2003 정보 및 제어 학술회의 (2003.11.21-22)

임베디드 시스템에서의 ECDSA(Elliptic Curve Digital Signature Algorithm) 구현

A Software Implementation of The Elliptic Curve Digital Signature Algorithm on a Embedded System.

김 현 익[1], 김 용 민[2], 정 석 원[3], 이 상 진[4], 정 창 훈[5]

[1] 고려대학교 정보보호 대학원(전화:(02)3290-4251 팩스:(02)928-9109 E-mail: kimhyunik@cist.korea.ac.kr)
[2] (주) 시큐어 텔레콤 (전화:(042)863-0053 팩스:(042)863-6433 E-mail: youngmink@securelink.co.kr)
[3] 고려대학교 정보보호 대학원(전화:(02)3290-4251 팩스:(02)928-9109 E-mail: jsw@cist.korea.ac.kr)
[4] 고려대학교 정보보호 대학원(전화:(02)3290-4251 팩스:(02)928-9109 E-mail: sangjin@korea.ac.kr)
[5] (주) 시큐어 텔레콤 (전화:(042)863-0053 팩스:(042)863-6433 E-mail: iqjump@securelink.co.kr)

Abstract : In this paper, after the crypto acceleration board of the server-termination type is designed, we implement the Elliptic Curve Digital Signature Algorithm on the board that serves data integrity and user authentication. For implementing ECDSA, we use crypto co-processor, MPC180, to reduce the computation burden of main processor (MPC860) on the board. By using crypto co-processor, the computation efficiency in case prime field is improved more between 90 and 100 times than the software library and between 20 and 90 times in case binary field. Our result is expect to apply for SSL acceleration board.

Keywords : crypto acceleration board, ECDSA, crypto co-processor.

Ⅰ. 서론

전자 기술 및 정보 통신 기술의 획기적인 진보로 인해 이전에는 누릴 수 없었던 다양한 서비스를 유, 무선 통신 채널을 통해 제공할 수 있게 되었다. 하지만 이러한 통신 채널은 누구나 접근할 수 있는 개방형 형태를 취하고 있기 때문에, 불법적인 침입자로 인한 보안상의 문제가 발생할 수 있으며, 이는 곧 유, 무형의 재산 피해로 이어질 수 있다. 따라서 권한이 없는 불법적인 침입자로부터 개인이나 조직이 소유한 정보를 보호하고, 통신 채널로 전송되는 데이터의 위, 변조 및 도청을 방지하기 위해 암호 시스템의 필요성이 크게 대두되고 있다. 또한 통신상에서 전송되는 문서나 기타 저작권에 속하는 음성 혹은 영상 데이터의 전송 시, 수신자의 신원을 확인하거나 정당성을 인증하여, 차후에 일어날 분쟁을 해결할 수 있는 메커니즘이 필요하게 되었다. 이는 암호 시스템을 이용한 서명 기법의 적용으로 차후에 일어날 분쟁의 해결책을 제공할 수 있게 되었다.

하지만 암호 시스템과 서명 기법은 수학적 이론에 근거한 복잡한 수식 연산을 요구하며, 자원 제약 측면에서 상대적으로 유리한 호스트 환경에서도 암호화를 위한 연산 수행은 큰 부담으로 작용하고 있다. 또한 수천 명의 사용자를 인증하고, 이들과의 통신을 수행해야 하는 웹 서버의 경우, 암호 시스템 및 서명 기법을 위한 연산은 전체 시스템의 효율 및 성능 저하로 이어질 수 있다,

본 논문에서는 서버 환경에서 각각의 클라이언트의 보안 요구 사항에 맞게 암호화 통신 및 서명을 이용한 인증을 별도의 하드웨어 모듈에서 수행하여 서버의 연산 부담을 줄여주는 server-termination[1] 방식의 암호 가속 보드의 설계 및 기능 검증을 수행하였다,

