通信情報保護學會誌
第2卷 第3號, 1992. 9

모듈러 지수 연산 알고리듬

이석래* · 염흥열** · 이만영*

요 약

본 논문에서는 암호알고리듬 실현을 위해 요구되는 계산량에 가장 큰 영향을 미치는 모듈러 지수(modular exponentiation)에 관한 여러가지 연산알고리듬을 분석 및 제시하고 그 예를 보인다. 본 논문에서 소개되는 연산알고리듬은 $x^n \pmod p$를 계산하기 위한 대표적 방식인 이진방식(binary method), 이진방식의 일반형인 m진방식(m-ary method), 합성수(n)에 적용 가능한 소인수분해 방식(factor method), 그리고 고리(chain)를 이용하는 파워트리 방식(power tree method) 및 가산고리방식(addition chain method)등을 포함한다.

1. 서 론

DES(data encryption standard)와 같은 관용 암호알고리듬은 키 분배시 비용이 많이 소요되고, 비밀키 저장이 어렵기 때문에 다수의 이용자를 갖는 컴퓨터 통신망에서의 응용에는 적합하지 않다. 따라서, 다수의 이용자를 포함하는 컴퓨터 통신망에 적합한 암호알고리듬으로서 ID(identity)를 이용한 공개키 암호알고리듬[1]이 응용될 전망이다. 하지만 이 암호알고리듬의 실현시 많은 양의 계산이 요구되어 고속 처리가 불가능하기 때문에 아직까지는 널리 적용되고 있지 않다. 따라서, 많은 암호학자들은 공개키 암호알고리듬의 보편적 사용을 위한 가장 핵심 요소 기술인 고속 연산 처리가 가능한 간단한 연산알고리듬의 고안에 심혈을 기울이고 있다.

본 논문에서는 암호알고리듬 실현을 위해 요구되는 계산량에 가장 큰 영향을 미치는 모듈러 지수(modular exponentiation)에 관한 여러가지 연산알고리듬을 분석 및 제시하고 그 예를 보인다. 본 논문에서 소개되는 연산알고리듬은 $x^n \pmod p$를 계산하기 위한 대표적 방식인 이진방식(binary method)[5,10], 이진방식의 일반형인 m진방식(m-ary method)[4,9,10], 합성수(n)에 적용 가능한 소인수분해 방식(factor method)[3], 그리고 고리(chain)를 이용하는 파워트리 방식(power tree method)[3,6,7,11]등을 포함한다.

2. 연산 알고리듬

2.1 이진방식

* 한양대학교 전자통신공학과
** 순천향대학교 전자공학과

이진방식은 x^n (mod p)를 계산하기 위한 대표적인 연산방식이다. 본 절에서는 세가지 연산알고리듬인 SX(square multiplication) 이진방식, 올림차순(right-to-lett) 이진방식, 그리고 내림차순(left-to-right) 이진방식등을 소개한다.

(1) SX 이진방식

SX 이진방식을 소개하고 그 예를 보인다.

-알고리듬 A-

① n을 이진수열(binary sequence)로 표현한다.

② 이진수열에서 '1'은 'SX', '0'은 'S'로 교체한다.

③ 이진수열의 가장 왼쪽편의 'SX'를 소거한다.

④ S는 중간결과에 대한 제곱연산으로, X는 x를 곱하는 연산으로 각각 해석하여 x^n (mod p) 연산을 수행한다.

(예제 1) x^{23} (mod p)를 SX 방식을 이용하여 계산하자.

① 23을 이진수열로 표현한다.

: n→(10111)

② 이진수열에서 '1'을 'SX', '0'을 'S'로 교체한다.

: (10111)→(SX, S, SX, SX, SX)

③ 수열의 가장 왼편의 'SX'를 소거한다.

: (SX, S, SX, SX, SX)→(S, S, X, S, X, S, X)

④ x^2 (mod p), x^4 (mod p), x^5 (mod p), x^{10} (mod p), x^{11} (mod p), x^{22} (mod p), x^{23} (mod p)를 계산한다.

따라서 x^{23} (mod p)를 계산하기 위해 요구되는 모듈러 곱셈 연산의 횟수는 7회가 된다.

(2) 올림차순 이진방식

올림차순 이진방식에 대한 모듈러 지수 연산알고리듬과 흐름도를 소개하고 그 예를 보인다. 올림차순 이진방식은 n을 이진수열로 표현했을 때, 알고리듬이 우선 최하위(LSB) 비트를 탐색하고 마지막으로 최상위(MSB) 비트를 탐색한다.

-알고리듬 B-

① (초기화 단계) N←n, Y←1, Z←x로 초기값을 부여한다.

② N이 짝수인지 홀수인지를 판별한다. N이 홀수이면 단계 ③을 수행하고 N이 짝수이면 단계 ⑥을 수행한다.

③ Y에 Z를 곱한 후, 이 연산 결과를 Y에 일시적으로 저장한다.

$$Y \leftarrow Y \times Z \pmod{p} \tag{1}$$

④ 다음의 연산을 수행한다.

$$N \leftarrow \lfloor N/2 \rfloor \tag{2}$$

여기서, $\lfloor x \rfloor$: x를 넘지 않는 최대의 양의 정수

⑤ N이 '0'이면 이 알고리듬을 종료하고, 그렇지 않으면 단계 ⑦을 수행한다.

⑥ 다음의 연산을 수행한다.

$$N \leftarrow \lfloor N/2 \rfloor \tag{3}$$

⑦ Z를 제곱하여 Z에 일시 저장한 후, 단계 ②을 수행한다.

$$Z \leftarrow Z \times Z \pmod{p} \tag{4}$$

알고리듬 B가 종료되면, Y는 x^n (mod p)의 계산 결과가 된다.

모듈러 지수 연산알고리듬 B에 대한 흐름도는 그림 1과 같다.

(예제 2) 표 1은 알고리듬 B를 이용하여 x^{23} (mod p)를 계산할 결과이다.

모듈러 곱셈 연산의 횟수는 8회 수행된다. 일반적으로 알고리듬 B를 이용했을 때 소요되는 모듈러 곱셈의 횟수는 $\lfloor \log_2 n \rfloor + \nu(n)$이다. 여기서, $\nu(n)$은 n의 이진수열 표현식에서의 '1'의 갯수, 즉 중(weight)과 같다.

(3) 내림차순 이진방식

내림차순 이진방식에 대한 모듈러 지수 연산알고리듬과 흐름도를 소개하고 그 예를 보인다.

