RNS상에서 시간 및 공간 복잡도 향상을 위한 병렬 모듈러 곱셈 알고리즘

(Parallel Modular Multiplication Algorithm to Improve Time and Space Complexity in Residue Number System)

박 희 주 † 김 현 성 †
(Hee-Joo Park) (Hyun-Sung Kim)

요 약 본 논문에서는 RNS 시스템 상에서 시간 및 공간 복잡도 향상을 위한 병렬 모듈러 곱셈 알고리즘을 제안한다. 모듈러 감소를 위해서 새로운 테이블 참조 방식을 사용한다. 테이블 참조시 RNS 시스템이 비 가중치 시스템이므로 대수 비교를 비교하기 위해서 MRS 시스템을 이용한다. 제안한 곱셈 알고리즘은 RNS 컴퓨터 상에서 상대적으로 계산하기 쉬운 MRS 시스템을 사용함으로써 대수 비교를 효율적으로 수행할 수 있다. 기존의 RNS 시스템 상에서 테이블 감소를 이용한 모듈러 곱셈 알고리즘과 비교시 전체 테이블의 크기를 1/2로 줄일 수 있고, 산술 연산도 $2l$ 개의 프로세서를 사용하여 $O(l)$ 만에 수행할 수 있다.

키워드 : RNS 시스템, 모듈러 곱셈, MRS 시스템, 테이블 감축, 암호화 시스템

Abstract In this paper, we present a novel method of parallelization of the modular multiplication algorithm to improve time and space complexity on RNS (Residue Number System). The parallel algorithm executes modular reduction using new table lookup based reduction method. MRS (Mixed Radix number System) is used because algebraic comparison is difficult in RNS which has a non-weighted number representation. Conversion from residue number system to certain MRS is relatively fast in residue computer. Therefore magnitude comparison is easily performed on MRS. By the analysis of the algorithm, it is known that it requires only 1/2 table size than previous approach. And it requires $O(l)$ arithmetic operations using $2l$ processors.

Key words : Residue Number System, Modular Multiplication, Mixed Radix System, Table Lookup Reduction, Crypto System

1. 서 론

1976년 W. Diffie와 M.E. Hellman은 공개키 암호시스템(public key cryptosystem) 개념을 처음으로 제안하였다[1]. 그 후, 상대적으로 구현이 쉽고 현재까지 안전성을 인증받고 있는 공개키 암호 시스템은 1978년 Rivest, Shamir 그리고 Adleman이 제안한 RSA 암호 시스템이다[2]. RSA 암호 시스템은 키의 분배와 관리가 쉽고, 디지털 서명(digital signature)과 인증(authentication)이 가능한 장점을 가지고 있다. 그러나 암호화와 복호화 시 512 비트 이상, 즉 아주 큰 수의 모듈러 지수 연산($M^e \bmod N$)을 수행하므로 처리 속도의 지연이 큰 문제가 되었다. 모듈러 지수 연산(modular exponentiation)은 $AB \bmod N$ 형태의 모듈러 곱셈(modular multiplication)을 반복 수행함으로써 얻을 수 있다. 그러므로 큰 수에 대해서 효율적으로 반복 수행하는 빠른 모듈러 곱셈 알고리즘을 설계하는 것이 고성능의 암호 및 복호 시스템을 개발하는데 있어서 가장 중요하다.

모듈러 곱셈을 효율적으로 수행하기 위해서 많은 알고리즘들이 제안되었다[3-13]. 대부분의 알고리즘들이 가중치 숫자 시스템(weighted number system)을 사용하여 모듈러 곱셈을 수행한다[12]. 반면에 RNS(Residue Number System) 시스템은 그 연산에 있어서 캐리를 고려할 필요가 없고(carry free) 병렬로 처리할 수 있는 특성 때문에 특별한 관심을 끌고 있다[3-11,13-15].

RNS 시스템은 각 디지트에 일치하는 특별한 가중치를 가진 디지트의 전후관계에 의존하는 수 시스템이 아

† 종신회원 : 경일대학교 컴퓨터공학과 교수
hjpark@kiu.ac.kr
kim@kiu.ac.kr
논문접수 : 2003년 3월 6일
심사완료 : 2003년 6월 10일

니다. 그래서 대수의 비교와 나눗셈 연산이 상당히 어렵다. 모듈러 곱셈에서는 곱셈 연산을 하고 나서 모듈러 감소 연산이 필요하다. 감소 연산을 위해서는 대수의 비교가 필요해서 RNS 상에서의 모듈러 곱셈 연산이 곱셈 연산에 잘 적용되지 않을 것처럼 보인다. 그러나 RNS 시스템과 관계있는 MRS(Mixed Radix Number System) 시스템 상에서의 연산을 통해서 RNS 형태의 병렬 모듈러 곱셈 알고리즘을 수행할 수 있다. MRS 시스템에서는 RNS 시스템에서 사용된 것과 똑 같은 모듈러를 사용한다.