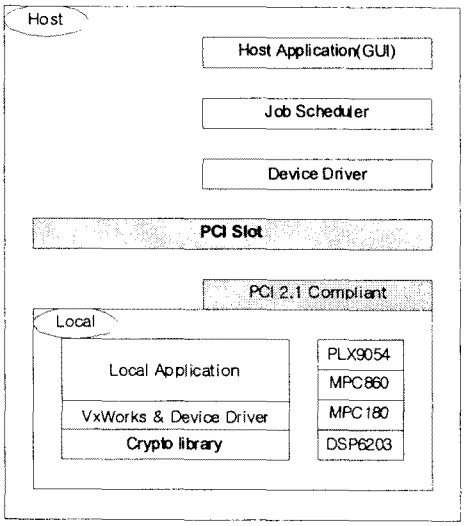
또한 암호 가속 보드 상에서 ECDSA(Elliptic Curve Digital Signature Algorithm)구현시 모토롤라사의 암호 보조 프로세서인 MPC180을 이용한 ECDSA(Elliptic Curve Digital Signature Algorithm)의 구현과 암호 가속 보드의 주 프로세서인 MPC860만을 이용한 소프트웨어 라이브러리와의 성능 비교를 통하여 최적 기능 구현을 위한 방안을 연구하였다.

Ⅱ. 암호 가속 보드의 구성

1. 전체 시스템 구성

암호 가속 보드의 기능 요구 사항은 키 생성, 서명 생성, 서명 검증의 서비스를 호스트 환경의 사용자에게 제공하는 것이다. 이러한 서비스를 제공하기 위해

암호 가속 보드는 크게 소프트웨어 구성 요소와 하드웨어 구성 요소로 설계되었다. 각 구성 요소는 내장형 컴퓨터(embedded computer system)가 갖추어야 하는 기본 요건[2]을 만족시키는 범위에서 구성되었으며, 통합 개발 환경을 제공한다.(그림 1)



(그림 1) 전체 시스템 구성.

호스트 어플리케이션은 시스템 성능 검증을 위해 설계되었으며, 네트워크 상에서 전달 받은 데이터 패킷은 job scheduler에 의해 전 처리 과정을 거친 후, 데이터에 대한 암, 복호 및 서명 생성 명령을 디바이스 드라이버를 통하여 암호 가속 보드에 전달하게 된다. 하드웨어 및 소프트웨어 구성 요소에 관한 내용은 다음 절에서 설명하도록 하겠다.

2. 하드웨어 구성 요소

암호 가속 보드의 하드웨어 구성 요소는 다음과 같다.

- PCI Ver2.1을 만족하는 인터페이스(PLX9054).
- PCI 로컬 영역의 서비스 분배 및 제어를 위한 프로세서(MPC 860)
- 암호 보조 프로세서(MPC180)
- 국내 표준 블록 암호 알고리즘을 지원하기 위한 fixed point DSP(TMS320C6203)

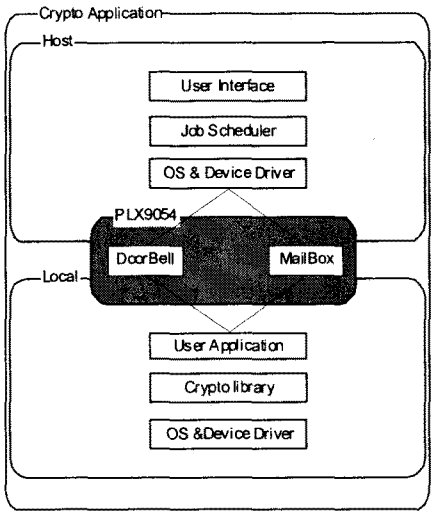
암호 가속 보드는 PCI 슬롯으로부터 전원을 공급받아 동작하며, PCI 2.1을 지원하는 로컬 마스터 칩인 PLX9054를 이용하여 PCI 로컬 버스를 구성하였다. 로컬 시스템의 OS로는 WindRiver사의 Vxworks를 사용하였으며, MPC860과 MPC180을 사용하여 호스트가 요청하는 암/복호 및 서명 동작을 지원하게 된다. 처리된 데이터의 결과는 PLX9054의 내부 레지스터를 이용하여 호스트에 전달한다.