표 1. 알고리듬 B를 이용한 x^{23}의 연산결과

루프 수	②에서의 N	③에서의 Y	④, ⑥에서의 N	⑦에서의 Z
1	23	$x \pmod p$	11	$x^2 \pmod p$
2	11	$x^3 \pmod p$	5	$x^4 \pmod p$
3	5	$x^7 \pmod p$	2	$x^8 \pmod p$
4	2		1	$x^{16} \pmod p$
5	1	$x^{23} \pmod p$	0	

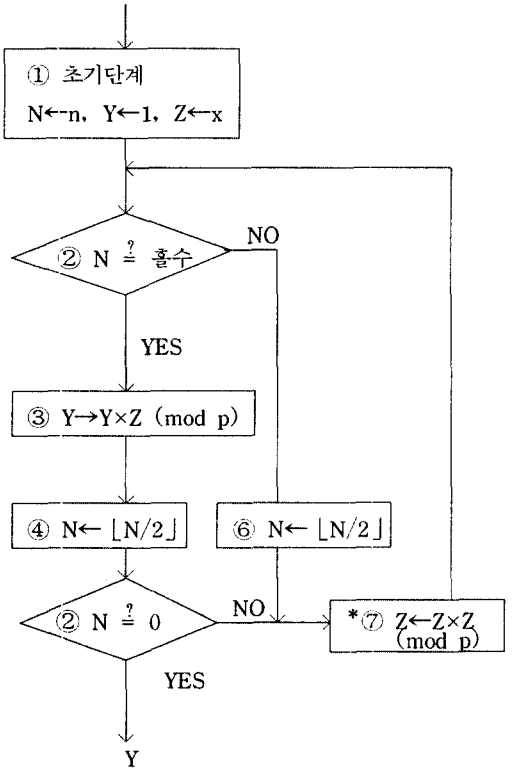


그림 1. 올림차순 이진방식의 흐름도

－알고리듬 C－

① (초기화 단계) n을 다음과 같은 이진수열로 표시한다.

$$n=(n[k-1], n[k-2], \cdots, n[1], n[0]),$$
$$k=\lceil \log_2 n \rceil \qquad (5)$$

변수 Y←x, i←k-2로 초기화 한다.

② n[i]가 '1'이면 단계 ③을 수행하고, '0'이면 단계 ⑤를 수행한다.

③ 다음과 같이 Y를 모듈러 제곱하여 Y에 일시 저장한다.

$$Y \leftarrow Y \times Y \pmod p \qquad (6)$$

④ Y에 x를 모듈러 곱셈하여 Y에 일시 저장한 후, 단계 ⑥을 수행한다.

$$Y \leftarrow Y \times x \pmod p \qquad (7)$$

⑤ 다음과 같이 Y를 모듈러 제곱하여 Y에 일시 저장한다.

$$Y \leftarrow Y \times Y \pmod p \qquad (8)$$

⑥ i가 0보다 작으면을 종료하고, 그렇지 않은 경우, i=i-1한 후 ②로 되돌아간다.

알고리듬 C가 종료되면 Y가 $x^n \pmod p$한 결과이다. 모듈러 지수 연산알고리듬 C에 대한 흐름도는 그림 2와 같다.

(예제 3) $x^{23} \pmod p$를 계산하기 위하여 내림차순 이진방식을 이용한 연산은 표 2와 같다.

표 2. 내림차순 이진방식을 이용한 x^{23}의 연산결과

n의 이진계열	i	Y
1	4	x
0	3	$x^2 \pmod p$
1	2	$x^5 \pmod p$
1	1	$x^{11} \pmod p$
1	0	$x^{23} \pmod p$

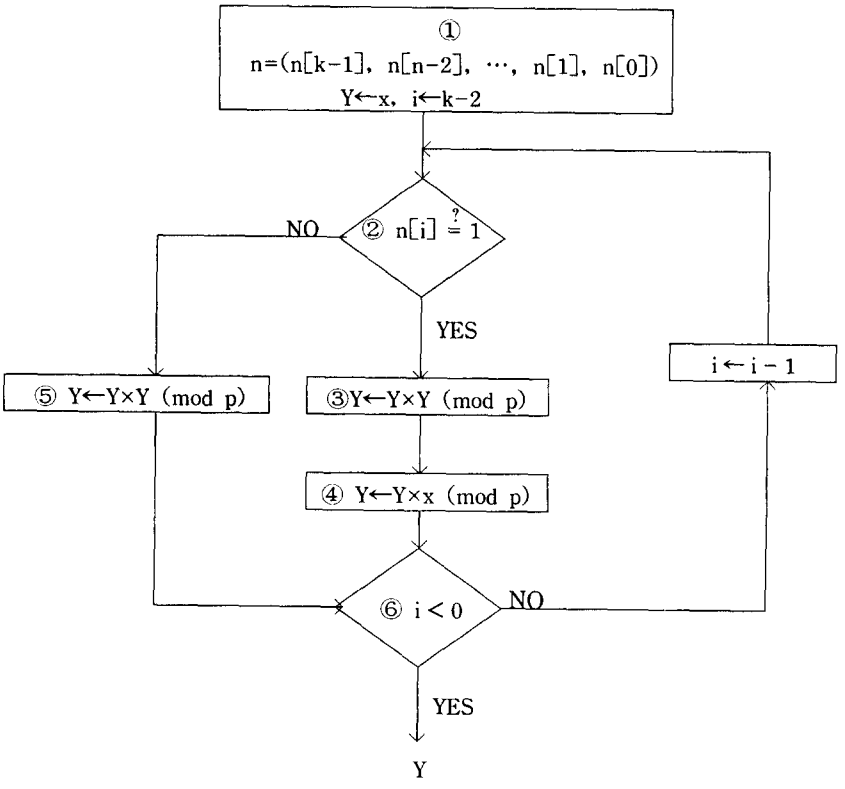


그림 2. 내림차순 이진방식을 이용한 연산의 흐름도

x^{23} (mod p)를 계산하는데 소요되는 모듈러 곱셈 연산 횟수는 7회이다. 이는 올림차순 이진방식 보다 하나 적다. 내림차순 이진방식을 이용하여 x^n을 계산하기 위해 요구되는 일반적인 모듈러 곱셈 연산 횟수는 $\lfloor \log_2 n \rfloor + v(n) - 1$이 된다.

