선계산을 이용한 고속 모듈라 멱승법

김종덕 * 박일환 † 이성재 ‡ 임종인 §

1998년 11월 27일

요 약

대부분의 공개키 암호시스템은 큰 수의 모듈라 멱승을 기본으로 한다. 본 논문에서는 대표적인 고속 모듈라 멱승 알고리즘인 몽고메리법에 대해 윈도우법을 결합하여 연산한 결과 효율성이 향상됨을 보였고, 이러한 결과를 윈도우법과 비교하여 실험하였다.

1 들어가며

1976년 Diffie와 Hellman에 의해 공개키 암호시스템이 처음 제안[1]된 후 많은 공개키 암호시스템이 제안되었다. 그런데 대부분의 공개키 암호시스템은 큰 수의 모듈라 멱승이 기본연산으로서 다뤄진다. 이러한 방식의 대표적인 암호시스템은 인수분해의 어려움과 이산대수의 어려움에 각각 근거한 RSA[2]와 ElGamal[3] 등이 있다. 그러므로 공개키 암호시스템은 모듈라 멱승법이 얼마나 고속으로 처리되는가에 따라 그 효율성을 판가름할 수 있다.
한편 모듈라 멱승법은 주어진 수에 대해 멱승(exponentiation)을 취하고 그 결과를 주어진 수에 대한 나머지 값을 취하는 (reduction)과정으로 이루어진다. 따라서 효율적인 모듈라 멱승법은 고속의 멱승법과 고속으로 잉여류를 계산하는 알고리즘을 원활히 결합할 때에만 가능하다.
본 논문에서는 대표적인 멱승법인 몽고메리법 [4]에 윈도우법을 결합하여 선계산되는 정보를 이용하여 모듈라 멱승을 수행하였고, 그 효율성을 윈도우법과 비교하였다.

2 다정도 수의 연산

일반적으로 다정도 (multi precision) 수는 적당한 진법을 이용한 배열로 처리된다. 이때 처리상의 잇점을 살리기 위해 2의 멱승에 대한 진법이 대체로 이용

*jdkim@math.korea.ac.kr 고려대학교 기초과학연구소
†ilhpark@etri.re.kr 전자통신연구원 부호기술부
‡lsj91@gauss.korea.ac.kr 고려대학교 대학원 수학과
§jilim@tiger.korea.ac.kr 고려대학교 자연과학대학 수학과

된다. 예컨대 b진법에 대한 다정도 수 n은 다음과 같이 표현될 수 있다.

$$n = \sum_{i=0}^{n-1} n_i b^i, \quad 0 \le n_i \le b-1.$$

실제 구현은 보통의 32비트 컴퓨터에서 원활히 운용될 수 있도록 진수 b를 2^{32}으로 취했다.
한편 두 n자리수를 종이와 연필을 사용하여 곱하면 전체 n^2번의 단정도 (single precision) 곱셈과 그만큼의 덧셈이 요구된다. 한편 같은 수를 제곱하는 경우에는 단정도 곱셈에 대하여 전체의 $\frac{1}{2}$의 정보가 필요하게 되므로 곱셈횟수를 대략 반으로 줄일 수 있다.
고속 곱셈법 중 대표적인 방식으로는 이른바 divide and conquer 방식에 근거한 Karatsuba의 방법이 있다. 즉, 전체 $2m$비트의 두 수 a, b를 곱할 경우 a, b의 상위 m비트와 하위 m비트를 각각 a_h, b_h와 a_l, b_l 이라 하면 $a \times b$는 다음과 같이 계산된다.

$$\begin{aligned} a \times b &= (a_h 2^m + a_l)(b_h 2^m + b_l) \\ &= a_h b_h 2^{2m} + ((a_h + a_l)(b_h + b_l) - a_h b_h - a_l b_l) 2^m + a_l b_l \end{aligned}$$

보통 덧셈과 뺄셈에 드는 비용은 곱셈에 대한 것과 비교할 때 상대적으로 미미하므로 논외로 한다면, 각 m비트 수들 간의 곱셈에는 m^2번의 단정도 곱셈이 요구되므로 전체 $3m^2$번의 곱셈이 수행된다. 일반적인 방식으로 곱셈을 할 경우 전체 곱셈횟수가 $(2m)^2 = 4m^2$이었던 것을 볼 때 $\frac{1}{4}$배의 속도향상을 꾀할 수 있다. 이런 방법을 재귀적 (recursive)으로 수행한다면 충분히 빠르게 곱셈법을 구현할 수 있다. (512 ~ 2048 비트에 대한 최적 반복횟수 (depth)에 대한 실험은 논문 [5] 참조.)

