

RSA 암호시스템에서 처리속도향상을 위한 모듈러 승산기 설계에 관한 연구

정 우 열*

A Study on the Modus Multiplier design on Enhancing Processing Speed in the RSA cryptosystem

Woo-Yeol Jeong*

요 약

네트워크의 발전은 통신망의 발전과 더불어 심각한 사회문제를 발생시킨다. 즉, 보안에 관련된 문제는 네트워크를 사용할 경우 해킹과 크래킹에 대하여 더욱 주의해야한다는 것이다. 이러한 해커나 크래커로부터 보안을 유지하기 위해서는 새로운 암호알고리즘을 개발하거나 키길이를 길게하여 정해진 시간안에 복호불가의 상태를 유지하는 방법이 일반적으로 사용되고 있다.

본 논문에서, RSA 암호시스템에서 제안된 몽고메리 승산기는 캐리부분만을 어레이 형태로 구성하였고 병목현상을 없애기 위하여 승산과정을 가변길이화로 구성하였다. 그러므로 제안된 몽고메리 승산기는 실시간 처리 및 외부의 크래킹을 막아낼 수 있는 기능을 강화시켰다.

Abstract

The development of network and the other communication-network can generate serious problems. So, it is highly required to control security of network. These problems related secu be developed and keep up to confront with anti-security part such as hacking, cracking. Th way to preserve security from hacker or cracker without developing new cryptographic algori 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 multiplicator for eliminating bottle neck effect with the RSA cryptosystem. Therefore, this proposed montgomery multiplicator enforce the real time processing and prevent outer cracking.

I. 서 론

고도의 정보통신의 발전은 다양한 정보의 전송이 필수 불가결하며 이러한 정보 전송은 네트워크의 발전을 의미 하며 네트워크의 발전은 정보보호기술의 대두를 가져온다. 이러한 네트워크의 정보보호를 위한 암호 알고리즘에는 대칭형 암호와 비대칭형 암호 알고리즘 두 가지가 있다. 네트워크에서의 보안은 대칭형 암호방식보다는 비대칭형 암호방식이 키관리 및 분배문제 등에서 매우 유리하다. 일반적으로 비대칭 암호 알고리즘은 수학적으로 해를 구하기 어려운 문제를 대상으로 전개되고 있다. 대표적인 비대칭 암호시스템 중의 하나인 RSA 암호시스템은 매우 큰 정수에 대한 소인수분해가 어렵고 많은 계산을 요구한다는 점을 이용한 암호시스템이다.

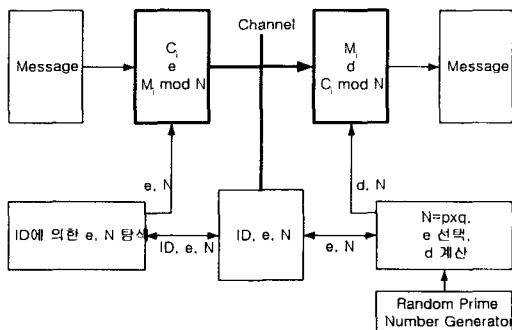


그림 1. 비대칭 암호시스템
Fig 1. Asymmetric cryptosystem

그림 1과 같은 비대칭 암호시스템에서 공개키와 비밀 키의 생성식은 식(1)과 같다.

$$\begin{aligned} \text{encryption: } & C = E(M) = M^e \pmod{N} \\ \text{decryption: } & M = D(C) = D(E(M)) = C^d \pmod{N} \end{aligned} \quad (1)$$

식 (1)과 같은 이유 때문에 RSA는 키의 분배와 관리 면에서 매우 효율적이며 인증(authentication)과 전자서

(digital signature)이 가능한 장점을 가지고 있으나 154자리(512 비트)이상의 큰 수에 대한 모듈러 지수연산이 요구되는 처리시간에 대한 단점을 가지고 있다.

RSA 암호시스템에서 요구되는 모듈러 지수연산의 고속화를 위하여 모듈러 승산의 횟수를 줄이는 방법과 승산부분을 고속화하는 방법이 주로 연구되어왔다. 이러한 연산은 모듈러 승산(modular multiplication)과 모듈러 선(modular reduction) 연산으로 구성되어 있으며 모듈러 승산이 전체연산의 대부분을 차지하게 된다.

