韓國컴퓨터情報學會 論文誌
第15卷 第11號, 2010. 11.

2010-15-11-3-5

# RFID/USN 환경에서 노드들간의 보안성 증대를 위한 RI-RSA 시스템 설계

이 선 근*

# RI-RSA system design to increase security between nodes in RFID/USN environments

Seon-Keun Lee*

## 요 약

IT발전으로 RFID/USN 환경은 매우 친숙한 통신수단이 되었다. 그러나 노드수의 증가와 보안, 크기제약 등으로 인하여 다양한 서비스를 구현하기에는 부족한 점이 존재한다. 그러므로 본 논문은 이러한 문제점을 해결하기 위하여 RFID/USN 환경에 적합한 비대칭형 암호시스템인 RI-RSA를 제안하였다. 제안된 RI-RSA 암호시스템은 구현하기 용이하다. 처리속도를 증가시킬 수 있도록 RSA의 병목현상이 발생되는 곱셈부를 2차원으로 세분화하여 RI-RSA를 제안하였고 이를 하드웨어 칩 레벨로 구현하였다. 모의실험 결과, 6% 정도의 회로 감소를 가져왔고, 처리시간은 기존 RSA와 비교하여 RI-RSA가 30% 빠름을 확인하였다.

## Abstract

Due to the IT development, RFID/USN became very familiar means of communication. However, because of increased number, security, and size constraints of nodes, it is insufficient to implement a variety of services. To solve these problems, this paper suggests RI-RSA, which is an appropriate asymmetric cryptographic system for RFID/USN environment. The proposed RI-RSA cryptographic system is easy to implement. To increase the processing speed, RI-RSA was suggested by subdividing the multiplication section into two-dimensional, where bottleneck phenomena occurs, and it was implemented in the hardware chip level. The simulation result verified that it caused 6% of circuit reduction, and for the processing speed, RI-RSA was 30% faster compare to the existing RSA.

▸ Keyword : RFID/USN, RSA, Asymmetric, Modular exponent, Cryptographic algorithm

• 제1저자 : 이선근
• 투고일 : 2010. 05. 28, 심사일 : 2010. 07. 12, 게재확정일 : 2010. 09. 14.
* 전북대학교 IT응용시스템공학부

# Ⅰ. 서 론

정보통신의 발달에 힘입어 RFID 및 USN 응용분야가 매우 활발해지고 있다. 그러나 이러한 RFID/USN 환경이 발달할수록 중요 정보의 보안성에 대한 경각은 비례적이지 않다. 거의 모든 산업분야에 적용되고 있는 RFID/USN 분야에 대한 보안성의 중요성은 정보통신이 발달할수록 더욱 발전되어야 할 분야임에도 불구하고 비약적인 발달이 되지 못하고 있는 것이 현 실정이다. 특히, 보안을 위하여 사용되는 다양한 방법들이 RFID/ USN 특성인 센서노드의 자원제약 때문에 많은 비용을 요구하는 비대칭형 방식의 키 교환을 직접 사용할 수 없다[1].

그러므로 본 논문에서는 이러한 문제점을 해결하기 위하여 RFID/USN 환경에 적합하도록 비대칭형 암호시스템의 자원제약을 최소화할 수 있는 방식을 제안한다. 특히, RSA와 같은 비대칭형 암호방식은 키의 분배와 관리면에서 매우 효율적이며 인증(authentication)과 전자서명(digital signature)이 가능한 장점을 가지고 있으나, 154자리(512 비트) 이상의 큰 수에 대한 모듈러 지수 연산 처리시간이 너무 오래 걸린다는 단점을 가지고 있다. 즉, RFID/USN 환경에 대한 자원제약이 매우 심해 실질적으로 이러한 환경에 RSA와 같은 비대칭형 암호방식을 사용하기에 무리가 따른다.

그러므로 본 연구에서는 이러한 자원제약을 감소시키기 위한 방법으로 모듈러 지수연산을 최소화할 수 있는 기법을 개발하여 RSA 암호시스템에 적용시켰다.

