ECC/ECDSA 암호 알고리즘을 장착한 자바 카드 기반의 인증서 발행 방법

김은환*, 박미옥*, 전문석*
*숭실대학교 컴퓨터학과
e-mail:ehkim@cherry.ssu.ac.kr

Certificate Issureing Method based on JavaCard with ECC/ECDSA Cryptography Algorithms

Eun-Hwan Kim*, Mi-Og Park*, Moon-Seog Jun*
*Dept of Computer Science, Soongsil University

요약

최근 일반인들을 대상으로 인터넷 서비스가 보편화되면서 인터넷 뱅킹이나 전자 상거래등을 통해 구매정보나 지불 정보 혹은 개인 신상 정보 등과 같은 중요한 정보의 전송에 보안의 중요성이 더욱 증대되어지고 있는 실정이다. 본 논문에서는 자바 카드와 ECC/ECDSA 알고리즘을 소개하고, 현재 자바 카드에서 지원하지 않는 ECC/ECDSA 알고리즘을 설계하여 자바 카드에 장착하였다. 그러므로 자바 카드를 이용하여 중요한 정보를 암호화/복호화 할 수 있도록 구현하였다. 또한, 인증서에서 사용하는 키들을 자바 스마트 카드를 사용하여 제공하는 방법을 제안하였다.

1. 서론

최근 일반인들을 대상으로 인터넷 서비스가 보편화되면서 인터넷이나 통신망을 통해서 중요한 데이터나 전자 상거래를 위한 개인 신용 정보 등을 교환하는 경우가 많아졌다. 그러므로 인터넷을 통한 중요한 정보의 전송에 보안의 중요성이 더욱 증대되어지고 있으며, 최근 이러한 보안 관리 대상이 되는 정보들을 안전하게 보관하고 관리할 수 있는 방법이 스마트 카드 시스템이다. 스마트 카드는 카드 내부에 프로세서와 메모리를 갖고 있기 때문에 중요한 정보를 저장하여 휴대를 할 수 있다[2].

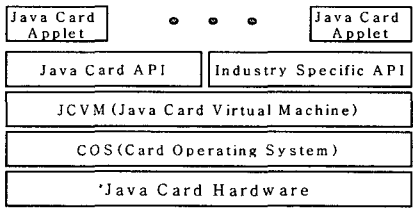
자바 카드는 스마트 카드 기술에 자바 기술을 접목시킨 것이다. 자바 카드는 하드웨어와 밀접한 연관이 되는 COS(Card Operating System)위에 JCVM(Java Card Virtual Machine)이 탑재되며, 그 위에 각종 API를 마련하여 사용자로 하여금 애플릿을 다운로드하여 사용할 수 있게 만든 것이다.

본 논문에서는 이러한 자바 카드 환경에서 구현되어 있지 않은 ECC/ECDSA 암호 알고리즘을 설계하고 구현하였으며, 실제 PKI를 이용한 인증서 발행에 대한 제안과 대부분 인증서에 사용되는 공개키인 RSA 알고리즘과 비교 분석했다. 논문의 구성은 2장에서 자바 카드와 타원곡선 암호 알고리즘에 대해서 설명하고 3장에서는 설계 및 구현에 대해서 논한다. 4장에서는 PKI에 사용할 인증서 발행을 제안하고, ECC/ECDSA 암호 알고리즘과 RSA 암호 알고리즘을 비교 분석하고 5장은 결론 및 향후 과제에 대해서 설명한다.

2. 관련 연구

2.1 자바 카드

자바 카드란 스마트 카드 기술을 기반으로 자바의 기술을 접목시킨 형태를 말한다. 전체적인 자바 카드의 구조는 (그림 1)과 같이 나타낸다[1].



(그림 1) 자바 카드 구조

위의 (그림 1)에서와 같이 자바 카드는 COS(Card Operating System)기반에서 JCVM(Java Card Virtual

Machine)이 장착되어 있는 구조의 스마트 카드를 말한다. 응용 프로그램을 개발하기 위해서는 자바 카드 애플릿을 작성하고 Java Card API나 Industry Specific API를 활용하여 JCVM상에서 수행한다.