모듈러 지수 연산은 이진방식, r -진방식, 누승테이블 방식, 가산고리방식 등이 있으며 이중에서 이진방식을 이용한 몽고메리 모듈러 지수연산(montgomery modular exponent)방식이 처리 속도면에서 가장 빠르다.

몽고메리 지수(Montgomery Exponent : ME(M, E, R))를 계산하기 위하여 다음 식 (2)와 같은 순서를 이용한다.

```

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

```

여기에서 입력은 M, e, N, R, R' ($R' = R^2 \bmod N$ pre-computation)이며 출력은 C 이다.

1e부터 6e까지는 몽고메리 모듈러 지수연산 알고리즘을 표현한 것으로써 R' 를 미리 계산하여 입력하고 모듈러 승산부분을 몽고메리 모듈러 승산으로 치환함으로서 알고리즘 자체가 간결하게 되며 이로 인하여 계산속도가 크게 향상된다. 그러나 모듈러 승산의 횟수는 지수부 이진수의 '1'의 개수와 같으므로 연산횟수를 줄이기 위하여 '1'의 개수를 줄이는 형태로써 연산속도의 고속화가 진행되고 있다.

몽고메리 곱셈(Montgomery Multiplication : MP(A, N, R))인 경우에는 다음과 같은 과정을 거치게 된다.

```

1m :  $n_0' \leftarrow -n_0' \bmod r$ 
2m :  $P_{-1} \leftarrow 0$ 

```

3m : for i ← 0 to n-1
 $P_i \leftarrow P_{i-1} + a_i \times B \times r^i$
 $m_i \leftarrow p_i \times n_0' \bmod r$
 $P_i \leftarrow P_i + m_i \times N \times r^i$
4m : $P \leftarrow P_i \text{ div } R$
5m : return P

입력은 A, B, N, R이며 출력은 $P = A \times B \times R^{-1} \bmod N$ 이다.

1m에서 5m까지의 과정에서 A, B, N, R, P는 정수이 a_i, p_i 는 정수 A, P의 i번째 자리에 해당하는 자릿수이다. 모듈러스의 배수 m_i 는 디지트값 p_i 에 의해 계산되고 그 결과를 누적된 P_i 의 계산에 이용하여 m_i 를 계산하기 위한 덧셈을 수행한다. 그럼 2, 3과 같이 몽고메리 알고리즘에서 모듈러스의 배수는 부분결과의 하위자리수에 의존 하므로 모듈러스의 배수와 캐리의 전달 진행방향은 동일하다. 즉, 모듈러 승산에 대한 병렬처리가 가능하기 때문에 RSA 암호시스템에서 몽고메리 알고리즘을 사용하여 고속의 암호화를 수행하게 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 암호화는 모듈러 연산의 고속수행을 저하시키는 요인으로 작용하게 된다.

그러므로 몽고메리(montgomery) 알고리즘을 적용한 RSA 암호시스템에서 기존에 사용된 모듈러 연산의 처리 시간 지연 및 고속처리 저해요인을 없애기 위하여 본 논문에서는 고속수행의 저해요인이 되는 가변길이 데이터와 대용량의 데이터들에 대한 처리시간의 지연을 없애고 모듈러 승산연산의 고속화를 위하여 톡업테이블을 사용하였으며, 전체구조는 시스토릭 어레이 구조를 사용하지 않고 모듈러 배수에 대한 캐리 생성 부분만을 어레이 방식의 구조를 채택하였다.

II. 몽고메리 매트릭스 알고리즘

(Montgomery Matrix Algorithm : MMA)

1m에서 5m까지의 과정 중에서 연산 반복부분은 식 (2)와 같다.

for $i \leftarrow 0 \text{ to } n-1$
 $P_i \leftarrow P_{i-1} + a_i \times B \times r^i$ (2)
 $m_i \leftarrow p_i \times n_0' \bmod r$
 $P_i \leftarrow P_i + m_i \times N \times r^i$