본 논문에서 제안한 병렬 모듈러 곱셈 알고리즘은 RNS 시스템 상에서 표현된다. 지금까지 고안된 RNS 곱셈 알고리즘에서는 시스템의 워드 크기가 고려되지 않았다. 본 논문에서는 RNS 시스템에서 필요한 모듈러의 크기를 워드(w)의 크기로 제한한다. 즉 degree(m_i)$\leq w$인 모듈러를 사용함으로써 좀 더 효율적인 계산을 하고자 한다. 워드 w의 크기는 모듈러 감소를 위해서 사용되는 테이블의 크기와 곱셈을 수행하는데 소요되는 전체 수행시간을 고려하여 결정하며, 일반적인 크기는 8이나 16으로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 RNS와 MRS 수 체계에 대해서 기술한다. 3장에서 모듈러 감소에서 사용되는 MRS 시스템을 이용한 테이블 생성 방법을 제안한다. 4장에서는 새로운 방법으로 생성된 테이블을 이용하여 모듈러 감소를 수행 할 수 있는 병렬 모듈러 곱셈 알고리즘을 제안한다. 그리고 5장에서는 제안된 알고리즘의 이해를 돕기 위해서 예제를 통하여 설명하고, 알고리즘을 분석한다. 마지막으로 6장에서 결론을 맺는다.

2. RNS와 MRS 시스템

이 장에서는 RNS와 MRS 수 시스템에 대해서 살펴본다. 먼저 두 수 시스템에서 공통적으로 사용하는 기호에 대하여 정의하고, 각 수 시스템에 대해서 살펴본다.

2.1 기호 정의

본 절에서는 본 논문에서 공통적으로 사용할 기호들을 정의한다.

A, B : 연산의 두 정수(고정기수 체계의 정수) 입력값

X : 고정기수 체계의 정수

N : 정수 연산에서 모듈러로 사용되는 기약수

$X_{RNS}=(x_1, x_2, \ldots, x_L)$: RNS상의 정수 X 표현

$X_{MRS}=\langle x'_L, x'_{L-1}, \ldots, x'_1\rangle$: MRS상의 정수 X 표현

r : 고정기수 체계의 기수

L : 고정기수 체계의 수를 표현하기 위한 디지트의 수

m_i : RNS와 MRS 상의 한 모듈러

k : 기약수 N의 degree

w : RNS나 MRS 상의 계수의 degree

l : RNS나 MRS 상에서 하나의 정수를 표현하기 위한 모듈러의 개수

$M=\prod_{i=1}^{L} m_i$: RNS 상에서 모듈러의 곱

$\overline{m_i}=\left|\frac{1}{m_i}\right|_{m_j}, j=i+1, L$: RNS 상에서 모듈러 m_i의 역원

$Q_w=I_w=\{0,1,2,\ldots,2^w-1\}$: w 비트로 표현 가능한 모든 집합

2.2 수 시스템

기수 r을 갖는 고정기수 체계의 정수 X는 다음과 같이 표현된다.

$$X=\sum_{i=0}^{L-1} a_i r^i;\ a_i \in Z_r$$

고정기수 r로 표현된 L개의 디지트를 갖는 정수 X는 r상에서 표현범위 $[0, r^L-1]$를 갖는다. $r=10$일 경우, 잘 알려진 10진수 체계가 된다. 고정기수 체계의 알려진 특징은 다음과 같다.

(1) 대수 비교가 쉽다.

(2) 동적인 표현범위 확장(디지트를 추가할 수 있음)이 가능하다.

(3) 간단한 shift 연산으로 곱셈(나눗셈) 가능하다.

(4) 오버플로우와 수의 부호 검사가 간단하다.

고정기수 체계, 즉 가중치를 갖는 수체계의 단점은 캐리정보가 반드시 왼쪽 디지트로부터 오른쪽 디지트로 전송되어야 한다는 것이다. 결과적으로 이 캐리를 처리하기 위해서 사용되는 캐리관련 연산에 사용되는 시스템으로 인하여 연산 속도가 늦어진다. 캐리 속도를 가속화시킬 수는 있으나 그렇게 하기 위해서는 추가적인 하드웨어를 필요로 한다.

그러나 RNS는 가중치를 갖지 않는 수 체계이기 때문에 캐리를 고려할 필요가 없고, 그렇기 때문에 연산에 있어서 매우 빠른 속성을 갖고 있다. 그러한 장점과는 반대로 수의 부호 검사나 오버플로우같은 대수 비교가 어렵고 나눗셈 연산을 계산하기 힘들다[3,4].

RNS의 속성은 다음과 같다.

(1) RNS 기저는 모듈러의 집합인 벡터 $\{m_1, m_2, \ldots, m_L\}$로 구성된다. 여기서 모든 m_i는 서로소, 즉 $GCD(m_i, m_j)=1,\ i\neq j$이다.