# Ⅱ. 몽고메리 승산 알고리즘

RSA 암호시스템에서 병목현상이 두드러지는 모듈러 연산은 크게 두 부분으로 나뉘어진다. RSA는 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 현 실정이다. 모듈러 지수 연산을 수행하기 위하여 개발된 방법은 이진 방식, $r$-진 방식, 누승테이블 방식, 가산고리 방식 등이 있으나 이중에서 이진 방식을 적용한 몽고메리 모듈러 지수연산(montgomery modular exponent) 방식이 개발된 방식 중에서 처리속도 측면만 고려할 때 가장 빠른 것으로 보고되고 있다. 다음은 몽고메리 지수(Montgomery Exponent) 부분을 계산하기 위한 과정을 나타내는 알고리즘이다.

1me : $M_{init}' \leftarrow MP(1, M, N, R')$ (MP(·) : Montgomery product)
2me : $C_{init}' \leftarrow MP(1, 1, N, R')$
3me : for i ← 0 to n-1
  if $e_i = 1$ then $C_i' \leftarrow MP(C_{i-1}', M_{i-1}', N, R')$
  if $i \leqq n-2$ then $M_i' \leftarrow MP(M_{i-1}', M_{i-1}', N, R')$
4me : $C \leftarrow MP(1, C_i', N, R')$
5me : $C \leftarrow C \bmod N$
6me : return C ($C = M^e \bmod N$)

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. 다음 알고리즘은 몽고메리 승산(Montgomery Multiplication)인 경우에 수행되는 연산과정을 보여준다[2-3].

1mm : $n_0' \leftarrow n_{setting\ value}' \bmod r$ $(n_{-0} = n_{setting\ value})$
2mm : $P_{init} \leftarrow 0$
3mm : for i ← 0 to n-1
  $P_i \leftarrow P_{i-1} + a_i \times B \times r^i$
  $m_i \leftarrow p_i \times n_0' \bmod r$
  $P_i \leftarrow P_i + m_i \times N \times r^i$
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 과정에서 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 디지트 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 부분 결과의 하위 자리수에 의존하는 형태이므로 모듈러의 배수와 캐리의 전달 진행방향은 동일하다. 이러한 이유로 모듈러 승산에 대한 어

레이 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 시스토릭 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[4].

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 RI(Reduced Iteration) 기법을 이용한 연산 수행 단계를 줄여 모듈러 승산 연산의 고속화를 시도하였다.

## III. 몽고메리 RI 승산 알고리즘

1mm에서 5mm까지의 연산과정 중에서 반복 수행되는 부분은 for 문에 해당하는 식 (1)과 같다.

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

식 (1)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 (1) 부분을 병렬처리가 가능하도록 변형하면 처리속도의 향상 및 시스템 효율을 증대시킬 수 있다. 그러므로 반복수행 연산과정을 병렬처리가 가능하도록 하기 위하여 식 (1)을 식 (2)와 같이 변형한다.

$$\begin{aligned}&\text{for } i \leftarrow 0 \ to \ n-1 \\ &\quad m_i \leftarrow ((P_{i-1} \bmod r) + a_i \times b_0) \times n_0{}' \bmod r \\ &\quad P_i \leftarrow (P_{i-1} + a_i \times B + m_i \times N) \ div \ r\end{aligned} \quad \cdots\cdots (2)$$

여기에서 $P_{i-1}$은 for 루프문장의 $i$번째 계산을 수행하기 이전 $P$에 대한 부분 결과값이다.

몽고메리 승산 정의에 의하여 식 (3)이 성립한다.

$$r^i P_i = a_i B + m_i N \quad \cdots\cdots (3)$$

그러므로 부분 결과값 $P$의 최종값 $P_n$은 식 (4)와 같이 된다.

$$r^n P_i = AB + MN \quad \cdots\cdots (4)$$

식 (3)과 식 (4)에 의하여 결과적으로 식 (5)를 유추할 수 있다.

$$P \equiv ABR^{-1} \bmod N \quad \cdots\cdots (5)$$

이때 한자리 정수에 대한 승산을 수행할 때, 식 (6)과 같이 한자리 정수에 대한 값은 부분 자리수와 같다고 정의할 수 있다.

$$M(a_i B) \equiv a_i B \quad \cdots\cdots (6)$$

그러므로 한자리 정수는 식 (6)과 같이 정의할 수 있으므로 부분 결과값 $P$는 식 (7)과 같이 정의될 수 있다.

