DOI: 10.3745/KIPSTC.2011.18C.1.001

$m-$진법 모듈러 지수연산

이 상 운†

요 약

암호학의 암호 생성과 해독, 소수판별법의 성능은 대부분 $a^b(\mathrm{mod}\,n)$의 모듈러 지수연산의 효율적 구현여부로 결정된다. 모듈러 지수연산법에는 표준 이진법이 최선의 선택으로 알려져 있다. 그러나 큰 자리수의 b에 대해서는 $d-ary, (d=2,3,4,5,6)$이 보다 효율적으로 적용된다. 본 논문에서는 $b\equiv 0(\mathrm{mod}\,m)$, $2\le m\le 16$인 경우 b를 m-진법으로 변환시켜 수행하는 방법과 m-진법 수행과정에서 결과 값이 1 또는 a가 발생하는 경우 곱셈 수행횟수를 획기적으로 줄이는 방법을 제안하였다.

키워드 : 소수 판별, 확률적 판별법, 모듈러 지수연산 법, 이진법, 반복 제곱법

Modular Exponentiation by m-Numeral System

Sang-Un Lee†

ABSTRACT

The performance and practicality of cryptosystem for encryption, decryption, and primality test is primarily determined by the implementation efficiency of the modular exponentiation of $a^b(\mathrm{mod}\,n)$. To compute $a^b(\mathrm{mod}\,n)$, the standard binary squaring still seems to be the best choice. But, the $d-ary, (d=2,3,4,5,6)$ method is more efficient in large b bits. This paper suggests m-numeral system modular exponentiation. This method can be apply to $b\equiv 0(\mathrm{mod}\,m)$, $2\le m\le 16$. And, also suggests the another method that is exit the algorithm in the case of the result is 1 or a.

Keywords : Primality Test, Cryptosystem, Modular Exponentiation, Binary Method, Repeated Squaring

1. 서 론

밑 a, 지수 b의 연산을 지수연산 (exponentiation)이라 하며, a^b을 n으로 나눈 나머지를 구하는 $a^b(\mathrm{mod}\,n)$을 모듈러 지수연산 (modular exponentiation)이라 한다. 모듈러 지수연산은 암호학 (cryptography) 분야의 암호 생성과 해독과 더불어 소수판별법 (primality test, PT)에도 널리 사용되고 있다[1,2].

모듈러 지수 연산법에는 $a^b(\mathrm{mod}\,n)$을 직접 계산하는 직접법 (direct method), $(a\times a)(\mathrm{mod}\,n)$을 $b-1$회 수행하는 곱셈법 (multiplication method), $b_{(10)}$을 $b_kb_{k-1}b_{k-2}\cdots b_1b_{0_{(2)}}$의 이진수로 변환시켜 1 비트씩 계산하는 이진법 (binary method), d 비트씩 계산하는 $d-ary$법, $a^b=(a^2)^{(b-1)/2}$ 형태로 반감시키면서 계산하는 반복 제곱법 (repeated squaring)이 있다. 또한, 고속 푸리에 변환 (Fast Fourier Transform, FFT) 곱셈법 또는 몽고메리 감소법 (Montgomery reduction)을 적용하기도 한다.[3-6]

일반적으로 RSA와 같이 큰 자리수의 암호 생성과 해독시 실수 연산을 하고 모듈러 연산을 하게 되면 메모리와 연산 속도가 현저히 떨어져 이진값을 이용하여 연산을 하게 되어 이진법을 표준 이진법 (standard binary method)이라 한다[7,8].

표준 이진법은 b_k에서 $c=a$로 초기치를 설정하고, b_{k-1}부터 b_0까지 각 비트에서 $c=(c\times c)(\mathrm{mod}\,n)$의 제곱 (squaring)을 수행하며, $b_i=1$인 비트 값에 대해 $c=(c\times a)(\mathrm{mod}\,n)$의 곱셈 (multiplication)을 수행한다. 즉, 이진법은 누승 (2^d승)과 곱셈을 수행하는 제곱-곱셈 (squaring-and-multiplication)법이다. 이진법은 이진수의 1 비트 (bit) 단위

† 정 회 원 : 강릉원주대학교 멀티미디어공학과 부교수
논문접수 : 2010년 9월 16일
수 정 일 : 1차 2010년 10월 19일
심사완료 : 2010년 10월 19일

로 수행하므로 $1-ary$ (1항)법이라 한다.

$d-ary$ (d항)법은 이진수의 d 비트 단위로 수행하는 방식으로 누승-곱셈법이라 할 수 있다. $d-ary$법은 2^d ($d>1$)에 대해 0과 1을 제외한 2^d-2개의 사전처리 (preprocessing)를 수행하여 저장하고, d 비트 단위로 d회 제곱을 한 후 비트 값에 대해 사전처리된 값을 곱하는 방식이다. 일반적으로 암호와 복호에 사용되는 큰 자리수에 대해서는 $d-ary$법이 표준 이진법 ($d=1$)보다 효율적인 것으로 알려져 있다[3,7,9,10].

