디지털 컨텐츠 보호를 위한 DTCP용 타원곡선 암호(ECC) 연산기의 구현

김 의 석, 류 태 규, 정 용 진
광운대학교 전자통신공학과
전화 : 02-940-5551 / 핸드폰 : 011-454-7194

Design of ECC Calculator for Digital Transmission Content Protection(DTCP)

Eui-Seok Kim, Tae-Gyu Ryu, Yong-Jin Jeong
Dept. of Electronics & Communications Engineering, Kwangwoon Univ.
E-mail : check95@explore.kw.ac.kr

Abstract

In this paper, we implement an Elliptic Curve Cryptosystem(ECC) processor for DTCP. Because DTCP(Digital Transmission Content Protection) uses GF(p), where p is a 160-bit prime integer, we design a scalar multiplier based on GF(p). The scalar multiplier consists of a modular multiplier and an adder. The multiplier uses montgomery algorithm which is implemented with CSA(Carry-save Adder) and CLA(Carry-lookahead Adder). Our new scalar multiplier has been synthesized using Samsung 0.18 um CMOS technology and the maximum operation frequency is estimated 98 MHz, with the size about 65,000 gates. The resulting performance is 29.6 kbps, that is, it takes 5.4 msec to process a 160-bit data frame. We assure that this performance is enough to be used for digital signature, encryption/decryption, and key exchanges in real time environments.

1. 서론

정보 통신 기술의 급격한 발전은 전자상거래 및 전자 문서 교환과 같은 사용자간의 높은 수준의 보안과 신뢰성을 요구하는 다양한 서비스를 제공하고 있다. 그러나 인터넷 환경에서 사용자간의 정보 및 디지털컨텐츠의 공유, 복사 및 유포가 가능하기 때문에 저작권 침해와 같은 많은 문제가 발생하고 있다. 이를 해결하기 위해 Hitachi, Intel, Matsushita, Sony, Toshiba 5개 회사가 모여 디지털 컨텐츠 보호를 위한 DTCP를 제정하였다. DTCP는 음성/영상 컨텐츠 보호를 위하여 불법 복제 및 위조를 방지한다. DTCP는 AKE(Authentication and Key exchange), Content Encryption, Copy Control Information, System Renewability의 4가지 주요 구성으로 나눌 수 있으며, 이 가운데 AKE에서 타원 곡선 암호 시스템이 사용된다[1].

본 논문에서는 DTCP를 위한 스칼라 곱셈기를 설계하였다. 스칼라 곱셈기는 Verilog HDL을 이용하여 설계하였으며 Altera(사)의 Excalibur칩을 사용하여 검증하였다. 빠른 스칼라 곱셈 연산을 위하여 몽고메리 모듈러 곱셈 알고리즘을 이용하였으며, 곱셈기에 사용되는 덧셈기는 CSA와 CLA를 사용하였다.

2. 타원 곡선 암호 알고리즘

타원 곡선 암호 알고리즘은 타원 곡선 이산 로그 문제(ECDLP : Elliptic Curve Discrete Logarithm Problem)에 근간을 두고 있다. ECDLP는 타원 곡선상의 임의의 한 점 P에 정수 k를 곱한 값이 $Q=kP$일 때, 점 Q와 P를 알고 있더라도 정수 k를 계산하기 어려움을 의미한다. 따라서 타원 곡선 암호 시스템의 핵심 연산은 스칼라 곱셈, 즉 $Q=kP$를 구하는 것이다.

* 본 연구는 광운대학교 산학연센터 및 IDEC의 툴 지원으로 이루어졌습니다.

소수체 $GF(p)$상에서의 타원곡선은 (x, y)인 점들로 구성되어지고, $y^2=x^3+ax+b$ 의 형태를 가진다[2]. 효율적인 스칼라 곱셈을 연산하기 위하여 Double and Add 알고리즘을 이용할 경우 동일한 두 점의 합을 구하는 두배점 연산(doubling one point)과 서로 다른 두 점의 합을 구하는 점덧셈 연산(adding two point)을 반복적으로 이용함으로 스칼라 곱셈을 연산할 수 있다. 타원곡선상의 임의의 두 점을 P_1, P_2라 하였을 때 P_1과 P_2의 합 P_3를 구하는 과정은 다음과 같다[2].