(2) $M=\prod_{i=1}^{L} m_i$

(3) 벡터 $\{x_1, x_2, \ldots, x_L\}$는 $X<M$인 정수 X의 RNS 표현이다.

$$x_i=|X|_{m_i}=X \bmod m_i$$

$AB \bmod N$ 연산에서 모듈러 감소시 필요한 모듈러

N의 degree를 k라 두면, 모듈러 곱셈에 필요한 입력 A와 B의 degree는 $k-1$이 된다. 나중에 기술될 알고리즘 분석을 위해서 $k=lw$라 두자. 알고리즘의 효율성을 위해서 본 논문에서는 x_i가 w비트나 1워드로 표현될 수 있도록 각 m_i를 선택한다. 즉 $\deg(x_i)=d_i<w$ 이다. 그러므로 본 논문에서 사용하는 RNS 시스템은 L개의 각 모듈러가 서로소이고, 각 degree가 w인 m_i로 구성되고, $\deg(M)=Lw \geq 2k$이다. $k=lw$이므로 $L \geq 2l$ 이다. RNS의 표현 범위를 입력 크기의 두배로 잡는 이유는 두 입력의 곱셈 결과를 중복없이 나타내기 위해서이다.

CRT(Chinese Remainder Theorem)에 근거하여 M보다 작은 어떠한 정수도 단지 하나의 RNS 표현을 가진다. RNS 시스템 상에서 덧셈, 뺄셈 그리고 곱셈 연산은 효율적으로 수행되나 나눗셈 연산은 쉬운 연산이 아니다. 만약 A, B 그리고 C가 다음과 같이 RNS로 표현된다면 :

$A_{RNS}=(a_1, a_2, \ldots, a_L)$;

$B_{RNS}=(b_1, b_2, \ldots, b_L)$;

$C_{RNS}=(c_1, c_2, \ldots, c_L)$;

RNS체계의 잉여수 덧셈, 뺄셈 그리고 곱셈 연산은 다음과 같이 정의된다.

(1) $C=A+B$는 잉여수 덧셈을 나타낸다 :

$c_i=a_i+b_i \bmod m_i,\ i=1,2,\ldots,L$

(2) $C=A-B$ 는 잉여수 뺄셈을 나타낸다 :

$c_i=a_i-b_i \bmod m_i,\ i=1,2,\ldots,L$

(3) $C=A\times B$ 는 잉여수 곱셈을 나타낸다 :

$c_i=a_i\times b_i \bmod m_i,\ i=1,2,\ldots,L$

덧셈 $C=A+B$ 연산이 수행될 때, 연산의 결과를 저장하는 변수 C의 degree는 A와 B의 최대 degree 보다 크지 않다. 즉 잉여수 연산은 $c_i=a_i+b_i \bmod m_i$ 연산으로 정확한 결과를 출력할 것이다. 그러나 곱셈 $C=A\times B$ 연산에서는 결과는 10진수 연산에서와 같이 연산 결과값의 degree가 곱셈의 입력 A와 B의 degree 보다 증가한다. 즉, 곱셈 연산 결과 C는 기약수(irrecucible) N을 이용한 모듈러 감소 연산이 필요하다. 그러므로, 모듈러 N 곱셈 연산을 RNS 상에서 수행할 때 곱셈 결과의 degree를 줄일 수 있는 방법이 필요하다.

또한 곱셈 결과의 degree를 결정하기 위해서는 대수의 크기 비교 연산이 필요하다. 그러나 RNS는 비가중치 수 체계이므로 대수의 크기 비교 연산이 불가능하다. 대수의 크기 비교 연산을 위해서 본 논문에서는 가중치 수체계인 MRS(Mixed-Radix System)를 이용하여 결과의 degree를 줄이기 위해서 다음장에서 설명할 새로운 TRM(Table lookup Reduction Method)을 사용한다.

MRS의 표현은 잉여수 계산에서 다음과 같은 중요한 2가지 특징을 갖는다.

(1) MRS는 가중치 시스템이므로 대수 비교 연산이 쉽다.

(2) RNS에서 MRS로의 변환이 잉여수 계산에서 빠르게 수행된다.

MRS에서의 기저는 RNS에서 사용된 기저의 모듈러 집합을 다시 사용한다. $\langle x'_L, x'_{L-1}, \ldots, x'_1\rangle$가 M보다 적은 수 X의 MRS 표현이라면 X는 다음과 같다.

$$X_{MRS}=x'_L m_{L-1} m_{L-2} \cdots m_1 + \cdots + x'_3 m_2 m_1 + x'_2 m_1 + x'_1$$

여기서 m_i는 기수이고 x'_i는 MRS의 계수이다($0 \leq x'_i < m_i$).

RNS 표현을 가중치 시스템인 MRS 표현으로 변환하기 위해서 MRC(Mixed Radix Number Conversion) 알고리즘을 사용한다.

