자원이 제약된 디바이스에서 효율적인 타원 멀티 스칼라 곱셈의 구현을 위한 유연한 접근

서석충, 김형찬, R.S. Ramakrishna
광주과학기술원 정보통신공학과

A Flexible Approach for Elliptic Multi-Scalar Multiplication on Resource-constrained Devices

Seog Chung Seo, Hyung Chan Kim, R.S. Ramakrishna
Department of Information and Communications,
Gwangju Institute of Science and Technology

요 약

Elliptic Curve Cryptosystem (ECC)은 작은 키 크기로 인하여 스마트카드, 센서 모트와 같은 메모리, 컴퓨팅 능력이 제약된 디바이스에서 사용하기에 적합하다. 본 논문에서는 이러한 디바이스에서 타원 곡선 서명 알고리즘 (ECDSA) 검증(Verification)의 주된 계산인 멀티 스칼라 곱셈을 (multi-scalar multiplication) 효율적으로 구현하기 위한 알고리즘을 제안한다. 제안 알고리즘은 어떠한 메모리 크기에서도 적용 가능할 뿐만 아니라 해당 메모리 크기에서 최적의 효율성을 제공한다. 또한 스칼라 리코딩 (Scalar recoding) 과정이 table lookup을 사용하지 않고 on-the-fly 하게 진행되기 때문에 기존의 다른 알고리즘에 비하여 더욱 메모리를 절약할 수 있다. 실험을 통하여 제안 알고리즘의 성능을 메모리 사용량, 효율성 측면에서 분석한다.

I. 서론

현재의 유비쿼터스 환경의 모바일 컴퓨터들은 스마트카드, 센서 모트와 같은 자원이 제약된 작은 디바이스들로 구성된다. 따라서 이러한 제약된 자원에서 충분한 보안성을 달성할 수 있는 암호 시스템이 필요하다. 타원곡선 암호 시스템 (ECC) [1]은 기존의 RSA나 DSA와 같은 암호 시스템과 비교하여 훨씬 작은 키 크기를 요구하기 때문에 자원이 제약된 디바이스에서 구현되기에 용이하다. 본 논문에서는 이러한 디바이스에서 ECDSA의 검증 (verification) 알고리즘의 주된 계산인 멀티 스칼라 곱셈 (uP + vQ, u와 v는 스칼라, P와 Q는 포인트)을 효율적으로 구현하기 위한 알고리즘을 제안한다. 구현 알고리즘의 목표는 계산 부하와 메모리 사용량을 최소화 하는 것이다. 이를 위하여 여러 알고리즘들이 제안되었으나 위의 두 측면을 충분히 만족시키지 못하였다. E. Dahmen [4]의 알고리즘은 기존의 알고리즘들 중에서 최적의 성능을 제공하고 메모리 사용량을 최소화하였지만 특정 메모리 크기에서만 적용가능하다. 제안 알고리즘은 위의 알고리즘에 비하여 거의 비슷한 성능을 제공하며, 메모리 절약과 확장성 측면에서는 보다 뛰어나다. 본 논문의 기여는 다음과 같다. 첫째, left-to-right recoding 알고리즘의 사용으로 recoding시에 필요한 메모리의 낭비를 제거하였다. 또한 recoding 알고리즘이 table lookup을 이용하지 않고, weighted sum을 이용하여 on-the-fly하게 계산이 되기 때문에 기존의 다른 recoding 알고리즘들에 비하여 메모리 사용량을 더욱 줄일 수 있다. 둘째, fractional window 개념의 도입으로 어떠한 메모리 크기에 대해서 낭비 없이 최적의 효율성을 제공한다. 셋째, 제안 알고리즘은 기존의 다른 알고리즘들에 비하여 늘어나는 메모리에 대하여 적절한 성능을 제공하는 확장성을 제공한다.

II. Multi-Scalar Multiplication

일반적으로 ECC의 성능은 스칼라 곱셈 (uP, 주어진 scalar u와 point P에 대하여)의 구현에 달려있다. 멀티 스칼라 곱셈 ($\sum_{j=1}^{k} d_j P_j$)은 두 개 이상의 스칼라 곱셈의 결과를 더한 것이다. 스칼라 곱셈을 계산하기 위하여 다음과 같은 binary method가 이용된다.

```
X ← O
for i = n − 1 down 0 do
    X ← ECDBL(X)
        if d[i] ≠ 0 then
            X ← ECADD(X, Q_d[i])
        endif
end for
return X
```

즉, scalar의 값이 0이면 point doubling (ECDBL) 만을 수행하고, 0이 아니면 doubling 연산 후에 해당 bit에 대응하는 point를 더한다 (ECADD). Binary method의 진행방향은 스칼라의 MSB에서 LSB 혹은 LSB에서 MSB 모두 가능하지만 MSB에서 LSB가 더욱 선호된다.