3. 소프트웨어 구성 요소

소프트웨어 구성 요소는 다음과 같다.(그림 2)

- 호스트용 그래픽 사용자 인터페이스.(Application)
- 호스트용 PCI 디바이스 드라이버.
- 로컬용 실시간 운영체제 및 디바이스 드라이버(BSP)
- 로컬용 암호 연산에 사용되는 라이브러리(MPC180)

최상위 사용자 프로그램인 GUI가 있고, 여러 사용자의 요청들을 직렬화 시키는 job scheduler가 동작한다. job scheduler에 의해 직렬화된 사용자 패킷들은 디바이스 드라이버를 통하여, PLX9054의 Mailbox 레지스터에 설정된다. 이는 로컬 시스템에 인터럽트를 유발하며, MPC860은 BSP에 설정된 기능 요구 사항에 따라, MPC180으로 사용자 서비스를 위한 연산을 수행하게 된다. 처리된 결과는 PLX9054의 DoorBell 레지스터를 이용하여 호스트로 전송된다. 호스트의 job scheduler는 받은 데이터를 취합하여, 사용자가 요구한 서비스를 제공한 후, 한 세션을 마치게 된다.



(그림 2) 소프트웨어 구성 요소.

다음 장에서는 ECDSA를 구현하기 위해 필요한 타원 곡선 암호 알고리즘 설명과 MPC180이 제공하는 API를 이용한 서명 구현 시, 유의해야 할 사항들을 설명하겠다.

Ⅲ. 암호 가속 보드를 이용한 ECC 서명 구현

1. 타원곡선 암호시스템의 개요

1985년 Neal Koblitz와 Victor Miller는 타원곡선군을 이용한 공개키 암호 시스템을 제안하였다. 표수가 p(p>3)인 유한체 GF(p)에 대하여 타원 곡선 E는 다음과 같이 정의된다.

$$E: y^2 = x^3 + ax + b \quad (1)$$

여기에서 a, b ∈ GF(p)이고 $4a^3 + 27b^2 \neq 0$을 만족한다. 그리고 타원 곡선 군을 아래 식과 같이 정의한다.

$$E(GF(p))=\{(x, y) \mid y^2 = x^3 + ax + b\} \cup \{O\} \quad (2)$$

여기에서 O는 무한 원점을 뜻한다. 또한 표수가 2인 유한체 $GF(2^n)$에 대하여 타원 곡선 E는 다음과 같이 정의된다.

$$E: y^2 + xy = x^3 + ax^2 + b \quad (3)$$

여기에서 a, b ∈ GF(2)이다. 이에 대한 타원곡선군을 아래 식과 같이 정의한다.

$$E(GF(2^n))=\{(x, y) \mid y^2 + xy = x^3 + ax^2 + b\} \cup \{O\} \quad (4)$$

유한체를 F라 할 때, E(F)는 좌표들의 덧셈에 대하여 덧셈 군(additive group)을 형성하게 된다. 이 때 E(F)의 총 원소의 개수를 타원 곡선 E의 차수(order)라 부르며 #E라 표기한다. 좌표 P에 대한 정수 k와의 스칼라 곱셈(Scalar Multiplication)은 좌표 P를 k번 더하는 것으로 정의된다. 수식으로 표현하면 아래 식과 같이 정의된다.[3]

$$Q = [k]P = P + P + \cdots + P + P \quad (5)$$

P는 커브상의 좌표이며, k는 $1 \leq k \leq ord(P)$의 범위를 가진다.

타원 곡선 암호 시스템의 안전성은 P와 [k]P가 주어졌을 때 k를 알아내는 것의 어려움에 바탕을 두고 있으며, 이를 ECDLP(Elliptic Curve Discrete Logarithm Problem)라 한다.[4]

2. 타원 곡선을 이용한 서명 알고리즘 구현

타원 곡선을 이용한 전자 서명은 크게 시스템 파라메터 설정, 키 생성, 키 분배, 서명 생성, 서명 검증으로 구성되며, 본 논문에서는 키 생성과 서명 생성 및 검증에 필요한 기능들을 소프트웨어 및 하드웨어 라이브러리로 구현하였다. 시스템 파라메터 설정에 필요한 도메인 상수(domain parameter)는 WTLS, SEC2, X9.62에서 권고한 커브의 도메인 상수를 사용하였으며, 키 생성, 서명 생성, 서명 검증의 알고리즘은 X9.62를 참조하였다.[5]