2.2 소인수분해 방식

소인수분해 방식은 x^n (mod p)에서 n이 합성수(composite number)이고 n이 소인수분해될 수 있을 경우에 적용 가능하다. n의 소인수(prime factor) 중에서 가장 작은 소수를 u라 하고, v를 1보다 큰 수라 하자. 만약 $n=u \cdot v$이라면, x^n (mod p)는 우선 $y=x^u$ (mod p)를 계산한 후, 다시 $z=y^v$ (mod p)를 계산함으로 얻을 수 있다. 그러나 n이 큰 소수인 경우, x^n을 x^{n-1}과 x로 분리한 후, n-1을 소인수분해하여 상기의 방식을 적용하여 x^{n-1} (mod p)의 값을 구한 후, 그 결과에 x를 곱하여 x^n (mod p)를 구할 수 있다.

(예제 4) x^{55} (mod p)를 계산하자.

계산과정을 요약하면 표 3과 같다. 여기서 적용된 연산방식은 내림차순 이진방식이다.

x^{55}를 소인수분해 방식으로 구하기 위해 요구되는 모듈러 곱셈 연산은 총 8회가 요구된다. 내림차순 이진방식을 적용했을 경우에 요구되는 모듈러 곱셈의 수가 $\lfloor \log_2 55 \rfloor + v(55) - 1 = 9$이므로 소인수분해 방식이 하나 작다. 일반적으로 소인수분해 방식은 이진방식 보다 작은 수의 모듈러 곱셈이 요구된다. 그러나 n의 중이 매우 작은 경우($v \leq 2$)는 이진방식에서 요

표 3. x^{55} (mod p) 계산 과정

$55 = 5 \times 11 \rightarrow u = 5,\ v = 11$

$y = x^5 \bmod p$

$= ((x^2 \bmod p)^2 \bmod p) \cdot x \bmod p \rightarrow$ 3회의 모듈러 곱셈 연산

$z = y^{11} \bmod p$

$= ((((y^2 \bmod p)^2 \bmod p) \cdot y \bmod p)^2 \bmod p) \cdot y \bmod p$

$\rightarrow$ 5회의 모듈러의 곱셈 연산

$z = x^{55}$ (mod p) $\rightarrow$ 8회의 모듈러의 곱셈 연산

구되는 연산 갯수가 소인수분해 방식 보다 작을 수 있다. 이에 대한 대표적인 예는 다음과 같다. x^{33} (mod p)를 계산할 때 이진방식은 6회, 소인수분해 방식은 7회의 연산이 요구된다.

2.3 m진방식

이진방식은 m진방식으로 일반화될 수 있다. n은 다음과 같이 radix가 m인 수로 표현될 수 있다.

$$n=d_0 \cdot m^t+d_1 \cdot m^{t-1}+ \cdots + d_{t-1} \cdot m^1+d_t \quad (9)$$

여기서, $0 \le d_j < m,\ 0 \le j \le t$

x, x^2, x^3, …, x^{m-1}을 미리 계산하여 표로 저장한다. 따라서 n의 m진수 표현에서 x^{di}는 표에 저장되어 있다. m진방식을 이용하여 x^n (mod p)를 계산하기 위한 과정은 다음과 같다.

① 표로부터 x^{d0}를 선택한 후, x^{d0}를 m 멱승한 $(x^{d0})^m$ (mod p)를 얻는다.

② $(x^{d0})^m$에 표부터 선택된 x^{d1}를 곱하여 $y_1=x^{d_0 \cdot m+d_1}$ (mod p)를 얻는다.

③ y_1을 m 멱승한 후 x^{d2}를 곱하여 $y_2=x^{d_0 \cdot m^2+d_1 \cdot m+d_2}$ (mod p)를 얻는다.

이와 같은 방법을 반복 수행하면 $y_t=x^{d_0 \cdot m^t+d_1 \cdot m^{t-1}+ \cdots +d_{t-1} \cdot m+d_t}$를 계산할 수 있다. 따라서 m진방식이 알고리듬은 다음과 같다.

－알고리듬 D－

① (초기화 단계) $Y_0 \leftarrow 1$, $i \leftarrow 0$, (x, x^2, x^3, …, x^{m-1})을 구하여 표에 구성한다.

② x^{ci}를 표에서 선택한 후, 이를 m 멱승하여$(x^{di})^m$ (mod p)를 구한다.

③ 다음을 계산한다.

$$Y_i=(Y_{i-1})^m \cdot x^{di} \quad (\text{mod } p) \quad (10)$$

④ i가 t+1이면 알고리듬을 종료하고, 그렇지 않으면 $i \leftarrow i+1$하고 단계 ②로 돌아간다.

그리고 m진방식에서 요구되는 모듈러 곱셈의 횟수는 $m=2^k$의 경우 일반적으로 $m-2+(k+1) \cdot t$가 되고, d_i가 '0'인 경우를 고려하면 $m-2+(k+1) \cdot t-\gamma(n)$이 된다. 여기서 $\gamma(n)$은 $d_i=0(i=0, 1, \cdots, t)$의 수이다. 한편 이 일반식을 이진연산의 모듈러 곱셈의 수와 비교하면 $m=2$, $\nu(n)+\gamma(n)=\lfloor \log_2 n \rfloor +1$로 부터 두식이 동일함을 알 수 있다.

2.4 파워트리 방식

x^n (mod p)를 계산하기 위한 방법으로 파워트리를 이용하는 방법이 있다. x^n (mod p)를 계산할 때 파워트리 방식을 적용하기 위해서는 그림 3과 같은 미리 구성되어 있는 파워트리에서 정수 n을 찾아야 한다. n에 대한 지수수열은 트리의 근원인 1에서 n까지의 경로(path) 상에 나타난 일련의 수열이다. 예를 들어 x^{23} (mod p)를 계산하려 할때, 그림 3의 파워트리로 부터 23을 찾은 후, 1에서 부터 23까지의 경로에 존재하는 수를 나열하여 (1, 2, 3, 5, 10, 13, 23)의 지수수열을 얻을 수 있다. 따라서, 6회의 모듈러 곱셈이 x^{23} (mod p)를 계산하는데 요구된다. 이와 같은 파워트리를 생성하는 방법은 다음과 같다. 우선 트리의 k번째 레벨(level)까지가 구성되었다고 가정할 때, (k+1)번째 레벨의 각 원소를 구성하기

위하여 먼저 k번째 레벨의 왼쪽부터 차례로 각 절점(node)에 있는 수 n를 취하여, 각 절점의 수 n에 대한 수열 $(n+1, n+a_1, n+a_2, \cdots, n+a_{k-1}=2n)$을 생성한다. 만약 수열 내의 수가 이미 트리에 나타나 있다면, 수열에서 삭제한다. 여기서 1, a_1, a_2, …, a_{k-1}은 1부터 n까지의 지수수열이다. 그리고 삭제되지 않은 수를 n 아래에 나열하여 절점 n에 대한 k+1번째 트리를 완성한다. 나머지 절점에 대해서도 동일한 방법으로 파워트리를 구성한다. 예를 들어 그림 3에서 4번째 레벨의 수는 5, 6, 8이다. 이 4번째 레벨의 5로 부터 5번째 레벨의 수를 구하기 위해, 우선 1에서 부터 5까지의 수 (1, 2, 3, 5)를 이용하여 5+1, 5+2, 5+3, 5+5를 구한다. 이때 6과 8은 트리에 나타나 있는 수이므로 삭제하고 7과 10을 절점 5에 부속된 5번째 레벨의 수로 간주하여 절점 5 아래에 부착한다.