여기에서 M(·)은 매트릭스 반복 부분인 식 (1)에 대한 식 (2)의 변형된 표현이므로 $R \equiv r^n$을 이용하여 $P$를 구할 필요 없이 $r$-진수로 표현된 디지트 레벨(digit level)로 모든 계산이 처리되도록 각각의 부분 결과값에서 나눗셈 연산을 수행하도록 함으로서 각 연산 시점에서의 의존 관계값의 수가 감소되어 병렬처리가 용이하게 된다.

$$\begin{aligned}r^i P_i &= M(a_i B) + M(m_i N) \quad \cdots\cdots (7) \\ r^n P_n &= M(AB) + M(MN) \\ \therefore P &\equiv M(ABR^{-1} \bmod N)\end{aligned}$$

식 (2)에 대한 식 (7)의 표현은 한자리 정수에 대한 병렬처리와 룩업 테이블을 이용한 매트릭스 함수를 이용하여 변형한 것이다. 식 (7)과 같이 변형한 이유는 병렬처리가 가능하도록 변형하였다 하더라도 식 (2)의 각각의 연산부분이 해결되기 전까지는 다음 단계로 넘어갈 수 없는 것이 기존 몽고메리 모듈러 승산 알고리즘이다. 그러므로 식 (7)과 같이 매트릭스 함수를 포함하게 되면 동시 다발적으로 연산이 수행되어 시스템 처리시간의 지연이 없어지게 된다. 이러한 이유로 식 (7)과 매트릭스 함수 M(·)를 이용하여 식 (2)를 다시 변형하면 식 (8)과 같이 정리할 수 있다.

$$\begin{aligned}&\text{for } i \leftarrow 0 \ to \ n-1 \\ &\quad m_i \leftarrow ((P_{i-1} \bmod r) + M(a_i \times b_0)) \times n_0{}' \bmod r \\ &\quad P_i \leftarrow (P_{i-1} + M(a_i \times B) + M(m_i \times N)) \ div \ r\end{aligned} \quad \cdots\cdots (8)$$

그러므로 식 (8)은 본 논문에서 제안하는 몽고메리 모듈러 승산을 위한 몽고메리 RI 승산 알고리즘(Montgomery Reduced Iteration Multiplication Algorithm)인 RI에 대한 정의식

이 되며 RI는 몽고메리 모듈러 승산의 반복연산에 해당하는 부분이 된다.

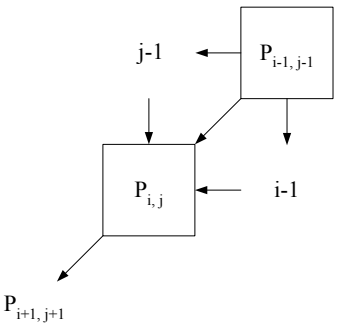


그림 1. 2차원 배열에 대한 승수 연산
Fig. 1 Multiplication in two-dimension array

그림 1에서 각 부분 승산값은 한 단계씩 아래방향으로 이동하며 가산을 수행하게 된다. 즉, 이차원 $(i, j)$ 인덱스 공간에서 a와 m은 (0, -1)방향으로, b와 N은 (-1, 0) 방향으로, P는 (-1, -1) 방향으로 값이 이동함을 의미한다. 그러므로 각 몽고메리 부분 결과값을 생성하는 것은 이전단계에서 계산된 P의 부분 결과값$P_{i-1,\,j-1}$이 된다.

이때 모듈러 배수 $m_i$는 앞 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 서로소인 조건에서 항상 1이 되므로 $m_i$의 계산에서 생략 가능하게 된다.