Algorithm 1 EC Point Addition(Affine Coordinate)

input : $P_1(x_1, y_1)$, $P_2(x_2, y_2)$
output : $P_3(x_3, y_3) = P_1 + P_2$

if $P_1 = 0$ **then** $P_3 = P_2$ and stop
if $P_2 = 0$ **then** $P_3 = P_1$ and stop
if $x_1 == x_2$ **then**
  if $y_1 == y_2$ **then** //doubling one point
    $\lambda=(y_1-y_2)/(x_1-x_2) \bmod p$
  else
    $P_3 = 0$
else //adding two point
  $\lambda=(3x_2^2+a)/(2y_2) \bmod p$
$x_3 = \lambda^2-x_1-x_2 \bmod p$
$y_3 = (x_2-x_3)\lambda-y_2 \bmod p$

위 알고리즘에서와 같이 타원 곡선에서 두 점의 합을 계산하기 위해서는 곱셈, 나눗셈 그리고 덧셈 연산이 필요하다. 이 가운데 나눗셈 연산이 가장 많은 시간과 자원을 필요로 하며, 스칼라 곱셈기의 성능을 좌우하게 된다. 그러나 투영 좌표 변환을 통하여 나눗셈 연산 제거가 가능하며, 다음과 같이 정의된다[2].

Algorithm 2 EC Point Addition (Projective Coordinate)

intput : $P_1(X_1,Y_1,Z_1)$, $P_2(X_2,Y_2,Z_2)$
output : $P3(X_3,Y_3,Z_3) = P_1+P_2$

if $X_1 == X_2$ **then**
  if $Y_1 == Y_2$ **then** //doubling one point
    $M = 3X_1^2 + aZ_1^4$;
    $Z_2 = 2Y_1Z_1$;
    $S = 4X_1Y_1^2$;
    $X_2 = M^2 - 2S$;
    $Y_2 = M(S - X_2) - T$;

  else
    $P_3 = 0$;
else //adding two point
    $U_0 = X_0Z_1^2$;
    $U_1 = X_1Z_0^2$;
    $S_0 = Y_0Z_1^3$;
    $S_1 = Y_1Z_0^3$;
    $W = U_0 - U_1$;
    $R = S_0 - S_1$;
    $Z_2 = Z_0Z_1W$;
    $X_2 = R^2 - TW^2$;
    $V = TW^2 - 2X_2$;
    $2Y_2 = VR - MW^3$;

알고리즘 2에서 볼 수 있듯이 알고리즘 1와 비교하여 나눗셈 연산이 제거되었지만 보다 많은 곱셈 연산을 필요로 한다. 따라서 모듈러 곱셈 연산이 스칼라 곱셈의 많은 부분을 차지하게 되며 성능 향상을 위하여 효율적인 모듈러 곱셈기의 설계를 필요로 한다. 본 논문에서는 이를 위해 몽고메리 모듈러 곱셈 알고리즘을 사용하였으며 다음과 같이 정의된다.

Algorithm 3 Montgomery Multiplication

intput : x, y, P
output : S = $Monpro(x, y) = xyR^{-1} \bmod P$

$S = 0$;
for i from 0 to $k-1$ **do**
  $m_i = t_0 \hat{} x_iy_0$;
  $S = (S + x_iy + m_iP) / 2$;
end

위 알고리즘에서 k는 피승수 y의 비트 사이즈를 의미한다. 위에서 알 수 있듯이 몽고메리 알고리즘은 y의 모든 비트에 대해 검색하여 덧셈 및 쉬프트 연산을 행하는 것이다. x와 y가 k-bit 일 경우 k번 오른쪽 쉬프트 연산이 이루어지기 때문에 결과 값 xy에 2^{-k} 작은 값이 출력된다. 따라서 연산 후, 올바른 값을 얻기 위하여 다음과 같은 후처리 과정이 필요하다.

