비대칭키 암호 알고리즘을 고속으로 수행하는 자바카드 구현 및 성능 평가

김 호 원, 최 용 제, 김 무 섭, 박 영 수
한국전자통신연구원 IC카드구조연구팀
전화 : 042-860-6228

Design of JavaCard with enhanced Public Key Cryptograhy and its performance Evaluation

Ho Won Kim, Yong Je Choi, Moo Seop Kim, and Young Soo Park
Electronics and Telecommunication Research Institute
E-mail : khw@etri.re.kr

Abstract

In this paper, we present the development of a JavaCard for public key crypto algorithms and its performance evaluation. To make a high performance for the public key crypto algorithm such as RSA and ECC on a JavaCard, we have implemented a crypto coprocessor in hardware and ported it to the card operating system and virtual machine environments. The performance of the public key crypto algorithms on the JavaCard shows that our JavaCard is suitable for IC card applications which needs high performance and high level of security.

I. 서론

최근 인터넷과 무선 통신과 같은 세계적인 통신 네트워크 기술의 발전으로 업무 및 일상 생활 부문은 많은 변화를 겪게 되었다. 특히, 기존의 오프 라인 업무 수행 방식은 온라인 업무 수행 방식으로 변화되고 있으며, 네트워크로 연결된 디지털 정보에 대한 의존성도 더욱 높아져가고 있다. 이에 디지털 정보에 대한 기밀성(confidentiality), 무결성(integrity), 상호 인증(authentication) 유지가 중요한 이슈로 등장하고 있다. 이를 위해서, 최근 대칭키 및 비대칭키 암호 알고리즘, 보안 프로토콜, 전자 상거래 모델 등에 관한 많은 연구가 수행되고 있다. 특히, IC카드(스마트카드)는 휴대하기 쉽고 사용하기 용이하며, 보안성이 우수한 특성이 있기 때문에, 보안 분야의 핵심 요소로 자리잡고 있다. 하지만, 기존의 IC카드는 카드 벤더 및 시스템 벤더에 따라, 서로 다른 카드와 터미널을 가지고 있기 때문에, 상호간에 호환성이 부족하다는 단점을 가진다[1]. 이러한 비호환성 문제를 해결하기 위해서 개방형 플랫폼(open platform) 카드 기술이 개발되고 있으며, 자바카드(Java Card)는 개방형 IC 카드 플랫폼의 대표적인 예이다[2].

개방형 특성을 가지는 자바카드는 단일 카드로 전자 화폐나 선불 카드, GSM SIM 카드, 뱅킹 카드, 보건증 등과 같이 다양한 용도로 사용할 수 있다. 자바카드는 자바 언어를 사용하므로, 기존의 특정 플랫폼에 의존적인 IC카드와는 달리, 쉽게 프로그램 할 수 있고, 발급후에도 새로운 응용 프로그램을 구현할 수 있다는 장점을 가진다.

자바카드는 자바카드 가상 머신(JCVM: JavaCard Virtual Machine)의 바이트코드 인터프리터(byte code interpreter)를 통하여, 원하는 명령을 수행한다. 이는 특정 프로세서에 최적인 실행 코드를 미리 발생한 후, 수행하는 컴파일러 방식보다 성능이 떨어진다는 단점

을 가진다. 또한, 최근 IC카드는 많은 연산을 필요로하는 공개키 암호 알고리즘을 필요로 하는 전자화폐 혹은 PKI(Public Key Infrastructure)의 응용에 많이 사용되므로, 공개키 암호 알고리즘을 고속으로 수행해야 할 필요가 있다.

본 연구팀이 개발하는 IC 카드는 32비트 RISC 타입 프로세서를 기반으로 하고 있으며, 암호 전용 처리 하드웨어를 가지기 때문에, RSA와 ECC와 같은 공개키 암호 알고리즘, Triple-DES, SEED, Rijndael과 같은 비밀키 암호 알고리즘, SHA-1과 MD5와 같은 서명/해쉬 함수를 고속으로 수행할 수 있다.

본 논문은 개방형 IC카드 플랫폼인 자바카드에서 RSA 및 타원곡선 암호 알고리즘(ECC:Elliptic Curve Cryptography)과 같은 비대칭키 암호 알고리즘을 고속으로 수행할 수 있는 하드웨어/소프트웨어 구현 및 성능 평가에 관한 연구이다. 현재 시판되고 있는 자바카드는 대부분 DES와 같은 대칭키 암호 알고리즘을 사용하여, 암호화 및 서명, 인증 등에 사용하였으나, EMV 및 CEPS 등과 같은 전자화폐 규격에서도 볼 수 있듯이, 안전도(level of security)를 강화하기 위해서 RSA 및 ECC와 같은 비대칭키 암호 알고리즘 수용이 불가피하다. 본 논문에서는 비대칭키 암호 알고리즘을 고속으로 수행하는 코프로세서를 개발하였으며, 개발된 코프로세서를 한국전자통신연구원에서 개발중인 자바카드 가상 머신(Java Card Virtual Machine), 암호 API(crypto API), 암호프로토콜 같은 자바카드와 연동하여 동작 및 성능을 검증한다.