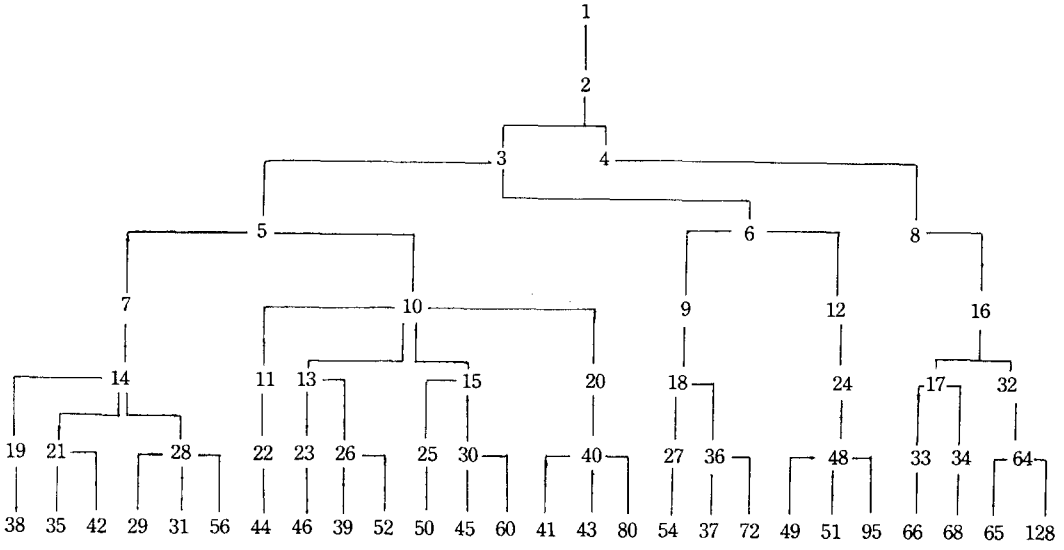


그림 3. 파워트리

컴퓨터 테스트는 파워트리가 그림 3에 목록된 모든 n에 대하여 최적의 결과를 얻을 수 있음을 보인다. 그러나 상당히 큰 n에 대하여 파워트리 방식이 항상 최적의 방법은 아니다.[3)]

2.5 가산고리 방식

x에 대한 모듈러 지수 연산은 지수부가 정수들의 덧셈으로 구성되어 있으므로 $x^n \pmod p$를 계산하는 문제는 n에 대한 가산고리를 구하는 문제에 귀착된다. n에 대한 가산고리는 다음과 같은 정수의 수열로 표현된다.

$$1=a_0, a_1, a_2, \cdots, a_r=n \qquad (11)$$

위의 수열에서 각 원소 다음의 특성을 만족한다.

$$a_i=a_j+a_k, \qquad (k\le j<i,\ i=1, 2, \cdots, r)$$

가산고리 수열의 정의에 의하면 a_1은 2이고, a_2는 2, 3, 4 중의 어느 하나가 될 수 있다. n에 대한 가산고리의 길이는 가산고리 수열에서 '1'을 제외한 원소의 갯수를 의미한다. n에 대한 가산고리의 가장 짧은 길이를 L(n)으로 표시한다. 가산고리의 길이는 $x^n \pmod p$를 계산하기 위한 곱셈 수와 밀접한 관계가 있으므로 가산고리의 길이가 짧을수록 곱셈 연산의 수가 작다. 모든 100이하의 정수 n을 가능한 짧은 가산고리의 길이를 갖는 트리는 그림 4와 같다. 하지만, 임의의 수에 대한 가산고리가 반드시 하나만

존재하는 것은 아니다. 예를 들어 77에 대한 가산고리는 다음과 같이 4가지로 나타낼 수 있다.

1,	2,	4,	8,	9,	17,	34,	43,	77
1,	2,	4,	8,	9,	17,	34,	68,	77
1,	2,	4,	5,	9,	18,	36,	41,	77
1,	2,	4,	5,	9,	18,	36,	72,	77

각 가산고리 수열의 길이는 L(77)=8이다.

L(n)을 결정하는 문제는 처음으로 1894년 H. Dellac에 의해 제안되었고, 소인수분해 방식을 언급했던 E.de Jonquières는 $p<200$인 모든 소수에 대하여 L(p)의 값을 구하였다.[3)]

m진수로 표현된 정수 $n(d_0 \cdot m^t+d_1 \cdot m^{t-1}+\cdots+d_{t-1} \cdot m^t+d_t)$에 대한 가산고리는 다음과 같다. $m=2^k$인 경우, 일반적인 가산고리는 다음과 같은 형태를 지닌다.[3)]