2.1 Shamir's Trick and Interleave Method

멀티 스칼라 곱셈을 계산하기 위하여 binary method의 변형인 Shamir's trick과 Interleave method가 제안되었다. Shamir's trick은 조인트 스칼라의 값이 0이 아닐 경우에 ECADD가 수행되는 것이 특징이다.

```
if (d_1[i], d_2[i]) ≠ (0,0) then
    X ← ECADD(X, d_1[i]P_1 + d_2[i]P_2)
```

반면에 Interleave method는 ECADD가 각각의 스칼라에 따라 독립적으로 수행된다.

```
for (j = 1; j ≤ 2; j++)
    if d_j[i] ≠ 0 then
        X ← ECADD(X, d_j[i]P_j)
```

즉, 위와 같이 for loop을 돌면서 해당 스칼라의 bit가 0이 아닐 경우에 ECADD가 수행된다. Shamir'trick과 interleave method의 성능을 비교해보면 (*Digit set* $D \in 0,1$) 아래와 같다.

	Operations	Precomputation
Shamir's trick	$nECDBL + \frac{3n}{4} ECADD$	1 (P+Q)
Interleave method	$nECDBL + nECADD$	0

비록, 스칼라의 Digit set이 0과 1로만 구성된 경우에 Shamir's trick의 성능이 Interleave method 보다 뛰어나지만 Digit set이 늘어남에 따라 precomputed table의 크기가 지수 승으로 증가하는 단점이 있다.

2.2 Signed Recoding Algorithms

Shamir's trick과 Interleave method에서 볼 수 있듯이 doubling의 수는 n번으로 고정되어 있다. 하지만 addition의 수를 줄임으로써 멀티 스칼라 곱셈의 성능을 높일 수 있다. Signed recoding의 목적은 입력으로 들어온 스칼라들에 대하여 같은 값을 유지하면서 signed digit set을 이용하여 0이 아닌 비트의 수를 줄이는 것이다. 다음과 같은 예를 들 수 있다. ($\bar{1} = -1$)

$127 = 1111111_2$	$7ECDBL + 7ECADD$
$127 = 1000000\bar{1}_2$	$7ECDBL + 2ECADD$

Signed digit set을 이용하는 recoding 과정을 통하여 5번의 point addition을 줄일 수 있다. 대표적인 signed recoding algorithm으로 wNAF(width-w Non-adjacent-form)과 wMOF (width-w Mutual-opposite-form)가 있다. 두 recoding 알고리즘의 비교는 다음과 같다. [2]

	Precomputation	Density	Direction
wNAF[5]	$2^{w-2}-1$	1/(w+1)	right-to-left
wMOF[2]	$2^{w-2}-1$	1/(w+1)	left-to-right

두 알고리즘의 성능과 메모리 요구량은 동일하지만 wNAF의 생성이 right-to-left로 진행되는 것에 비하여 wMOF는 left-to-right로 생성되기 때문에 on-the-fly하게 binary method와 통합 (merge) 될 수 있다. Density 1/(w+1)은 연속적인 w의 비트들중에서 기껏해야 오직 하나의 non-zero 비트만이 존재함 (1번의 addition)을 의미한다. 따라서 window의 크기를 증가시킬수록 성능을 높일 수 있다. 하지만 그에 따라 digit set의 크기가 증가하기 때문에 precomputed table의 크기도 증가한다. wNAF와 wMOF의 digit set은 $\{\pm 1, \pm 3, ..., \pm 2^{w-1}-1\}$이며, 오직 홀수로만 구성된다. 따라서 precomputated table의 크기는 $2^{w-2}-1$이 된다.

2.3 Precomputed Table

2.2절에서 recoding algorithm에 사용되는 window size의 크기를 증가시킬수록 성능은 향상시킬 수 있으나, precomputed table의 크기는

증가함을 확인할 수 있었다. 즉, 성능과 메모리의 tradeoff를 적절히 조율해야한다. Shamir's trick과 Interleave method를 signed recoding algorithm과 결합하였을 경우의 precomputed table의 크기는 다음과 같다 (recoding algorithm은 wMOF로 가정한다).

	Precomputation
Shamir's Trick	$\frac{\lvert D\rvert^k - 1}{2} - k, D = digit\ set$
Interleave Method	$k \cdot \frac{(\lvert D\rvert - 1)}{2} - k$

위 테이블에서 확인할 수 있듯이, Shamir's trick에서는 signed recoding algorithm의 digit set이 증가할수록, precomputation table의 크기는 지수 승으로 증가하는 것에 반하여, Interleave method의 precomputation table 크기는 linear하게 증가한다. 따라서 멀티 스칼라 곱셈을 구현하는 것에 있어서 Interleave method가 Shamir's trick보다 확장성과 유연성 면에서 뛰어난 것을 확인할 수 있다.

