Montgomery 알고리듬을 이용한 새로운 곱셈 방식

오중효[0], 하재철, 이국희, 문상재
경북대학교 전자전기공학부

A New Multiplication Method Using Montgomery Algorithm

Joong-Hyo Oh, Jae-Cheol Ha, Kook-Heui Lee, and Sang-Jae Moon
School of Electronics and Electrical Engineering, Kyungpook National Univ.

본 연구는 정보통신연구관리단의 국책기술개발사업 지원에 의해서 이루어졌습니다.

요 약 문

Montgomery 알고리듬은 모듈라 연산을 고속으로 수행하는 방법이다. 그러나 이는 연산할 수를 n-residue로 변환하는 전처리 단계가 필요하다. 이러한 residue 변환에 필요한 오버헤드로 인해 한번의 곱셈에는 비효율적이다. 본 논문에서는 Montgomery 알고리듬을 사용하여 한번의 곱셈을 효율적으로 수행하는 방법을 제안한다.

1. 서 론

공개키 방식에 근거한 암호 시스템에서 실시간 구현을 위해서는 모듈라 곱셈을 빠르게 수행해야 한다. 모듈라 곱셈은 큰 수에 대한 곱셈과 곱한 결과를 모듈라 하는 두 동작으로 이루어져 있으며 모듈라 연산이 곱셈의 속도를 좌우할 수 있다. 모듈라 연산 방법에는 고전적인 방법, Barrett, Yang, Takagi 및 Montgomery 알고리듬 등이 있으며 이 중 Montgomery 알고리듬이 고속이며 널리 이용된다[1-6].

Montgomery 알고리듬은 연산할 수를 n-residue로 변환하여 모듈라 연산을 수행하고 결과를 다시 일반 residue로 변환한다. 이로 인해 적은 수의 곱셈에서는 다른 모듈라 감소 알고리듬에 비해 느리다. 그러나 residue 변환으로 인한 오버헤드를 보상할 수 있는 멱승 연산의 경우에는 빠르게 모듈라 연산을 수행할 수 있다.

본 논문에서는 Montgomery 알고리듬을 이용하여 residue 변환 없이 한번의 곱셈을 효율적으로 수행할 수 있는 방법을 제안한다. 제안한 곱셈 방식을 구현하여 기존의 곱셈 방식과 성능을 비교 검토한다.

2. Montgomery 알고리듬

모듈라 감소는 다음과 같이 정의된 값에 대해 $c \bmod n$을 계산하는 것이라 가정한다. 여기서 c는 최대 k자리인 두 수의 곱이며 b는 radix를 의미한다.

$$n = n_{k-1}b^{k-1} + \cdots + n_1 b + n_0$$
$$c = c_{j-1}b^{j-1} + \cdots + c_1 b + c_0 \quad where\ k < j \le 2k$$

Montgomery는 고전적인 방법과 달리 반복적인 나눗셈이 없는 모듈라 감소 알고리듬을 제안했다[6]. Montgomery 알고리듬은 모듈라 n 잉여집합을 n-residue라 정의하고 이 집합 내에서 빠른 모듈라 감

소를 수행한다.

임의의 정수 x의 R에 대한 n-residue는 식 (1)과 같이 나타낼 수 있으며 완전한 잉여계를 형성한다.

$$\{x \cdot R \bmod n \mid 0 \le x < n\} \tag{1}$$

Montgomery 알고리듬의 수행절차는 다음과 같다. 먼저 $R > n$이고 $gcd(R, n) = 1$인 R을 선택한다. 이때 R은 기수 b에 대해 $R = b^k$이다. 모듈라 감소는 n-residue내의 임의의 정수 $t(0 \le t < Rn)$에 대해서 $t' = REDC(t) = \frac{t + m \cdot n}{R} = tR^{-1} \bmod n$을 계산하는 것과 같다. 여기서 m은 $t \cdot n' \bmod R$이다. R^{-1}와 n'는 $R \cdot R^{-1} - n \cdot n' = 1$을 만족하는 값이며 extended Euclidian 알고리듬을 이용하여 쉽게 계산할 수 있다. 이때 t'는 $0 \le t' < 2n$을 만족하며 정확한 값을 위해 한번의 뺄셈이 필요하다.