II. 자바카드 개요

자바카드는 자바 프로그램 언어를 수행할 수 있는 스마트카드다[2]. 현재 자바카드 2.1 사양이 선 마이크로시스템사의 홈페이지에서 구할 수 있다[3]. 이 곳에서 자바카드 가상 머신에 관한 규격과 API에 관한 규격을 구할 수 있다.

자바카드 가상 머신은 오프카드(off-card)와 온카드(on-card)상에서 수행되는 부분으로 나눠진다. 클래스 로딩이나, 바이트코드 검증, 링크 등과 같이 수행 시간이 중요한 요소로 고려되지 않는 기능들은 카드 단말기나 PC, 워크스테이션 등과 같은 오프카드상에서 수행되며, 바이트코드 인터프리터 기능은 온카드상의 JCRE(JavaCard Runtime Environment)에서 수행 된다.

일반적으로 작은 크기의 메모리와 저성능 프로세서를 가지는 IC카드의 특성과 IC카드 응용 프로그램의 특성 때문에, 자바카드는 자바 언어가 지원하는 모든 특성을 지원하지 않는다. 예를 들어, long형이나 double형, float형. string형, 다차원 배열과 같은 데이터 타입과 동적 클래스 로딩, 쓰레드(thread), garbage collection 등과 같은 기능을 지원하지 않는다[3]

자바카드와 카드 단말기간의 통신은 ISO 7816 표준에 기술되어져 있다. 통신 메시지는 APDU(Application Protocol Data Unit) 프로토콜 형태로 전송되며, 자바카드의 응용 프로그램인 애플릿은 특정 라이프사이클(life cycle)을 가진다[2].

자바카드 애플릿은 VisualCafe나 JDK와 같은 기존의 자바 프로그램 개발 툴을 사용하여, class 파일을 만든 후, 자바카드 개발 툴에서 제공하는 converter를 사용하여, CAP 파일을 생성한다. CAP 파일은 자바카드에 다운로드 되며, 다운로드 된 CAP 파일은 카드내의 JCRE에서 수행된다.

III. 공개키 암호 알고리즘을 고속으로 수행하는 자바카드

그림 1은 본 연구팀에서 개발중인 자바카드의 구조를 보여주고 있다. 이는 프로세서와 메모리, I/O 모듈, 암호 프로세서 등과 같은 기반이 되는 하드웨어와 COS(Card Operating System), JCVM, API, Applets 등과 같은 소프트웨어로 구성된다. RSA와 ECC 비대칭키 암호 알고리즘을 고속으로 수행하기 위한 암호 프로세서는 COS내의 device driver에 의해 인식되며, COS내의 native function에 의해 제어된다. 이는 JCVM에 의해서 API와 연결되는데, 암호 프로세서와 관련된 API는 키 설정 및 데이터 암/복호화 지정, 입출력 데이터 설정 등이 있다.

자바카드 가상 머신은 특정 카드에 의존하는 하드웨어 및 COS에 대하여, 정형화된 인터페이스를 제공해 줌으로서, 자바카드의 개방형 특성을 제공한다. 자바카드 프레임워크는 응용 소프트웨어에서 필요로하는 API를 제공해준다. VOP(Visa Open Platform)과 EMV는 API 상위 계층에 구현되어, 카드의 운영 절차나 보안 모델, 뱅킹 서비스 등을 제공해 준다.

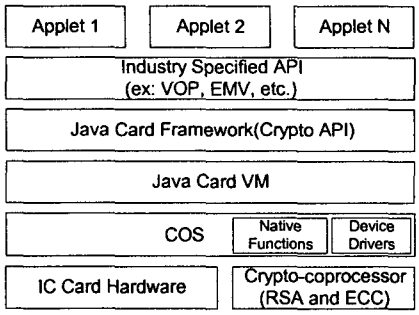

그림 1 비대칭키 암호 알고리즘을 고속으로 수행하는 개선된 자바카드의 구조

본 연구팀은 보안성이 강화된 IC카드를 개발하기 위해서, DES 및 SEED와 같은 비밀키 암호 알고리즘과 SHA-1, MD5 해쉬 함수를 소프트웨어로 구현하였다. 비밀키 암호 알고리즘은 공개키 암호 알고리즘과 달리, 소프트웨어로 구현했을 경우에도 IC카드 응용 소프트웨어에서 만족할 만한 수준의 성능을 제공한다.