자바 카드의 특징은 플랫폼의 독립성, 보안성 및 객체 지향성과 같은 자바 언어의 특징을 그대로 지니고 있다. 스마트 카드의 관점에서 살펴볼 때 다중 응용 프로그램을 다운 받아 여러 가지 용도로 사용할 수 있으며 ISO 7816과 같은 국제 표준과의 호환성을 제공하고 있다.

개발자는 애플릿을 생성하고 컴파일 하여 자바 카드에 다운로드하고 다운로드 된 애플릿과의 모든 통신은 apdutool을 사용하여 행하게 된다. 통신은 명령(Command) APDU와 응답(Response) APDU로 행해진다. 다음 (그림 2)는 APDU구조를 나타낸 것이다[3].

명령(Command) APDU

CLA	INS	P1	P2	Lc	Data	Le

응답(Response) APDU

Response Data	SW1	SW2

(그림 2) APDU의 구조

CLA는 자바 카드가 apdutool로부터 명령 APDU를 받았을 때 수행할 애플릿을 의미한다. INS는 명령어의 종류를 의미한다. P1과 P2는 INS가 가리키는 명령어들에 대한 파라메터를 나타낸다. Lc는 명령 APDU의 데이터의 길이를 의미하고, Le는 응답할 데이터의 길이를 나타낸다. 응답 APDU에서는 명령 APDU로부터 명령을 수행하고 결과를 Response Data에 저장하고 SW1의 값이 90, SW2의 값이 00이면 명령이 성공적으로 수행했다는 것을 의미한다.

2.2 ECC 암호 알고리즘

타원 곡선 암호 시스템(Elliptic Curve Cryptosystem, ECC)은 1985년 Neil Koblitz와 Victor Miller가 타원곡선 상의 점들에 대한 이산 대수 문제에 기반을 두고 각자 독립적으로 제안하고 있는 암호 시스템이다.

타원 곡선은 임의의 유한체(finite Field) F_p-상에서 정의하고 있다.

$\{(x,y) | y^2 \bmod p = (x^3+ax+b) \bmod p\} \cup O$

여기에서 $a, b \in Fp$를 만족해야 하며 또한 $4a^3+27b^2 \neq 0 \bmod p$인 상수이어야 한다. 그리고 타원곡선의 무한 원점(point at infinity)이라고 하는 원소 O를 포함한다. 타원곡선 암호 알고리즘의 성질은 다음과 같다.

1. $P + O = O + P = P(\forall P \in E(Z_p))$
2. $P=(x,y) \in E(Z_p)$일 때,
   $P - P = O$, $(x,y) + (x,-y)=0$
3. $P=(x_1,y_1) \in E(Z_p)$, $Q=(x_2,y_2) \in E(Z_p)$ 이고 $<P \neq Q>$ 일 때, $P+Q=(x_3,y_3)$
   여기서, $x3= \lambda 2 - x1 - x2$
   $y3= \lambda (x1 - x3) - y1$ 이고,

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$$

2.3 ECDSA 암호 알고리즘

ECDSA(Elliptic Curve Digital Signature Algorithm)는 DSA(Digital Signature Algorithm)를 타원 곡선 알고리즘으로 변형시킨 것이다[4].

다음은 ECDSA 암호 알고리즘의 동작 과정이다.

(1) 키 생성

Entity A는 키를 생성한다.

㉠ Z_p에서 정의한 타원곡선 E를 선택한다. $E(Z_p)$는 상당히 큰 소수 n에 의해서 나누어져야 한다.

㉡ order n인 점 $P \in E(Z_p)$를 선택한다.

㉢ 구간 [2, n-2]에서 통계적으로 유일하고 예측 불가능한 정수 d를 선택한다.

㉣ $Q = dP$를 계산한다.

㉤ 그러므로 Entity A의 공개키는 (E, P, n, Q)가 되고, 비밀키는 d가 된다.

(2) 서명 단계

메시지 m에 Entity A는 서명을 한다고 가정한다.

