효율적인 모듈러 멱승 연산을 위한 그래프 모델링 방법[1)]

박치성[O1] 김지은[2] 김동규[1]
[1] 부산대학교 컴퓨터공학과
[2] 한국전자통신연구원
[1]{cspark, dkkim}@islab.ce.pusan.ac.kr
[2]jekim@etri.re.kr

Graph Modeling Method for Efficient Computation of Modular Exponentiation

Chi Seong Park[O1] Ji Eun Kim[2] Dong Kyue Kim[1]
[1] Pusan National University, Pusan 609-735, Korea
[2] Electronics and Telecommunications Research Institute, Daejeon, 305-700, Korea

요 약

모듈러 멱승은 양수 x, E, N에 대하여 $x^E \bmod N$으로 정의된다. 모듈러 멱승 연산은 대부분의 공개키 암호화 알고리즘과 전자서명 프로토콜에서 핵심적인 연산으로 사용되고 있으므로, 그 효율성은 암호 프로토콜의 성능에 직접적인 영향을 미친다. 따라서 모듈러 멱승 연산에 필요한 곱셈 수를 감소시키기 위하여, 슬라이딩 윈도우를 적용한 CLNW 방법이나 VLNW 방법이 가장 널리 사용되고 있다. 본 논문에서는 조합론(combinatorics)에서 많이 응용되는 그래프 모델을 모듈러 멱승 연산에 적용할 수 있음을 보이고, 일반화된 그래프 모델을 통하여 VLNW 방법보다 더 적은 곱셈 수로 모듈러 멱승을 수행하는 방법을 설명한다. 본 논문이 제안하는 방법은 전체 곱셈 수를 감소시키는 새로운 블록들을 일반화된 그래프 모델의 초기 블록 테이블에 추가할 수 있는 초기 블록 테이블의 두 가지 확장 방법들로써, 접두사 블록의 확장과 덧셈 사슬 블록의 확장이다. 이 방법들은 새로운 블록을 초기 블록 테이블에 추가하기 위해 필요한 곱셈의 수와 추가한 뒤의 전체 곱셈 수를 비교하면서 초기 블록 테이블을 제한적으로 확장하므로, 지수 E에 non-zero bit가 많이 나타날수록 VLNW 방법에 비해 좋은 성능을 보이며 이는 실험을 통하여 검증하였다.

1. 서 론

모듈러 멱승은 주어진 양수 x, E, N에 대하여 $x^E \bmod N$으로 정의된다. RSA[1], Elgamal 서명 시스템[2], DSS[3] 등 빠른 연산이 요구되는 대부분의 공개키 암호화 알고리즘과 전자서명 프로토콜에서 이를 핵심적인 연산으로 사용되고 있으므로, 모듈러 멱승 연산의 효율성은 암호 프로토콜의 성능에 직접적인 영향을 미친다. 하지만 이때의 모듈러 멱승은 매우 큰 수의 E를 필요로 할 뿐만 아니라, 한 번의 곱셈 연산 자체에도 매우 큰 cost가 필요하다. 따라서 모듈러 멱승 연산의 효율성은 크게 곱셈 자체의 속도 증가, 혹은 최대 $E-1$번이었던 곱셈 수의 감소, 이렇게 두 방향으로 연구되어 왔다. 본 논문은 후자를 따라 곱셈 수를 감소시키는 방향으로 접근한다.

모듈러 멱승 연산에 필요한 곱셈 수를 감소시키기 위해 새로운 곱셈 순서를 찾는 기존의 방법들로 이진 방법[4], M-ary 방법[4], 덧셈 사슬 방법[5, 6], 슬라이딩 윈도우[7] 등이 있으며 이들 중 슬라이딩 윈도우를 적용한 CLNW 방법과 VLNW 방법이 가장 널리 사용되고 있다. VLNW 방법은 CLNW 방법보다 좋은 성능을 보이지만, CLNW 방법을 비롯한 기존의 다른 방법들과 달리 두 개의 인자를 사용함으로써 E의 특징에 따른 인자의 최적 값을 하나 더 고려해야 하는 어려움이 있다.

본 논문은 VLNW 방법을 포함한 기존의 알고리즘들을 하나의 인자로 일반화할 수 있는 그래프 모델을 설명하고, 전체 곱셈 수를 감소시키는 새로운 블록들을 일반화된 그래프 모델의 초기 블록 테이블에 추가할 수 있는 초기 블록 테이블의 두 가지 확장 방법을 제안한다. 이 방법들을 통해 지수 E에 non-zero bit가 많이 나타날수록 VLNW 방법에 비해 좋은 성능을 보이며, 그 타당성은 non-zero bit의 비율을 조정하면서 임의로 생성한 270개의 E에 대한 실험 결과로 제시한다.

