

---

# RSA 암호시스템의 모듈러 승산기 처리속도 향상을 위한 연구

이선근\* · 정우열\*\*

A Study on the Modulus Multiplier Speed-up Throughput in the RSA Cryptosystem

Seon-keun Lee\* · Woo-Yeol Jeung\*\*

## 요 약

최근, 다양한 네트워크의 발달은 심각한 사회문제를 발생시킨다. 그러므로 네트워크의 보안성을 통제할 수 있는 방법이 요구되어진다. 보안과 관련된 이러한 문제들은 해킹, 크래킹과 같은 비 보안 분야에 직면해 있다. 새로운 암호알고리즘의 개발없이 해커나 크래커로부터 안전성을 보장받기 위한 방법은 확장된 키 길이를 통한 비 암호해석법을 유지시키는 것이다.

본 논문에서는 RSA 암호시스템에서 병목현상을 제거하기 위해서 가변길이 곱셈, 캐리 생성 부분을 하나의 어레이 방식을 사용하는 몽고메리 곱셈기 구조를 제안하였다. 그러므로 제안된 몽고메리 곱셈기는 크래킹으로부터 안전성을 제공하게 되며 실시간 처리가 가능해질 것이다.

## ABSTRACT

Recently, the development of the various network method can generate serious social problems. So, it is highly required to control security of network. These problems related security will be developed and keep up to confront with anti-security field such as hacking, cracking. The way to preserve security from hacker or cracker without developing new cryptographic algorithm is keeping the state of anti-cryptanalysis in a prescribed time by means of extending key-length.

In this paper, the proposed montgomery multiplication structured unit array method in carry generated part and variable length multiplication for eliminating bottle neck effect with the RSA cryptosystem. Therefore, this proposed montgomery multiplier enforce the real time processing and prevent outer cracking.

## 키워드

Asymmetric cryptographic, RSA Cryptosystem, Montgomery multiplication, Lookup-table

## 1. 서 론

일반적으로 공개키 암호 알고리즘은 수학적인 해를

구하기 어려운 문제를 대상으로 암호화를 전개하고 있기 때문에 해를 구하는 것이 바로 암호분야의 핵심이 된다. 대표적인 비대칭 암호시스템 중의 하나인 RSA

---

\* (주)울시스템즈  
접수일자 : 2009. 07. 27

\* 한려대학교 멀티미디어정보통신공학과  
심사완료일자 : 2009. 09. 07

암호시스템은 매우 큰 정수에 대한 소인수분해가 어렵고 해를 찾아내는데 많은 계산을 요구한다는 점을 이용한 암호시스템이다[1][2].

비대칭 암호시스템에서 사용되어지는 키는 네트워크 환경에 노출시키는 공개키와 사용자가 독립적으로 가지는 비공개 비밀키의 두 종류가 사용되어지며 이러한 두 종류의 키를 동시에 사용함으로써 암호화를 수행하게 된다.

비대칭 암호화에 대한 수식을 키와 수학적인 과정을 거치게 되면 원 신호를 얻을 수 있으며 키의 공개와 비밀성 유지와 같은 키 메카니즘에 의하여 사용자의 수가 매우 크다. 그렇기 때문에 RSA와 같은 비대칭형 암호 방식은 키의 분배와 관리면에서 매우 효율적이며 인증(authentication)과 전자서명(digital signature)이 가능한 장점을 가지고 있으나 154자리(512 비트) 이상의 큰 수에 대한 모듈러 지수연산 처리시간이 너무 오래 걸린다는 단점을 가지고 있다. 즉, 비대칭형 암호방식은 수학적인 일반해에 대하여 키에 관련된 특수해를 찾기 때문에 특정 환경이 형성되지 않으면 그 해를 구할 수가 없게 되고 특수환경이 조성되어도 처리시간이 매우 길어지는 단점을 가진다[3][4].

