Window Non-Adajcent Form method를 이용한 타원곡선 암호시스템의 고속 스칼라 곱셈기 설계 및 구현

안 경 문, 김 종 태
성균관 대학교 전기전자 및 컴퓨터 공학부
전화 : 031-290-7173 / 핸드폰 : 016-411-1064

Design and Implementation of Fast Scalar Multiplier of Elliptic Curve Cryptosystem using Window Non-Adjacent Form method

Kyoung-Moon Ahn, Jong Tae Kim
School of Electrical and Computer Engineering, Sungkyunkwan University
E-mail : tofly@chollian.net

Abstract

This paper presents new fast scalar multiplier of elliptic curve cryptosystem that is regarded as next generation public-key crypto processor. For fast operation of scalar multiplication a finite field multiplier is designed with LFSR type of bit serial structure and a finite field inversion operator uses extended binary euclidean algorithm for reducing one multiplying operation on point operation. Also the use of the window non-adjacent form (WNAF) method can reduce addition operation of each other different points.

I. 서론

타원곡선 암호시스템(elliptic curve cryptosystem : ECC)은 1985년에 N. Koblitz와 V. Miller에 의해 독립적으로 제안되었다. 공개키 암호시스템의 한 종류인 ECC는 타원곡선 이산대수 문제(elliptic curve discrete logarithm problem : ECDLP)에 기반한다. 다른 공개키 암호시스템보다 비트 당 안전도가 더 높은 ECC는 서명, 인증 등 빠른 속도와 제한된 대역폭 등이 요구되는 분야와 스마트 카드처럼 메모리와 연산력에 제한이 있는 분야에 적용 가능하다.

ECC의 기본 연산은 타원곡선 상의 한 점을 정수배 하는 스칼라 곱셈이며, ECC 실행시 가장 많은 시간을 소비한다.

본 논문에서는 스칼라 곱셈기의 구현 파라미터로써 $GF(2^m)$형태의 유한체 $GF(2^{193})$, 기저로는 polynomial basis, 좌표계는 affine 좌표계, 모듈라 연산을 수행하기 위한 기약다항식 $f(x) = x^{193} + x^{15} + 1$을 선택하였다. 스칼라 곱셈의 고속화를 위해 window non-adjacent form(WNAF) method를 적용하여 설계 및 검증하고, 이를 구현하였다.

II. 유한체 연산기

2.1 유한체($GF(2^m)$) 가산기

유한체 $GF(2^m)$에서의 덧셈 연산은 각각 대응하는 비트들의 XOR 연산을 통하여 구현할 수 있다. 다른 유한체 연산기들에 비해 유한체 덧셈 연산기는 가장 적은 비용으로써 구현 가능하다. 두 입력에 대한 덧셈은 다음과 같이 나타낼 수 있다.

$$A(\alpha) = a_0 + a_1\alpha + \cdots + a_{m-1}\alpha^{m-1} \quad (1)$$

또는 $$\mathbf{A} = (a_0, a_1, \cdots, a_{m-1}) \quad (2)$$

$$B(\alpha) = b_0 + b_1\alpha + \cdots + b_{m-1}\alpha^{m-1} \quad (3)$$

또는 $\mathbf{B} = (b_0, b_1, \cdots, b_{m-1})$ (4)

$$A(\alpha) + B(\alpha) = (a_0+b_0) + (a_1+b_1)\alpha + \cdots + (a_{m-1}+b_{m-1})\alpha^{m-1} \quad (5)$$

또는

$$\mathbf{A} + \mathbf{B} = (a_0+b_0, a_1+b_1, \cdots, a_{m-1}+b_{m-1}) \quad (6)$$

이에 대한 구조는 직렬과 병렬로써 구현할 수 있다.[1]

2.2 유한체(GF(2^m)) 곱셈기

유한체 GF(2^m)에서 두 개의 입력(A(α), B(α))에 대한 곱(Z(α))은 다음과 같이 나타낼 수 있다.