III. 제안 알고리즘

기존의 recoding algorithm들은 메모리에 대한 유연성 (flexibility)이 떨어진다는 단점이 있다. 예를 들어, wMOF의 경우 window가 3, 4, 5, 6,...인 경우에 대하여 1, 3, 7, 15, ...의 precomputed table을 요구한다. 만약 사용가능한 table의 크기가 10이라면 3만큼의 메모리를 낭비하게 되는 것이다. 따라서 어떤 메모리에서도 적용 가능한 recoding알고리즘이 필요하다.

3.1 Fractional Window

메모리에 대한 최적의 유연성을 제공하기 위하여 wMOF에 fractional window [3]를 적용한 fracwMOF를 제안한다. 실제로 [6]에서 wMOF에 fractional window개념을 적용한 알고리즘이 제안되었으나 MOF에서 wMOF로의 변환이 table lookup을 이용하는 형태이기 때문에 추가적인 메모리를 요구한다. 제안 알고리즘에서는 이러한 recoding과정이 weighted sum을 이용하여 on-the-fly하게 진행되기 때문에 더욱 메모리를 절약할 수 있다.

멀티 스칼라 곱셈에서 사용하는 메모리 크기와 계산 부하를 최소화하고 여기에 메모리에

```
INPUT: a fractional width w=w0+w1 u-bit nonzero scalars d_j, j = 1,2
  Points P, Q, R ← O, V for checking reduction
OUTPUT: kP + lQ
d_j[-1] ← 0; d_j[n] ← 0, j = 1, 2
Compute iP, iQ for all i ∈ [3, (1 + w1) * 2^(w0-1) − 1]
upper ← (1 + w1) * 2^(w0-1) − 1
for i = n to 0 do
  R ← ECDBL(R)
  for j = 1 to 2 do
    if δ_j = 0 and d_j[i] ≠ d_j[i − 1] then
      δ_j ← fracwMOF(V, upper, d_j[i-1]-d_j[i], d_j[i-2]-d_j[i-1],
      ..., d_j[i-w0-1]-d_j[i-w0])
      if V = true then
        W ← w0
      else
        W ← w0 + 1
      end if
    end if
    if δ_j[W − 1] ≠ 0 then
      R ← ECADD(R, δ_j[w0]*P_j)
    end if
    δ_j << 1
  end for
end for
return (R)
```

알고리즘 1: Interleave method using fracwMOF

```
INPUT: V check for reduction, upper bound, w0-bit MOF strings
OUTPUT: fractionalwMOF code
check ← true, multiplier ← 1, divisor ← 1, SUM ← 0,
position ← 0, w ← w0 + 1, RUN ← true
while RUN do
  for i ← w to 0 do
    if check && δ_i - δ_(i−1) then
      position ← i; check ← false; divisor ← 2^(w−position)
    end if
    SUM ← SUM + multiplier * (d_i − d_(i−1))
    multiplier ← multiplier * 2
    fracwMOF[n] ← 0
  end for
  SUM ← SUM/divisor
  if SUM > upper then
    w ← w − 1, divisor ← 1, multiplier ← 1, SUM ← 0,
    check ← true, V ← false
  else
    fracwMOF[position] ← SUM
    RUN ← false, V ← true
  end if
end while
return (fracwMOF[w], V)
```

알고리즘 2: Generation of fractional wMOF: fracwMOF (on-the-fly)

대한 유연성을 제공하기 위하여 Interleave method using fractional wMOF를 제안한다.

전체적인 알고리즘의 동작은 다음과 같다. 먼저 사용가능한 precomputed table의 크기 q와 기본 윈도우 크기 w_0로부터 fractional window w_1의 크기를 구한다.

$$wastedmemory = availablememory - tablesizeofwMOF$$
$$r = q - (2^{w_0-2} - 1) = q - 2^{w_0-2} + 1, w_1 = \frac{r}{2^{w_0-2}}$$

w_0와 w_1을 모두 얻은 후에 알고리즘1의 step5에서 fractional wMOF가 가질 수 있는 upper bound를 계산한다. 만약 알고리즘 2의 step 5~13을 걸쳐 계산된 값이 upper bound보다 크면 윈도우의 크기가 $w = w_0 + w_1$에서 기본 윈