몽고메리 부분 결과값$P_{i,j}$를 계산하는 과정에서 발생되는 캐리는 별도의 계산 수행없이 출력 잉여분의 MSB에 추가시켜 출력한다[4-5].

식 (5)를 수행하기 위해서 설정해 놓은 2진 비트레벨에 대한 룩업 테이블은 0(0×0)~81(9×9)까지의 2진 데이터들로 구성된다. 각 정수들에 대하여 비트 레벨로 미리 설정해 둠으로서 입력이 유입되는 순간 바로 승산을 수행할 수 있도록 하기 위한 부분이다.

그림 2는 한자리 정수에 대한 몽고메리 매트릭스 승산블록이다. 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.

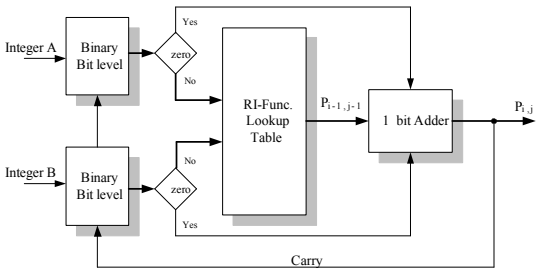


그림 2. 한자리 정수에 대한 몽고메리 RI 승산 블록
Fig. 2 Montgomery RI multiplication for single integer

기존 몽고메리 승산 알고리즘은 초기값과 계산과정에서 산출되어지는 중간단계의 값을 저장하기 위한 매체인 버퍼가 별도로 필요하였으나 본 논문에서 제안한 몽고메리 RI 승산 알고리즘은 별도의 저장매체가 불필요하며 단지 2진 비트 레벨들에 대한 1회의 승산연산을 수행하기 위한 룩업 테이블만 필요하게 된다.

따라서 RI는 기존 몽고메리 승산 알고리즘의 반복계산이 필요했던 식 (1)에 대하여 식 (8)을 정의하였고 정의된 식 (8)에 대하여 전체적인 RI는 그림 2와 식 (9)로서 정의될 수 있다.

식 (9)에서 RI 연산은 매우 단순한 과정을 보인다. 즉, 입력 정수중의 어느 하나가 0 또는 1인 경우에는 $P_{i,j}$값은 룩업 테이블을 거치지 않고 즉시 출력하도록 하였다. 또한 병렬처리가 가능하도록 하기 위하여 시스템 구성은 정수값 가중치에 대한 어레이 구조를 채택하였다.

이런 결과로 입력 정수들이 가변 데이터량에 의한 연산수행에 있어서 병목현상으로 인한 처리시간 지연을 효과적으로 감소할 수 있도록 연산이 수행된다. 또한 모듈러 승산연산의 고속수행을 위하여 룩업 테이블을 사용함으로서 1회의 연산횟수를 가지고 처리하도록 하였다. 그리고 전체 승산기를 시스토릭 어레이 구조를 사용하지 않고 단지 모듈러 배수에 대한 캐리 생성 부분만을 어레이 방식으로 채택함으로서 154자리 이상의 정수가 입력되더라도 외부 인터페이스에 해당하는 어레이 부분만을 확장함으로서 주위환경에 대한 아무런 제약조건이 발생하지 않도록 하였다.