본 논문은 총 5장으로 구성되는데, 먼저 2장에서 기존 알고리즘들을 살펴보고, 3장에서 일반화된 그래프 모델을 설명하며, 4장에서 일반화된 그래프 모델의 초기 블록 테이블 확장 방법을 제안하며, 5장에서 실험을 통해 본 논문이 제안하는 방법과 기존 알고리즘들의 성능을 비교한다.

2. 기존 알고리즘 개요

본 장에서는 모듈러 멱승 연산에 필요한 곱셈 수를 감소시키기 위해 새로운 곱셈 순서를 찾는 기존의 몇 가지 알고리즘들을 살펴본다.

2.1 이진(Binary) 방법

가장 기본적인 이진 방법은 E를 이진수 E_B로 표현하고 왼쪽에서 오른쪽 방향으로 한 bit씩 읽으면서 현재 값을 제곱하고 non-zero bit일 때 현재 값에 x를 곱하는 방법이다. E를 이진수로 표현하여 곱셈 수를 $E-1$에서 kn(k는 상수, $n=\log_2 E$)로 감소시켰다.

2.2 M-ary 방법

이진 방법을 $M \geq 2$으로 일반화한 M-ary 방법은 $d=\log_2 M$ bits의 블록 단위로 E_B를 읽으며 현재 값을 M승, 즉 d번 제곱하면서 현재 값에 블록의 값을 곱하는 연산이다. M개의 초기 블록을 미리 계산해야 하지만 이 계산 결과를 반복되는 E_B의 블록들에 재사용할 수 있으므로, 적절한 d 인자를 선택하여 곱셈 수를 감소시킬 수 있다.

2.3 슬라이딩 윈도우(Sliding Window)

zero bit로만 이루어진 블록을 zero 블록, 그 외의 블록을 non-zero 블록이라 할 때, M-ary 방법에서 zero 블록은 블록

1) 이 논문은 2005년도 교육인적자원부 지방연구중심대학 육성사업의 지원에 의하여 연구되었음

의 값을 곱하는 부가적인 곱셈을 필요로 하지 않았다. 슬라이딩 윈도우는 zero 블록의 가변 길이를 허용하여 zero 블록들의 평균적인 길이를 늘인다. non-zero 블록의 가변 길이 여부에 따라 CLNW(Constant Length Non-zero Window) 방법과 VLNW(Variable Length Non-zero Window) 방법으로 나뉘는데, VLNW 방법은 non-zero 블록의 길이를 각각 고정되지 않은 $l \le d$로 결정하고 d 인자 외에 부가적인 q 인자를 통해 non-zero 블록의 처음과 마지막 bit를 non-zero로 보장한다. 따라서 non-zero 블록이 모두 홀수이므로 M-ary 방법과 d 인자가 같을 때 계산해야 할 초기 블록의 수는 $M/2$개로 줄어든다. VLNW 방법은 $128 \le n \le 2048$에 대하여 최적의 인자를 적용했을 때 M-ary 방법보다 약 5-8% 정도 곱셈의 수를 더 줄일 수 있는 것으로 알려져 있다.

이상의 방법들을 통해 전체 곱셈 수 c를 다음과 같이 정의할 수 있다.

$c(E_B, param) = i(param) + s(E_B, param) + m(E_B, param)$
$i(param)$: d 인자로 결정되는 초기 블록의 개수
$s(E_B, param)$: E_B를 읽을 때 필요한 제곱 $n-l$ (l은 첫 번째 블록의 길이)
$m(E_B, param)$: 첫 번째 블록을 제외한 non-zero 블록의 개수

$\therefore d \propto i(param),\ d \propto 1/s(E_B, param),\ d \propto 1/m(E_B, param)$

3. 일반화된 그래프 모델(Generalized Graph Model)

본 장에서는 d 인자 하나로 기존 알고리즘들을 일반화할 뿐 아니라, 그들 이상의 성능도 기대할 수 있는 새로운 모델링 방법을 설명한다.