## II. 몽고메리 승산 알고리즘

비대칭형 암호방식은 네트워크 환경속에서 키분배 및 관리측면에서 매우 탁월한 성능을 발휘하기 때문에 네트워크 환경에 적합한 암호방식이다. 그러나 비대칭형 암호시스템은 수학적인 해를 구하는 방법을 이용하여 암호화를 수행하기 때문에 처리시간이 매우 길다는 단점을 가진다. 비대칭형 암호시스템의 대표적인 방식인 RSA 역시 수학적인 해를 찾기 위해서는 매우 오랜 시간이 소요된다. 그러므로 RSA 암호시스템에서 요구되는 모듈러 지수연산부의 고속화를 위하여 모듈러 승산 횟수를 줄이는 방법과 승산부분을 고속화하는 방법이 주로 진행되어 왔다. RSA 암호시스템에서 사용되어지는 모듈러 연산은 크게 두 부분으로 나뉘어진다. 즉, 모듈러 승산(modular multiplication) 연산부와 모듈러 리덕션(modular reduction) 연산부로 구성되어 있으며 이들 중에서 모듈러 승산 연산부분이 전체 모듈러 연산 처리시간의 거의 대부분을 차지하고 있는 것이 현실적

이다. 모듈러 지수 연산을 수행하기 위하여 개발된 방법은 이진방식,  $\gamma$ -진방식, 누승테이블방식, 가산고리방식 등이 있으나 이중에서 이진방식을 적용한 몽고메리 모듈러 지수연산(montgomery modular exponent) 방식이 개발된 방식 중에서 처리속도에서 가장 빠른 것으로 보고되고 있다.

다음 식(1)은 몽고메리 지수(Montgomery Exponent) 부분을 계산하기 위한 과정을 나타내는 순서이다.

$$\begin{aligned}
 1me : M_{init}' &\leftarrow MP(1, M, N, R') \\
 &\quad (MP(\cdot)) : \text{Montgomery product} \\
 2me : C_{init}' &\leftarrow MP(1, 1, N, R') \\
 3me : \text{for } i &\leftarrow 0 \text{ to } n-1 \\
 &\quad \text{if } e_i = 1 \text{ then} \\
 &\quad \quad C_i' &\leftarrow MP(C_{i-1}', M_{i-1}', N, R') \\
 &\quad \text{if } i \leq n-2 \text{ then} \\
 &\quad \quad M_i' &\leftarrow MP(M_{i-1}', M_{i-1}', N, R') \\
 4me : C' &\leftarrow MP(1, C_i', N, R') \\
 5me : C &\leftarrow C' \bmod N \\
 6me : \text{return } C &\quad (C = M^e \bmod N)
 \end{aligned} \tag{1}$$

여기에서 입력은 M, e, N, R( $R' = R^e \bmod N$ )이며 출력은 암호문 C이다.

1me부터 6me 까지는 몽고메리 지수연산 알고리즘을 순서대로 표현한 것으로서 R'는 전처리 과정을 수행하고 모듈러 승산부분을 몽고메리 모듈러 승산으로 대체하면 알고리즘 자체가 간결하게 되는데 이런 이유로 계산속도가 다소 향상된다. 그러나 모듈러 승산의 횟수는 지수부 이진수의 '1'의 개수와 같으므로 연산횟수를 줄이기 위하여 최근에는 '1'의 개수를 줄이는 형태로서 연산속도의 고속화가 진행되고 있다. 또한 몽고메리 승산(Montgomery Multiplication)인 경우에는 다음 식(2)와 같은 과정을 거치게 된다.

$$\begin{aligned}
 1mm : n_0' &\leftarrow n_{\text{setting value}}' \bmod r \\
 &\quad (n_{-0} = n_{\text{setting value}}) \\
 2mm : P_{init} &\leftarrow -0 \\
 3mm : \text{for } i &\leftarrow 0 \text{ to } n-1 \\
 &\quad P_i \leftarrow P_{i-1} + a_i \times B \times r^i
 \end{aligned}$$

$$m_i \leftarrow p_i \times n_0' \bmod r \quad (2)$$

$$P_i \leftarrow P_i + m_i \times N \times r^i$$

4mm :  $P \leftarrow P_i \div R$

5mm : return P

여기에서 입력은 A, B, N, R이며 출력은  $P = A \times B \times R^{-1} \bmod N$ 이 된다. 1mm에서 5mm까지의 과정이서 사용되는 파라미터 A, B, N, R, P는 정수이며  $a_i, p_i$ 는 정수 A와 P의  $i$ 번째 자리에 해당하는 자릿수를 의미한다. 모듈러 배수  $m_i$ 는 디지털 값  $p_i$ 에 의해 계산되고 이에 대한 결과를 누적된  $P_i$ 의 계산에 이용하여  $m_i$ 를 계산하기 위한 다음 단계의 덧셈을 수행한다.