$$Z(\alpha) = \sum_{i=0}^{m-1} b_i(\alpha^i A(\alpha)) \quad (7)$$

설계 시 승산기의 구조는 속도 이득을 얻기 위해 bit-serial 구조의 LFSR 형태로 구현하였다. 그림 1은 본 논문에서 구현한 승산기구조를 나타낸다.[1]

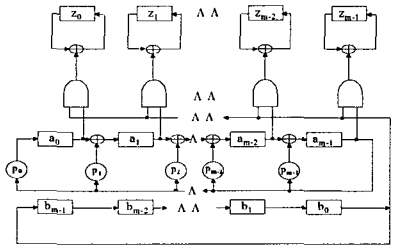

그림 3 유한체 LFSR 승산기

2.3 유한체(GF(2^m)) 역원기

기저가 polynomial인 GF(2^m)에서 역원 연산은 곱셈 연산에 비해 8 ~ 10배의 수행시간을 소비한다. 역원 연산을 위한 알고리즘은 보통 extended euclidean algorithm, parallel euclidean algorithm, almost inverse algorithm, inversion using multiplication algorithm이 사용되며, 본 논문에서는 extended euclidean algorithm을 수정한 extended binary euclidean algorithm을 사용하였다.[2]

Algorithm 1은 나눗셈 연산을 위한 extended binary euclidean algorithm을 나타낸다.

Input : W : a, the element of GF(2^m) that is to be inverted
X : N, the binary representation of the field polynomial f(x)
Y : b, the element of GF(2^m) that is to be multiplied by the computed inverse
Z : 0, just plain old zere
Output : Z = b/a
Algorithm : While (W ≠ 0)
While (W_0 == 0)
W = W/2;
Y = (Y + Y_0 · N)/2;
EndWhile
While (X_0 == 0)
X = X/2;
Z = (Z + Z_0 · N)/2;
EndWhile
If (W ≥ X)
W = W + X;
Y = Y + Z;
Else
X = W + X;
Z = Y + Z;
EndIf
EndWhile

Algorithm 1 : extended binary euclidean algorithm

스칼라 곱셈 연산에서 θ값을 계산하는 과정은 역원과 곱셈 연산이 필요하다. 본 논문에서는 extended binary euclidean algorithm을 적용함으로써 역원 계산 후 실행하는 곱셈 연산을 함께 처리할 수 있다. 즉, Y에 1이 아닌 임의의 값이 입력되는 경우 결과값은 W의 역원과 Y의 값을 곱한 값으로써 Y/W한 값이다. 단순히 W의 역원을 구하기 위해서는 Y에 1을 입력하면 된다. 이는 스칼라 곱셈 연산과정 중 한번의 곱셈 연산을 생략할 수 있고, 약 18% 정도의 clock수를 줄일 수 있다.[2]

Ⅲ. 스칼라 곱셈

3.1 points addition

서로 다른 두점을 더하기 위한 식은 다음과 같다.

$P = (x_1, y_1)$, $Q = (x_2, y_2)$,
$E = y^2 + xy = x^3 + ax^2 + b$
$P + Q = (x_3, y_3)$

$$x_3 = \theta^2 + \theta + x_1 + x_2 + a \qquad (8)$$
$$y_3 = \theta(x_1 + x_3) + x_3 + y_1$$
$$\theta = \frac{y_1 + y_2}{x_1 + x_2}$$

θ는 점 P와 Q의 직선의 기울기이다. 그림 2는 서로 다른 두 점에 대한 points addition 블럭도이다. 입력은 P(in_x1, in_y1), Q(in_x2, in_y2), irre_poly(기약다항식), curve(타원곡선의 계수)이고, 출력은 R(out_x, out_y)이다.