본 논문에서는 $a^b \pmod n$ 계산시 b의 값에 따라 표준 이진법보다 다른 진법을 적용하면 곱셈 (누승+곱셈) 횟수를 감소시킬 수 있음을 보인다. 따라서 본 논문에서는 $b \equiv 0 \pmod m, (m=3,5,6,7,10,12)$이면 $m-$진법을, 그렇지 않으면 $d-ary(2^d, d=1,2,3,4)$을 적용한 일반화된 $m-$진법을 제안한다. 2장에서는 $d-ary$법을 고찰한다. 3장에서는 표준 이진법을 일반화한 $m-$진법을 제안한다. 4장에서는 제안된 방법의 적합성을 검증한다.

2. $d-ary$ 모듈러 지수연산법

암호체계의 암호와 복호에 적용되는 $a^b \pmod n$에서 $a<n$, $b<n$을 적용하며, 소수판별법에서는 $a<n$, $b=n-1<n$을 적용한다. $a^b \pmod n$, $a<n, b<n$에 대해 a와 b가 매우 큰 수일 경우, a^b을 직접 계산할 수 없어 $a^b \pmod n$의 해를 한 번에 얻기는 현실적으로 불가능하다. 지수연산은 $(a^b)^c = a^{b\times c}, a^b \times a^c = a^{b+c}$로 계산된다.

표준 이진법은 10진수 b를 2진수 $b_k b_{k-1} \cdots b_2 b_{1_{(2)}}$으로 변경시켜 (그림 1)과 같이 수행한다[7].

```
[방법 1] - a^b (mod n)
Modular-Exponentiation (a, b, n)
    c = 1
    b = (b_k, b_{k-1}, ..., b_1, b_0)_2
    for   i = k down to 0 do
          c = (c × c) mod n /* square
          if   b_i = 1 then c = (c × a) mod n
    return c
[방법 2] - a^b (mod n)
Modular-Exponentiation (a, b, n)
    c = a, b = (b_k, b_{k-1}, ..., b_1, b_0)_2
    for   i = k-1 down to 0 do
          c = (c × c) mod n
          if   b_i = 1 then c = (c × a) mod n
    return c
```

(그림 1) 표준 이진법

(그림 1)에서 이진법은 $2^{k+1} > n > 2^k$의 k에서는 $c=a$로 초기치를 설정하고, $k-1$부터 0까지 k회의 제곱 (d승)과 $b_i=1$인 d진법의 비트 값 개수 $l(b)$회의 곱셈을 수행한다. 즉, MSB에서 LSB로 좌측에서 우측 (L2R: Left-to-Right)으로 계산한다. [방법 2]가 [방법 1]에 비해 곱셈횟수가 1회 적기 때문에 본 논문에서는 [방법 2]를 적용한다.

표준 이진법을 보다 효율적으로 처리하는 방법으로 (그림 2)의 $d-ary$ 방법이 있다. 이 방법은 d 비트씩 묶어 처리한다. $d-ary$에는 2비트씩 묶는 Quaternary법 (4진법), 3비트씩 묶는 Octal법 (8진법) 등이 있다. 이진법은 $1-ary$라 할 수 있다.

$a^b, b=250=11111010_{(2)}$에 대해 표준 이진법과 $2-ary$, $3-ary$를 비교한 결과는 <표 1>에 제시되어 있다. 표준 이진법은 곱셈을 12회 수행하였다. $2-ary$는 사전처리에서 $2^m-2=2$회와 9회의 곱셈을 수행하여 11회를 수행한다. $3-ary$는 $2^m-2=6$회의 사전처리와 8회의 곱셈으로 14회를 수행한다. 따라서 $2-ary$가 가장 좋은 결과를 얻는다.

```
a^b (mod n), b = (b_k, b_{k-1}, ..., b_1, b_0)_2
Preprocessing (사전 처리)
    a_0 = 1
    for   i = 1 to (2^k - 1) do
          a_i = (a_{i-1} × a) mod n
c = 1
Modular-Exponentiation (a, b, n)
    for   i = n down to 0 do
          for   j = 0 to k-1

                c = (c × c) mod n /* power
          if   b_i ≠ 0 then c = (c × a_{b_i}) mod n
    return d
```

(그림 2) $d-ary$법

〈표 1〉 $d-ary$ 지수연산법

(a) $1-ary$ (이진법) : $1-1-1-1-1-1-0-1-0$

[방법 2]		누승	곱셈 (비트 값)
b_7	1	초기치 : $c=a$	
b_6	1	$(a^1)^2 = a^{1\times 2} = a^2$	$(a^2)(a^1) = a^{2+1} = a^3$
b_5	1	$(a^3)^2 = a^{3\times 2} = a^6$	$(a^6)(a^1) = a^{6+1} = a^7$
b_4	1	$(a^7)^2 = a^{7\times 2} = a^{14}$	$(a^{14})(a^1) = a^{14+1} = a^{15}$
b_3	1	$(a^{15})^2 = a^{15\times 2} = a^{30}$	$(a^{30})(a^1) = a^{30+1} = a^{31}$
b_2	0	$(a^{31})^2 = a^{31\times 2} = a^{62}$	
b_1	1	$(a^{62})^2 = a^{62\times 2} = a^{124}$	$(a^{124})(a^1) = a^{124+1} = a^{125}$
b_0	0	$(a^{125})^2 = a^{125\times 2} = a^{250}$	