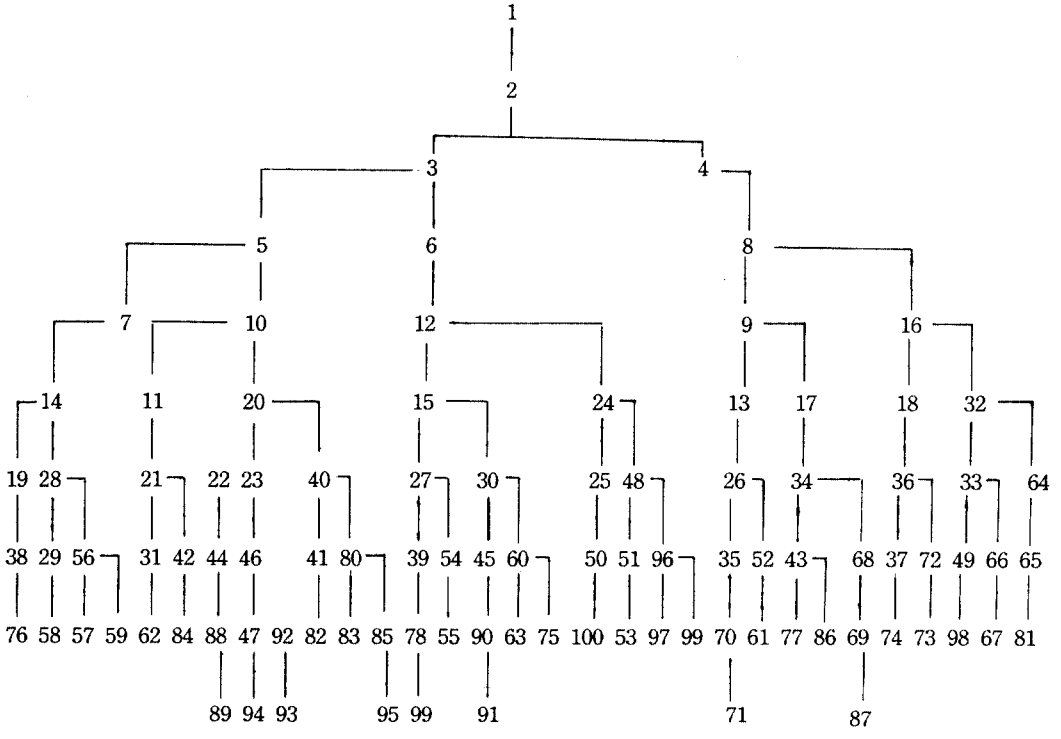


그림 4. $n \leq 100$에 대한 곱셈 수를 최소화한 트리

$$
\begin{aligned}
&1,\ 2,\ 3,\ \cdots,\ m-2,\ m-1,\\
&2d_0,\ 4d_0,\ \cdots,\ m \cdot d_0,\ m \cdot d_0+d_1,\\
&2(m \cdot d_0+d_1),\ 4(m \cdot d_0+d_1),\ \cdots,\ m(m \cdot d_0+d_1),\ m^2 \cdot d_0+m \cdot d_1+d_2,\\
&\quad \cdot \qquad \cdot \qquad \cdot\\
&\quad \cdot \qquad \cdot \qquad \cdot\ ,\ d_0 \cdot m^t+d_1 \cdot m^{t-1}+\cdots+d_t
\end{aligned}
\tag{12}
$$

(예제 5) 만약 $m=2^2$인 경우, 정수 n=156에 대한 m진수 가산고리는 다음과 같이 구할 수 있다. t=3, k=2, $d_0=2$, $d_1=1$, $d_2=3$, $d_3=0$이 된다. 따라서 가산고리는 아래와 같다.

1, 2, 3
$2\cdot 2$, $4\cdot 2$, $4\cdot 2+1$,
$2(4\cdot 2+1)$, $4(4\cdot 2+1)$, $4^2\cdot 2+4\cdot 1+3$,
$2(4^2\cdot 2+4\cdot 1+3)$, $4(4^2\cdot 2+4\cdot 1+3)$

결국 156에 대한 가산고리 수열은(1, 2, 3, 4, 8, 9, 18, 36, 39, 78, 156)이 된다. 156에 대한 가산고리 길이는 L(156)=10이다. 만약 $m=2^k$인 경우, m진수로 표현된 n에 대한 일반적인 가산고리의 길이는 $m-2+(k+1)\cdot t-\gamma(n)$이 된다. 여기서, $\gamma(n)$은 $d_j=0(0\le j\le t)$의 갯수이다.

2진수로 표현된 (즉, m=2) 정수 n에 대한 가산고리의 길이는 다음과 같다.

$$L(2^{e0}+2^{e1}+\cdots+2^{et})\le e_0+t,\quad (e_0>e_1>\cdots>e_t\ge 0) \tag{13}$$

$e_0=\lfloor \log_2 n\rfloor$이고, e_i는 n을 radix를 2로 하는 다항식 표현의 지수부이다. 표현의 편의를 의해 다음과 같은 식을 정의한다.

$\lambda(n)=\lfloor \log_2 n\rfloor$
$\nu(n)$=n의 이진 표현에서의 '1'의 갯수 (14)

예를 들어 $\lambda(17)=4$이고, $\nu(17)=\nu(10001)=2$가 된다. 위의 일반적인 m진법으로 표시된 n에 대한 가산고리의 길이 L(n)으로부터 이진수로 표현된 가산고리의 길이는 $\lambda(n)+\nu(n)-1$이 된다. 따라서 이진수로 표시된 정수 n에 대한 가산고리의 길이와 $\lambda(n)$ 및 $\nu(n)$과의 관계는 다음과 같다.

$$L(n)\le\lambda(n)+\nu(n)-1 \tag{15}$$

(1) 가산고리를 위한 용어정의

일반적으로 가산고리는 다음과 같은 식이 만족되므로 '상승(acscending)' 특성을 갖는다.

$$1=a_0<a_1<a_2<\cdots<a_r=n \tag{16}$$

왜냐하면 두개의 원소 a_i, a_j (i<j)가 동일하면 그들 중의 하나는 수열에서 소거되고, n 보다 큰 항도 소거되어 그 수열은 다시 올림차순(ascending order)으로 재배열되기 때문이다. 이후 기술될 가산고리는 상승 가산고리만을 의미한다. 가산고리의 정의에 의해, $1\le i\le r$에 대하여 a_i는 다음과 같이 표현될 수 있다.

$$a_i=a_j+a_k \tag{17}$$

가산고리와 연관된 특별한 용어들을 정의한다.

1) doubling

만약 j=k=i-1이 성립하면, (즉 j와 k가 i-1과 동일하면) 식 (16)의 단계 i는 'doubling'이라고 한다. 이 경우 a_i는 최대값으로 $2a_{i-1}$을 갖는다.

2) star step

만약 j가 i-1이면, 단계 i는 'star step'이라 한다.

3) small step

$\lambda(a_i)=\lambda(a_{i-1})$이면, 단계 i는 'small step'이라고 한다.

일반적으로 원소 a_i는 $a_{i-1}<a_i\le 2a_{i-1}$의 관계가 성립하므로 $\lambda(a_i)$는 항상 $\lambda(a_{i-1})$이거나 $\lambda(a_{i-1})+1$이 된다. 식 (16)과 같은 임의의 가산고리에 대한 길이 r은 $\lambda(n)$+(small step의 수)이 된다. 가산고리의 각 단계(step)간의 기본 특성은 다음과 같다.

① 단계 1은 항상 'doubling'이다.

② doubling은 명백히 star step이지만 결코 small step이 아니다.

③ 단계 i가 doubling이면 단계 i+1은 star step이 된다.

④ 단계 i가 small step이 아니면, 단계 i+1은 small step, star step, 혹은 양자가 될 수 있다.