$$
\begin{aligned}
&\text{for } all\, 0 \leqq i \leqq n-1,\ 0 \leqq j \leqq n+1\\
&\quad \text{if } A \text{ or } B = 0 \text{ then}\\
&\qquad \text{if } i = 0 \text{ then}\\
&\qquad\quad P_{i,j} \leftarrow 0\\
&\qquad \text{else}\\
&\qquad\quad b_{i,j} \leftarrow b_{i-1,j}\\
&\qquad\quad n_{i,j} \leftarrow n_{i-1,j}\\
&\qquad \text{end if}\\
&\qquad \text{if } j = n+1 \text{ then}\\
&\qquad\quad P_{i,j} \leftarrow 0\\
&\qquad \text{else}\\
&\qquad\quad P_{i,j} \leftarrow P_{i-1,j+1}\\
&\qquad \text{end if}\\
&\quad elsif\ A \text{ or } B = 1 \text{ then}\\
&\qquad \text{if } i = 0 \text{ then}\\
&\qquad\quad P_{i,j} \leftarrow A \text{ or } B\\
&\qquad \text{else}\\
&\qquad\quad b_{i,j} \leftarrow b_{i-1,j-1}\\
&\qquad\quad n_{i,j} \leftarrow n_{i-1,j-1}\\
&\qquad endif\\
&\qquad \text{if } j = n+1 \text{ then}\\
&\qquad\quad P_{i,j} \leftarrow A \text{ or } B\\
&\qquad \text{else}\\
&\qquad\quad P_{i,j} \leftarrow P_{i-1,j+1}\\
&\qquad endif\\
&\quad elsif\ A \text{ or } B \neq 0 \text{ or } 1 \text{ then}\\
&\qquad \text{if } j = 0 \text{ then}\\
&\qquad\quad a_{i,j} \leftarrow a_i\\
&\qquad\quad carry(MSB) \leftarrow 0\\
&\qquad\quad m_{i,j} \leftarrow ((P_{i,j} \bmod r) + M(a_i \times b_{i-1,j})) \bmod r\\
&\qquad \text{else}\\
&\qquad\quad a_{i,j} \leftarrow a_{i,j-1}\\
&\qquad\quad m_{i,j} \leftarrow m_{i,j-1}\\
&\qquad\quad carry(MSB) \leftarrow carry(MSB-1)\\
&\qquad endif\\
&\quad endif\\
&end \text{ for}
\end{aligned}
\qquad \cdots\cdots (9)
$$

또한 제안된 RI는 알고리즘 전체 중 일부만이 어레이 구조로 되어있고 나머지 부분은 룩업 테이블을 사용하여 계산하는 구조로 되어 있으므로 하드웨어 구현시 가장 큰 단점이었던 외부로부터의 크래킹에 의한 공격으로부터 더욱 자유로울 수 있다는 것이며, RFID/USN과 같이 외부제약이 많은 분야에 보다 쉽게 적용하기 용이하다는 특징을 가진다.

## Ⅳ. 모의실험 및 고찰

RFID/USN 환경에서 처리시간, 크기 등과 같은 자원제약으로 인하여 많은 장점에도 불구하고 비대칭형 암호알고리즘을 적용하기 어려웠었다. 그러므로 본 논문에서는 이러한 문제점을 해결하기 위하여 RI를 개발하여 RSA 암호시스템에 적용하여 설계하였다.

RI-RSA 암호알고리즘의 구현은 VHDL을 이용하여 Top-down 방식으로 진행하였으며, 설계환경은 Synopsys Design Analyser Ver. 1999.10, QUARTUS 7.0을 사용하였고, 모의실험에 사용된 툴은 Synopsys VHDL Debegger, ModelSim 5.8C를 사용하였다. 구현을 위한 테스트베드는 ALTERA Cyclone II EP2C35 디바이스를 사용하였다.

그림 3은 RI블록을 회로합성한 그림으로서 그림 2와 식 (9)의 기능을 수행하는 실제 회로이다. 그림 4는 RI-RSA 전체 시스템의 회로합성으로서 RI 블록을 포함하는 RI-RSA 전체시스템이다.