㉠ 구간 [2, n-2]에서 통계적으로 유일하고 예측 불가능한 정수 k를 선택한다.

㉡ $kP = (x_1, y_1)$을 계산하고 $r = x_1 \bmod n$을 계산한다. 이때, r=0이면 ㉠단계로 되돌아간다.

㉢ $k^{-1} \bmod n$을 계산한다.

㉣ $e = h(m)$으로 메시지를 해쉬한다. 해쉬 함수는 보통 SHA-1을 사용한다.

㉤ $s = k^{-1}\{e + dr\} \bmod n$을 계산한다. 이때, s=0이면 ㉠단계로 되돌아간다.

㉥ 메시지 m에 대한 서명은 (r, s)가 된다.

(3) 서명 검증 단계

Entity B는 Entity A의 서명 (r, s)을 검증한다고 가정한다.

㉠ Entity A의 인증된 공개키 (E, P, n, Q)를 얻는다.

㉡ r과 s가 구간 [1, n-1]의 범위에 있는지 확인한다.

㉢ $w = s^{-1} \bmod n$과 $e = h(m)$을 계산한다.

㉣ $u_1 = ew \bmod n$과 $u_2 = rw \bmod n$을 계산한다.

㉤ $u_1P + u_2Q = (x_2, y_2)$를 계산하고, $v = x_2 \bmod n$을 계산한다.

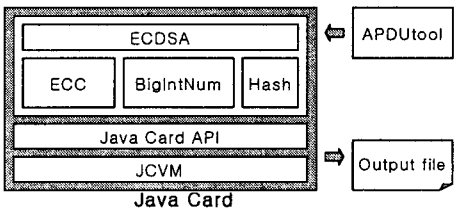
㉥ $v = r$이면 Entity A의 메시지 m에 대한 서명임을 확인한다.

3. 설계 및 구현

본 장에서는 자바 카드에서 사용 할 ECC/ECDSA 암호 알고리즘을 애플릿으로 작성하였다. 또한, ECC 암호 알고리즘을 구현하기 위해서 충분히 큰 수를 처리하고 저장할 수 있는 데이터 타입인 BigIntNum 클래스를 별도로 개발하여 API형태로 사용하였다.

3.1 개발 시스템 구성 및 실험

ECC/ECDSA 암호 알고리즘을 다운로드하여 수행하려면 실제 자바 카드와 CAD등이 있어야 하지만 Sun에서 제공하는 JavaCard Development Kit를 이용하여 개발 환경을 구성하였다. 전체적인 시스템 구성은 (그림 3)과 같다.



(그림 3) 시스템 구성

(그림 3)에서 자바 카드 부분에 해당하는 것은 실제 시스템의 메모리 상에서 동작하게 된다. ECC, ECDSA, BigIntNum, Hash는 애플릿 형태로 프로그램 하여 자바 카드에 다운로드 했다. 구현한 애플릿은 JCVM상에서 자바 카드 API를 활용하여 동작한다. 전체적인 동작은 SUN에서 제공하는 APDUtool이라는 툴킷을 이용하여 자바 카드에 명령을 한다. 이 명령을 받아 애플릿이 동작하고 그 결과를 Output file형태로 출력한다.

3.2 BigIntNum 클래스

ECC 암호 알고리즘을 자바 카드에서 수행하기 위해서는 충분히 큰 수를 처리 할 수 있는 각종 연산 메쏘드와 모듈러 연산등을 수행하는 메쏘드들이 필요하다. 이런 상당히 큰 수를 처리하는 BigIntNum 클래스와 메소드를 개발하여 자바 카드에 API 형태로 탑재하여 사용했다. BigIntNum 클래스는 자바에서 제공하는 BigInteger 클래스를 자바 카드에 맞게 재 구성한 것이다.