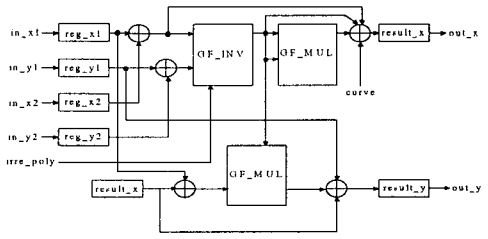

그림 2 points addition 블럭도

3.2 point doubling

한 점을 두 배 하기 위한 식은 다음과 같다.

$$P = (x_1, y_1)$$
$$E = y^2 + xy = x^3 + ax^2 + b$$
$$2P = P + P = (x_3, y_3)$$
$$x_3 = \theta^2 + \theta + a \qquad (9)$$
$$y_3 = x_1^2 + (\theta + 1)x_3$$
$$\theta = x_1 + \frac{y_1}{x_1}$$

그림 3은 입력 P(in_x, in_y)에 대한 point doubling 블럭도이다.

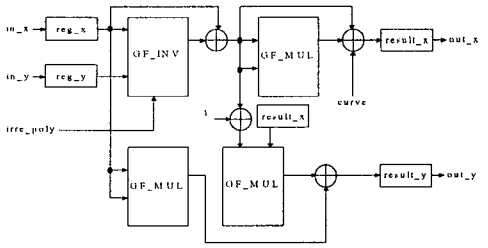

그림 3 point doubling 블럭도

3.3 computing the $NAF_w(k)$

스칼라 곱셈은 입력값 정수 k를 변환하고, 변환값에 의해 연산을 수행하는 방법에 따라 크게 3가지 binary method, non-adjacent form(NAF) method, window NAF(WNAF) method가 있다.[3] 정수 k의 변환 형태는 binary method의 경우 비트열, NAF method는 {-1, 0, 1}로 이루어진 열[4], WNAF method는 0과 정수 u로 이루어진 열로 변환된다. 본 논문에서는 변환값 u의 최상위 비트를 부호 비트로 처리하여 스칼라 곱셈 연산 시 참조하기 위해 필요한 레지스터의 개수를 줄이는 효과를 얻었다. 이때 u의 조건은 다음과 같다.

$$u \equiv k \pmod{2^w} \text{ and } -2^{w-1} \le u < 2^{w-1} \qquad (10)$$

Algorithm 2는 정수 k의 $NAF_w(k)$ 형태로의 변환을 나타낸다.

Input : a positive integer k
Output : $NAF_w(k)$
Algorithm : i ← 0.
While k ≥ 1 do
If k is odd then:
$k_i \leftarrow k \bmod 2^w$,
$k \leftarrow k - k_i$;
Else : $k_i \leftarrow 0$;
$k \leftarrow k/2$, $i \leftarrow i + 1$.
Return($(k_{i-1}, k_{i-2}, \cdots, k_1, k_0)$)

Algorithm 2 : computing the $NAF_w(k)$

3.4 WNAF method

Window w = 4인 경우 $NAF_w(k)$는 {0, ±1, ±3, ±5, ±7}의 값을 갖는다. 스칼라 곱셈 연산은 $NAF_w(k)$의 값에 따라 서로 다른 두 점을 더하는 points addition 연산과 같은 점을 더하는 point doubling 연산으로 이루어진다. $NAF_w(k)$의 값이 (-)인 경우, 레지스터에 저장된 점 P(x, y)는 다음과 같이 변환되어 연산에 사용된다.[5]

$$-(x, y) = (x, x + y) \qquad (11)$$

Alorithm 3은 WNAF를 이용한 스칼라 곱셈을 나타낸다.

Input : Window width w,
$$NAF_w(k) = \sum_{i=0}^{l-1} k_i 2^i, \quad P \in E(F_p)$$

Output : kP
Algorithm : Compute $P_i = iP$
($i \in \{1, 3, 5, \cdots, 2^{w-1}-1\}$)
$Q \leftarrow 0$.
For i from l-1 downto 0 do
$Q \leftarrow 2Q$.
If $k_i \neq 0$ then:
If $k_i > 0$ then $Q \leftarrow Q + P_{ki}$
Else $Q \leftarrow Q - P_{ki}$
Return(Q)
Algorithm 3 : WNAF method

3.5 스칼라 곱셈기 블럭도

그림 4는 points addition과 point doubling 연산, 사전 연산에 의해 계산된 점 P의 홀수배 들을 저장하는 레지스터로 구성된 스칼라 곱셈기 블럭도이다.