식 (2)에 대한 반복수행 연산과정을 병렬처리가 가능하도록 하기 위하여 변형하면

for $i \leftarrow 0 \text{ to } n-1$
 $m_i \leftarrow ((P_{i-1} \bmod r) + a_i \times b_0) \times n_0' \bmod r$ (3)
 $P_i \leftarrow (P_{i-1} + a_i \times B + m_i \times N) \text{ div } r$

여기서 $P_{-1} = 0$ 은 P 의 초기값이고, P_{i-1} 은 for 루프 문장의 i번째 계산을 수행하기 이전 P 에 대한 부분결과 값이다. 몽고메리 승산 정의에 따라 $r^i P_i = a_i B + m_i N$ 이 성립하며 P 의 최종값 P_n 은 $r^n P_i = AB + MN$ 이 되어 결과적으로 $P = ABR^{-1} \bmod N$ 이 된다. 이때 한자리 정수에 대한 승산을 수행함에 있어 $M(a, B) \equiv a_i B$ 로 정의할 수 있으므로

$r^i P_i = M(a, B) + M(m, N)$
 $r^n P_n = M(AB) + M(MN)$ (4)
 $\therefore P \equiv M(ABR^{-1} \bmod N)$

이 된다.

식 (2)에 대한 식 (3)의 변형된 표현은 $R = r^n$ 을 이용

하여 P 를 구할 필요없이 r 진수로 표현된 디지트 레벨(digit level)로 모든 계산이 처리되도록 각각의 부분결과값에서 나눗셈 연산을 수행하므로 각 연산점에서의 의존 관계값의 수가 축소되어 병렬처리가 용이하다는 점이다. 식 (3)에 대한 식 (4)의 표현은 한자리 정수에 대한 병렬처리와 툭업테이블을 이용한 매트릭스 함수를 이용하여 변형한 것이다. 식 (4)와 같이 변형한 이유는 병렬처리가 가능하도록 변형하였다 하더라도 식 (3)의 각각의 연산부분이 해결되기 전까지는 다음단계로 넘어갈 수 없는 것이 기존 몽고메리 모듈러 승산 알고리즘이다. 그러므로 식 (4)와 같이 매트릭스 함수를 포함하게 되면 동시에 별도로 연산이 수행되어 처리시간의 지연이 없어지게 된다. 이러한 이유로 식 (4)와 매트릭스 함수 $M(\cdot)$ 를 이용하여 식 (3)을 다시 변형하면

$$\begin{aligned} \text{for } i \leftarrow 0 \text{ to } n-1 \\ m_i &\leftarrow ((P_{i-1} \bmod r) + M(a_i \times b_0)) \times n_0' \bmod r \\ P_i &\leftarrow (P_{i-1} + M(a_i \times B) + M(m_i \times N)) \bmod r \end{aligned} \quad (5)$$

그러므로 식 (5)는 본 논문에서 제안하는 몽고메리 모듈러 승산을 위한 몽고메리 매트릭스 승산 알고리즘의 반복연산에 해당하는 부분이 된다.

$$\begin{array}{lcl} 0101(5) & = & A(\text{피승수}) \\ \times 1010(10) & = & B(\text{승수}) \\ \hline 0000 & = & a_0 B \times r^0 \\ 0101 & = & a_1 B \times r^1 \\ 0000 & = & a_2 B \times r^2 \\ 0101 & = & a_3 B \times r^3 \\ \hline 0110010(50) & = & A \times B(\text{integer product}) \end{array}$$

그림 2. 2차원 배열에 대한 승수연산
Fig 2. Two-dimension array for multiplication

2진수로 표현되는 비트레벨에서 P 를 생성하기 위하여 요구되는 입력값들에서 피승수 A 와 모듈러스의 배수 M 은 반복루프에 해당하는 i 인덱스에 두고 승수 B 와 모듈러스 N 은 단일 비트의 위치에 해당하는 j 인덱스로 두어 인덱스공간을 2차원으로 배열한다.

그림 2에서 각부분 승산값은 한 단계씩 아래로 이동되어 더해진다. 즉, 이차원 (i, j) 인덱스공간에서 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 부분결과 값을 생성하는 것은 이전단계에서 계산된 P 의 부분결과 값 $P_{i-1, j+1}$ 이 된다.