$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^6 \rightarrow a^7 \rightarrow a^{14} \rightarrow a^{15} \rightarrow a^{30} \rightarrow a^{31} \rightarrow a^{62} \rightarrow a^{124} \rightarrow a^{125} \rightarrow a^{250}$

(b) $2-ary$ (4진법) : $11-11-10-10$

사전처리		$10 : a \rightarrow a^2,\ 11 : a^2 \rightarrow a^3$	
구분		누승	곱셈 (비트 값)
$b_7 b_6$	11	a^3 /* 사전처리에서 구한 a^3 사용	
$b_5 b_4$	11	$(a^3)^4 = a^{3\times 4} = a^{12}$ $(a^3 \rightarrow a^6 \rightarrow a^{12})$	$(a^{12})(a^3) = a^{12+3} = a^{15}$ $(a^{12} \rightarrow a^{15})$
$b_3 b_2$	10	$(a^{15})^4 = a^{15\times 4} = a^{60}$ $(a^{15} \rightarrow a^{30} \rightarrow a^{60})$	$(a^{60})(a^2) = a^{60+2} = a^{62}$ $(a^{60} \rightarrow a^{62})$
$b_1 b_0$	10	$(a^{62})^4 = a^{62\times 4} = a^{248}$ $(a^{62} \rightarrow a^{124} \rightarrow a^{248})$	$(a^{248})(a^2) = a^{248+2} = a^{250}$ $(a^{248} \rightarrow a^{250})$

$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^3 \rightarrow a^6 \rightarrow a^{12} \rightarrow a^{15} \rightarrow a^{30} \rightarrow a^{60} \rightarrow a^{62} \rightarrow a^{124} \rightarrow a^{248} \rightarrow a^{250}$

(c) $3-ary$ (8진법) : $011-111-010$

사전처리		$010 : a^1 \rightarrow a^2,\ 011 : a^2 \rightarrow a^3$ $100 : a^3 \rightarrow a^4,\ 101 : a^4 \rightarrow a^5$ $110 : a^5 \rightarrow a^6,\ 111 : a^6 \rightarrow a^7$	
구분		누승	곱셈 (비트 값)
$b_8b_7b_6$	011	a^3 /* 사전처리에서 구한 a^3 사용	
$b_5b_4b_3$	111	$(a^3)^8 = a^{3\times 8} = a^{24}$ $(a^3 \rightarrow a^6 \rightarrow a^{12} \rightarrow a^{24})$	$(a^{24})(a^7) = a^{24+7} = a^{31}$ $(a^{24} \rightarrow a^{31})$
$b_2b_1b_0$	010	$(a^{31})^8 = a^{31\times 8} = a^{248}$ $(a^{31} \rightarrow a^{62} \rightarrow a^{124} \rightarrow a^{248})$	$(a^{248})(a^2) = a^{248+2} = a^{250}$ $(a^{248} \rightarrow a^{250})$

$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^4 \rightarrow a^5 \rightarrow a^6 \rightarrow a^7$
$a^3 \rightarrow a^6 \rightarrow a^{12} \rightarrow a^{24} \rightarrow a^{31} \rightarrow a^{62} \rightarrow a^{124} \rightarrow a^{248} \rightarrow a^{250}$

표준 이진법은 $a^b \pmod n, b = b_k b_{k-1} b_{k-2} \cdots b_1 b_{0_{(2)}}$에 대해 $b_i = 1$이 평균 $\left\lceil \frac{k}{2} \right\rceil$개 존재한다고 가정하면, k회의 제곱과 $\left\lceil \frac{k}{2} \right\rceil$회의 곱셈을 수행하여 총 곱셈 수행 횟수는 $k + \left\lceil \frac{k}{2} \right\rceil$회이다. $d-ary$는 2^d-2회의 사전처리, $(\frac{k}{d}-1) \cdot d = k-d$회의 누승과 $(\frac{k}{d}-1) \cdot (\frac{2^d-1}{2^d})$회의 곱셈을 수행한다. 여기서 $(\frac{2^d-1}{2^d})$은 2^d개의 가능한 비트 값 경우수 중 0을 제외한 2^d-1개에 대해서만 곱셈이 수행되기 때문이다. 이러한 공식을 적용하면 이론적으로는 특정 k에 대해 최적인 d를 결정할 수 있으며, 이는 Henriquez[10]로부터 인용된 자료로 <표 2>에 제시하였다. 그러나 실제로는 $b_i = 1$의 개수에 따라 영향을 받는다. 결국, 모듈러 지수 연산법에는 2진법, 4진법, 8진법, 16진법 등 2^d-진법이 적용되고 있다.

〈표 2〉 k에 대해 최적인 $d-ary$

k	곱셈 수행횟수		d
	이진법	$d-ary$	
8	11	10	2
16	23	21	2
32	47	43	2, 3
64	95	85	3
128	191	167	3, 4
256	383	325	4
512	767	635	5
1024	1535	1246	5
2048	3071	2439	6

암호 생성과 해독에 적용되는 $d-ary$는 b의 이진 자리수에 영향을 받는다. $2 \le k \le 2048$에 대해 계산한 결과 최적의 k 범위는 <표 3>과 같이 얻었다. 따라서 $d-ary$를 적용할 경우에는 b 값의 2진 자리수 k에 따라 <표 3>을 적용하면 수행횟수를 감소시킬 수 있다.