몽고메리 알고리즘에서 모듈러 배수는 부분결과의 하위 자리수에 의존하는 형태이므로 모듈러의 배수와 캐리의 전달 진행방향은 동일하다. 이러한 이유로 모듈러 승산에 대한 어레이 방식이 가능하며 병렬처리가 가능하기 때문에 RSA 암호시스템에서 몽고메리 알고리즘을 사용하여 고속의 암호화를 수행할 수 있게 된다. 몽고메리 알고리즘을 사용하여 모듈러 연산을 수행하기 위한 고속구현에는 여러 가지 방법들이 모색되어 있다. 승산의 고속구현을 위하여 DSP 및 시스템 어레이 구조 등의 방법을 사용하고 있지만 보호해야 할 데이터량의 증가와 일정하지 않은 길이를 가진 데이터들에 대한 연속적인 암호화 수행은 잦은 버퍼의 포화상태 유발 및 병목현상으로 인하여 모듈러 연산의 고속수행을 저하시키는 요인으로 작용하게 된다.[5][6][7]

그러므로 몽고메리 알고리즘을 적용한 RSA 암호시스템에서 기존에 사용된 모듈러 연산의 처리시간 지연 및 고속처리 저해요인인 버퍼포화 및 병목현상, 가변길이 데이터와 대용량의 데이터 그리고 계산상의 지연 등을 없애기 위하여 본 논문에서는 병렬처리의 방해요인의 제거를 위한 룩업 테이블 사용 및 모듈러 배수 연산에 대한 캐리 생성 부분만을 어레이 방식의 구조로 채택하는 것과 더불어 MM 방식을 이용한 연산 수행 단계를 줄여 모듈러 승산 연산이 고속으로 수행될 수 있도록 하였다.

### III. 매트릭스 승산 알고리즘

식(2)에서 1mm부터 5mm까지의 연산과정 중에서 반복 수행되는 부분은 3mm인 for 문에 해당하는 식(3)과 같다.

for  $i \leftarrow 0$  to  $n-1$  (3)

$$P_i \leftarrow P_{i-1} + a_i \times B \times r^i$$

$$m_i \leftarrow p_i \times n_0' \bmod r$$

$$P_i \leftarrow P_i + m_i \times N \times r^i$$

식(3)은 몽고메리 승산연산의 세 번째 과정에 해당하는 부분으로서 연산수행 과정 중 반복되는 부분이기 때문에 이 부분의 연산속도가 늦어지면 병목현상과 버퍼포화 현상이 발생하여 전체 시스템의 성능이 크게 저하된다. 그러므로 식(3) 부분을 병렬처리가 가능하도록 변형하면 처리속도의 향상 및 시스템 효율을 증대시킬 수 있다. 그러므로 반복수행 연산과정을 병렬처리가 가능하도록 하기 위하여 변형하면 식(4)와 같이 표현된다.

for  $i \leftarrow 0$  to  $n-1$  (4)

$$m_i \leftarrow ((P_{i-1} \bmod r) + a_i \times b_0) \times n_0' \bmod r$$

$$P_i \leftarrow (P_{i-1} + a_i \times B + m_i \times N) \text{ div } r$$

여기에서  $P_{i-1}$ 은 for 루프문장의  $i$ 번째 계산을 수행하기 이전 P에 대한 부분 결과값이다. 몽고메리 승산 정의에 따라서 식(5)가 성립한다.

$$r^i P_i = a_i B + m_i N \quad (5)$$

그러므로 부분결과값 P의 최종값  $P_n$ 은 식(6)과 같이 된다. 이때 식(5)와 식(6)을 이용하면 식(7)과 같은 결과를 유추할 수 있다.