$$S = S * R \bmod P = S * 2^k \bmod P \quad (1)$$

그러나 스칼라 곱셈 연산 가운데 모든 곱셈 연산 후에 위 식과 같은 후처리 과정을 해준다면 비효율적이다. 따라서 ECC의 초기 좌표 값에 2^{2k}으로 미리 몽고메리 알고리즘을 이용하여 곱하는 전처리 과정을 행한 후 스칼라 곱셈 연산을 하며, 최종 결과 값에 후처리 과정을 행함으로 중간 결과 값에 대한 후처리 과정을 제거할 수 있다. 이 때 전처리 과정에서 2^{2k}을 곱하였기 때문에 후처리 과정에서는 1을 곱해야 한다. $S=A*B \bmod P$를 연산할 때 전처리 과정 및 후처리 과정을 포함하는 전체 모듈러 곱셈 연산 과정을 나타내면 다음과 같다.

Pre-Processing
$A^* = Monpro(A, 2^{2k}) = A*2^{2k}*2^{-k} \bmod P$
$= A*2^k \bmod P$
$B^* = Monpro(B, 2^{2k}) = B*2^{2k}*2^{-k} \bmod P$
$= B*2^k \bmod P$
Main-Processing (2)
$S^* = Monpro(A^*, B^*) = A*2^k*B*2^k*2^{-k} \bmod P$
$= A*B*2^k \bmod P$
Post-Processing
$S = Monpro(S^*, 1) = A*B*2^k*2^{-k}$
$= A*B \bmod P$

3. 하드웨어 구현

소수체인 GF(p)에서의 스칼라 곱셈기는 몽고메리

알고리즘을 이용한 모듈러 곱셈기와 모듈러 덧셈/뺄셈기 그리고 연산의 결과 값을 저장하기 위한 레지스터와 컨트롤러로 구성되어 있다. 본 논문에서 구현한 스칼라 곱셈기의 전체 구조는 그림 1과 같다.

연산을 위하여 타원 곡선의 변수로서 p, a가 입력되며, 초기 좌표 x, y, 스칼라 곱셈 연산을 위한 r값이 입력된다. 또한 몽고메리 알고리즘에서 식 (2)에서와 같이 초기 좌표 값에 대한 전처리 과정에 필요한 mon_2k값이 입력되며, 스칼라 곱셈의 결과 값으로 X_2, Y_2값이 데이터 버스로 출력 된다.

모듈러 곱셈기는 몽고메리 알고리즘을 이용하였으며 전체 구조는 그림 2와 같다. 알고리즘 3에서와 같이 몽고메리 모듈러 곱셈연산에서 덧셈 연산은 중간 결과 값 Sum, Multiplier B 그리고 모듈러 연산을 위한 P, 3개의 입력 값을 필요로 하기 때문에 연산 시간을 단축하기 위하여 CSA와 CLA를 사용하였다.

모듈러 덧셈/뺄셈기의 구조는 그림 3과 같다. 곱셈기와 마찬가지로 모듈러 덧셈/뺄셈기는 161-bit에서 연산된다. 하지만 뺄셈 연산을 하기 위하여 Sign bit 가 필요하며 덧셈 연산에서의 캐리로 인해 Sign bit 가 수정되는 것을 피하기 위하여 여기에서는 163-bit CLA를 사용한다.

4. 검증 및 성능 분석

본 논문에서 구현한 스칼라 곱셈기는 Verilog HDL을 이용하여 설계하였고, ALTERA(사)의 Excalibur를 사용하여 성능을 검증하였다. 전체 검증은 IEEE P1363a 표준 문서에 제시되어 있는 테스트 벡터를 이용하였으며, 전체 시스템 검증을 위하여 EC-DSA 및 EC-DH를 수행하여 획득한 키 값을 이용하여 텍스트 파일을 암호화 한 후 복호화 하여 비교하는 방법으로 검증하였다.