일반화된 그래프 모델 $G=(V, E_g)$는 슬라이딩 윈도우와 동일한 $M/2$ 크기의 초기 블록 테이블, E_B의 각 bit에 대응하는 n개의 정점(node)들과 부가적인 2개의 source, sink 정점, 그리고 정점 u의 E_B 내 인덱스를 $i(u)$라고 할 때 다음과 같은 간선(edge)들 E_g로 정의된다.

$$E_g = \{(u, v) \mid |i(v) - i(u)| \le d, \forall (u, v) \in E_g\}$$

또한 모든 간선의 가중 값은 간선이 표현하는 블록 계산에 필요한 곱셈의 수로 정의된다. 예를 들어 초기 블록 테이블에 존재하는 길이 $l (\le d)$의 블록을 표현하는 간선의 가중 값은 $l+1$, 초기 블록 테이블에 존재하지 않는다면 $l+2$이다.

이상의 정의에 의한 일반화된 그래프 모델의 예는 그림 1의 (a)와 같다. (b)는 블록 011_2을 나타내는 간선의 예이다.

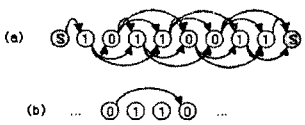

그림 1. d=3일 때 일반화된 그래프 모델의 예

G의 source 정점부터 sink 정점까지 선택되는 하나의 경로(path)는 E_B를 블록들로 나누는 방법, 즉 하나의 곱셈 순서를 나타낸다. 또한 하나의 경로가 가지는 가중 값의 합은 해당 곱셈 순서의 $s(E_B, param) + m(E_B, param)$가 된다. 따라서 G는 동일한 d 인자일 때의 기존 알고리즘을 모두 포함하는 일반화된 모델이며, G의 source 정점부터 sink 정점까지 최단 경로(Shortest Path)[10]는 동일한 d 인자일 때의 기존 알고리즘들 중 최소의 곱셈 수이거나 혹은 그들은 발견할 수 없는 그 이하의 곱셈 수이다.

4. 초기 블록 테이블을 이용한 모듈러 멱승 연산

본 장에서는 d 인자의 영향을 최소화하면서, 즉 $i(param)$를 유지하면서 G의 초기 블록 테이블을 확장하는 두 가지 방법을 제안한다. 초기 블록 테이블 내의 모든 초기 블록은 G에 간선으로 반영되므로, 길이가 길고 E_B 내에 빈번하게 나타나는 초기 블록을 통해 $s(E_B, param)$와 $m(E_B, param)$을 줄일 수 있다. 이제 그러한 블록을 발견하여 초기 블록 테이블에 추가하는 접두사 블록의 확장과 덧셈 사슬 블록의 확장, 그리고 두 가지 확장 방법의 통합에 대해서 설명한다.

4.1 접두사 블록의 확장

E_B의 접두사 블록을 적당한 길이까지 CLNW 방법으로 연산하며 중간 결과들을 초기 블록 테이블에 추가하는 방법이다. 연산된 접두사 블록은 이미 E_B 내에 존재하는 첫 번째 블록이라는 장점을 가지지만 블록 자체의 연산에 필요한 곱셈 수를 감쇄할 수 있는 것이어야 한다. 따라서 다음과 같이 확장한다.

확장된 접두사 블록이 홀수일 경우마다 확장된 블록에 해당하는 간선들을 추가하기 전과 후의 최단 경로를 비교한다. 홀수 접두사 블록이 자체 연산에 소요한 곱셈 수 이상 최단 경로를 줄이지 못한다면 이전 홀수 접두사 블록 확장 직후의 상태로 돌아가 접두사 블록의 확장을 종료한다.

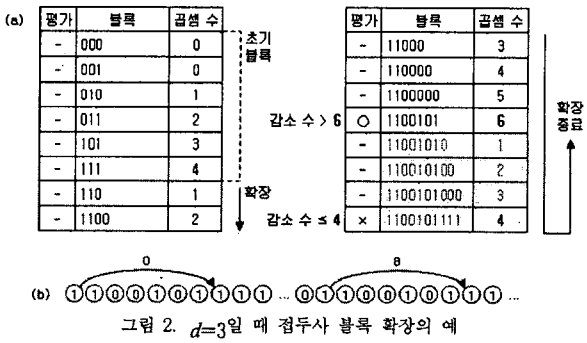

그림 2. d=3일 때 접두사 블록 확장의 예

그림 2의 (a)는 접두사 블록 확장의 예이다. 1100101111_2의 확장 시 감소되는 곱셈 수가 4보다 적으므로 1100101_2 확장 직후의 상태로 돌아가 접두사 블록의 확장을 종료한다. (b)는 가장 긴 접두사 블록 1100101_2의 간선이 G에 추가된 예이다.