$$r^n P_i = AB + MN \quad (6)$$

$$P \equiv ABR^{-1} \bmod N \quad (7)$$

이때 한자리 정수에 대한 승산을 수행함에 있어서, 식 (8)과 같이 한자리 정수에 대한 값은 부분자리수와 같다고 할 수 있다.

$$M(a_i B) \equiv a_i B \quad (8)$$

그러므로 한자리 정수는 식(8)과 같이 정의할 수 있으므로 부분 결과값  $P$ 는 식(9)와 같이 정의될 수 있다.

$$\begin{aligned} r^j P_i &= M(a_i B) + M(m_i N) \\ r^n P_n &= M(AB) + M(MN) \\ \therefore P &\equiv M(ABR^{-1} \bmod N) \end{aligned} \quad (9)$$

반복 부분인 식(3)에 대한 식(4)의 변형된 표현은  $R \equiv r^n$ 을 이용하여  $P$ 를 구할 필요 없이  $r$ -진수로 표현된 디지털 레벨(digit level)로 모든 계산이 처리되도록 각각의 부분 결과값에서 나눗셈 연산을 수행하므로 각 연산 시점에서의 의존 관계값의 수가 감소되어 병렬 처리가 용이하게 된다. 식(4)에 대한 식(9)의 표현은 한자리 정수에 대한 병렬처리와 룩업 테이블을 이용한 매트릭스 함수를 이용하여 변형한 것이다. 식(9)과 같이 변형한 이유는 병렬처리가 가능하도록 변형하였다 하더라도 식(4)의 각각의 연산부분이 해결되기 전까지는 다음 단계로 넘어갈 수 없는 것이 기존 몽고메리 모듈러 승산 알고리즘이다. 그러므로 식(9)과 같이 매트릭스 함수를 포함하게 되면 동시 다발적으로 연산이 수행되어 시스템 처리시간의 지연이 없어지게 된다. 이러한 이유로 식(9)와 매트릭스 함수  $MM(\cdot)$ 를 이용하여 식(4)를 다시 변형하면 식(10)과 같이 정리할 수 있다.

$$\begin{aligned} \text{for } i \leftarrow 0 \text{ to } n-1 \quad (10) \\ m_i &\leftarrow ((P_{i-1} \bmod r) + M(a_i \times b_0)) \times n_0' \bmod r \\ P_i &\leftarrow (P_{i-1} + M(a_i \times B) + M(m_i \times N)) \text{ div } r \end{aligned}$$

그러므로 식(10)은 본 논문에서 제안하는 몽고메리 모듈러 승산을 위한 몽고메리 매트릭스 승산(Matrix Multiplication : MM) 알고리즘에 대한 정의식이 되며 MM은 몽고메리 모듈러 승산의 반복연산에 해당하는 부분이 된다.

그림 1에서 각 부분 승산 값은 한 단계씩 아래로 이동함과 동시에 왼쪽 쉬프트(left shifting)를 수행하여 가산연산을 수행하게 된다. 즉, 이차원  $(i, j)$  인덱스 공간에서  $a$ 와  $m$ 은  $(0, -1)$ 방향으로,  $b$ 와  $N$ 은  $(-1, 0)$  방향으로,  $P$ 는  $(-1, -1)$ 방향으로 값이 이동함을 의미한다.

그러므로 각 몽고메리 부분 결과값을 생성하는 것은 이전단계에서 계산된  $P$ 의 부분 결과값  $P_{i-1, j-1}$ 이 된다.

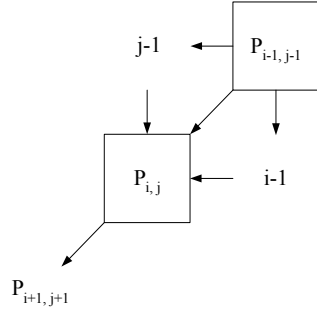


그림 1. 2차원 배열에 대한 승수 연산  
Fig. 1 Multiplication in two-dimension array

이때 모듈러의 배수  $m_i$ 는 이전 단계의 부분 결과값  $P_{i-1, j-1}$ 과  $a_i \times b_{i-1,0}$ 을 합한 결과의 최하위 비트에 의존하기 때문에  $j=1$ 인 인덱스에서만 계산되어진다. 또한  $m_i$ 의 계산을 위해 요구되는  $n_0'$ 는  $r=2$ 와 서로 소인 조건에서 항상 1이 되므로  $m_i$ 의 계산에서 생략 가능하게 된다.