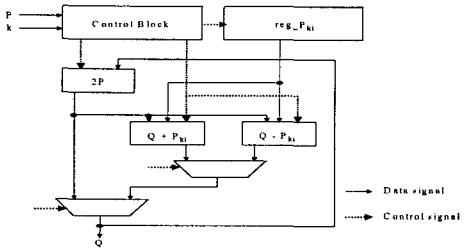

그림 4 스칼라 곱셈기 블럭도

Ⅳ. 합성 결과 및 비교

표 1은 C언어를 사용하여 입력 정수 k에 대해 세가지 알고리즘을 적용, 검증하여 비교한 결과이다. k값의 크기에 따라 WNAF method가 binary method에 비해 18.6 ~ 22.5%, NAF method에 비해 8.2 ~ 11.3%의 시간 이득을 보았다

k(bits)	Binary	NAF	WNAF (window = 4)
131	1,420	1,280	1,156
163	2,790	2,562	2,272
193	4,964	4,218	3,902
223	9,634	8,140	7,470

Pentium III(450MHz), Visual C++ (단위 : clock)

표 5 정수 k에 대한 시뮬레이션 결과

표 2는 C 언어로 알고리즘을 검증한 후 VHDL을 이용하여 RTL 레벨에서 설계 및 functional simulation을 한 후 SYNOPSYS사의 Design Complier를 이용한 합성 결과이다. 공정 라이브러리는 lsi_10k를 사용하였다. 단위로 사용한 cell count의 cell은 2-input NAND gate의 면적과 등가이다. Clock frequency는 34MHz이다.

	Combination	Noncombination	Total
GF_MUL	3,387	5,892	9,279
GF_INV	13,531	9,738	23,269
$NAF_w(k)$	5,547	4,020	9,567
Addition	28,758	23,462	52,220
Doubling	26,541	27,414	53,955

표 6 합성 결과 (단위 : cell count)

Ⅴ. 결론

본 논문에서는 타원곡선 암호시스템의 스칼라 곱셈기의 고속화를 위해 bit-serial 구조의 LFSR 형태로 유한체 승산기를 구현하였고, point 연산 시 extended binary euclidean algorithm을 사용함으로써 한 번의 곱셈 연산을 생략할 수 있었다. WNAF method의 적용은 다른 방법에 비해 수행시간의 이득을 얻을 수 있었다. 정수 k의 변환된 형태에서 최상위 비트를 부호 비트로 처리하였다. 이는 스칼라 곱셈 연산 시 참조하는 레지스터의 개수를 줄이는 효과와 입력 P점에 대한 홀수배를 계산하기 위한 연산량을 줄이는 효과를 보았다.

이와 같이 구현된 스칼라 곱셈기의 적용으로 향후 타원곡선 암호시스템 구현 시 성능 향상이 기대된다.

참고문헌(또는 Reference)

[1] Man Young Rhee, "Cryptography and Secure Communications", McGRAW-HILL, pp. 209-224, 1994
[2] 이상호. "차세대 공개키 암호프로세서용 표준 다항식 기저의 고속 타원곡선 암호시스템 하드웨어 설계", 성균관대학교, 2001
[3] M. Brown, D. Hankerson, J. Lopez, A. Menezes, Software Implementation of the NIST Elliptic Curves Over Prime Fields", CT-RSA, pp. 250-265, 2001
[4] M. Rosing, "Implementing Elliptic Curve Cryptography", Manning, pp. 120-126, 1999
[5] J. Solinas, "Improved Algorithms for Arithmetic on Anomalous Binary Curves", corrected and updated version of CRYPTO'97 paper, Technical Report of CACR, University of Waterloo, 1999