우선 MPC180을 이용한 스칼라 곱셈 구현 시 유의해야 할 사항은 정확한 기능 레지스터 설정과 조건에 맞는 데이터 입력이다. 각 라이브러리의 연산 모듈은 데이터 길이를 일괄적으로 고정하여 입력 데이터를 레지스터에 저장하기 때문에, 길이 고정값은 법(modulus)의 크기로 선택하는 것이 바람직하다. 길이 설정은 16비트를 하나의 단위로 인식하여 설정해준다. 연산은 사영좌표계(projective coordinate system)상에서 진행되므로, 최종 연산 결과 값도 사영좌표계에 존재하게 된다. 따라서 이를 원래의 입력 좌표가 존재하는 affine 좌표계로 변환시켜 주는 연산이 필요하며, 소수 체와 이진 체에 대해서 각각 두 번의 역원 연산이 필요하게 된다. 역원 연산을 위한 함수는 소프트웨어 라이브러리를 사용하였다.

스칼라 곱셈 시 필요한 입력은 base point 좌표, 도메인 파라메터 a와 b, 법 n, R^2 (mod n)가 필요하며, R^2값이 필요한 이유는 각 연산 모듈이 몽고메리 타입의 연산 기능을 수행하기 때문이다. R^2 (mod n)의 값을 구하기 위해 소수체의 경우 mpc180PkhaR2() 함수를 이용하여 데이터를 구할 수 있지만 이진 체에 대해선 기능을 제공하고 있지 않다. 대신 모듈러 곱셈 연산 함수를 이용하여 이진 체 상에서 R^2(mod n)을 구할 수 있는 함수를 구성하였으며, 함수 구성의 대략적인 개요는 다음과 같다.

```
S_INT COM_R2MODN_GF2N
(PKHA_SUBNUM MOD,
PKHA_SUBNUM R2MODN,
IRR *n,
S_INT length)
{
  ……..
  for(i=0;i<=length;i++){
    if(i == length){
        A[i] = 0x00000001;
        B[i] = 0x00000001;
    }
    else
    {
        A[i] = 0x00000000;
        B[i] = 0x00000000;
    }
  }
……..
mod_size = (U_LONG)(2*(length+1));
mpc180PkhaClearMemory();
mpc180PkhaLoadModSize(mod_size);
mpc180PkhaLoadSubReg(MOD,PKHA_REGN0);
mpc180PkhaLoadSubReg(A,PKHA_REGA0);
mpc180PkhaLoadSubReg(B,PKHA_REGB0);
mpc180EccModularMultiply2(0,0,0,PKHA_F2M);
mpc180PkhaReadSubReg(R2MODN,PKHA_REGB0);
……..

}
```

(그림 3) 함수 구성 예

변수 A와 B를 법의 사이즈에 따라 각각 1로 세팅 한 후 모듈러 곱셈 연산을 수행하면 $1/R^2$을 결과 값으로 얻을 수 있다. 결과 값을 이용하여 한 번의 역원 연산을 수행하면 스칼라 곱셈에 필요한 R^2(mod n)을 얻을 수 있게 된다.

mpc180PkhaR2() 함수의 기능을 사용할 때 유의해야 할 사항은 법의 레지스터 저장 시 상위 16비트가 모두 0이면 계산상의 오동작으로 잘못된 연산 결과를 출력한다는 점이다. 정확한 연산을 위해 법의 상위 16비트가 0일 경우를 처리해 줄 수 있는 에러 처리 루틴을 고려하여 구현을 하였다. 또한 모듈러 곱셈, 덧셈 연산을 위한 라이브러리를 사용할 경우에도 각 입력 데이터가 법보다 크지 않아야 하며, 이 경우에도 동일하게 잘못된 연산 결과 값이 산출된다.

키 생성, 서명 생성, 서명 검증 구현 시 필요한 난수 생성기(random number generator)는 MPC180이 제공하는 하드웨어 라이브러리를 사용하였다. 이 난수 생성기는 LFSR(Linear Feedback shift register)와 CASR(Cellular automata shift register)이 병렬로 동작하여 32비트 단위

로 난수를 생성하며, 사용자가 요청한 크기에 따라 난수를 생성한다. 난수 생성은, 타원 곡선 위수의 길이보다 8바이트 정도 큰 난수를 생성한 후, 타원 곡선 위수로 모듈러 연산을 수행한 결과 값을 사용하였다. 이는 난수 생성에 임의성을 부가하기 위해서 법보다 긴 수를 선택할 필요가 있기 때문이다.