표 1은 자바카드에서 구현된 비대칭키 암호 알고리즘 규격을 보여 주고 있다. 이들은 카드상에서 하드웨어 모듈로 구현되었다. 또한 해당 native function과 device driver를 가진다. 이들은 자바 가상 머신을 통하여, crypto API에 연결되어 사용자 응용 소프트웨어에 사용된다. crypto API는 javacardx.crypto와 javacard.security 클래스에 의해 정의된다. RSA crypto API는 javacard 규격에 따라 구현되어졌다. 하지만, ECC인 경우에는 규격에 정의 되어 있지 않기 때문에, 본 연구팀에서 새롭게 정의하였다.

표 1 자바카드에서 하드웨어로 구현된 비대칭키 암호 알고리즘의 규격

알고리즘	비트 길이	규 격
RSA	1024	지수승 수준 RSA 동작 구현, Montgomery 알고리즘 사용, 지수 e=17에서의 성능 평가
ECC	163	다항식 기반 타원곡선, 스칼라 곱셈 수준 ECC 동작 구현, SEC2 제안 곡선 사용

표 2는 RSA 및 ECC crypto API와 이의 사용법을 보여주고 있다. 표에서 보는 바와 같이, ECC crypto API는 RSA crypto API와 유사한 구조를 가지는데, 이는 RSA 암호 프로세서와 ECC 암호 프로세서가 서로 비슷한 인터페이스 특성(RSA: exponentiation mode, ECC: scalar multiplication mode)을 가지기 때문이다.

표 2 RSA, ECC용 crypto API 형태 및 사용법

알고리즘	Crypto API usage

RSA	// Build Key Set
	publicRSAKey =
	KeyBuilder.buildkey(TYPE_RSA_PUBLIC, 1024, true);
	
	// Initialize RSA public key
	publicRSAKey.setModulus(n, testRSAOffset, 1024/8)
	publicRSAKey.setExponent(e, 0, e.length);
	.
	//Get cipher RSA instance
	cRSA_NO_PAD = Cipher.getInstance((byte)0x0C, false);
	
	// RSA Encryption with public key
	cRSA_NO_PAD.init(publicRSAKey,
	Cipher.MODE_ENCRYPT);
	cRSA_NO_PAD.doFinal(·);
	.
	// RSA Decryption with private key
	cRSA_NO_PAD.init(privateRSAKey,
	Cipher.MODE_DECRYPT);
	cRSA_NO_PAD.doFinal(·);

ECC	// Build Key Set
	ECC_k_Key =
	KeyBuilder.buildkey(TYPE_ECC_POLY, 163, true);
	
	// Initialize ECC parameters
	ECC_k_Key.setScalar_k (k, testRSAOffset, 163/8)
	ECC_k_Key.setP1(x1, y1);
	ECC_k_Key.setA2(A2_xval, A2_yval);
	.
	//Get cipher ECC instance
	cECC_NO_PAD = Cipher.getInstance((byte)0x0C, false);
	
	// ECC Encryption with k value
	cECC_NO_PAD.init(ECC_k_Key,
	Cipher.MODE_ENCRYPT);
	cECC_NO_PAD.doFinal(·);
	.
	// ECC Decryption with s value
	cECC_NO_PAD.init(ECC_s_Key,
	Cipher.MODE_DECRYPT);
	cECC_NO_PAD.doFinal(·);

Ⅳ. 성능 평가

본 연구팀이 개발하는 IC카드의 공개키 암호 알고리즘 동작 성능을 측정하기 위해서, ECDSA 프로토콜[5]을 구현하였다. ECDSA 프로토콜은 서명 생성 부분과 서명 검증 부분으로 나눌 수 있으며, 이들은 각각 알고리즘 1과 2에 나타나있다.