〈표 3〉 k 범위에 대해 최적인 $d-ary$

k	$d-ary$
[2, 6]	$1-ary$
[7, 34]	$2-ary$
[35,121]	$3-ary$
[122,368]	$4-ary$
[369,1043]	$5-ary$
[1044,2048]	$6-ary$

3. 빠른 모듈러 지수연산법

$2 \le m \le 16$에 대해 a^m의 누승 (m승)을 얻는 곱셈 수행 횟수는 <표 4>에 제시되어 있다. 표에서 $d-ary$인 2, 4, 8과 16진법을 제외하고 $4-ary$인 16진법의 a^{16}을 얻는 누승 곱셈횟수 4회 이내인 3, 5, 6, 7, 10과 12 진법을 "m-진법"이라 하자.

〈표 4〉 m-진법의 사전처리와 a^m 누승 곱셈 수행횟수

진법	사전처리	a^m 누승 곱셈 횟수
2 ($1-ary$)	0	$a \rightarrow a^2$ (1회)
3	1	$a \rightarrow a^2 \rightarrow a^3$ (2회)
4 ($2-ary$)	2	$a \rightarrow a^2 \rightarrow a^4$ (2회)
5	3	$a \rightarrow a^2 \rightarrow a^4 \rightarrow a^5$ (3회)
6	4	$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^6$ (3회)
7	5	$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^6 \rightarrow a^7$ (4회)
8 ($3-ary$)	6	$a \rightarrow a^2 \rightarrow a^4 \rightarrow a^8$ (3회)
9	7	$a \rightarrow a^2 \rightarrow a^4 \rightarrow a^8 \rightarrow a^9$ (4회)
10	8	$a \rightarrow a^2 \rightarrow a^4 \rightarrow a^5 \rightarrow a^{10}$ (4회)
11	9	$a \rightarrow a^2 \rightarrow a^4 \rightarrow a^5 \rightarrow a^{10} \rightarrow a^{11}$ (5회)
12	10	$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^6 \rightarrow a^{12}$ (4회)
13	11	$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^6 \rightarrow a^{12} \rightarrow a^{13}$ (5회)
14	12	$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^6 \rightarrow a^7 \rightarrow a^{14}$ (5회)
15	13	$a \rightarrow a^2 \rightarrow a^3 \rightarrow a^6 \rightarrow a^7 \rightarrow a^{14} \rightarrow a^{15}$ (6회)
16 ($4-ary$)	14	$a \rightarrow a^2 \rightarrow a^4 \rightarrow a^8 \rightarrow a^{16}$ (4회)

표준 이진법의 곱셈 수행 횟수를 줄일 수 있는 방법은 다음과 같이 m-진법을 적용하고 특정 상황이 발생하면 알고리즘을 종료시키는 방법을 적용한다. 이를 "빠른 지수연산법"이라 하자.

(1) a^b에 대해 $b \equiv 0 \pmod m$인 m-진법을 적용하면 이진법보다 곱셈 수행 횟수를 줄일 수도 있다.

예를 들면 a^{15}의 경우, 이진법은 $15 = 1111_{(2)}$로 $a \rightarrow (a^2 \rightarrow a^3) \rightarrow (a^6 \rightarrow a^7) \rightarrow (a^{14} \rightarrow a^{15})$로 6회의 곱셈을 수행한다. 5진법은 $15 = 30_{(5)}$으로 $(a^2 \rightarrow a^4 \rightarrow a^5) \rightarrow (a^5)^2 = a^{10} \rightarrow (a^{10} \cdot a^5) = a^{15}$로 5회로 줄일 수 있다. 3진법은 $15 = 120_{(3)}$으로 $(a^2 \rightarrow a^3) \rightarrow (a^3)^2 = a^6 \rightarrow (a^6)^2 = a^{12} \rightarrow (a^{12} \cdot a^3) = a^{15}$으로 5

회를 수행한다. 7진법은 $15=21_{(7)}$으로 $(a^2\rightarrow a^3\rightarrow a^6\rightarrow a^7)\rightarrow(a^7)^2=a^{14}\rightarrow a^{14}\bullet a=a^{15}$으로 6회를 수행한다. 따라서 3진법이나 5진법을 적용하면 이진법보다 1회를 줄일 수 있다.