Montgomery 알고리듬을 이용할 경우 곱할 두 수를 n-residue로 변환하는 과정이 필요한데 $R' = R^2 \bmod n$을 사전에 계산해 두면 $x \cdot R \bmod n$을 $REDC(x \cdot R')$로 쉽게 구할 수 있다.

Montgomery 모듈라 감소는 t에 n의 적당한 배수를 더하여 그 값이 R의 배수가 되도록 하는 것이다. 즉, 하위 k자리가 0이 되도록 하면 된다. 그러므로 한번에 m을 계산하지 않고 각 자리에 대해서 $n_0' = -n_0^{-1} \bmod b$를 구하고 $m_i = t_i \cdot n_0' \bmod b$를 계산하여 더하는 것이 효율적이다[7]. 이때 필요한 곱셈수는 $k(k+1)$ 정도이며 $n_0^{-1} = -1$이 되도록 n을 선택하면 k^2번 정도의 곱셈이 필요하다. 이를 나타낸 것이 그림 1이다.

$REDC(t)$:
  for $i = 0$ *to* $k-1$ *step* 1
    $m = t_i \cdot n_0' \bmod b$
    $t = t + m \cdot n \cdot b^i$
  $t = t$ *div* R
  if $t \ge n$ *then return* $t - n$
    else return t

Fig. 1. Montgomery modular algorithm.

Montgomery 알고리듬을 이용하여 모듈라 곱셈을 수행할 경우 곱하는 수를 n-residue로 변환하여야 하고 곱셈 결과를 모듈라 감소 후 다시 일반 residue로 변환하여야 한다. 즉, $c = x \cdot y \bmod n$은 다음과 같이 이루어진다.

$$x' = REDC(x \cdot R') = x \cdot R \bmod n$$
$$y' = REDC(y \cdot R') = y \cdot R \bmod n$$
$$c' = REDC(x' \cdot y') = xyR \bmod n$$
$$c = REDC(c') = xy \bmod n$$

또한 다음과 같이 한 수만 residue 변환시켜 곱셈을 수행할 수도 있다. 이 경우는 마지막의 residue 변환을 생략할 수 있다.

$$x' = REDC(x \cdot R') = x \cdot R \bmod n$$
$$c = REDC(x' \cdot y) = xy \bmod n$$

3. 제안하는 곱셈 방식

다음과 같이 수행하면 한번의 곱셈에서 n-residue로의 변환 없이 Montgomery 알고리듬을 효과적으로 이용할 수 있다.

k자리의 두 수 x, $y(< n)$의 모듈라 곱셈을 수행한다고 하자.

$$x = \sum_{i=0}^{k-1} x_i b^i, \quad y = \sum_{i=0}^{k-1} y_i b^i \quad \text{where } b \text{ is a radix}, \quad 0 \le x_i, y_i < b$$

두 수의 모듈라 곱, $c = x \times y \bmod n$은 식 (2)와 같이 나타낼 수 있다.

$$\begin{aligned} c = xy \bmod n &= (c_H b^k + c_L) \bmod n \\ &= ((c_H b^k \bmod n) + (c_L \bmod n)) \bmod n \\ &\text{where } c_H = \sum_{i=k}^{2k-1} c_i b^{i-k}, \; c_L = \sum_{i=0}^{k-1} c_i b^i \end{aligned} \tag{2}$$

위의 식에서 $c_H b^k \bmod n$은 Montgomery 알고리듬에서 사용되는 $R(= b^k)$을 이용하여 $c_H R \bmod n$으로 나타낼 수 있다. 그리고 $c_H R \bmod n$은 c_H를 n 잉여집합으로 변환하는 것과 같은 연산이다. 그러므로 Montgomery 알고리듬을 이용하여 $c_H R \bmod n$을 $REDC(c_H \cdot R')$로 쉽게 구할 수 있다. 여기서 R'는 사전에 계산되어진 $R^2 \bmod n$이다. 이를 이용하여 모듈라 곱셈을 하는 구체적인 알고리듬은 그림 2와 같다.

```
Mont_Mul(x, y, c) :
        c = x × y
        c_H = c div R ,  c_L = c mod R
        c_H' = REDC(c_H · R')
        c = c_H' + c_L
        while (c > n) do
            c = c − n
        return c
```

Fig. 2. A new modular multiplication using Montgomery algorithm.