Mixed Radix Number Conversion Algorithm

입력 : $X_{RNS}=(x_1, x_2, \ldots, x_L)$

출력 : $X_{MRS}=\langle x'_L, x'_{L-1}, \ldots, x'_1\rangle$

$$= x'_L m_{L-1} m_{L-2} \cdots m_1 + \cdots + x'_3 m_2 m_1 + x'_2 m_1 + x'_1$$

보조값 : $\overline{m_i}=\left|\frac{1}{m_i}\right|_{m_j}, j=i+1, L$

Step 1. for i=1 to L-1 do

Step 2. $X=X-x[i] \bmod m$

Step 3. $X=X\times\overline{m_i} \bmod m$

Step 4. return X_{MRS}

Step 2와 3의 X는 고정기수 체계의 정수값이 아닌 RNS 시스템에서 MRS 시스템으로 변환하기 위해서 사용되는 임시 변수이다. 즉, Step 1의 for 루프를 종료한 후의 X 값이 X_{MRS}가 된다. 변환 과정은 3장의 예제를 통해서 자세히 살펴본다.

3. 테이블을 이용한 모듈러 감소

본 논문에서는 모듈러 감소를 위해서 모듈러 N의 배수를 저장하고 있는 테이블을 사용한다. N의 degree를 s, 즉 $\deg(N)=s$라 두면 테이블을 구성하기 위해서는 $s+w$ 보다 작은 degree를 갖는 N의 배수를 구하면 된다. w 비트로 표현할 수 있는 모든 집합 I_w를 고려하면 다음과 같다.

$$Q_w=I_w=\{0,1,2,\ldots,2^w-1\}$$

모든 인덱스 i는 $i \in I_w$, 몫 q_i는 $q_i \in Q_w$이다. 또,

$v_i = q_i N,\ i \in I_w$로 정의한다. 즉, $v_i = n_i \bmod m_i, i = 1,2,\ldots,L$이다. v_i는 degree가 $s+w$보다 적고, $(l+1)$개의 워드를 갖는 수로 구성된다.

$$r_i = \langle R_{i,l}\ R_{i,l-1}\ldots R_{i,1}\ R_{i,0}\rangle$$

본 논문에서는 값이 0이 아닌 v_i의 가장 최상위 워드(w비트)를 테이블의 인덱스로 사용하여 2^w개의 열을 갖는 테이블 T를 생성한다. 즉, $i \in I_w$에 대하여

$$T(V_{i,l}) = \langle V_{i,l-1}\ldots V_{i,1}\ V_{i,0}\rangle$$

로 저장한다. 중요한 점은 모든 최상위 비트 $V_{i,l}$, $i \in I_w$는 단 하나만 존재한다는 것이다.

MRS 시스템 상에서 구성된 테이블 T는 모듈러 연산의 결과값에 모듈러 감소 연산을 적용하기 위해서 사용된다. 만약 결과값 c가 $lw+w$ 길이를 갖고, $c = \langle C_l\ C_{l-1}\ldots C_1\ C_0\rangle$로 표현된다고 하자. 모듈러 감소 알고리즘은 lw길이를 갖는 $c \bmod N$을 계산한다. 결과값 c를 계산하기 위해서 생성된 테이블 T의 목록에서 $C_l = V_l$을 인덱스로 이용하여 $\langle V_{i,l-1}\ldots V_{i,1}V_{i,0}\rangle$ 항목을 선택한다. T의 위치 C_l에 잉여수 $\langle V_{i,l-1}\ldots V_{i,1}\ V_{i,0}\rangle$가 저장되어 있기 때문에

$$c = \langle C_{l-1}\ C_{l-2}\ldots C_1\ C_0\rangle - \langle V_{l-1}\ V_{l-2}\ldots V_1\ V_0\rangle = \langle C'_{l-1}C'_{l-2}\ldots C'_1C'_0\rangle$$

여기서 $C'_j = C_j + V_j,\ j = 0,1,2,\ldots,l-1$이다. 예를 들어 $N=201$이고, $m_1=13$, $m_2=15$, $m_3=16$, $m_4=11$일 경우 $q_2N = 2\times 201 = 402$이고, 모듈러 $m_i, i=1,2,3,4$에 의해서 RNS 시스템 표현하면 결과는 다음과 같다.

402 mod 13 = 12, 402 mod 15 = 12

402 mod 16 = 2, 402 mod 11 = 6

즉 $X_{RNS} = 402_{RNS} = (12,12,2,6)$이다. 여기에서 구해진 RNS 표현을 앞장에서 설명한 MRC 알고리즘을 이용하여 MRS 시스템으로 변환한다. 그 변환 과정은 다음과 같다.