몽고메리 부분 결과값  $P_{i,j}$ 를 계산하는 과정에서 발생하는 캐리는 별도의 계산 수행없이 출력 잉여분의 MSB에 추가시켜 출력한다[8][9].

표 1은 식(6)을 수행하기 위해서 설정해 놓은 2진 비트 레벨에 대한 룩업 테이블이다. 각 정수들에 대하여 비트 레벨로 미리 설정해 놓는 이유는 입력이 들어가면 즉시 승산을 수행할 수 있도록 하기 위한 부분이다.

표 1. MM(·) 함수에 대한 룩업 테이블  
Table 1. Lookup table for MM(·) Func.

| x | 0      | 1      | 2      | 3      | 4      | 5      | 6      | 7      | 8      | 9      |
|---|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 |
| 1 | 000000 | 100001 | 100010 | 000011 | 100010 | 000010 | 000010 | 100011 | 100000 | 000010 |
| 2 | 000000 | 100010 | 100010 | 000010 | 100010 | 000010 | 000010 | 100011 | 100000 | 000010 |
| 3 | 000000 | 000011 | 000010 | 000010 | 000010 | 000011 | 000010 | 100010 | 001000 | 001011 |
| 4 | 000000 | 100010 | 100010 | 000010 | 100000 | 000010 | 001000 | 100110 | 100000 | 000010 |
| 5 | 000000 | 000010 | 000010 | 000011 | 000010 | 100100 | 000010 | 100001 | 000000 | 000110 |
| 6 | 000000 | 000010 | 000010 | 000010 | 000010 | 000010 | 000010 | 100010 | 001000 | 001010 |
| 7 | 000000 | 100011 | 100010 | 100010 | 100010 | 100001 | 100010 | 101000 | 101100 | 001111 |
| 8 | 000000 | 100010 | 100000 | 000100 | 101000 | 000100 | 001000 | 101100 | 100000 | 000100 |
| 9 | 000000 | 000010 | 000010 | 000101 | 000010 | 000101 | 001010 | 001111 | 000000 | 100001 |

그림 2는 한자리 정수에 대한 몽고메리 매트릭스 승산 블록이다. 1회의 승산에 필요한 연산횟수는 단지 1회에 국한되므로 이에 해당하는 비트연산만을 수행할 수 있도록 설정된 블록이다. 또한  $P_{i,j}$ 에는 내부연산에서 발생된 캐리를 포함하게 되는데 이때 발생된 캐리는 여러개로 구성된 몽고메리 매트릭스 블록의 2진 비트 레벨부분으로 피드백 되어 다시 연산을 수행하게 된다.

기존 몽고메리 승산 알고리즘은 초기값과 계산과정에서 산출되어지는 중간단계의 값을 저장하기 위한 매체인 버퍼가 별도로 필요하였으나, 본 논문에서 제안한 몽고메리 매트릭스 승산 알고리즘(MM)은 별도의 저장 매체가 불필요하며 단지 2진 비트 레벨들에 대한 1회의 승산연산을 수행하기 위한 룩업 테이블 만이 필요하게 된다.

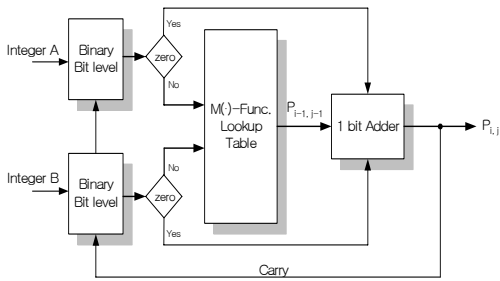


그림 2 한자리 정수에 대한 몽고메리 매트릭스 승산 블록  
Fig. 2 Montgomery matrix multiplication for single integer

따라서 MM 방식은 기존 몽고메리 승산 알고리즘의 반복계산이 필요했던 식(3)에 대하여 식(10)를 정의하였고 정의된 식(10)에 대하여 전체적인 MM 방식은 그림 2와 식(11)로서 정의될 수 있다.