표 1은 $GF(p)$에서 스칼라 곱셈기를 구현한 대표적인 논문인 Sıddıka의 논문[3]과 비교한 표이다. Sıddıka의 스칼라 곱셈기는 파이프 라인 구조를 이용하여 모듈러 곱셈기를 설계하였기 때문에 연산을 위해 삽입되는 레지스터에 의해 최대 지연 시간이 짧아지는 장점이 있지만 단일 곱셈 연산을 위해 많은 클럭 수를 필요로 한다. 본 논문에서 설계한 스칼라 곱셈기는 모듈러 곱셈기의 CSA와 162-bit CLA를 한 클럭에 연산하기 때문에 최대 지연 시간은 상대적으로 길지만, 전체 연산을 위해 필요로 하는 총 클럭 수가 적기 때문에 전체 성능이 빨라진다.

표 1에서와 같이 FPGA에서 검증된 전체 성능은 Sıddıka의 스칼라 곱셈기가 9.4 kbps로 더 좋은 성능을 나타내지만 Xilinx의 FPGA를 사용하였기 때문에

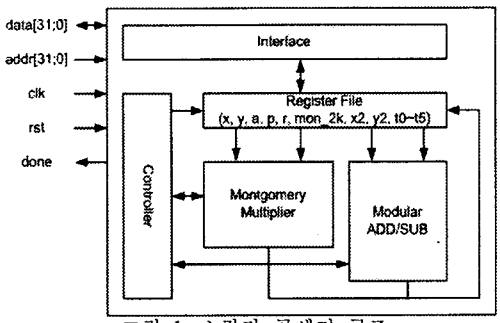

그림 1. 스칼라 곱셈기 구조

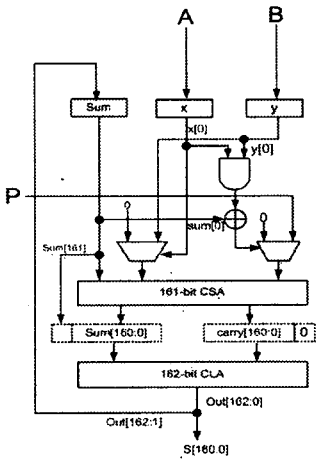

그림 2 몽고메리 곱셈기 구조

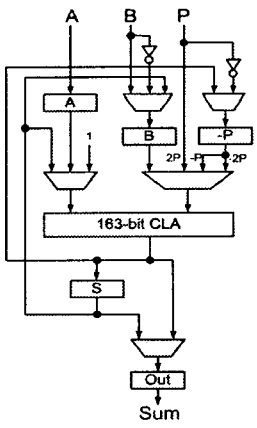

그림 3. 모듈러 덧셈/뺄셈기 구조

Altera의 FPGA를 사용한 본 논문에 비하여 처리 속도가 높게 측정되었다. 정확한 성능 비교를 위하여 연산을 위한 전체 클럭 수와 최대 지연 시간을 고려할 경우 Sıddıka의 스칼라 곱셈기는 1,552,840개의 전체 클럭 수와 10개 게이트의 최대 지연 시간을 가지지만,

표 1. Siddika 스칼라 곱셈기와의 성능 비교

	Siddika	본 논문
Normal to Montgomery	$12n+16$	$4(n+2)$
EC doubling	$36n + 38$	$10n + 29$
EC adding	$42n + 56$	$16n + 38$
m(EC point double) + m/2(EC point adding)	$m(57n+66)$	$m(18n+48)$
Projective to Affine	$3n^2 + 16n + 16$	$HW(p-2)\ (n+2) + (n-1)(n+2)+4(n+2)$
Montgoemry to normal	$6n + 8$	$2(n+2)$
Montgoemry Multiplication	$3n + 4$	$n+2$
Modular Add/Sub	$2n+1$	3
Total number of clock	$60n^2+105n+40$ = 1,552,840 clock	$19.5n^2+60n+18$ ($HW(p-2) = n/2$) = 508,818 clock
Total Area(gate)	115,520	64,539 (Samsung STD-130)
Critical Path	$2T_{FA}+T_{HA} = 10T_{gate}$	$T_{CLA} + T_{FA} + 4T_{gate} = 25T_{gate}$
Performance	91.308 MHz, 9.4 kbps (Xilinx V1000E-BG-560-8)	21.3 MHz, 7 kbps (Altera Excalibur EPXA1020C2)