4.2 덧셈 사슬 블록의 확장

임의의 두 초기 블록을 한 번 더해서 얻는 새로운 덧셈 사슬 블록을 초기 블록 테이블에 추가하는 방법이다. 덧셈 사슬 블록은 한 번의 곱셈으로 얻을 수 있는 새로운 초기 블록이지만 블록 자체의 연산에 필요한 한 번의 곱셈을 감쇄할 수 있는 것이어야 한다. 따라서 다음과 같이 확장한다.

현재 초기 블록 테이블 내에 있는 임의의 두 블록을 한 번 더해 홀수일 때만 후보(candidate) 블록 목록에 추가한다. 각각의 후보 블록에 해당하는 간선을 추가하기 전과 후의 최단 경로를 비교하여 최단 경로를 2 이상 줄이는 덧셈 사슬 블록에 한해서만 초기 블록 테이블로 옮기고, 나머지는 버린다. 초기 블록의 수가 늘어났다면 후보 블록 목록을 갱신하면서 후보 블록 목록이 빌 때까지 이 과정을 반복한다. 그림 3은 덧셈 사슬 블록 확장의 예이다.

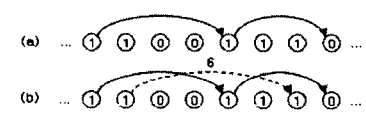

그림 3. 덧셈 사슬($1100_2+111_2=10011_2$) 블록 확장의 예

4.3 접두사 블록 확장과 덧셈 사슬 블록 확장의 통합

초기 블록들이 다양할수록 덧셈 사슬 블록의 확장으로 다양한 블록을 생성할 수 있으며 접두사 블록의 확장이 그것을 가능케 한다. 따라서 본 논문은 그림 4와 같이 두 가지 형태로 두 가지 확장 방법을 통합한다.

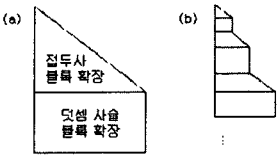

그림 4. 두 가지 확장 방법을 통합하는 두 가지 형태

그림 4의 (a)는 두 확장 방법을 순서대로 사용하는 형태로서, 불필요한 접두사 블록의 확장을 최소화한다. 반면 (b)는 하나의 접두사 블록을 확장할 때마다 덧셈 사슬 블록들을 확장하는 형태로서, 접두사 블록 자체의 평가를 덧셈 사슬 블록의 확장에 미루는 대신 (a)의 방법으로는 얻기 힘든 덧셈 사슬 블록을 얻을 수 있다.

5. 실험 결과

512bit, 1024bit, 2048bit의 n에 대하여 E_B 내의 non-zero bit 비율을 변화시키면서 10개씩 생성된 총 270개의 임의의 E에 대하여 VLNW 방법, 일반화된 그래프 모델, 일반화된 그래프 모델에 본 논문이 제시하는 방법을 적용한 "Graph + ExD", "Graph + ExM"을 비교하였다. "Graph"가 일반화된 그래프 모델이고, "Graph + ExD"는 그림 4 (a), "Graph + ExM"은 그림 4 (b)의 통합 형태이다.

