타원 곡선 암호 알고리즘의 네배점 스칼라 연산

문상국, 허창우, 유광열
목원대학교 전자정보보호공학부

Point Quadruple Operation on Elliptic Curve Cryptography Algorithm

Sangook Moon, Chang Wu Hur, and Kwang Ryol Ryu
Mokwon University, School of Electronics & Information Engineering
E-mail : smoon@mokwon.ac.kr

요 약

타원곡선 암호시스템에서의 가장 줄기가 되는 연산은 스칼라 곱셈 연산이다. 본 논문에서는 기존의 두배점-덧셈 (*double-and-add*) 알고리즘으로 처리하였던 스칼라 곱셈 연산을 개선하여 네배점-덧셈 (*quad-and-add*) 알고리즘을 사용하기 위하여 네배점 (point quadruple) 연산을 유도한다. 유도된 식은 C 프로그램을 사용하여 실제 계산에 응용하여 증명하였다. 네배점 스칼라 연산은 타원곡선 암호시스템의 효율적이고 빠른 연산을 처리하는데 응용될 수 있다.

ABSTRACT

The most time-consuming back-bone operation in an elliptic curve cryptosystem is scalar multiplication. In this paper, we propose a method of inducing a GF operation named point quadruple operation to be used in the *quad-and-add* algorithm, which was achieved by refining the traditional *double-and-add* algorithm. Induced expression of the algorithm was verified and proven by C program in a real model of calculation. The point quadruple operation can be used in fast and efficient implementation of scalar multiplication operation.

키워드
point quadruple, elliptic curve cryptography (ECC), GF, cryptosystem

Ⅰ. 서 론

타원 곡선 암호 응용 계산을 적용할 때 가장 시간이 많이 걸리면서 대부분을 차지하는 동작은 다음 식과 같이 점 덧셈 연산을 반복하여 수행하는 스칼라 곱셈이다. 여기서 k는 $GF(2^m)$ 상의 임의의 정수이고 P는 $GF(2^m)$ 상에서 정의된 타원 곡선 위의 임의의 점이다.

$$kP = P + P + \ldots + P \qquad (1)$$

일반적인 연산 방식에서 스칼라 곱셈을 한 번 수행할 때는 여러 번의 점 덧셈 연산과 (더하고자 하는 두 점이 다를 때) 두배점 연산이 (더하고자 하는 두 점이 같을 때) 필요하고, 무엇보다도 식 (1)을 빠르게 처리할 수 있는 방법이 중요하다. 식 (1)을 빠르게 처리하는 방법은 전통적으로는 *double-and-add* 방식이 사용되어 왔지만, 최근 *quad-and-add* 방식이 새롭게 제안되었다. 본 논문에서는 *quad-and-add* 방식에서 사용될 수 있는 point quadruple 연산의 구현에 대해서 식으로 유도하고, 실제 어플리케이션으로 검증하여 본다. 유도된 수식은 C 프로그램으로 증명되어 성능을 평가하였다.

Ⅱ. 타원곡선 연산

타원 곡선 암호 시스템에서 스칼라 곱셈을 구현할 때 가장 많이 사용되면서 기본적인 방법이 *double-and-add* 알고리즘을 사용한 방식이다 [1]. 이 방식은 RSA 암호 시스템에서 정수의 법 연산으로 지수 연산을 구현할 때 사용하는 *square-and-multiply* 방식과 유사하다고 할 수 있는데,

$k = \sum_{i=0}^{m-1} b_i 2^i \ (bi \in 0,1)$ 일 때 알고리즘은 아래와 같다. 알고리즘 내에서 점 덧셈 연산은 *add*(), 두배점 연산은 *double*() 이라고 표기한다.

Double-and-add algorithm for computing kP

kP :

$k = \sum_{i=0}^{m-1} b_i 2^i \ (bi \in 0,1)$
$P := P(x_1, y_1)$
$Q := P$
for i from m-1 downto 0 do (2)
  $Q := double(Q)$
  if $b_i = 1$ then
  $Q := add(P, Q)$
end $(Q = kP)$

이 알고리즘에서 중요한 것은 최소 m-1번의 두배점 연산에 더하여 k를 이진수로 표현했을 때의 Hamming weight만큼의 점 덧셈 연산이 필요하다는 것이다. 이 방식을 개선하기 위해, 몇 가지 방식이 제안되었다. 그 중 하나가 Non Adjacent Format (NAF)을 이용한 방식인데, 알고리즘은 다음과 같다 [2].