$b=2301=3\times767, 13\times177$인 경우, 이진법을 적용하면 $2301=100011111101_{(2)}$로 11회의 제곱과 7회의 곱셈으로 18회를 수행한다. 3진법을 적용하면 $2301=10011020_{(3)}$으로 사전에 a^2와 $a^2\rightarrow a^3$를 얻고 $(a^3)^3=((a^3)^2\times a^3)\rightarrow(a^9)^3a^1=((a^9)^2\times a^9\times a^1)\rightarrow(a^{28})^3a^1=((a^{28})^2\times a^{28}\times a^1)\rightarrow(a^{85})^3=((a^{85})^2\times a^{85})\rightarrow(a^{255})^3a^2=((a^{255})^2\times a^{255}\times a^2)\rightarrow(a^{763})^3=((a^{767})^2\times a^{767})=a^{2301}$로 17회 수행된다. 13진법을 적용하면 $2301=1080_{(13)}$으로 사전에 $a^2\rightarrow a^3\rightarrow a^6\rightarrow a^{12}\rightarrow a^{13}$ (5회)와 $a^6a^2=a^8$ (1회)를 얻고, a^{13}에서 시작하여 $b_3b_2=10$에서는 a^{13}으로 0회, $b_1=8:(a^{13})^{13}(a^8)=(a^{13})^2\rightarrow(a^{26})(a^{13})\rightarrow(a^{39})^2\rightarrow(a^{78})^2\rightarrow(a^{156})(a^{13})\rightarrow(a^{169})(a^8)=a^{177}$의 6회, $b_0=0$에서 $(a^{177})^{13}=(a^{177})^2\rightarrow(a^{354})(a^{177})\rightarrow(a^{531})^2\rightarrow(a^{1062})^2\rightarrow(a^{2124})(a^{177})=a^{2301}$에서 5회로 총 5+1+0+6+5=17회가 된다. 따라서 3진법과 13진법은 이진법에 비해 1회를 축소시킬 수 있다.

일반적으로 m-진법은 (그림 3)과 같이 수행된다.

```
a^b (mod n), b = (b_k, b_{k-1}, ..., b_1, b_0)_m
if    b ≡ 0 (mod m) then m-진법 적용
      (m = 3,5,6,7,10,12)
else d-ary법 적용.
Preprocessing (사전 처리)
      a^m 누승 계산: a→a^2→ ··· →a^m /* <표 4> 누승 참조
      b_{k-1}, ..., b_1, b_{0(m)}에서 누승계산에 없는 비트 값 곱셈
                  수행
c = a^m
Modular-Exponentiation (a, b, n)
      if    b_k ≠ 1 then c = c^{b_k} (mod n)
            /* c→c^2→ ··· →c^{b_k - 1} c 계산
      else if b_k = 1 then c = c_0.
      if    b_{k-1} ≠ 0 then c = (c × a^{b_{k-1}}) mod n
      for   i = k-2 down to 0 do
            c = c^m (mod n) /* c→c^2→ ··· →c^{m-1} c 계산
            if      b_i ≠ 0 then c = (c × a^{b_i}) mod n
      return c
```

(그림 3) m-진법

(2) $a^b(\mathrm{mod}\,n)$의 m-진법 곱셈 수행 과정에서 $a^x(\mathrm{mod}\,n)=1$을 얻는 경우 $b=qx+r$ 또는 $x\equiv r(\mathrm{mod}\,b)$를 계산하여 $a^r(\mathrm{mod}\,n)$을 결과값으로 얻으면 곱셈 수행 횟수를 줄일 수 있다.

예를 들면 $7^{2301}(\mathrm{mod}\,10)$의 경우 $7^4(\mathrm{mod}\,10)=1$이 된다. 이진법을 수행하면 $2301=100011111101_{(2)}$로 11회의 제곱과 $b_i=1$의 개수인 7회의 곱셈을 수행하여 총 18회를 수행한다. 그러나 $7^2\rightarrow7^4$의 2회 제곱을 수행하면 $7^4(\mathrm{mod}\,10)=1$이 계산되므로 $2301=4\times575+1$을 추가로 계산하여 비트 값 1에 대해 $7^1(\mathrm{mod}\,10)=1$을 적용하면 더 이상의 곱셈을 수행하지 않아도 되므로 3회로 축소시킬 수 있다.

(3) $a^x(\mathrm{mod}\,n)=a$가 계속 발생하는 경우 결과값은 a로 더 이상 곱셈을 수행하지 않아도 된다.

예를 들면, $85^{103}(\mathrm{mod}\,119)$, $103=1100111_{(2)}$을 이진법으로 수행하면 6회의 제곱과 4회의 곱셈으로 총 10회의 곱셈을 수행한다. 그러나 $b_5=1$에서의 제곱 $85^2(\mathrm{mod}\,119)=85$과 곱셈 $85^3(\mathrm{mod}\,119)=85$에서 계속하여 a값을 얻는다. 이는 더 이상의 곱셈을 수행하여도 동일한 값을 얻으므로 2회 수행으로 알고리즘을 종료시킬 수 있다.

4. 적용 및 결과 분석

4.1 $b\equiv0(\mathrm{mod}\,m)$인 경우 m-진법 적용

$b=250=11111010_{(2)}$에 대해 이진법은 12회, $2-ary$는 11회, $3-ary$는 14회의 곱셈을 수행하였다. 5진법을 적용하면 $250=2000_{(5)}$로 사전처리에서 $a\rightarrow a^2\rightarrow a^4\rightarrow a^5$의 3회 곱셈을 수행하고, 초기치 a^5에 대해 $b_3b_2=20$에서 $(a^5)^2=a^{10}$의 1회 제곱, $b_1=0$에서 $(a^{10})^5=(a^{10})^2\rightarrow(a^{20})^2\rightarrow(a^{40})a^{10}=a^{50}$의 3회 곱셈 (제곱 2회와 곱셈 1회)을, $b_0=0$에서 $(a^{50})^5=(a^{50})^2\rightarrow(a^{100})^2\rightarrow(a^{200})a^{50}=a^{250}$의 3회를 수행하면 해를 얻는다. 따라서 곱셈 횟수는 3+1+3+3=10회를 수행한다. 결국, $d-ary$에서 가장 좋은 결과를 얻은 $2-ary$의 11회보다도 좋은 결과를 얻는다.