식(11)에서 MM 방식의 연산과정은 매우 단순하다. 즉, 입력 정수중의 어느 하나가 0 또는 1인 경우에는  $P_{i,j}$ 값은 룩업테이블을 거치지 않고 즉시 출력하도록 하였다. 또한 병렬처리가 가능하도록 하기 위하여 시스템 구성은 정수값 가중치에 대한 어레이 구조를 채택하였다.

```

for all  $0 \leq i \leq n-1, 0 \leq j \leq n+1$ 
  if A or B = 0 then
    if i = 0 then
       $P_{i,j} \leftarrow 0$ 
    else
       $b_{i,j} \leftarrow b_{i-1,j}$ 
       $n_{i,j} \leftarrow n_{i-1,j}$ 
    end if
  if j = n+1 then
     $P_{i,j} \leftarrow 0$ 
  else
     $P_{i,j} \leftarrow P_{i-1,j+1}$ 
  end if
  elif A or B = 1 then
    if i = 0 then
       $P_{i,j} \leftarrow A$  or  $B$ 
    else
       $b_{i,j} \leftarrow b_{i-1,j-1}$ 
       $n_{i,j} \leftarrow n_{i-1,j-1}$ 
    end if
  if j = n+1 then
     $P_{i,j} \leftarrow A$  or  $B$ 
  else
     $P_{i,j} \leftarrow P_{i-1,j+1}$ 
  end if
  elif A or B  $\neq$  0 or 1 then
    if j = 0 then
       $a_{i,j} \leftarrow a_i$ 
       $carry(MSB) \leftarrow 0$ 
       $m_{i,j} \leftarrow ((P_{i,j} \bmod r) + M(a_i \times b_{i-1,j})) \bmod r$ 
    else
       $a_{i,j} \leftarrow a_{i,j-1}$ 
       $m_{i,j} \leftarrow m_{i,j-1}$ 
       $carry(MSB) \leftarrow carry(MSB-1)$ 
    end if
  end if
end for
    
```

이런 결과로 입력 정수들이 가변 데이터량에 의한 연산 수행에 있어서 병목현상으로 인한 처리시간 지연을 효과적으로 감소할 수 있도록 연산이 수행된다. 또한 모듈러 승산연산의 고속수행을 위하여 룩업 테이블을 사용함으로써 1회의 연산횟수를 가지고 처리하도록 하였다. 그리고 전체 승산기를 시스템 어레이 구조를 사용하지 않고 단지 모듈러 배수에 대한 캐리 생성부분만을 어레이 방식으로 채택함으로써 154 자리 이상의 정수가 입력되더라도 외부 인터페이스에 해당하는 어레이 부분만을 확장함으로써 주위환경에 대한 아무런 제약조건이 발생하지 않도록 하였다. 또한 제안된 MM 방식은 알고리즘 전체 중 일부만이 어레이 구조로 되어 있고 나머지 부분은 룩업 테이블을 사용하여 계산하는

구조로 되어 있으므로 하드웨어 구현시 가장 큰 단점이었던 외부로부터의 크래킹에 의한 공격으로부터 더욱 자유로울 수 있다는 것이다.

설계된 MM 방식의 블록은 연산속도를 증가시키기 위하여 룩업 테이블을 사용하였으며 입력되는 데이터의 스트림 값에 따라 어레이 구조를 사용할 수 있도록 가변적으로 설계하였다.

그림 3은 MM 방식을 적용하여 설계한 RSA 암호시스템이다. 그림 3에서 RSA의 구성은 몽고메리 매트릭스 승산부와 외부 메모리를 이용하여 처리속도를 향상시킬 수 있도록 하였으며 이에 관한 제어는 별도의 제어블록을 이용하여 제어된다.