Binary NAF method for computing kP

$NAF(k) = \sum_{i=0}^{t-1} k_i \cdot 2^i$

kP :
  $Q := 0$
  for i from t-1 downto 0 do
    $Q := 2Q$
    if $k_i = 1$ then $Q := Q + P$ (3)
    if $k_i = -1$ then $Q := Q - P$
  end $(Q = kP)$

위 방식은 kP를 계산함에 있어 k의 이진 표현에 대한 redundancy를 사용한다. 하지만 k를 NAF 형태로 미리 바꾸어야 한다는 단점이 있다. 이보다 발전된 형태는 *quad-and-add* 라는 이름의 알고리즘으로 다음 식 (4)와 같다.

Quad-and-add algorithm using radix-4 redundancy

kP :

$k = \sum_{i=0}^{\frac{m}{2}-1} r_i 4^i$ (r_i는 booth recoding 값)

$P := P(x_1, y_1)$
$2P := double(P)$
$Q := \{0P, +P, +2P, -P, -2P\}$ 중 하나
  for i from $\frac{m}{2} - 1$ downto 0 do
  $Q := quad(Q)$
  if (r_i == +P) then
  $Q := add(P, Q)$
  if (r_i == +2P) then
  $Q := add(2P, Q)$
  if (r_i == -P) then
  $tempP := neg(P)$ (4)
  $Q := add(tempP, Q)$
  if (r_i == -2P) then
  $tempP := neg(2P)$
  $Q := add(tempP, Q)$
end $(Q = kP)$

여기서, 타원 곡선의 네 배가 되는 점을 구하기 위해 한 연산 단계에 두 번씩 *double*() 연산을 수행하는 것을 개선시키기 위해서, 점 덧셈 연산과 두배점 연산을 이용하여, 네배점 연산 (point quadruple; *quad*())을 유도하여 다음과 같은 식 (5)를 얻을 수 있다.

Point quadruple operation (quad())

· $P(x_1, y_1) = Q(x_1, y_1)$ is the same point on an EC.
· if x_1 = 0, the result $4P$ is O (zero at infinity)
· if $x_1 \neq 0$, the result $4P(x_1, y_1) = R(x_3, y_3)$, x_3 and y_3 are as follows,

$$
\begin{aligned}
x_3 &= \lambda'^2 + \lambda' + a, \\
y_3 &= x_1^2 + (\lambda' + 1)x_3, \\
\lambda' &= x_2 + \lambda + 1 + \frac{x_1^2}{x_2}, \\
x_2 &= \lambda^2 + \lambda + a, \\
\lambda &= (x_1 + \frac{y_1}{x_1}).
\end{aligned}
\tag{5}
$$

III. 4배점 연산의 기능 검증

유도된 타원 곡선의 4배점 연산을 검증하기 위해서, 스칼라 곱셈 연산을 세부 블록으로 나누어 검증하였다. 세부 블록 검증은 가장 하위 레벨의 유한체 곱셈기, 유한체 나눗셈기로 나누어 검증하고 전체 블록은 상태 머신 (state machine)을 통하여 설계된 제어 회로와 임시 레지스터 블록 등

을 통합하여 검증하였다. 회로의 전체적인 구조와 확장성 등은 수식을 통해 알고리즘을 유도하여 C 프로그램으로 모의실험 (simulation) 하여 검증하였다.

먼저 유한체 곱셈기를 검증하기 위해서 동작이 검증된 *Mastrovito*의 직렬 곱셈기 [3]를 C 언어로 구현하고 테스트 벡터 (test vector)로서 193비트 임의의 난수를 발생시켜 입력하였다. C 언어로 기술된 직렬 곱셈기는 알고리즘에 기초하여 기술되었기 때문에 오류를 발생시키지 않으며 결과는 알고리즘의 흐름에 따른 부분 결과를 살핌으로써 확인되었다. 또한, [1]에서 검증된 나눗셈기를 사용하여 검증에 활용하였다. 전체 스칼라 곱셈을 검증하기 위해서는 타원 곡선 연산의 특성으로 생성점 (generator)에 타원 곡선의 위수를 곱하면 무한 원점 O를 얻는다는 사실을 이용하여 $n \cdot G = O$를 계산하여 얻음으로써 검증을 수행하였다 [4]. 그림 1과 그림 2는 각각 *double-and-add* 알고리즘과 *quad-and-add* 알고리즘을 사용하고 제안된 4배점 연산인 *quad()* 연산을 사용하여 192번째 스텝에서 예측 결과 값인 무한 원점 O를 얻는 것을 보여주는 결과이다.