Moduli :	13	15	16	11	
X_{RNS}	$a_1 = 12$	12	2	6	$a_1 = \lvert X\rvert_{13} = 12$
Subtract a_1	12	12	12	1	
$X - a_1$	0	0	6	5	
Multiply by $\lvert\frac{1}{13}\rvert_{m_i}$		7	5	6	
$\frac{X-a_1}{13}$	$a_2 = 0$	14	8		$a_2 = \lvert\lfloor\frac{X}{m_1}\rfloor\rvert_{m_2}$
Subtract a_2		0	0	0	$= \lvert\lfloor\frac{X}{13}\rfloor\rvert_{15} = 0$
$\frac{X-a_1}{13} - a_2$		0	14	8	
Multiply by $\lvert\frac{1}{15}\rvert_{m_i}$			15	3	
$\frac{X-a_1}{13} - a_2$ / 15		$a_3 = 2$	2		$a_3 = \lvert\lfloor\frac{X}{m_1 m_2}\rfloor\rvert_{m_3}$
Subtract a_3			2	2	$= \lvert\lfloor\frac{X}{195}\rfloor\rvert_{16} = 2$
$\frac{X-a_1}{13} - a_2$ / 15 $- a_3$			0	0	

그러므로 $X_{RNS} = 402_{RNS} = (12,12,2,6)$의 MRS 표현은 $X_{MRS} = \langle 0,2,0,12\rangle$이다. 위의 계산에서 MRS의 최상위 워드의 결과는 그 이전 워드의 결과값이 0이기 때문에 더 이상 계산할 필요가 없다. 이렇게 모듈러 N에 모든 $q_i \in Q_w$를 곱해서 Table 1을 만들 수 있다.

Table 1의 MRS 표현을 보면 첫 번째 워드는 모두 0임을 알 수 있다. 그러므로 Table 1을 구현할 때 값이 0이 아닌 최상위 워드인 두 번째 워드를 인덱스로 활용해서 다음 Table 2와 같이 저장할 수 있다.

모듈러 감소 연산을 위한 또 하나의 다른 방법은 모든 모듈러 감소 연산을 위한 테이블을 미리 다 만들어 놓는 것이다. 모든 곱셈 연산에서 모듈러 감소는 최대 2번이면 되므로 고려될 테이블의 크기는 Table 2에서 제안된 2배의 크기를 갖게 될 것이다. 그러나 곱셈 알고리

Table 1 모듈러 감소 테이블 T

$i,\ Q_i$	$T(i)$	$i,\ Q_i$	$T(i)$
0	< 0, 0, 0, 0 >	8	< 0, 8, 3, 9 >
1	< 0, 1, 0, 6 >	9	< 0, 9, 4, 2 >
2	< 0, 2, 0, 12 >	10	< 0, 10, 4, 8 >
3	< 0, 3, 1, 5 >	11	< 0, 11, 5, 1 >
4	< 0, 4, 1, 11 >	12	< 0, 12, 5, 7 >
5	< 0, 5, 2, 4 >	13	< 0, 13, 6, 0 >
6	< 0, 6, 2, 10 >	14	< 0, 14, 6, 6 >
7	< 0, 7, 3, 3 >	15	< 0, 15, 6, 12 >

Table 2 수정된 모듈러 감소 테이블 T

$i,\ Q_i$	$T(i)$	$i,\ Q_i$	$T(i)$	$i,\ Q_i$	$T(i)$	$i,\ Q_i$	$T(i)$
0	< 0, 0 >	4	< 1, 11 >	8	< 3, 9 >	12	< 5, 7 >
1	< 0, 6 >	5	< 2, 4 >	9	< 4, 2 >	13	< 6, 0 >
2	< 0, 12 >	6	< 2, 10 >	10	< 4, 8 >	14	< 6, 6 >
3	< 1, 5 >	7	< 3, 3 >	11	< 5, 1 >	15	< 6, 12 >

즘에서 필요한 shift 연산은 필요 없다. 테이블의 크기는 워드의 크기를 증가시킬수록 커진다. 그러므로 컴퓨터 시스템의 메모리 용량에 따라서 모듈러 감소 연산의 방법을 선택할 수 있다.

4. RNS를 이용한 병렬 모듈러 곱셈 알고리즘

이 장에서는 본 논문에서 제안한 잉여수 연산과 테이블 모듈러 감소 방법을 이용한 병렬 모듈러 곱셈 알고리즘을 제시한다. 테이블 기반의 모듈러 감소 연산을 위해서 앞 장에서 MRS 시스템상에서 구현된 테이블이 사용된다. 잉여수 연산을 이용한 병렬 모듈러 곱셈 알고리즘은 다음과 같다.

RNS Based Multiplication Algorithm

입력 : $A_{RNS}=(a_1, a_2, \ldots, a_L)$,
$B_{RNS}=(b_1, b_2, \ldots, b_L)$

출력 : $C_{MRS}=\langle c'_L, c'_{L-1}, \ldots, c'_1\rangle$
$= x'_L m_{L-1} m_{L-2} \ldots m_1$
$+ \ldots + x'_3 m_2 m_1 + x'_2 m_1 + x'_1$

보조값 : $\overline{m_i}, T, MAX[i]$

Step 1. $C_{RNS}= A_{RNS}\times B_{RNS}$
Step 2. for i=1 to $L-1$ do
Step 3. $C= C-c[i] \bmod m$
Step 4. $C= C\times \overline{m_i} \bmod m$
Step 5. for $i=2l-1$ downto l do
Step 6. $C= C- T[c[i]]\times MAX[i]$
Step 7. return C_{MRS}