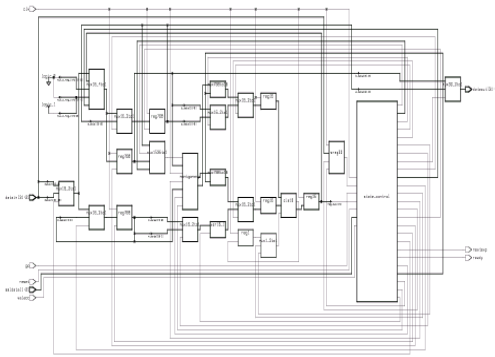


그림 3. MM 방식을 적용한 RSA 시스템  
Fig. 3 RSA system adopted MM algorithm

그림 4는 MM 방식을 적용한 RSA 암호시스템에 대한 모의실험 결과이다. 주클럭 @50MHz에 대하여 입력 데이터에 대한 출력과형이 기존 RSA 암호시스템에 비하여 약 30%의 응답시간이 빠르게 출력됨을 보이고 있다.

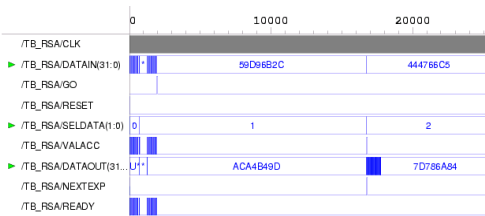


그림 4. MM을 적용한 RSA 암호시스템의 모의실험  
Fig. 4 Simulation in RSA cryptosystem using MM

#### IV. 결 론

본 논문에서는 RSA와 같은 비대칭 암호시스템에서 성능을 좌우하게 되는 모듈러 승산기와 지수 승산기에 대하여 매트릭스 룩업 테이블을 이용한 MM 방식을 제안하였으며 MM 방식을 적용한 RSA 암호시스템을 설계하였다. 설계된 MM-RSA는 기존 몽고메리 승산기를 적용한 RSA 암호시스템에 비하여 어레이 배열로 인한 면적과 외부 메모리 사용으로 인한 단점이 있지만 처리 속도 면에서 기존 RSA에 비하여 MM-RSA는 응답시간이 30% 향상되었음을 확인하였다. 이러한 결과는 암호시스템의 특성상 처리속도가 우선임을 고려할 때 네트워크 기반 통신시스템에 매우 적합할 것으로 생각된다.

#### 참고 문헌

- [1] T. ElGamal, "A Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms", IEEE Trans. Information Theory, Vol. 31, pp.469-472, 1985.
- [2] O. Goldreich, "Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme", In Proceeding CRYPTO'86, Lecture Notes in Computer Science, No. 263, Springer-verlag, pp.104-110, 1987.
- [3] O. Goldreich, H. Krawczyk and M. Luby, "On the existence of pseudorandom generators", SIAMJ, on Computing, Vol. 22(6), pp.1163-1175, 1993.
- [4] O. Goldreich and L. A. Levin, "A Hard-core Predicate for All One-Way Functions", Proceedings of the 21st ACM Symposium on Theory of Computing, pp.25-32, 1989
- [5] P. L. Montgomery, "Modular multiplication without trial division", Mathematics of computation, Vol. 44, pp.519-521, 1985.
- [6] C. N. Zhang, H. L. Martin and D. Y. Yun, "Parallel algorithms and systolic array designs for RSA cryptosystem", Intel Confer. on systolic arrays, pp.341-350, 1988
- [7] K. Iwamura, T. Matsumoto and H. Imai, "Systolic arrays for modular exponentiation using montgomery method", Proc. EUROCRYPT'92, pp.477-481, 1992.
- [8] S. E. Eldridge and C. D. Walter, "Hardware

implementation of montgomery's modular multiplication algorithm", IEEE Trans. Comput., Vol. 42, No. 6, pp.693-699, 1993.

- [9] C. D. Walter, "Systolic modular multiplication", IEEE Trans. Comput., Vol. 42, No. 3, pp.376-378, 1993.

### 저자 소개



**이선근(Seon-keun Lee)**

원광대 전자공학과 박사  
(주)율시스템즈 연구소장

※ 관심분야 : 암호시스템, 프로세서 SoC설계, 유전자알고리즘



**정우열(Woo-yeol Jeong)**

원광대 전자공학과 박사  
한려대학교 멀티미디어정보통신공학과 교수

※ 관심분야 : 이동통신시스템, 암호

시스템, VLSI설계