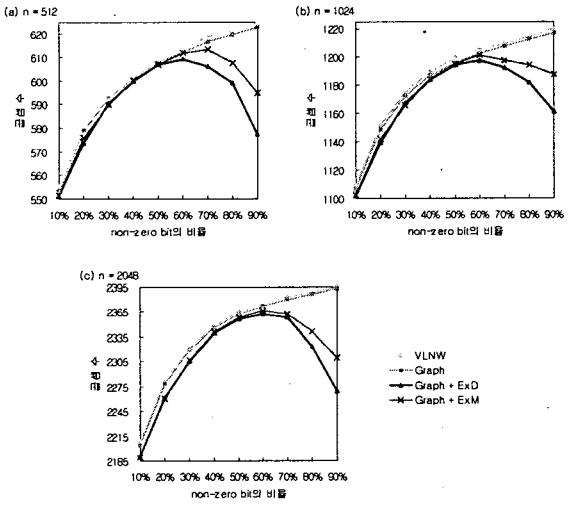

그림 5. n={512, 1024, 2048}일 때 non-zero bit 비율에 따른 평균 곱셈 수의 비교

그림 5는 좌측 상단부터 시계방향으로 n이 각각 512, 1024, 2048일 때의 비교 결과로서 가로축은 E_B 내의 non-zero bit 비율, 세로축은 각각의 방법이 찾은 평균 곱셈 수이다. non-zero bit의 비율이 증가할수록 본 논문이 제시하는 방법이 VLNW 방법에 비해 좋은 성능을 보인다. "Graph + ExD"과 "Graph + ExM"의 차이를 통해 non-zero bit의 비율이 증가할수록 덧셈 사슬 블록의 확장보다는 접두사 블록의 확장이 성능을 좌우한다는 것을 알 수 있다. 대부분의 경우에서 "Graph + ExD"의 성능이 가장 좋지만, non-zero bit의 비율이 낮은 몇몇 경우에서 "Graph + ExM"가 "Graph + ExD"보다 좋은 성능을 보였는데, 이때가 "Graph + ExD"로는 얻을 수 없는 덧셈 사슬 블록을 얻은 때임을 알 수 있다. 표 1은 n=512일 때 non-zero bit 비율에 따른 VLNW 방법과 "Graph + ExD"의 평균 곱셈 수 차이이다.

표 1. n=512일 때 VLNW 방법과 "Graph + ExD"의 평균 곱셈 수 차이

방법 \ 비율	10%	20%	30%	40%	50%	60%	70%	80%	90%
VLNW	555	582	595	603	610	614	619	621	625
Graph	553	579	593	601	608	612	617	620	623
Graph + ExD	551	574	590	600	607	609	606	599	577
VLNW와의 차이	-4	-8	-5	-3	-4	-5	-7	-22	-48

6. 결론

모듈러 멱승 연산에 필요한 곱셈 수는 적은 수의 감소만으로도 암호 프로토콜의 성능에 큰 향상을 가져온다. 가장 널리 사용되는 CLNW 방법과 VLNW 방법 중 VLNW 방법이 더 좋은 성능을 보이지만, CLNW 방법을 비롯한 기존의 다른 방법들과 달리 두 개의 인자를 사용함으로써 E의 특징에 따른 인자의 최적 값을 하나 더 고려해야 하는 어려움이 있었다.

본 논문은 기존의 알고리즘들을 d 인자 하나로 일반화할 수 있는 그래프 모델을 설명하고, d 인자의 영향을 최소화하면서 전체 곱셈 수를 감소시키는 새로운 블록들을 일반화된 그래프 모델의 초기 블록 테이블에 추가할 수 있는 초기 블록 테이블의 두 가지 확장 방법을 제안한다. 하나는 E_B의 접두사를 적절한 길이만큼 연산해 초기 블록 테이블에 추가하는 접두사 블록 확장 방법이며, 다른 하나는 임의의 두 초기 블록에 대한 한 번의 덧셈 사슬로 얻는 새로운 블록을 초기 블록 테이블에 추가하는 방법이다. 이 방법들은 새로운 블록을 초기 블록 테이블에 추가하기 위해 필요한 곱셈의 수와 추가한 뒤의 전체 곱셈 수를 비교하면서 초기 블록 테이블을 제한적으로 확장하므로, E_B에 non-zero bit가 많이 나타날수록 VLNW 방법에 비해 좋은 성능을 보인다.

참고문헌

[1] R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, Vol.21, No.2, pp.120-126, 1978
[2] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory, Vol.31, No.4, pp.469-472, 1985
[3] National Institute for Standards and Technology, Digital Signature Standard(DSS), Federal Register, Vol.56, pp.169, 1991
[4] D. E. Knuth, The art of computer programming: Seminumerical algorithms, Vol.2, Addison-Wesley, pp.461-485, 1981
[5] J. Bos, M. Coster, Addition chain heuristics, In Proc. Crypto '89, Lecture Notes in Computer Science, Vol.435, pp.400-407, 1990
[6] P. Downey, B. Leong and R. Sethi, Computing sequences with addition chains, SIAM J. Comp., Vol.10, No.3, pp.638- 646, 1981
[7] C. K. Koç, Analysis of Sliding Window Techniques for Exponentiation, Computer and Mathematics with Applications, Vol.30, No.10, pp.17-24, 1995
[8] 김지은, 김동규, 모듈러 멱승을 계산하는 일반화된 모델, 2003년 한국멀티미디어학회 춘계학술발표대회논문집, 6권, 1호, pp.1-4, 2003
[9] 김지은, 김동규, 효율적인 초기 블록 테이블 구축을 통한 모듈러 멱승 계산, 2003년 한국멀티미디어학회 추계학술발표대회논문집(상), 6권, 2호, pp.112-115, 2003
[10] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest, Introduction to algorithms, The MIT Press, pp.514-531, 1990