도우 크기인 w_0로 줄어든다 (reduction). 즉, upper bound를 넘지 않는 범위 내에서 기본 윈도우 보다 큰 윈도우를 사용하고, 그렇지 않은 경우에는 기본 윈도우 크기를 사용한다. MOF에서 fracwMOF를 계산하는 과정은 weighted sum ($=\sum_{i=0}^{w} 2^i \cdot (d_{w-i} - d_{w-i-1})$) 을 통하여 이루어진다. Main Interelave method는 완성된 fractional window들을 (두개의 스칼라에 대한) 이용하여 ECDBL과 ECADD를 수행한다. 제안된 Interleave method using fracwMOF은 다음과 같은 성능과 메모리 소비량을 제공한다. (두 개의 스칼라가 같은 윈도우 크기를 사용한다고 가정)

Operation	Precomputed table
$nECDBL + \frac{2n}{w_0 + w_1} ECADD$	$2\{(1+w_1)2^{w_0-1} - 1\}$

IV. 실험 및 분석

제시한 방법의 타당성을 확인하기 위하여 10000만의 실험에 대한 평균을 내었다. 랜덤하게 생성된 163 비트 스칼라에 대하여 기존의 4가지 알고리즘과 제안 알고리즘인 Interleave-fracwMOF를 구현하여 적용하였다. 그림1은 각 알고리즘의 non-zero bit density와 이에 대응하는 precomputed table의 크기에 대한 상관관계를 표현한다. 실험을 통하여 각 알고리즘의 공통된 특성이 확인되었다. 즉, non-zero bit density가 낮아질수록 멀티 스칼라 곱셈의 성능은 향상되지만 precomputed table의 크기가 커지기 때문에 메모리 소비량은 커졌다. 또한 제안 알고리즘을 제외한 나머지 알고리즘들은 특정한 메모리 크기에서만 적용이 가능하였다. 뿐만 아니라 non-zero density가 낮아질수록 특정 메모리 크기들 사이의 간격이 증가하였다. 이로부터 기존의 알고리즘들은 메모리에 대한 유연성이 떨어진다는 것을 확인할 수 있었다. 반면 제안 알고리즘은 기존의 알고리즘들과 같은 성능을 유지하면서 어떠한 메모리 크기에서도 적용 가능하였다. 비록 제안 알고리즘의 성능이 사용가능한 precomputed table의 크기가 10일 때 "Shamir-Erik" [4]의 성능에 근소하게 뒤지긴 하지만 메모리에 대한 유연성과 확장성면에서 뛰어나다.

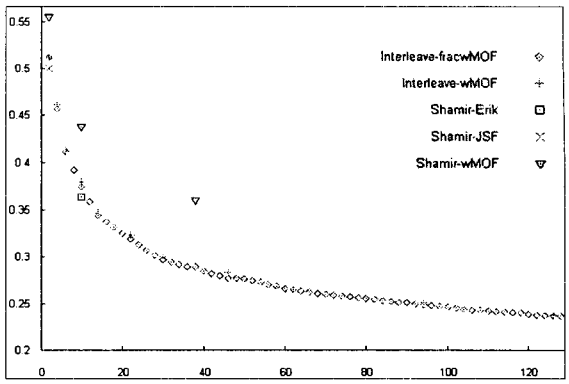

그림 1: 기존의 알고리즘과 제안 알고리즘의 비교
(x축: precomputed table size, y축: non-zero density)

V. 결론

본 논문에서는 ECDSA의 검증과정에서 주된 계산 부하인 멀티 스칼라 곱셈을 위한 효율적인 알고리즘을 제시하였다. 제안 알고리즘은 자원이 제약된 디바이스에서도 충분히 사용될 수 있도록 메모리 사용량과 계산 부하를 최소화하였으며 어떠한 메모리 크기에서도 메모리의 낭비 없이 최적의 효율성을 제공한다. 실험을 통하여 기존의 다른 알고리즘에 비하여 메모리 확장성과 성능향상 면에서 우수함을 확인하였다.

[참고문헌]

[1] Miller, V.S.: Use of Elliptic Curves in Cryptography. CRYPTO'85, LNCS218, (1986). pp. 417-426

[2] K. Okeya, and et al.: Signed Binary Representation Revisited. CRYPTO 2004. LNCS3152. (2004). pp. 123-139

[3] Bodo Moller: Fractional Windows Revisited: Improved Signed-Digit Representation for Efficient Exponentiation. ICISC 2004. LNCS3506. (2004). pp. 137-153

[4] E. Dahmen, and et al.: An Advanced Method for Joint Scalar Multiplications on Memory Constraint Devices. Proc of ESAS 2005, LNCS3813. (2005). pp. 189-204

[5] J. Solinas: Efficient Arithmetic on Koblitz Curves. Design, Codes and Cryptography. (2000). 19:195-249

[6] K. Schmidt-Samoa, and et al.: Analysis of Fractional Window Recoding Methods and Their Application to Elliptic Curve Cryptosystems. IEEE Transaction on computers. Vol. 55 (Jan, 2006)