본 논문의 스칼라 곱셈기는 508,818개의 전체 클럭 수와 25개 게이트의 최대 지연 시간을 가지기 때문에 본 논문의 스칼라 곱셈기가 더 좋은 성능을 가진다. 표 1에서 T_{FA}는 Full Adder의 지연 시간, T_{HA}는 Half Adder의 지연 시간, T_{CLA}는 CLA의 지연시간, T_{gate}는 1개의 게이트의 지연 시간을 의미한다. 이 가운데 T_{CLA}는 162-bit CLA의 지연 시간을 의미하며, 본 논문에서는 4-level CLA를 사용하였기 때문에 17 gate의 지연시간을 의미한다.

본 논문에서 제안한 스칼라 곱셈기는 하드웨어 사이즈면에서도 Siddika의 스칼라 곱셈기에 비하여 우수하다. 표 1에서와 같이 ASIC을 위하여 합성하였을 경우 Siddika의 스칼라 곱셈기는 115,520 게이트를 필요로 한다. 그러나 본 논문에서 제안한 스칼라 곱셈기를 삼성 STD 130 셀 라이브러리에서 Synopsys를 이용하여 합성하였을 때 64,539 게이트를 필요로 한다. 이 때 동작 속도는 98 MHz이며, 데이터 처리속도는 29.6 kbps로 160-bit 프레임 당 5.4 ms 걸린다. 이러한 성능은 메모리와 하드웨어 사이즈가 제한적인 휴대용 보안 장치에서 디지털 서명, 암호화 및 복호화 그리고 키 교환 등에 효율적으로 사용될 수 있을 것으로 여겨진다.

5. 결론

본 논문에서는 $GF(p)$에서 몽고메리 모듈러 곱셈 알고리즘을 이용하여 DTCP를 위한 타원 곡선 스칼라 곱셈기를 설계하였으며, DTCP의 전자 인증 및 키 교환에서 사용될 수 있다. 스칼라 곱셈 연산에서 가장 많은 자원과 시간을 필요로 하는 모듈러 곱셈 연산을 효율적으로 처리하기 위하여 몽고메리 모듈러 곱셈 알고리즘을 이용하였으며, 전처리 과정과 후처리 과정을 이용한 효율적인 몽고메리 모듈러 곱셈 연산을 제안하였다. 본 논문에서는 제안한 스칼라 곱셈기는 파이프라인 구조를 이용한 Siddika의 스칼라 곱셈기[3]에 비하여 최대 동작 주파수는 느리지만 하드웨어 사이즈가 작으며 연산을 위한 전체 클럭 수가 작아 전체 성능이 좋다. 휴대용 보안장치와 같은 임베디드 시스템에서 높은 동작 속도를 요구하지 않기 때문에 본 논문에서 구현한 스칼라 곱셈기는 휴대용 보안 장치의 보안 코프로세서로 사용하기에 적합하다고 여겨진다.

참고 문헌

[1] DTCP Specification, "5C Digital Transmission Content Protection Specification Volume 1(Information Version)," July, 2000.

[2] IEEE P1363a/D5(Draft Version 5). Standard Specifications for Public Key Cryptography : Additional Techniques, August 16 2000

[3] S. Berna Örs, L. Batina, B. Preneel and J. Vandewalle, "Hardware Implementation of an Elliptic Curve Processor over GF(p)," Proceedings of the Application-Specific Systems, Architectures, and Processor, IEEE 2003.