⑤ 단계 i+1이 small step도 star step도 아니면, 단계 i는 small step이다.

4) star chain

이 고리는 오로지 star step만으로 이루어진 가산고리이다. 즉, 이 가산고리의 각각의 항 a_i는 a_{i-1}과 이전의 $a_k(k\le i-1)$의 합임을 의미한다. n에 대한 star chain의 최소 길이가 $L^*(n)$이라면 식 (18)이 성립한다.

$$L(n) \le L^*(n) \qquad (18)$$

[정리 1]

임의의 정수 n에 대한 가산고리는 d개의 doubling과 f(=r-d)개의 nondoubling으로 구성된다. 정수 n은 다음의 관계식을 만족한다.

$$n \le 2^{d-1} \cdot F_{f+3} \qquad (19)$$

여기서, F_{f+3} : Fibonacci 수
($F_1=1$, $F_2=1$, $F_n=F_{n-1}+F_{n-2}$)

(증 명)

r=d+f가 성립하므로, r=1일 경우, d=1, f=0, $F_3=2$가 되어 n=2가 됨으로서 식 (19)를 만족한다. r>1일때, 다음의 세가지의 경우가 존재한다.

① 단계 r이 doubling이면, $a_{r-1}=\frac{1}{2} \cdot n \le 2^{d-2} \cdot F_{f+3}$을 만족한다.

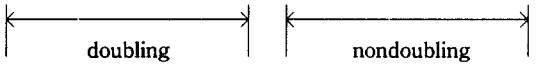
② 단계 r과 r-1이 모두 nondoubling이면, $a_{r-1} \le 2^{d-1} \cdot F_{f+2}$이고 $a_{r-2} \le 2^{d-1} \cdot F_{f+1}$이 된다. 그러므로 Fibonacci 수열의 정의에 의해 $n=a_r \le a_{r-1}+a_{r-2} \le 2^{d-1}(F_{f+2}+F_{f+1}=)2^{d-1} \cdot F_{f+3}$이 성립한다.

③ 단계 r은 nondoubling이지만, 단계 r-1은 doubling이면, $a_{r-2} \le 2^{d-2} \cdot F_{f+2}$이고 $n=a_r \le a_{r-1}+a_{r-2}=3a_{r-2}$이다. 따라서 $2F_{f+3}-3F_{f+2} \ge 0$을 고려하면 $n \le 3a_{r-2} \le 3 \cdot 2^{d-2} \cdot F_{f+2} \le 2^{d-1} \cdot F_{f+3}$을 만족한다.

위의 가능한 세가지에 대하여 $n \le 2^{d-1} \cdot F_{f+3}$을 만족한다. (증명완료)

따라서 연속 d개의 doubling과 f개의 nondoubling을 가지는 정수 n에 대한 가산고리는 다음과 같이 표현될 수 있다.

$$1, 2, \cdots, 2^{d-1}, 2^{d-1} \cdot F_3, \quad 2^{d-1} \cdot F_4, \cdots, 2^{d-1} \cdot F_{f+3} \qquad (20)$$



[따름정리 1]

임의의 정수 n에 대한 가산고리가 f개의 nondoubling과 s개의 small step으로 구성된 s와 f는 다음의 관계식을 만족한다.

$$s \le f \le 3.271s \qquad (21)$$

(증명)

nondoubling의 수는 small step의 수를 포함하므로 $s \le f$이다. 가산고리 길이는 $\lambda(n)+s$이므로, $f \ge 0$에 대해 $d+f=\lambda(n)+s$가 성립한다. 또 다음의 식이 성립한다.

$$\begin{aligned} F_{f+3} &= \phi^{f+3} \cdot 1/\sqrt{5} \\ &= \phi^f \cdot \{\tfrac{1}{2}(1+\sqrt{5})\}^3 \cdot 1/\sqrt{5} \\ &= 1.8\phi^f \\ &\le 2\phi^f \end{aligned} \qquad (22)$$

여기서, ϕ는 golden ratio : $\frac{1}{2}(1+\sqrt{5})$를 의미한다.

따라서 식 (22)에 의하여 $2^{\lambda(n)} \le n \le 2^{d-1} \cdot F_{f+3} \le 2^d \cdot \phi^f = 2^{\lambda(n)+s} \cdot (\phi/2)^f$가 성립한다. 그러므로 여기에 ln을 취하여 $0 \le s \cdot \ln 2 + f \cdot \ln(\phi/2)$을 얻을 수 있고, $\ln 2/\ln(2/\phi)=3.2706$이므로 식 (21)은 성립한다.[2,3] (증명완료)

(2) 특별한 n에 대한 L(n)의 값

임의의 가산고리 수열에서 임의의 원소 $a_i \le 2^i$, $\log_2 n \le r$을 만족하므로, $L(n) \ge \lceil \log_2 n \rceil$이다. $n=2^A$, 2^A+2^B인 경우의 가산고리의 길이의 최소값(하한값)은 각각 다음과 같다.

$$\begin{aligned} &L(2^A)=A \\ &L(2^A+2^B)=A+1, \quad (A>B) \end{aligned} \qquad (23)$$

따라서 임의의 수 n에 대한 $v(n) \le 2$일 경우, 최소 길이가 상기와 같으므로 곱셈수는 최소가 된다. 이는 [정리 2]에 의해 $v(n)=3$인 경우까지 확장될 수 있다.

[정리 2]

$$L(2^A+2^B+2^C)=A+2, \quad (A>B>C) \qquad (24)$$

이 성립한다.

(증 명)

하나의 small step을 가지는 모든 가산고리 다음 6가지 형태 중의 하나로 나타낼 수 있다. 단, '…'은 'doubling'을 표시하고 밑줄친 부분이 small step를 의미한다.

① 1, …, $\underline{2^A, 2^A+2^B}$, …, $2^{A+C}+2^{B+C}$; A>B≥0, C≥0.

② 1, …, $\underline{2^A, 2^A+2^B}$, $2^{A+1}+2^B$, …, $2^{A+C+1}+2^{B+C}$; A>B≥0, C≥0.

③ 1, …, $\underline{2^A, 2^A+2^{A-1}}$, $2^{A+1}+2^{A-1}$, 2^{A+2}, …, 2^{A+C} ; A>0, C≥2.

④ 1, …, $\underline{2^A, 2^A+2^{A-1}}$, $2^{A+1}+2^A$, 2^{A+2}, …, 2^{A+C} ; A>0, C≥2. (25)

⑤ 1, …, $\underline{2^A, 2^A+2^{A-1}}$, …, $2^{A+C}+2^{A+C-1}$, $2^{A+C-1}+2^{A+C+1}$, …, $2^{A+C+D+1}+2^{A+C+D-2}$; A>0, C>0, D≥0.