2.1 모듈라 감소법

a를 m으로 나눈 나머지 $(a \bmod m)$ 는 유일하게 정해지는 몫과 나머지 $q, r(a = aq + r,\ 0 \le r \le a)$에 대해 r을 구하는 과정이다. 이때 n을 a로 나눈 후의 나머지(r)가 최소잉여류에 속해야 한다.
그런데 나눗셈은 기본적인 다정도 연산중에서 가장 복잡하고, 따라서 비용이 많이 든다. 그렇기 때문에 나눗셈을 직접적으로 이용한 모듈라 감소법 (고전적인 감소법[6])은 효과적이지 않다.
이런 기본적인 감소법 이외에 속도가 빠른 감소법에는 몽고메리 감소법[4], Barret 감소법[7] 등이 있는데 일반적으로 큰 수 (> 512)의 감소에는 몽고메리 알고리듬이 가장 효율적이라고 알려져 있다. [8]
몽고메리 감소법은 고전적인 모듈라 감소를 수행하지 않고 효율적으로 모듈라 곱셈을 가능하게 해주는 방식이다.
양의 정수 m에 대하여 두 수 n, R이 $m < R,\ (m, R) = 1,\ 0 \le n < mR$를 만족한다고 하자. 그러면 서로소인 두 수 m, R에 대해 확장된 유클리드 알고리듬[9]에 의해 법 m에 대한 R의 곱셈에 대한 역원 R^{-1}을 구할 수 있다. 그리고 $U \equiv -nm^{-1} \bmod R$이라 하면 $n + Um \equiv 0 \bmod R$이므로 다음의 식을 정의할 수 있다.

$$MontRed(n, m) \equiv \frac{n + Um}{R} \equiv nR^{-1} \bmod m \qquad (1)$$

여기서 R은 진수의 멱승인 수를 택하게 되는데 그 이유는 법 R에 대한 연산이 수월하게 하기 위함이다. 이 방법을 이용해서 몽고메리 곱셈법과 멱승법을 효과적으로 수행할 수 있다.

2.2 고속 멱승법

멱승법은 여러가지가 있으나 약간의 사전계산 (pre-computation)을 이용하는 윈도우법과 몽고메리법이 대표적으로 사용되는 고속 멱승법이다.
윈도우 멱승법은 주어진 윈도우의 크기에 따라 선계산을 하고, 이를 이용하여 멱승을 취함으로서 평균 곱셈횟수를 줄이는 알고리즘이다. 지수가 $e = (e_t e_{t-1} \cdots e_1 e_0)$일 때 g^e를 계산한다고 가정하자. 이때 주어진 윈도우 크기(k)에 대하여 지수가 홀수이면서 k를 넘지않는 모든 멱승 $(g^1, g^3, g^5, \cdots, g^{2^k-1})$을 사전계산한다. 그리고 이 계산결과를 이용하여 지수를 상위비트로부터 하위비트에 이르기까지 k비트보다 작으면서 홀수인 블럭으로 나누고 먼저 계산된 결과를 토대로 하여 자릿수를 올려주는 방식으로 셈한다.[10]
한편 몽고메리 멱승법을 이용하기 위해 식 (1)을 이용하여 다음의 식을 정의하자.

$$Mont(u, v, m) \equiv MontRed(uv, m) \tag{2}$$

그러면 $x^e \bmod m$을 구한다고 할 때 식 (2)를 이용하여 $x^e R \bmod m$을 구한 후, 몽고메리 감소법에 의해 $MontRed(x^e R, m)$을 계산하면 원하는 결과를 얻을 수 있다.
이때 길이 l의 지수 e에 대하여 통상 1비트씩 멱승을 수행하게 되는데, 양의 정수 k에 대하여 가능한 모든 경우인 $2^k - 1$가지의 멱승을 테이블로 미리 계산하면 $\frac{l}{k}$만큼 연산 수행 횟수를 줄일 수 있다.