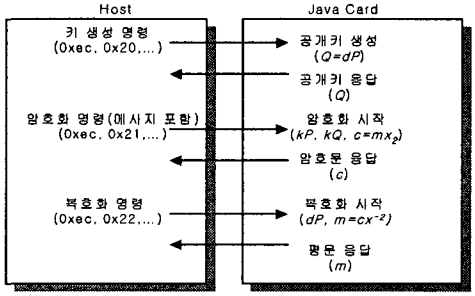
3.3 실험 및 결과

ECC/ECDSA 암호 알고리즘을 실험하기 위해서는 자바 카드의 특성을 고려하여 몇 가지 가정을 만들었다. 첫째, 자바 카드 내에서 사용하는 비밀키를 상수화 했다. 비밀키는 상당히 큰 값을 임의로 선택했다. 둘째, 타원곡선 값을 임의의 값으로 자바 카드 내에 저장했다. 셋째, 타원 곡선 상의 한 점 값인 P 값을 상당히 큰 소수를 선택하여 자바 카드 내에 저장했다. 일반적으로 자바 카드는 휴대가 가능하기 때문에 타원곡선 암호 알고리즘을 수행하기 위한 몇 가지 파라메터들을 자바 카드 내에 위치해 놓음으로써 안전하게 사용할 수 있다.

실험은 ECC 암호 알고리즘을 이용한 암호/복호화 과정과 전자 서명을 위한 ECDSA 알고리즘에 대한 검증 절차를 따로 만들어서 진행했다.

(1) ECC를 이용한 암호화/복호화 과정

다음 (그림 4)는 ECC 암호 알고리즘을 이용한 데이터 암호화/복호화 과정이다.



(그림 4) 암호화/복호화 과정

앞에서도 언급했듯이 타원곡선 E, 곡선 위의 임의의 한 점 P와 비밀키(d)는 이미 자바 카드 안에 저장되어 있는 상황이기 때문에 암호화 복호화 과정은 간단하게 수행된다.

(2) ECDSA의 전자 서명과 검증 과정

다음 (그림 5)는 ECDSA 전자 서명 알고리즘을 이용해서 데이터에 대한 전자 서명과 검증 과정이다.

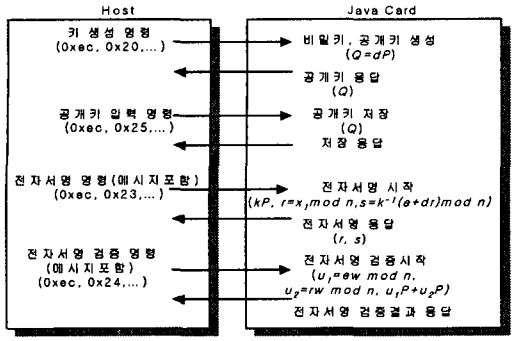
앞에서도 언급했듯이 타원곡선 E, 곡선 위의 임의의 한 점 P와 비밀키(d)는 이미 자바 카드 안에 저장되어 있는 상황이기 때문에 전자 서명과 전자 서명 검증 과정은 간단하게 수행된다.

전자 서명을 위해서 메시지 원본과 상대방의 공개키(Q)를 함께 자바 카드로 전송하여 전자 서명을 하고, 다시 메시지와 함께 전자 서명을 자바 카드로 전송하여 전자 서명을 검증하였다.

4. 활용 및 보안성 제안

자바 스마트 카드를 활용하여 응용 부분에 이용할 경우에 가장 큰 장점은 다음과 같다.

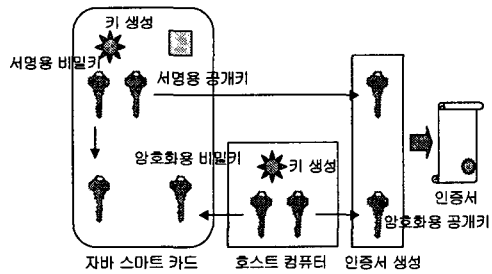
첫째는 키 생성 기능이다. 보통은 키 생성을 일반 컴퓨



(그림 5) 전자 서명/검증 과정

터에서 생성하여 비밀키를 저장하고 공개키를 일반에 공개한다. 나쁜 의도에 의해서 저장된 비밀키가 공개될 수도 있지만, 자바 스마트 카드는 키의 생성을 카드안에서 수행하기 때문에 비밀키의 누출에 대한 걱정이 없다.