⑥ 1, …, $\underline{2^A, 2^A+2^B}$, 2^{A+1}, …, 2^{A+C} ; A>B≥0, C≥1.

이 6가지의 형태는 $\nu(n)>2$를 갖지 않으므로 $L(2^A+2^B+2^C)$는 A+1 보다 커야만 하고 이진방식의 상한선에 의해 $L(2^A+2^B+2^C)\leq A+2$이므로 $L(2^A+2^B+2^C)=A+2$이다. (증명완료)

또 A>B>C>D일때, $L(2^A+2^B+2^C+2^D)$는 이진방식에 의해 기껏해야 A+3이고 정리 B에 의해 적어도 A+2이다. 따라서 $\nu(n)=4$일때, L(n)은 다음의 정리에 의해 결정된다.

[정리 3]

$\nu(n)\geq 4$일때, 다음의 4가지 경우를 제외하면 $L(n)\geq\lambda(n)+3$이다.

① A-B=C-D	(예 : n= 15) (26)
② A-B=C-D+1	(예 : n= 23)
③ A-B=3, C-D=1	(예 : n= 39)
④ A-B=5, B-C=C-D=1	(예 : n=135)

특히 상기의 4가지 경우의 n에 대한 L(n)=A+2이다.

(3) 가산수열 구성을 위한 알고리듬

본 절에서는 임의의 수에 대한 가산수열(addition sequence)을 구성할 수 있는 알고리듬을 소개한다. 이 알고리듬은 {1, 2, n}으로 구성되어 있는 기본수열(protosequence)로 부터 시작된다. 이 기본수열은 이후 기술될 알고리듬을 이용하여 각 단계마다 다른 변형된 수열로 변경되고, 각 단계에서의 기본수열의 마지막 수는 이전 단계에서의 기본 수열의 마지막 수보다 작다. 따라서 아래에 제시될 알고리듬을 수행하면, 결국 1, 2에서 출발하여 n에서 종료되는 가산고리가 구성되며, 이 때의 기본 수열은 {1, 2, …, f_2, f_1, f} (여기서, f=n)로 표현된다. 다음으로 가산고리의 구성을 위해 요구되는 2가지 알고리듬을 도입한다.

-알고리듬 E-

임의의 수 f가 작은 소수 p(=3, 5, 7)에 의해 나누어 진다면, 기본수열에 다음의 삽입수열을 포함시킨다.

$$f/p,\ 2f/p,\ \cdots,\ f \qquad (27)$$

(예제 6) f=48인 경우의 가산고리 수열을 구해보자.

f=48은 3으로 나누어지므로 p=3이 된다. 기본수열은 {1, 2, 48}이고, f/p=48/3=16, 2f/p=32이므로 기본수열에 16, 32를 삽입하여야 한다. 1단계 변형된 수열 {1, 2, 16, 32, 48}을 구한다. 다시 {1, 2, 16}을 기본수열로 하여 2단계 변형된 수열을 구해야 한다. 그런데 16은 2의 멱승이므로 간단히 4, 8을 삽입하여, 2단계 변형된 수열 {1, 2, 4, 8, 16}을 얻을 수 있다. 1단계에서 구한 수열과 2단계에서 구한 수열을 모으면 48에 대한 가산고리 수열 {1, 2, 4, 8, 16, 32, 48}을 얻을 수 있다. 따라서 48에 대한 가산수열의 길이는 6이 됨을 알 수 있다. 이 결과는

이진방식에서의 $\lambda(48)=5$, $\nu(48)=2$로 부터 구한 $L(48)=6$과 일치한다. 알고리듬 E를 이용하여 구한 길이가 이진방식에서 구한 길이와 동일한 것은 이진방식에서의 최소 길이를 가지기 위한 조건이 $\nu(n)\leq 2$이므로 상기 값이 이 조건을 만족하기 때문이다.

다음의 예제는 알고리듬 E에서 구한 가산고리가 이진방식에서 구한 길이 보다 짧음을 나타낸다.

(예제 7) f=117인 경우의 가산고리 수열을 구해 보자.

f=117은 3으로 나누어지므로 p=3이 된다. 기본수열은 {1, 2, 117}이고, t/p=117/39이므로 기본수열에 39, 78을 삽입한다. 1단계 변형된 수열은 {1, 2, 39, 78, 117}이 되고, 다시 {1, 2, 39}를 기본수열로 하여 2단계 변형된 수열을 구한다. 39/3=13이므로, 13, 26을 삽입하여 {1, 2, 13, 26, 39}을 구성한다. 다시 1, 2, 13으로 3단계 변형된 수열을 구하면, {1, 2, 3, 6, 12, 13}을 구성할 수 있다. 결국 가산고리 수열은 {1, 2, 3, 6, 12, 13, 26, 39, 78, 117}이 된다. 따라서 117에 대한 가산고리 수열의 길이가 9임을 알 수 있다. 이는 이진방식에서 $\lambda(117)=6$, $\nu(117)=5$로 부터 구한 $L(117)=10$ 보다 하나 작다.

-알고리듬 F-

임의의 작은 수 s를 선택하여, 다음 조건을 만족하는 k, u를 구한 후, 다음과 같이 삽입수열을 구성하여 기본수열에 삽입한다.

조 건 : $f/s\geq 2^u$, $\lfloor f/2^u \rfloor = k$

삽입수열 : $d=f-k\cdot 2^u$, $f-d=k\cdot 2^u$, $k\cdot 2^{u-1}$, …, $k/2$, k (28)

(예제 8) f=79인 경우의 가산고리 수열을 구해 보자.

임의의 s=5를 선택하면, $f/s=79/5=15.8\geq 2^u$로 부터 u=3, $\lfloor f/2^u \rfloor = \lfloor 79/2^3 \rfloor = 9$이므로, k=9를 구할 수 있다. 따라서 삽입수열은 $79-9\cdot 2^3=7$, 79-7=72, 36, 18, 9가 되어 1단계 변형된 수열은 {1, 2, 7, 9, 18, 36, 72, 79}가 된다. 기본수열 {1, 2, 7}로부터 2단계로 변형된 수열은 3, 4를 삽입하여 {1, 2, 3, 4, 7}이 된다. 결국 79에 대한 가산고리 수열은 {1, 2, 3, 4, 7, 9, 18, 36, 72, 79}가 된다. 이 수열의 길이가 9이므로 이진방식에서의 $\lambda(79)=6$, $\nu(79)=5$로 부터 구한 $L(79)=10$ 보다 하나 작다.

