16)& 0x000000ff)] ^
SS3[(short)((T0>>>24)& 0x000000ff)];

K[1] = SS0[(short)(T1& 0x000000ff)] ^ SS1[(short)((T1>>>8)&
0x000000ff)] ^ SS2[(short)((T1>>>16)& 0x000000ff)] ^
SS3[(short)((T1>>>24)&0x000000ff)];

T0 =A;
A = (A>>8) ^ (B<<24);
B = (B>>8) ^ (T0<<24);
T0 = A + C - KC1;
T1 = B + KC1 - D;

K[2] = SS0[(short)(T0 & 0x000000ff)] ^ SS1[(short)((T0>>8) &
0x000000ff)] ^
        SS2[(short)((T0>>16) & 0x000000ff)] ^
SS3[(short)((T0>>24) & 0x000000ff)];

K[3] = SS0[(short)(T1 & 0x000000ff)] ^ SS1[(short)((T1>>8) &
0x000000ff)] ^
        SS2[(short)((T1>>16) & 0x000000ff)] ^
SS3[(short)((T1>>24) & 0x000000ff)];
---------- 중략 ----------
T0 =A;
A = (A>>8) ^ (B<<24);
B = (B>>8) ^ (T0<<24);
T0 = A + C - KC15;
T1 = B + KC15 - D;
K[30] = SS0[(short)(T0 & 0x000000ff)] ^ SS1[(short)((T0>>8)
& 0x000000ff)] ^
        SS2[(short)((T0>>16) & 0x000000ff)] ^
SS3[(short)((T0>>24) & 0x000000ff)];

K[31] = SS0[(short)(T1 & 0x000000ff)] ^ SS1[(short)((T1>>8)
& 0x000000ff)] ^
        SS2[(short)((T1>>16) & 0x000000ff)] ^
SS3[(short)((T1>>24) & 0x000000ff)];
// 생성된 암호 키와 복호화 키 저장
encryptKeys = K;
decryptKeys = SeedDecRoundKey(encryptKeys);
}
```

V. 결 론

인터넷과 같은 컴퓨터 네트워크 기술의 급속한 발전은 이를 사용하는 사람들의 생활방식에 많은 영향을 미쳐 기업 간 거래 또는 기업과 소비자 간의 거래를 종래의 직접 구매방식에서 인터넷을 이용하는 전자적 거래 방식으로 바꾸어 놓았다. 그러나 이러한 거래는 사이버 공간을 통해 이루어지는 것이기 때문에 상대방의 신원을 확인할 수 없다는 점, 신용카드 번호와 같은 거래정보가 타인에게 누출될 우려가 있다는 점, 거래정보가 타인에 의해 위·변조될 수 있다는 점, 그리고 상대방이 거래 사실을 부인할 경우 이를 방지하기 어렵다는 점과 같은 문제점을 안고 있다. 따라서 이를 위한 해결책으로 암호화 기술과 전자서명 기술을 활용한 인증, 무결성, 비밀성, 부인방지 서비스 등이 등장하고 있다.

정보 보호를 목적으로 하는 인증 및 전자 서명 기능은 기존에 서로 만나서 행해야 했던 계약, 양방향 서명 등을 원격으로 그리고 안전하게 행할 수 있게 해준다. 그러나 이를 위해서는 개인 암호 키와 같은 비밀 정보를 안전하게 저장하고 관리해줄 수 있는 방안이 필요하므로 자바 카드와 같은 스마트카드를 인터넷에 활용하여 보안 문제를 해결하고자 하는 노력이 점차 활성화되고 있다. 본 논문은 이러한 연구에 조금이나마 도움이 되고자 자바 카드 기반의 SEED 알고리즘을 구현하였다.

▌참고문헌▐

[1] SUN Microsystems, Javacard 2.1.1 virtual machine specification 5, 18, 2000

[2] "128-bit Symmetric Block Cipher(SEED)", 한국정보통신기술협회, 1999.

[3] 최용락, 소우영, 이재광, 이임영, "통신망 정보 보호", 그린출판사, 1995.

[4] Bruce Schneier, "Applied Cryptography", John Wiley & Sons Inc., 1996.

[5] Elliotte Rusty Harold, "Java Network

Programing", O'REILLY, 1997.
[6] Feistel, H. "Cryptography and Computer Privacy" Scientific American, May 1973.
[7] Jonathan Knudesen, "Java Cryptography", O'REILLY, 1998.
[8] Scott Oaks, "Java Security", O'REILLY, 1998.
[9] William Stallings, "Cryptography and Network Security", Prentice-Hall, Inc. 1999.
[10] SUN Microsystems, "JavaCard 2.1 Specification", 1999.
[11] SUN Microsystems, "JavaCard Developer's Guide", 1999.