입력 A와 B의 degree를 최대 $k-1$이라고 가정하면 Step 1을 수행한 후 곱셈 결과를 저장하는 변수 C의 degree는 최대 $2(k-1)$가 된다. M의 degree가 기껏해야 $2k$이기 때문에 결과값 C는 유일하게 표현될 수 있고, MRC 알고리즘에서 생성된 결과도 유일하게 표현될 수 있다. 그러나 이 결과값 C의 degree가 k보다 크기 때문에 이 결과를 그대로 다른 곱셈의 입력으로 사용할 수는 없다. Step 1을 구현하기 위해서는 이미 유용한 RNS 곱셈기 중에서 하나를 선택할 수 있다. 현재 유용한 RNS 곱셈기로서는 테이블 검색 곱셈기(Table Lookup Multiplier), Quarter Square 곱셈기(Quarter Square Multiplier), 인덱스 변환 곱셈기(Index Calculus Multiplier) 그리고 어레이 곱셈기(Array Multiplier) 등이 있다[3,5,6,10]. 이러한 곱셈기 중에서 시간대 공간비를 구해서 시스템에 적합한 곱셈기를 선택할 수 있다. Step 1의 결과를 시스템이 원하는 degree인 k보다 적게 만들기 위해서는 모듈러 감수인 N을 생성하고 결과를 N보다 적게 만들어야 될 필요성이 있다. Step 2-6에서 모듈러 감소 연산을 수행한다. 먼저 MRC 알고리즘을 이용해서 결과 C를 가중치 숫자 시스템인 MRS 시스템으로 변환한다. 그리고 앞 장에서 생성된 테이블을 이용하여 곱셈 결과인 C의 degree를 $k-1$로 줄인다. Step 6에서 $MAX[2l-1]$에는 모듈러 중에서 가장 큰 값을 저장하고, 나머지 배열에는 숫자 1을 저장한다. RNS 컴퓨터에서 임의의 수와 모듈러의 곱셈은 shift 연산으로 구현할 수 있다. $MAX[i]$를 테이블 T에서 가져온 값과 곱하는 이유는 테이블 T에서 가져온 값의 최고 상위 워드가 한 자리수 왼쪽으로 이동했을때에도 같은 값을 유지하기 위해서이다. 그래야 결과값의 모듈러 감소 연산을 효율적으로 수행 할 수 있기 때문이다. 또다른 방법으로는 모듈러 감소를 위해서 곱셈한 결과값을 미리 테이블에 저장하는 방법이 있다. 이 방법도 테이블 T를 작성한 방법과 비슷한 방법으로 수행할 수 있다. 즉 Table 1의 최상위 워드의 표현을 갖는 인덱스 $i \in I_w$로 갖는 방법이다.

5. 알고리즘 수행 예제 및 분석

이 장에서는 앞 장에서 제시한 병렬 모듈러 곱셈 알고리즘의 좀더 효율적인 이해를 위해서 곱셈 알고리즘의 수행을 예를 들어서 설명한다. 또한 제안된 알고리즘을 분석하고, [15]에서 Halbutogullari가 제안한 알고리즘과 본 논문에서 제안한 알고리즘의 성능을 비교 분석한다.

5.1 알고리즘 수행 예제

이 절에서는 $k=8$일 경우 RNS 시스템 기반의 병렬 곱셈 알고리즘의 예에 대해서 다룬다. 모듈러 $N=201$일 때, $k=8, w=4$ 그리고 $l=k/w=2$이다. 그러므로 $L=2l=4$, 즉 RNS 시스템을 구성하기 위해서 degree가 4인 4개의 각 모듈러가 서로소인 모듈러 m_i가 필요하다. 예를 들기 위해서 w에 가장 근접한 서로소를 다음과 같이 $m_1=13$, $m_2=15$, $m_3=16$, $m_4=11$ 선택했다. M은 다음과 같이 구할 수 있다.

$$M= m_1 m_2 m_3 m_4 = 13\times 15\times 16\times 11 = 34320$$

즉, 입력된 두 수의 곱셈의 결과가 34320보다 적은 값일 경우에 RNS 시스템 상에서 결과값의 표현이 유일하다는 의미이다.

지금부터 $C=AB \bmod N$ 연산을 위한 RNS 시스템 기반의 병렬 곱셈 알고리즘의 각 Step별 계산 과정에 대해서 살펴보자. $A=180$이고 $B=163$일 경우의 예를 살펴보자. 먼저 A와 B를 RNS 시스템 상의 표현으로 바꾸면 다음과 같다.

$$A_{RNS}=(11,0,4,4),\quad B_{RNS}=(7,13,3,9)$$

RNS 시스템 기반의 병렬 모듈러 곱셈 알고리즘은 결

과값 $C=195$를 찾기 위해서 다음과 같은 과정을 수행한다.