(예제 9) f=382인 경우의 가산고리 수열을 구해 보자.

임의의 작은 수 s=14을 선택하면, $f/s=382/14=27.3\geq 2^u$에서 u=4, $\lfloor f/2^u \rfloor = \lfloor 382/2^4 \rfloor = 23$이므로 k=23을 구할 수 있다. 따라서 삽입수열은 $382-23\cdot 2^4=14$, 382-23=368, 184, 92, 46, 23가 되어 1단계 변형된 수열은 {1, 2, 14, 23, 46, 92, 184, 368, 382}가 된다. 기본수열 {1, 2, 14}로 부터 2단계 변형된 수열은 {3, 4, 7}를 삽입하여 {1, 2, 3, 4, 7, 14}가 된다. 결국 382에 대한 가산고리 수열은 {1, 2, 3, 4, 7, 14, 23, 46, 92, 184, 368, 382}로 길이가 11이 된다. 이 수열의 길이가 11이므로 이진방식에서의 $\lambda(382)=8$, $\nu(79)=7$로 부터 구한 길이 $L(79)=14$ 보다 3이 작다.

따라서 알고리듬 E, F를 이용하여 구한 임의의 정수 n에 대한 가산고리수열의 길이는 2진 방식에서 구한 길이 보다 일반적으로 짧음을 알 수 있다.

3. 결 론

본 논문에서는 연산 횟수 측면에서 $x^n \pmod p$를 계산하기 위한 최적의 모듈러 지수 연산알고리듬을 찾으려는 의도에서, 이진방식중 올림차순 이진방식과 내림차순 이진방식, 합성수의 모듈러 지수 연산을 좀 더 효율적으로 계산하기 위한 소인수 분해 방식, 3진방식 및 4진방식을 적용 가능케 하기 위한 m진방식, 또 고리를 이용한 파워트리 및 가산고리 방식을

소개했다. 이진방식은 알고리듬이 다른 방식에 비하여 간단하지만 중(weight)이 3 이상인 경우는 연산 횟수 측면에서 최적이 아니다. 소인수 분해 방식은 합성수를 소인수 분해하여 이진방식의 연산알고리듬을 그대로 적용하기 때문에 중이 큰 경우 연산 횟수를 감소시킬 수 있다. 3진 및 4진 방식은 radix가 크기 때문에 연산횟수가 경우에 따라 감소한다. 파워트리 방식은 고리의 구성 방법은 쉽지만 고리를 계산해야 하고 n이 큰 경우 연산 횟수 측면에서 최적이 아니다. 가산고리수열의 모듈러 곱셈 연산 횟수는 λ(n)+(small step의 수)이므로 small step의 수를 가능한 줄이면 짧은 가산고리 수열을 얻을 수 있고, 결과적으로 곱셈 연산 횟수를 줄일 수 있다. 일반적으로 가산고리 방식은 모듈러 곱셈 연산 횟수가 다른 방식 보다 많이 감소하지만 가산수열의 구성이 어렵다. 따라서 효율적인 가산수열의 구성을 위한 알고리듬에 대한 연구가 추후에 수행되어야 할 것이다.

참 고 문 헌

1. A. Shamir, "Identity-based Cryptosystem and Signature Scheme," Lecture Notes in Computer Science. 196, Advances in Cryptology-CRYPTO'84, Springer-Verlag, pp.47-53, 1985.

2. D.E. Knuth, *The Art of Computer Programming*, Addison-Wesley Company, Second Edition, Vol. 1, pp.78-83, 1981.

3. D.E. Knuth, *The Art of Computer Programming*, Addison-Wesley Company, Second Edition, Vol. 2, pp.441-466, 1981.

4. Hikaru Morita, "A Fast Modular-Multiplication Algorithm based on a Higher Radix," Lecture Notes in Computer Science. 435, Advances in Cryptology-CRYPTO'89, Springer-Verlag, pp.387-399, 1990.

5. G.B. Agnew, R.C. Mullin and S.A. Vanstone, "Fast Exponentiation in $GF(2^n)$," Lecture Notes in Computer Science. 330, Advances in Cryptology-EUROCRYPT'88, Springer-Verlag, pp.251-255, 1989.

6. Jurjen Bos and Matthijs Coster, "Addition Chain Heuristics," Lecture Notes in Computer Science. 435, Advances in Cryptology-CRYPTO'89, Springer-Verlag, pp.400-407, 1990.

7. J.V. Leeuwen, *Algorithms and Complexity*, The MIT Press, pp.637, 1990.

8. P.A. Findlay and B.A. Johnson, "Modular Exponentiation Using Recursive Sums of Residues," Lecture Notes in Computer Science. 435, Advances in Cryptology-CRYPTO'89, Springer-Verlag, pp.371-386, 1990.

9. S.I. Kawamura and K. Hirano, "A Fast Modular Arithmetic Algorithm Using a Residue Table," Lecture Notes in Computer Science. 330, Advances in Cryptology-EUROCRYPT'88, Springer-Verlag, pp.245-250, 1989.

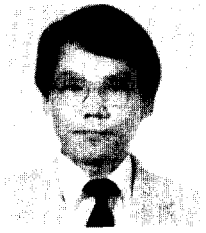
10. S. Kawamura, K. Takbayashi and A. Shimbo, "A Fast Modular Exponentiation Algorithm," IEICE Tran. Vol.E 74, No. 8, August, 1991.

11. Y. Yacobi, "Exponentiating Faster with Addition Chains," Lecture Notes in Computer Science. 473, Advances in Cryptology-EUROCRYPT'90, Springer-Verlag, pp.222-229, 1991.

□ 著者紹介

李 錫 來

1992年 漢陽大學校 工科大學 電子通信工學科 卒業
1992年 3月～現在 漢陽大學校 大學院



廉 興 烈

1981年 漢陽大學校 工科大學 電子工學科 卒業(工學士)
1983年 漢陽大學校 大學院 電子工學科 卒業(工學碩士)
1990年 漢陽大學校 大學院 電子工學科 卒業(工學博士)
1982年 12月～1990年 9月 韓國電子通信研究所 先任研究員
1990年 9月～現在 順天鄕大學校 工科大學 電子工學科
主 關心分野：暗號理論, 符號理論, 데이타通信 分野

李 晩 榮

1924年 11月 30日生
漢陽大學校 名譽教授
韓國通信情報保護學會 會長
受賞：大韓民國 學術院賞
著書：Error Correcting Coding Theory, McGraw-Hill, New York, 1989.