그림 3. RI 승산 합성도
Fig. 3 Synthesized RI multiplication circuit

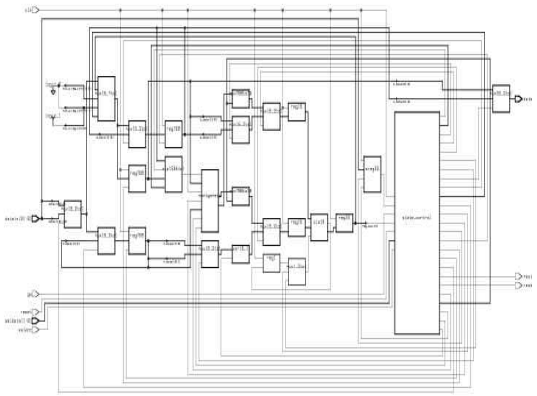

그림 4. RI 블록을 포함하는 RI-RSA 시스템
Fig. 4 RI-RSA system including RI block

표 1은 기존 RSA와 RI-RSA를 비교분석한 표이다[6-9]. RI-RSA에 사용된 룩업 테이블은 MACRO-cell (memory 영역)을 이용하였기 때문에 별도의 회로 증가가 없었으며 전체적으로 6%정도의 회로 감소를 가져왔고, 처리시간은 50MHz에서 기존 RSA와 비교하여 RI-RSA가 30% 빠름을 확인할 수 있었다.

표 1. RI-RSA와 RSA 비교분석표
Table 1. Comparison between RSA and RI-RSA

| 50@Mhz | RI-RSA | RSA |
|---|---|---|
| gate count | 175K | 186K |
| Throughput | 122Kbps | 94Kbps |

룩업테이블에 저장되는 RI 데이터는 실제로 수 바이트 정도이므로 RFID/USN의 ID 및 기본 정보저장소에 룩업테이블 내용을 할애한다면 그림 2의 RI 승산부의 부하가 감소되어 RI-RSA의 성능은 더욱 증가될 것으로 생각된다.

## V. 결론

현대사회는 정보통신의 발달에 힘입어 smart-x, RFID/USN 시대가 본격화되고 있다. 이러한 시점에서 RFID/USN 분야는 현대사회에 비중이 증대되고 있지만 한정된 자원제약 때문에 적절한 보안알고리즘의 개발이 매우 중요하다.

그러므로 본 논문에서는 노드들간의 보안성을 증대시키고 보다 높은 효율을 유지하기 위한 방편으로 비대칭형 암호알고리즘인 RSA에 RI기법을 적용한 RI-RSA를 설계하였다. 즉, RSA와 같은 비대칭 암호시스템에서 성능을 좌우하게 되는 모듈러 승산기와 지수 승산기에 대하여 룩업 테이블을 이용한 RI 알고리즘을 제안하였으며 RI-RSA 암호시스템을 설계하였다. 설계된 RI-RSA는 기존 RSA에 비하여 응답시간이 30% 향상되었음을 확인하였다. 이러한 결과는 암호시스템의 특성상 처리속도가 우선임을 고려할 때 네트워크 기반 RFID/USN 시스템에 매우 적합할 것으로 사료된다.

## 참고문헌

[1] 김창환, “유비쿼터스 환경에서의 정보보호기술,” *http://www.eic.re.kr*, 2008년 11월.

[2] 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.

[3] P. L. Montgomery, "Modular multiplication without trial division," mathematics of computation, Vol. 44, pp. 519-521, 1985.

[4] S. E. Eldridge, C. D. Walter, "Hardware implementation of montgomery's modular multiplication algorithm," IEEE Trans. Computer, Vol. 42, No. 6, pp. 693-699, 1993.

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

[6] 이선근, 정우열, “휴대용 보안시스템에 적합한 MIT-Serpnet 암호알고리즘 설계에 관한 연구,” 한국컴퓨터정보학회논문지, 제 13권 제 6호, 195-201쪽, 2008년 11월.

[7] *http://www.redbooks.ibm.com/redbooks/pdfs/sg247070.pdf*

[8] *http://www.rsa.com/rsalabs/node.asp?id=2100* (The RSA Laboratories Secret-Key Challenge)

[9] *http://discovery.bits-pilani.ac.in/discipline/csis/tsbs/Main/Publish/comparch/presilicon_Fpt04.pdf*, 2004.

## 저 자 소 개

이 선 근

1997 : 원광대학교 공학석사
2003 : 원광대학교 공학박사
2010 - 현재 : 전북대학교 IT응용시스템공학부 시간강사
관심분야 : 정보보호 및 암호알고리즘 설계, 보안시스템 SoC 설계, 임베디드 시스템 설계