$7^{560}(\mathrm{mod}\,561)$의 7^{560}에 대해 560은 2, 4, 5, 7, 8, 10, 14와 16으로 나누어진다. $560=1000110000_{(2)}$으로 이진법 11회, $2-ary$는 11회, $3-ary$는 16회, $4-ary$는 23회의 곱셈을 수행한다. 5진법은 $560=4220_{(5)}$로 사전에 $a^2\rightarrow a^4\rightarrow a^5$의 3회 곱셈을 수행하고, a^5에서 시작하여 $[(a^5)^4a^2=(a^5)^2\rightarrow(a^{10})^2\rightarrow(a^{20})a^2=a^{22}]\rightarrow[(a^{22})^5a^2=(a^{22})^2\rightarrow(a^{44})^2\rightarrow(a^{88})a^{22}\rightarrow a^{110}a^2=a^{112}]\rightarrow[(a^{112})^5=(a^{112})^2\rightarrow(a^{224})^2\rightarrow(a^{448})a^{112}=a^{560}]$이 되어 13회를 수행한다. 이 경우는 $b_i=1$의 개수가 9개중 2개로 매우 희박한 경우로 이진법이나 $2-ary$가 가장 효과적이다.

4.2 $d-ary$ 적용시 $a^x(\mathrm{mod}\,n)\equiv1$이 발생하는 경우

$23^{391}(\mathrm{mod}\,55)$, $b=391=110000111_{(2)}$에 대해 알고리즘을 수행한 결과는 (그림 4)에 제시되어 있다.

표준 이진법은 8회의 제곱과 4회의 곱셈으로 12회를 수행하는데 반해 제안된 알고리즘은 3회의 제곱과 1회 곱셈에서 1을 얻었다. 이때 23^{12} 또는 4회 수행으로 23^4과 같다. $23^{391}=23^{12\times32+7}$로 $23^7(mod\,55)=12$를 추가로 계산해야

초기치: $d=23$
1. $b_7=1,\ d=(23\times23)\pmod{55}=34\ :\ 23^2$
2. $d=(34\times23)\pmod{55}=12\ :\ 23^3$
3. $b_6=0,\ d=(12\times12)\pmod{55}=34\ :\ 23^6$
4. $b_5=0,\ d=(34\times34)\pmod{55}=1\ :\ 23^{12}$
5. $b_4=0,\ d=(1\times1)\pmod{55}=1$
6. $b_3=0,\ d=(1\times1)\pmod{55}=1$
7. $b_2=1,\ d=(1\times1)\pmod{55}=1$
8. $d=(1\times23)\pmod{55}=23$
9. $b_1=1,\ d=(23\times23)\pmod{55}=34$
10. $d=(34\times23)\pmod{55}=12$
11. $b_0=1,\ d=(12\times12)\pmod{55}=34$
12. $d=(34\times23)\pmod{55}=12$

(a) 이진법

초기치: $d=23$
1. $b_7=1,\ d=(23\times23)\pmod{55}=34\ :\ 23^2$
2. $d=(34\times23)\pmod{55}=12\ :\ 23^3$
3. $b_6=0,\ d=(12\times12)\pmod{55}=34\ :\ 23^6$
4. $b_5=0,\ d=(34\times34)\pmod{55}=1\ :\ 23^{12}$
5. $23^{391}=23^{12\times32+7},\ 23^7\pmod{55}=12$
$23^7=23^6\times23$
or 4번째 수행, $23^{391}=23^{4\times97+3}$
$23^3\pmod{55}=12$

(b) 빠른 지수연산법

(그림 4) $23^{391}\pmod{55}$ 비교

하나 $b_6=0$에서 23^6을 얻었으므로 추가로 $23^7=23^6\times23$의 1회 곱셈을 수행한다. 만약, 23^4을 적용하면 $23^{391}=23^{4\times97+3}$으로 $23^3\pmod{55}=12$을 계산한다. 23^3은 $b_7=1$에서 얻었으므로 추가적인 곱셈을 수행하지 않아도 된다. 따라서 빠른 이진법은 5회 또는 4회로 단축시킬 수 있다.

4.3 $d-ary$ 적용시 $a^x\pmod{n}\equiv a$가 발생하는 경우

$85^{103}\pmod{119}$, $103=1100111_{(2)}$에 대해 알고리즘을 적용한 결과는 (그림 5)에 제시하였다. 표준화된 이진법은 6회의 제곱과 4회의 곱셈 수행으로 총 10회를 수행하였다. 반면에 빠른 이진법은 첫 번째 제곱에서 85를 얻어 단지 1회로 단축시킬 수 있었다.