알고리즘 1. ECDSA 서명 생성 알고리즘
To sign a message m, a signer A does the following: 1. Select a random integer k from $[1, n-1]$ 2. Compute $kG^* = (x_1, y_1)$ and $r = x_1 \bmod n$ 3. Compute $k^{-1} \bmod n$ 4. Compute e = SHA-1(m) 5. Compute $s = k^{-1}\{e + d^{\dagger}r\} \bmod n$ If $s = 0$ then go to step 1 6. A's signature for the message m is (r,s)

* G is a base point on $E(GF(2^m))$
† d is a random integer from $[1, n-1]$ and A's private key

알고리즘 2. ECDSA 서명 검증 알고리즘
To verify A's signature (r,s) on m, a verifier B should do the following:
1. Verify that r and s are integers in $[1, n-1]$
2. Compute $e = SHA\text{-}1(m)$
3. Compute $w = s^{-1} \bmod n$
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$
5. Compute $u_1G + u_2Q^{\ddagger} = (x_1, y_1)$
6. Compute $v = x_1 \bmod n$
7. Accept the signature if and only if $v = r$

‡ $Q = dG$ is a A's public key

구현한 ECC 암호 알고리즘은 $GF(2^{163})$에서 정의된다. SEC2[6]에서 권장하는 기저 다항식($F(X) = X_{163} + X_7 + X_3 + 1$)과 곡선을 사용하였다. ECC를 구현할 때, 유한체 곱셈에서는 shift-and-add 방법을 사용하였고, 스칼라 곱셈에 대해서는 left-to-right binary 방법[8]을 사용하였다. 이 방법은 구현시 작은 크기의 하드웨어를 필요로 한다.

표 3 ARM7M 프로세서를 장착한 IC카드 프로세서와 암호 코프로세서를 장착한 자바카드상에서의 비대칭키 암호 알고리즘 성능 평가

알고리즘	프로토콜	ARM7M @33MHZ	암호 코프로세서 @20MHz	성능 향상 효과 (암호 코프로세서 /ARM7M)
RSA	암호화	1.8 sec	56msec	32
ECC	스칼라 곱셈	1.004sec	15.6msec	64.4
	ECDSA (signature generation)	1.032sec	16msec	64.5
	ECDSA (signature verification)	2.255sec	31.9msec	70.6

표 3은 ECDSA 프로토콜이 33MHz ARM7M 프로세서에서 동작하는 경우와 본 연구팀에서 개발한 비대칭키 암호 코프로세서에서 동작하는 경우에 대한 성능 측정 결과를 보여주고 있다. 표 3을 보면 알 수 있듯이, ECDSA 프로토콜 수행시, 대부분의 시간 지연은 스칼라 곱셈을 수행할 때(알고리즘 1에서 kG 연산, 알고리즘 2에서 $u_1G + u_2Q$ 연산) 발생한다는 것을 알 수 있다. ECDSA의 서명 검증시의 소요 시간은 서명 생성 때보다 약 2배 정도 시간이 더 소요된다는 것을 알 수 있다. ECDSA 프로토콜에서 modular reduction 단계와 inversion 계산 단계, SHA-1 수행 단계는 스칼라 곱셈에 비해 적은 시간을 소요한다.
20MHz로 동작하는 IC카드용 암호 코프로세서는 33MHz로 동작하는 RISC 형태의 ARM7M 프로세서보다 약 60배에서 70배 정도의 성능 향상 효과를 가진다.

V. 결론

본 논문에서는 RSA와 ECC를 고속으로 수행할 수 있는 자바카드 구현에 관한 내용이 기술 되어 있다. 본 연구팀에서 개발하는 자바카드는 많은 연산 시간을 필요로 하는 공개키 암호 알고리즘(RSA, ECC)을 고속으로 수행하는 암호 코프로세서를 가진다. 구현된 암호 코프로세서의 성능 측정 결과, 본 연구팀이 개발하는 자바카드는 높은 보안성을 필요로하는 다양한 IC카드 응용 부문에서 충분한 성능을 제공해 줄 수 있을 것으로 보인다.

참고문헌

[1] Reto Hermann Dirk Husemann, and Peter Trommler, *The OpenCard Framework* , proc. of the Third Smart Card Research and Advanced Application Conference, 1998.

[2] Zhiqun Chen, *JavaCard Technology for Smart Cards Architecture and programmers Guide*, Addison-Wesley, 2000.

[3] Sun Microsystems Inc., *JavaCard 2.1 Virtual Machine Specification*, available at http://java.sun.com/products/javacard

[4] ISO/IEC 7816-5: Integrated circuit(s) cards with contacts Part 5: Numbering system and registration procedure for application identifiers.

[5] Don B. Johnson, Alfred J. Menezes, and Scott Vanstone, *Elliptic curve digital signature algorithm(ECDSA)*, available at http://www.certicom.com

[6] Certicom Corp., *SEC 2: Recommended elliptic curve domain parameters*, September 2000, Version 1.0.

[7] TTA, *128-bit Symmetric Block Cipher(SEED)*, Telecommunication Technology Association(TTA), Seoul, Korea, June 1999.

[8] Darrel Hankerson, Julio Lopez Hernandez and Alfred Menezes, *Software Implementation of Elliptic Curve Cryptography over Binary Field*, CHES 2000, August, 2000.