3. 구현 결과

서명 생성 및 검증의 가장 기본적인 로직이라 할 수 있는 스칼라 곱셈의 소프트웨어 구현과 MPC180이 제공하는 하드웨어 라이브러리를 사용한 구현 결과의 성능 분석을 아래 표에서 비트 사이즈 별로 비교 분석 하였다.

비트 수	Hardware(ms)	Software(ms)
112	48	3312
192	96	5856
256	128	8176

[표 1] 소수 체에서의 스칼라 곱셈 연산 속도 비교

비트 수	Hardware(ms)	Software(ms)
113	16	208
239	32	576
409	64	2080

[표 2] 이진 체에서의 스칼라 곱셈 연산 속도 비교

소수 체(prime field)에서는 하드웨어 라이브러리를 사용하여 약 60~80배의 성능 개선이 이루어졌으며, 이진 체(binary field)에서는 10~30배의 성능 개선이 이루어졌다.

다음으로 서명 생성, 서명 검증의 소프트웨어 구현과 하드웨어 라이브러리를 사용한 구현 결과의 성능을 각 비트 사이즈 별로 비교 분석 하였다.

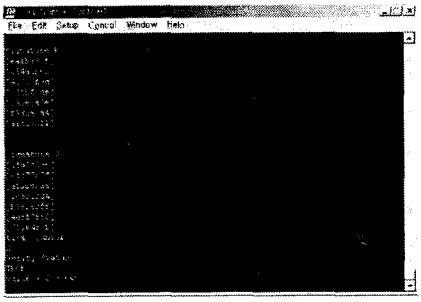
비트 수	Hardware(ms)	Software(ms)
112	48/60/144	3632/3472/6928
192	96/144/162	11024/10752/22176
256	144/135/234	21168/19936/41904

[표 3] 소수 체에서의 키 생성/서명 생성/검증 연산 속도 비교

비트 수	Hardware(ms)	Software(ms)
113	16/48/64	880/928/1776
239	32/64/128	2176/2112/4240
409	208/320/560	21104/20320/40272

[표 4] 이진 체에서의 키 생성/서명 생성/검증 연산 속도 비교

통상적으로 서명 검증이 서명 생성보다 더 많은 계산 시간이 요구되는데, 이는 두 번의 스칼라 곱셈과 한번의 point addition연산 때문이다. 소수 체(prime field)에서는 하드웨어 라이브러리를 사용하여 약 50~100배의 성능 개선 효과가 이루어졌으며, 이진 체(binary field)에서는 20~90배의 성능 개선 효과가 있음을 알 수 있다.



(그림 4) 서명 생성 및 검증 예.

Ⅳ. 결론 및 향후 연구과제

본 연구는 암호 가속 보드 설계 시, 암호 보조 프로세서의 사용을 통하여 얻을 수 있는 시스템 성능 개선 효과를 측정하기 위해 이루어졌다. 키 생성, 서명 생성, 서명 검증 연산의 핵심을 이루는 스칼라 곱셈과 여타의 연산 모듈을 전용 프로세서가 처리함으로써 현저한 성능 개선을 이룰 수 있음을 본 연구를 통해 알 수 있다. 향후 제한된 자원을 가지는 임베디드 환경에서, 시스템의 성능을 보다 개선하기 위해서는 암호 프리미티브에 대한 효율적인 구현 방안 연구가 선행되어야 하겠으며, 로컬 시스템의 프로세서 연산 부담을 효율적으로 경감시킬 수 있는 암호 보조 프로세서의 구조에 대한 연구도 병행되어야 하겠다.

참고문헌

[1] 최승복, 임채훈 "SSL 가속 기술의 분류와 제품 비교", (주) 퓨처시스템, 9. 2002.

[2] Philip J. Koopman Jr, "Embedded System Design Issues", Proceedings of the international Conference on Computer Design, 1996.

[3] "The Elliptic Curve Cryptosystem", Certicom Cooperation, 3. 1997.

[4] I.F.Blake, G.Seroussi, N.P Smart, "Elliptic Curves in Cryptography" Cambridge University Press. 1999.

[5] "American National Standard X9.62" American Banker Association, 1998.