Step 1 : $C_{RNS}=A_{RNS}\times B_{RNS}$
$=(11,0,4,4)\times(7,13,3,9)$
$=(12,0,12,3)$

Step 3 : $C=C-c[i] \bmod m$

Step 4 : $C=C\times\overline{m_i} \bmod m$
$=(12,0,12,3)-(12,12,12,12)=(0,3,0,2)$
$=(0,3,0,2)\times(0,7,5,6)=(0,6,0,1)$
$=(0,6,0,1)-(0,6,6,6)=(0,0,10,6)$
$=(0,0,10,6)\times(0,0,15,3)=(0,0,6,7)$
$=(0,0,6,7)-(0,0,6,6)=(0,0,0,1)$
$=(0,0,0,1)\times(0,0,0,9)=(0,0,0,9)$
$=\langle 9,6,6,12\rangle$

Step 6 : $C=C-T[C_i]\times MAX[i]$
$=\langle 9,6,6,12\rangle-\langle 0,9,4,2\rangle\times 16$
$=\langle 9,6,6,12\rangle-\langle 9,4,6,6\rangle=\langle 0,2,0,6\rangle$
$=\langle 0,2,0,6\rangle-\langle 0,1,0,6\rangle\times 1=\langle 0,1,0,0\rangle$

즉, 결과값 $C_{MRS}=\langle 0,1,0,0\rangle=0\times(13\times15\times16)+1\times(13\times15)+0\times(13)+0=195$ 이다.

5.2 알고리즘 분석

이 절에서는 본 논문에서 제안한 테이블 기반의 모듈러 감소와 병렬 모듈러 곱셈 알고리즘을 수행하는데 필요한 수행시간과 프로세서 요구사항에 대해서 분석한다.

먼저 테이블에 관한 연산을 고려하고, 테이블의 크기를 계산한다. 테이블과 관련된 연산은 테이블을 읽는 연산 뿐이다. 테이블 T로부터 테이블을 읽는 연산을 TREAD라고 표시하고, TREAD연산의 전체 회수를 계산한다. TREAD 연산은 병렬 모듈러 곱셈 알고리즘의 Step 6에서만 발생한다. 그러므로 TREAD 연산횟수에 관해서는 이 절의 후반부에 다루기로 한다. 테이블의 크기를 고려하면 테이블 T는 2^w행을 갖고 각 행은 k 길이를 갖는 수를 저장한다. 그러므로 테이블의 전체 크기는 $2^w\times k$ 비트가 된다. 테이블의 크기는 워드의 크기를 증가시킬수록 커진다. 그러므로 고정된 크기의 k에 대해서 워드의 크기 w는 컴퓨터 시스템의 메모리 용량에 따라서 조절할 수 있다.

다음으로 본 논문에서 제안된 RNS 시스템 기반의 모듈러 곱셈 알고리즘을 상세하게 분석한다. 2장에서 가정한대로 degree $(m_i)=w$이고 모듈러 곱셈에서 모듈러 m_i 감소는 테이블을 이용하여 수행한다. L개의 서로다른 모듈러를 사용하기 때문에 각각의 산술 연산의 선택된 모듈러 연산을 수행하기 위해서 $L=2l$개의 프로세서를 갖는다고 가정하자. 알고리즘 분석을 효율적으로 하기 위해서 워드 단위의 덧셈과 뺄셈 연산은 XOR로 나타내고, 워드 단위의 곱셈 연산은 MUL로 표현한다.

Table 3 병렬 모듈러 곱셈 알고리즘 분석

	MUL	XOR	TREAD
Step 1	1		
Step 3($2l$)		1	
Step 4($2l$)	1		
Step 6(l)	1	1	1
Total	$3l+1$	$3l$	l

Table 4 모듈러 곱셈 알고리즘들의 비교

	기존의 방법[15]	제안된 방법
Table size	$2\times2^w\times k$	$2^w\times k$
TREAD	$2l$	l
MUL	$2l+1$	$3l+1$
XOR	$3l-1+l\log_2(2l)$	$3l$

제안된 RNS 시스템 기반의 모듈러 곱셈 알고리즘의 분석은 다음 Table 3과 같다.

기존에 제안된 논문은 GF (2^k) 상에서 모듈러 곱셈을 위해서 정수와 RNS 시스템을 동시에 이용하는 방법을 사용하였다. 또한 모듈러 감소 연산을 위해서 RNS 시스템과 정수 시스템을 동시에 사용함으로서 더 큰 테이블을 필요로 하였다[15]. 본 논문에서 제안된 RNS 시스템 기반의 병렬 모듈러 곱셈 알고리즘과 기존에 방법을 비교하면 다음 Table 4와 같다.

Table 4에서 보여준 바와 같이 논문 [15]에서는 모듈러 곱셈을 위한 정수 테이블과 RNS 테이블, 즉 2개의 테이블이 동시에 필요하였으나, 제안한 논문에서는 MRS 수 시스템을 이용함으로서 테이블의 수와 연산의 수를 효율적으로 줄일 수 있었다.