3 실험과 그 결과

실험방식은 길이가 같은 3가지 난수 a, e, n을 발생하고 $a^e \bmod n$을 계산하였는데, 이때 실험의 공정성을 기하기 위해 주어진 컴퓨터 상에서 각각 1,000,000번씩 반복수행한 후의 평균 속도를 구하고, 그 결과를 표 1과 같이 정리하였다.
멱승법은 윈도우법의 경우 윈도우의 크기를 각각 2,4,8비트로, 그리고 몽고메리법은 각각 1,2,4,8비트로 운용되었다.
이때 사용된 컴퓨터와 운영체제는 다음과 같다. 각 기종 모두 gcc를 이용하여 컴파일하였다.

Linux Redhat Linux 4.0, Pentium 120 Mhz

Solaris SunOS 5.5.1, Ultra Sparc 167 Mhz

Pen233 Windows 95, Pentium 233 Mhz

Pen333 Windows 98, Pentium II 333 Mhz

컴퓨터의 성능이나 프로그래밍의 기술에 따라 약간의 결과가 달라지고, 속도가 더 빨라질 수 있겠으나 상대적인 속도는 몽고메리 방식보다는 윈도우 방식이 더 빠르다는 점과, 선 계산량을 많이 취할수록 고속화되기 보다는 어느 정도의 한계가 주어져서 윈도우법이나 몽고메리법 모두 윈도우의 크기가 4일 때 가장 효율이 좋아짐을 발견할 수 있었다.

Platform	비트수 / 연산법	1024	2048	4096	8192
Pen120	E2	0.42	8.31	8.34	8.56
	E4	0.38	7.81	7.87	7.94
	E8	0.51	8.62	8.68	8.70
	M1	1.29	23.91	29.34	40.98
	M2	1.19	22.14	27.11	38.75
	M4	1.09	20.78	25.49	36.46
	M8	1.38	*	*	*
Solaris	E2	0.72	35.82	51.00	51.21
	E4	0.64	33.42	46.42	46.45
	E8	0.85	40.65	48.15	48.14
	M1	1.55	73.13	129.49	146.19
	M2	1.43	68.48	120.43	136.33
	M4	1.30	64.26	111.34	127.16
	M8	1.77	*	*	*
Pen233	E2	0.21	3.86	5.52	5.62
	E4	0.19	3.63	5.23	5.69
	E8	0.25	4.56	5.79	5.77
	M1	0.65	11.75	20.02	60.42
	M2	0.59	11.04	18.93	88.42
	M4	0.54	10.19	16.49	34.05
	M8	0.70	*	*	*
Pen333	E2	0.15	2.92	2.91	2.74
	E4	0.13	2.74	2.74	2.57
	E8	0.17	3.02	3.02	2.83
	M1	0.45	8.55	8.56	12.98
	M2	0.41	7.88	8.08	11.97
	M4	0.38	7.02	7.67	11.33
	M8	0.48	*	*	*

표 1: 1회 평균 모듈라 멱승 시간 (단위 : sec)

4 결론

본 논문에서는 윈도우법과 몽고메리법에 대해 선계산량에 차등을 두어 모듈라 멱승을 수행하였다. 전체적으로 볼 때 윈도우법이 몽고메리법보다 더 효율적이었다.
그리고 몽고메리법에 선계산한 테이블을 이용하면 그렇지 않은 경우보다 고속화함을 보였다.
한편 선계산량에 비례해서 고속화되기 보다는 주어진 수의 크기와 선계산을 허용하는 비트수는 어느정도 상관관계가 있음을 알 수 있었고, 본 실험에서는 4비트를 선계산한 결과가 가장 효율적이었다.

참조 서적

[1] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, November 1976.

[2] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[3] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[4] P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[5] 황효선 and 임채훈. 공개키 암호시스템의 고속 구현. In *Conferance on Information Security and Cryptography 97*, 1997.

[6] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 1997. Third edition, 1997.

[7] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In A.M. Odlyzko, editor, *Proc. Crypto 86*, pages 311–323. Springer-Verlag, 1987. Lecture Notes in Computer Science No. 263.

[8] Antoon Bosselaers, René Govaerts, and Joos Vandewalle. Comparison of three modular reduction functions. In Douglas R. Stinson, editor, *Proc. Crypto 93*, pages 175–186. Springer, 1994. Lecture Notes in Computer Science No. 773.

[9] K.H. Rosen. *Elementary Number Theory and its Applications, (Third Edition)*. Addison Wesley, 1993.

[10] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC press, 1997.