초기치: $d=85$
1. $b_5=1,\ d=(85\times85)\pmod{119}=85$
2. $d=(85\times85)\pmod{119}=85$
3. $b_4=0,\ d=(85\times85)\pmod{119}=85$
4. $b_3=0,\ d=(85\times85)\pmod{119}=85$
5. $b_2=1,\ d=(85\times85)\pmod{119}=85$
6. $d=(85\times85)\pmod{119}=85$
7. $b_1=1,\ d=(85\times85)\pmod{119}=85$
8. $d=(85\times85)\pmod{119}=85$
9. $b_0=1,\ d=(85\times85)\pmod{119}=85$
10. $d=(85\times85)\pmod{119}=85$

(a) 이진법

초기치: $c_0=85$
1. $b_5=1,\ d=(85\times85)\pmod{119}=85$

(b) 빠른 지수연산법

(그림 5) $85^{103}\pmod{119}$ 비교

4.4 $d-ary$ 적용시 $a^x\pmod{n}\equiv a$ or 1 미발생인 경우

$b=3243679$인 경우, $b\equiv0\pmod{m}$이 존재하지 않아 $d-ary$를 적용하는 것이 가장 효율적이며 최적의 $d-ary$는 <표 3>에 따라 $2-ary$이다. 이진법은 $3243679=110001011\ 1111010011111_{(2)}$로 21+14=35회의 곱셈을 수행한다. $2-ary$는 사전처리 2회, 누승 20회와 곱셈 9회로 31회를 수행한다. 3진법은 $3243679=20002210111021_{(3)}$으로 2+1+24+8=33회를 수행한다. $3-ary$는 사전처리 5회, 누승 21회, 곱셈 7회로 5+21+ 7=33회를 수행한다.

$2^{18206926}\pmod{18206927}$, $18206926=100101011101000\ 0110011_{(2)}$에 대해 이진법은 21회의 제곱과 10회의 곱셈으로 31회를 수행한다. $2-ary$는 2회의 사전처리, $10\times2=20$회의 누승과 7회의 곱셈으로 29회를, $3-ary$는 6회의 사전처리, $7\times3=21$회의 누승과 6회의 곱셈으로 33회를 수행한다. 이는 <표 3>에서 $k=[7,34]$는 이론상으로는 $2-ary$를 적용하는 것이 가장 효율적인 결과와 일치한다. 18206926은 $2\le m\le16$중에서 2이외에는 나눌 수 없다. 따라서 $d-ary$이외에는 적용이 불가한 것으로 판단된다. 그러나 임의로 5진법을 적용해 보자. $18206926=14130110201_{(5)}$에 대해 사전처리는 $a\to a^2\to a^4\to a^5, a^2\to a^3$로 4회, $b_{10}b_9=14$에서 $b_{10}=1$로 처리하지 않으면 $(a^5)a^4=a^9$으로 1회, 나머지 b_8부터 b_0까지 9개 각각에 대한 누승에 3회로 27회, $b_i\neq0$인 6개에 대한 6회 곱셈을 수행하여 4+1+27+6=38회를 수행하여 $d-ary$보다 효율적이지 못하다. $2^{18206926}\pmod{18206927}$은 (그림 6)에서 알 수 있듯이 이진법을 수행하는 과정에서 알고리즘을 종료시킬 수 있는 경우가 발생하지 않아 더 이상의 곱셈 수행횟수를 단축시킬 수 없다.

$2^{18206926}\pmod{18206927}$,
$18206926=10010101110100001100 11_{(2)}$
초기치: $d=2$
1. $b_{20}=0,\ d=(2\times2)\pmod{n}=4$
2. $b_{19}=0,\ d=(4\times4)\pmod{n}=16$
3. $b_{18}=1,\ d=(16\times16)\pmod{n}=256$
4. $d=(256\times2)\pmod{n}=512$
5. $b_{17}=0,\ d=(512\times512)\pmod{n}=262144$
6. $b_{16}=1,\ d=(262144\times262144)\pmod{n}=6534238$
7. $d=(6534238\times2)\pmod{n}=13068476$
8. $b_{15}=0,\ d=13068476^2\pmod{n}=11350928$
9. $b_{14}=1,\ d=11350928^2\pmod{n}=8093663$
10. $d=(8093663\times2)\pmod{n}=16187326$
11. $b_{13}=1,\ d=16187326^2\pmod{n}=17791880$
12. $d=(17791880\times2)\pmod{n}=17376833$
13. $b_{12}=1,\ d=17376833^2\pmod{n}=14896521$
14. $d=(14896521\times2)\pmod{n}=11586115$
15. $b_{11}=0,\ d=11586115^2\pmod{n}=8438728$
16. $b_{10}=1,\ d=8438728^2\pmod{n}=13925329$
17. $d=(13925329\times2)\pmod{n}=9643731$
18. $b_9=0,\ d=9643731^2\pmod{n}=69624$
19. $b_8=0,\ d=69624^2\pmod{n}=4458794$
20. $b_7=0,\ d=4458794^2\pmod{n}=8479910$