둘째는 PKI등을 이용하여 자바 스마트 카드를 사용하는 경우 비밀키들과 인증서를 카드안에 입력하고 휴대할 수 있다는 점이다. 그러므로 어느곳에서든 비밀키들과 인증서를 이용하여 인증을 하고 필요한 작업을 수행할 수 있다. 본 연구에서 (그림 6)에서와 같이 ECC/ECDSA를 장착한 자바 스마트 카드를 이용하여 PKI 인증서에 사용할 키들에 대해 제안한다.



(그림 6) 인증서 생성 과정

서명용 키는 스마트 카드 내부에서 만들어진다. 서명용 비밀키는 스마트 카드 안에 존재하고 서명용 공개키는 스마트 카드 밖으로 인출되어 인증서를 만드는 재료로 사용된다.

암호화용 키는 스마트 카드 외부에서 만들어진다. 암호화용 비밀키는 외부에서 만들어진 후 스마트 카드 안으로 입력되어 카드 안에 저장하고, 암호화용 공개키는 인증서를 만드는 재료로 사용된다.

또한, 현재 사용하고 있는 인증서의 대부분은 공개키 암호 알고리즘인 RSA와 서명용 알고리즘인 DSA/DSS를 이용하여 인증서를 만든다. <표 1>은 ECC와 ECDSA를 이용하여 제안하고 있는 시스템을 사용하여 만든 인증서와의 비교 분석 결과이다.

〈표 1〉 인증서 생성 방식의 비교

구분	기존 인증서 발행	제안한 인증서 발행
키 생성	호스트 컴퓨터	자바 스마트 카드 /호스트 컴퓨터
서명용키 복구	복구 가능	복구 불가능
암호용키 복구	복구 가능	복구 가능
이동성	휴대 불가능	휴대 가능
키 길이	1024비트(RSA)	160비트(ECC)
인증서 크기	약 500 바이트(RSA)	약 300 바이트(ECC)
사용처	인터넷 웹	인터넷 웹/모바일 웹
추가 장비	없음	카드 리더 필요

서명용 키의 복구에 있어서 제안한 인증서는 비밀키의 생성을 자바 스마트 카드 내에서 생성했기 때문에 실제 카드의 분실로 인해서 키가 손실된 경우에는 키의 복구가 불가능하다. 그렇지만 암호용 키에 대해서는 중요한 문서일 경우 복호가 가능하게 하기 위해서 키의 생성을 호스트 컴퓨터에서 수행하고 자바 카드에 비밀키를 저장하여 사용하기 때문에 카드를 분실할 경우 호스트 컴퓨터에 저장된 비밀키로 복구가 가능하다.

제안한 인증서 시스템에서 단점으로 지적할 수 있는 것은 실제 스마트 카드를 사용할 경우 컴퓨터마다 카드를 읽을 수 있은 카드 리더가 필요하다. 그러므로 추가적인 비용이 소용된다.

5. 결론

본 논문에서는 스마트 카드 형태인 자바 카드를 이용하여 ECC/ECDSA 암호 알고리즘을 구현하고 PKI 인증서에서 사용할 키들에 대해 제안하고 비교 분석했다.

ECC/ECDSA 암호 알고리즘은 전자 상거래나 인터넷 보안을 위해서 스마트 카드와 같은 응용 도구를 사용하는 경우에 적용할 수 있으며 보다 높은 안전성을 제공할 수 있다.

스마트 카드와 같은 형태로 사용 할 경우 하드웨어로 구현하여 보다 빠르고 안전하게 사용 할 수 있도록 수정, 보완되어야 한다.

참고문헌

[1] Chen, Zhiqun, "Java Card Technology for smart Cards", Addison-Wesley, 2000.

[2] W. Rankl, W. Effing, "Smart Card Handbook", John Wiley & Sons, Ltd., 2000.

[3] http://java.sun.com/products/javacard/

[4] ANSI X9.62, "The Elliptic Curve Digital Signature Algorithm(ECDSA)", Working draft, October 1997.