6. 결 론

본 논문에서는 RNS 시스템 상에서 병렬 모듈러 곱셈 알고리즘을 제안하였다. 지금까지 고려된 대부분의 RNS 곱셈 알고리즘에서는 시스템의 워드 크기를 고려하지 않았다. 그러나 본 논문에서는 RNS 시스템에서 필요한 모듈러의 크기를 워드(w)의 크기로, 즉 degree$(m_i)\le w$로 제한하였다. 워드 w의 크기는 모듈러 감소를 위해서 사용되는 테이블의 크기와 곱셈을 수행하는데 소요되는 전체 수행시간을 고려하여 결정되며, 일반적인 크기는 8이나 16으로 한다.

제안된 병렬 곱셈 알고리즘은 RNS 컴퓨터 상에서 상대적으로 계산하기 쉬운 MRS 시스템을 사용함으로써 대수 비교를 효율적으로 수행할 수 있었다. MRS 시스템은 가중치를 갖는 수 시스템이므로, 모듈러 감소 연산

을 수행하는 데 효율적이었다. 기존의 RNS 시스템 상에서 테이블 감소 방법을 이용한 모듈러 곱셈 알고리즘과 비교시 전체 테이블의 크기를 $\frac{1}{2}$로 줄일 수 있었고, 산술 연산도 $2l$개의 프로세서를 사용하여 $O(l)$ 만에 수행할 수 있었다.

본 논문에서 제안된 알고리즘은 앞에서 기술한 것처럼 RNS 컴퓨터 상에서 기존에 제안된 알고리즘보다 좀 더 효율적인 공간복잡도와 시간복잡도를 가진다. 제안된 알고리즘을 이용하여 좀 더 빠른 암호화 시스템을 구축할 수 있을 것이다. 현재 제안한 알고리즘을 개선하는 연구와 하드웨어 구현 연구가 진행중에 있다.

참 고 문 헌

[1] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. on Info. Theory*, vol. IT-22(6) pp. 644-654, 1976.

[2] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Commu. ACM*. vol. 21, pp. 120-126, 1978.

[3] K.M. Elleithy and M.A. Bayoumi, "A Systolic Architecture for Modulo Multiplication," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 42, no. 11, pp. 725-729, Nov. 1995.

[4] J.C Bajard, L.S. Didier, and P. Kornerup, "An RNS Montgomery Modular Multiplication Algorithm," *IEEE Trans. on Computers*, vol. 47, no. 7, pp. 766-776, July 1998.

[5] D. Radhakrishnan and Y. Yuan, "Novel Approaches to the Design of VLSI RNS Multipliers," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, no. 1, pp. 52-57, Jan. 1992.

[6] G. Alia and E. Martinelli, "A VLSI Modulo m Multiplier," *IEEE Trans. on Computers*, vol. 40, no. 7, pp. 873-878, July 1991.

[7] L.L Yang and L. Hanzo, "Residue Number System Arithmetic Assisted M-ary Modulation," *IEEE Communications Letters*, vol. 3, no. 2, pp. 28-30, Feb. 1999.

[8] F.J. Taylor, "A VLSI Residue Arithmetic Multiplier," *IEEE Trans. on Computers*, vol. C-31, no. 6, pp. 540-546, June 1982.

[9] G.A. Jullien, "Implementation of Multiplication, Modulo a Prime Number, with Applications to Number Theoretic Transforms," *IEEE Trans. on Computers*, vol. C-29, no. 10, pp. 899-905, Oct. 1980.

[10] M. Soderstrand, W.K. Jenkins, G.A. Jullian and F.J. Taylor, *Residue Number Systems: Modern Applications in Digital Signal Processing*, New York, IEEE, 1986.

[11] V.S. Dimitrov, G.A. Jullien and W.C. Miller, "A Residue Number System Implementation of Real Orthogonal Transforms," *IEEE Trans. on Signal Processing*, vol. 46, no. 3, pp. 563-570, March 1998.

[12] H.S. Kim, K.J. Lee, J.J. Kim and K.Y. Yoo, "Partitioned Systolic Multiplier for GF(2^m)," *Proc. of the 1999 ICPP Workshops on IWSEC*, pp. 192-197, 1999.

[13] A. Halbutogullari and C.K. Koc, "Parallel Multiplication in GF(2^k) using Polynomial Residue Arithmetic," *Design, Codes and Cryptography*, to appear, 1999.

[14] N.S. Szabo and R.I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, McGraw-Hill, New York, 1967.

[15] F.J. Taylor, "Residue Arithmetic: A Tutorial with Examples," *Computer*, pp. 50-62, May 1984.

박 희 주

1978년 2월 영남대학교 전자공학과 공학사. 1981년 2월 영남대학교 전자계산기공학과 공학석사. 1995년 2월 대구가톨릭대학교 전산통계학과 이학박사. 1982년 3월~현재 경일대학교 컴퓨터공학과 교수 관심분야는 신경회로망, 패턴인식, 정보보호, 병렬처리

김 현 성

정보과학회논문지 : 시스템 및 이론 제 30 권 제 2 호 참조