4. 실험 및 고찰

Montgomery 알고리듬을 사용하여 기존의 방식대로 곱할 두수를 n-residue로 변환하여 곱셈을 수행

하면 $7k^2+4k$번 정도의 단정도 곱셈이 필요하며 두 수 가운데 한쪽만 변환하여 곱셈을 수행하면 $4k^2+2k$번의 단정도 곱셈이 필요하다. 그러나 제안된 방식으로 모듈라 곱셈을 수행하면 $3k^2+k$번 정도의 단정도 곱셈이 필요하다.

제안한 곱셈 방식을 C 언어로 구현하여 고전적인 모듈라 감소 방법을 사용한 곱셈 및 기존의 Montgomery 곱셈 방식과 성능을 비교 분석하였다. 사용 시스템은 IBM PC 100MHz이며 Borland C 컴파일러를 사용하였다.

각 길이에 따라 $x \times y \bmod n$을 수행하여 결과를 표 1에 나타내었다. Mont_1은 원래의 방식과 같이 곱할 두 수 모두를 residue 변환한 방식이며 Mont_2는 한쪽만 residue변환한 방식이다.

Table 1. Comparison of execution time for one multiplication.

[msec]

method \ |bit|	160	256	512	768	1024
Classical	0.2	0.4	1.6	3.6	6.4
Mont_1	0.3	0.8	3.1	6.8	12.1
Mont_2	0.2	0.4	1.8	4.1	7.1
Proposed	0.1	0.4	1.4	3.1	5.5

기존의 Montgomery 알고리듬을 이용한 곱셈은 n-residue로의 변환이 필요하므로 고전적인 모듈라 감소 방법을 이용한 곱셈에 비해 느리다. 그러나 표 1에 나타난 결과에서 제안한 곱셈 방식은 residue 변환이 없으므로 고전적인 모듈라 감소를 이용한 곱셈보다 빠른 성능을 가진다.

5. 결 론

Montgomery 알고리듬의 단점인 한번의 곱셈에서 효율이 떨어지는 점을 개선시키는 곱셈방법을 제시하였고 제안한 곱셈 방식의 수행시간을 기존의 방식들과 비교하였다. 제안한 곱셈 방식을 이용하면 Montgomery 알고리듬을 멱승과 곱셈에 효과적으로 적용할 수 있다.

참 고 문 헌

[1] David Naccache, David M'Raihi, "Arithmetic Co-processors for Public -key Cryptography : The State of Art," *CARDIS '96,* pp.39-58, 1996.

[2] D.E.Knuth, *The Art of Computer Programming, vol.2, Seminumerical Algorithms*, 2nd Edition, Addison-Wesley, 1981.

[3] H.Morita and C.H.Yang, "A Modular Multiplication Algorithm Using Lookahead Determination," *IEICE Trans. Fundamentals*, vol.E76-A, no.1 Jan. 1993.

[4] N. Takagi, "A Multiple-Precision Modular Multiplication Algorithm with Triangle Additions," *IEICE Trans. Inform. and syst.* vol.E78-D, pp.1313-1315, Oct. 1995.

[5] P.Barrett, "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor," *In Advances in Cryptology - CRYPTO '86*, pp.311-323, Springer -Verlag, 1987.

[6] P.L.Montgomery, "Modular Multiplication without Trial Division," *Math. Comp.*, vol.44, pp.519-521, 1985.

[7] S.R.Dusse and B.S.Kaliski Jr., "A Cryptographic Library for the Motorola DSP56000," *In Advances in Cryptology - EUROCRYPT '90*, pp.230-244, Springer-Verlag, 1991.