double-and-add

step# 0:double-and-add
0_D9B67D19_2E0367C8_03F39E1A_7EB2CA14_A651350A_AE617EBF
1_CE943356_07C304AC_29E7DEFB_D9CA01F5_96F92722_4CDEOF6C
step# 1:double
1_756FF0DC_810F7866_023C5F5C_B14481F3_A688572B_B1513DA3
1_07188367_5B3044A9_217AD3AC_A9EFBCDC_89CDEBA2_3F931652
step# 2:double
1_1549FE34_2A8980E6_C932AF6F_4C81D415_00B09840_85F3B447
1_C0DDD61E_0CD1960A_59F7FE63_A8660A53_4D9F431E_4BC9839F
⋮
step#190:double-and-add
1_2654EB57_653586DB_05FD2EBC_511BC95F_2D995691_E0E95F9F
0_9C3BCACD_837A6A81_97F97238_3D20828E_1797902E_5829F927
step#191:double
1_5AE7384C_9954F598_6475718C_069EE793_3F2AA29E_2465FBE7
1_3BC5521A_6D7AE739_4E5E2DF9_FA26FB66_2DB6D58D_13BC8CAA
step#192:double-and-add
0_00000000_00000000_00000000_00000000_00000000_00000000
0_00000000_00000000_00000000_00000000_00000000_00000000

그림 1. *double-and-add* 방식으로의 계산

quad-and-operation

step# 0
quad-and-add(p)
0_D9B67D19_2E0367C8_03F39E1A_7EB2CA14_A651350A_AE617EBF
1_CE943356_07C304AC_29E7DEFB_D9CA01F5_96F92722_4CDEOF6C
step# 1: quad
1_1549FE34_2A8980E6_C932AF6F_4C81D415_00B09840_85F3B447
1_C0DDD61E_0CD1960A_59F7FE63_A8660A53_4D9F431E_4BC9839F
⋮
step# 94 : quad-and-add(p)
0_24771C2C_8E33F4A9_81965AC9_5FBC8DE2_4A0FC903_6208E77D
0_905C0FEB_0E90B8D0_259C0682_C561E98C_43935E74_9A872F1D
step# 95 : quad-and-add(p)
1_2654EB57_653586DB_05FD2EBC_511BC95F_2D995691_E0E95F9F
0_9C3BCACD_837A6A81_97F97238_3D20828E_1797902E_5829F927
step# 96 : quad-and-add(p)
0_00000000_00000000_00000000_00000000_00000000_00000000
0_00000000_00000000_00000000_00000000_00000000_00000000

그림 2. *quad-and-add* 방식으로의 계산

IV. 비교 및 성능 평가

성능 평가는 타원 곡선 암호 시스템, 즉 유한체 상에서 가장 상위 레벨 연산인 스칼라 곱셈에 대하여 수행하였고, 새로운 *quad()* 연산에 중점을 맞추어 진행하였다. 제안된 *quad()* 연산을 사용하여 모의 실험된 암호용 프로세서는 SEG 2 [4]에서 제안하는 193 비트 타원 곡선 파라미터인 *sect193r2*를 적용하였을 때 타원 곡선 점의 암호화 단계인 스칼라 곱셈 수행 시 기존 방식 (*double-and-add*)에 비해 유한체 곱셈 연산이 30% 이상 줄고 유한체 제곱 연산과 유한체 덧셈 연산이 소폭 증가하여 전체적으로는 약 30%의 성능 향상을 얻을 수 있었다.

V. 결 론

본 논문에서는 Elgamal 타원 곡선 암호 시스템에서 사용될 수 있는 스칼라 곱셈 연산에서의 *quad()* 연산을 유도하고 제안하였다. 설계된 프로세서의 검증은 C 프로그램을 이용하여 수행하였다. 결과는 기존 방법에 비해 약 30%의 성능 향상을 보였다.

스칼라 곱셈은 Elgamal 타원 곡선 암호 시스템의 키를 교환하는데에도 사용되지만, 타원 곡선을 이용한 암호화, 혹은 타원 곡선을 이용하여 전자서명을 교환하는 데도 사용되는 등, 타원 곡선을 사용하는 거의 모든 알고리즘에 적용이 되는 매우 중요한 자리를 차지하고 있는 연산이다.

본 논문에서 제안된 방법은 IC 카드와 같은 정보보호기기의 복잡한 연산 처리 속도를 향상시키는 데 필수적인 역할을 할 것이며, 모든 유한체 연산을 수행하는데 매우 효율적으로 사용될 수 있을 것이다.

참고문헌

[1] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer-Verlag, 1991.

[2] D. Hankerson, J. L. Hernandez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields", *Crypto95*.

[3] E. D. Mastrovito, "VLSI Architectures for Computations in Galois Fields", *Linkoping Studies in Science and Technology Dessertations*, No. 242, 1991.

[4] Certicom research, "SEC 2 : Recommended Elliptic Curve Domain Parameters", October 1999.