21. $b_6 = 0,\ d = 8479910^2 \pmod n = 14593009$
22. $b_5 = 1,\ d = 14593009^2 \pmod n = 10158887$
23. $\quad d = (10158887 \times 2) \pmod n = 2110847$
24. $b_4 = 1,\ d = 2110847^2 \pmod n = 3054261$
25. $\quad d = (3054261 \times 2) \pmod n = 6108522$
26. $b_3 = 0,\ d = 6108522^2 \pmod n = 139750$
27. $b_2 = 0,\ d = 139750^2 \pmod n = 12236756$
28. $b_1 = 1,\ d = 12236756^2 \pmod n = 5472275$
29. $\quad d = (5472275 \times 2) \pmod n = 10944550$
30. $b_0 = 1,\ d = 10944550^2 \pmod n = 2245697$
31. $\quad d = (2245697 \times 2) \pmod n = 4491394$

(그림 6) $2^{18206926}$의 이진법 곱셈 수행횟수

결론적으로, $b \equiv 0 \pmod m, 2 \le m \le 16$을 계산하여 $d-ary$ 이외의 3,5,6,7,9,10,11,12,13,14,15중에서 누승 곱셈횟수가 4까지인 3,5,6,7,9,10,12의 m-진법을 적용하면 이진법의 곱셈 수행횟수를 줄일 수 있다. 그렇지 않은 경우 <표 3>의 k의 범위 기준에 근거하여 최적의 $d-ary$를 적용한다. 만약 알고리즘 수행 과정에서 추가로 2가지 조건이 발생하면 알고리즘을 종료시켜 곱셈 수행횟수를 단축시킬 수 있을 것이다.

5. 결 론

본 논문은 암호학의 암호 생성과 해독, 소수 판별법에 일반적으로 적용되는 $d-ary$법의 곱셈 횟수를 줄일 수 있는 방법을 제안하였다.

첫 번째로, 표준 이진법 $(1-ary)$의 곱셈 횟수를 줄일 수 있는 효율적으로 $d-ary$법을 제안하였다. 이는 a^b에서 b의 이진 자리수 k에 영향을 받음을 알 수 있었으며, 자리수 범위별로 이론상으로 최적의 $d-ary$법을 도출하였다.

두 번째로, $b \equiv 0 \pmod m$, $(m = 3,5,6,7,10,12)$인 경우, $d-ary$에 비해 m-진법을 적용하면 $d-ary$의 곱셈 횟수를 축소시킬 수 있음을 보였다.

세 번째로, $d-ary$를 적용하는 과정에서 $a^x \pmod n \equiv 1$ 또는 $a^x \pmod n \equiv a$가 발생하면 곱셈 횟수를 보다 축소 시킬 수 있음을 보였다.

제안된 알고리즘은 기존의 2^d-진법의 $d-ary$법에서 $2 \le m \le 16$의 m-진법을 적용할 수 있도록 일반화된 알고리즘으로 확장하였다. 또한, $d-ary$법 적용과정에서 알고리즘을 보다 빨리 종료시킬 수 있는 방법도 제시하였다.

참 고 문 헌

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," 2nd Edition, McGrew-Hill Book Company, 2005.

[2] M. Alfred, P. C. Oorschot, AND S. A. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1996.

[3] S. T. Klein, "Should One Always Use Repeated Squaring for Modular Exponentiation?." Information Processing Letters, Vol. 106, Issue. 6, pp. 232-237, 2008.

[4] D. M. Gordon, "A Survey of Fast Exponentiation Methods," Journal of Algorithms, Vol. 27, No. 1, pp. 129-146, 1998.

[5] P. Montgomery, "Modular Multiplication Without Trial Division," Math. Computation, Vol. 44, pp. 519 - 521, 1985.

[6] G. Saldamli and C. K. Koc, "Spectral Modular Exponentiation," Proc. of the 18th IEEE Symposium on Computer Arithmetic, pp. 123-132, 2007.

[7] V. Gopal, J. Guilford, E. Ozturk, W. Feghali, G. Wolrich, and M. Dixon, "Fast and Constant-Time Implementation of Modular Exponentiation," 28th International Symposium on Reliable Distributed Systems, Niagara Falls, New York, U.S.A., http://www.cse.buffalo.edu/srds2009/escs2009_submission_Gopal.pdf, 2009.

[8] L. Zhong, "Modular Exponentiation Algorithm Analysis for Energy Consumption and Performance," Technical Report CE-01-ZJL, Dept. of Electrical Engineering, Princeton University, 2001.

[9] N. Nedjah and L. M. Mourelle, "Efficient Pre-Processing for Large Window-Based Modular Exponentiation Using Ant Colony," Informatica, Vol. 29, pp. 151-161, 2005.

[10] F. R. Henriquez, "Modular Exponentiation," Arithmética Computacional, http://delta.cs.cinvestav.mx/~francisco/arith/ expo.pdf.

이 상 운

e-mail : sulee@gwnu.ac.kr

1983년~1987년 한국항공대학교 항공전자공학과(학사)

1995년~1997년 경상대학교 컴퓨터과학과(석사)

1998년~2001년 경상대학교 컴퓨터과학과(박사)

2003년 강원도립대학 컴퓨터응용과 전임강사

2004년~2007년 2월 국립 원주대학 여성교양과 조교수

2007년 3월~현 재 강릉원주대학교 멀티미디어공학과 부교수

관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도 (소프트웨어 규모, 개발노력, 개발기간, 팀 규모), 분석과 설계 방법론, 소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