이때 모듈러스의 배수 m_i 는 앞단계의 부분결과 값 $P_{i-1, j+1}$ 과 $a_i \times b_{i-1, 0}$ 를 합한 결과의 최하위 비트에 의존하기 때문에 $j=0$ 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 가능하게된다.

몽고메리 부분결과 값 $P_{i,j}$ 를 계산하는 과정에서 발생되는 캐리는 별도의 계산수행없이 출력 잉여분의 MSB에 추가시켜 출력한다.

표 1은 식 (5)를 수행하기 위해서 설정해 놓은 2진 비트레벨에 대한 툭업테이블이다. 각 정수들에 대하여 비트레벨로 미리 설정해 놓으므로써 입력이 들어가면 바로 승산을 수행할 수 있도록 하기 위한 부분이다.

그림 3은 한자리 정수에 대한 몽고메리 매트릭스 승산 블록이다. 1회의 승산에 필요한 연산횟수는 단지 1회만이 필요하게 설정된 블록이다. 또한 $P_{i,j}$ 에는 내부연산에서 발생된 캐리를 포함하게 되는데 이때 발생된 캐리는 여러 개로 구성된 몽고메리 매트릭스 블록의 2진 비트레벨부분으로 피드백되어 다시 연산을 수행하게 된다.

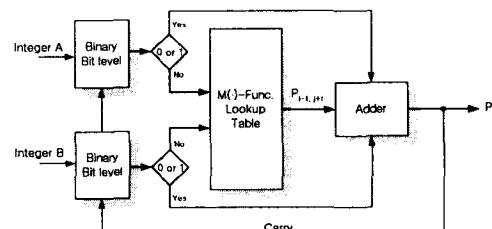


그림 3. 한자리 정수에 대한 몽고메리 매트릭스 승산 블록
Fig 3. Montgomery matrix multiplication for single

기존 몽고메리 승산 알고리즘은 초기값과 계산과정에서 산출되어지는 이전단계의 값을 저장하기 위한 매체가 별도로 필요하였으나 본 논문에서 제안한 몽고메리 매트릭스 승산 알고리즘(MMA)은 별도의 저장매체가 불필요하며 단지 2진 비트레벨들에 대한 1회의 승산연산을 수행하기 위한 툭업테이블만이 필요하게 된다.

따라서 MMA는 기존 몽고메리 승산 알고리즘의 반복 계산이 필요했던 식 (2)에 대하여 식 (5)를 정의하였고 정의된 식 (5)에 대하여 전체적인 MMA는 그림 3과 식 (6)과 같다.

표 1. $M(\cdot)$ 함수에 대한 룩업테이블
Table 1. Lookup table for $M(\cdot)$ Func.

\times	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

식 (6)에서 MMA의 연산은 매우 단순한 과정을 보인다. 즉, 입력정수 중의 어느 하나가 0 또는 1인 경우에는 $P_{i,j}$ 값은 룩업테이블을 거치지 않고 바로 출력하도록 하였다. 또한 별별처리가 가능하도록 정수값의 가중치에 대한 어레이 구조를 채택하였다.

이런 결과로 입력 정수들이 가변 데이터량을 가지므로써 연산수행에 있어서 병목현상으로 인한 처리시간 지연을 효과적으로 수행 할 수 있으며 모듈러 승산연산의 고속수행을 위하여 룩업테이블을 사용함으로서 1회의 연산 횟수를 가지고 처리하도록 하였다. 또한 전체 승산기를 시스토릭 어레이구조를 사용하지 않고 단지 모듈러스 배수에 대한 캐리 생성부분만을 어레이 방식으로 채택하므로써 154자리 이상의 정수가 입력되더라도 어레이부분만을 확장함으로서 아무런 제약조건이 발생하지 않도록 하였다. 또한 제안된 MMA는 알고리즘 전체중 일부만이 어레이 구조로 되어있고 나머지 부분은 룩업테이블을 사용하여 계산하는 구조로 되어 있으므로 하드웨어 구현시 가장 큰 단점이었던 외부로부터의 크래킹에 의한 공격으로부터 더욱 자유로울 수 있다는 것이다.

```

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
  elseif  $A$  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
  elseif  $A$  or  $B \neq 0$  or  $1$  then
    if  $j = 0$  then
       $a_{i,j} \leftarrow a_i$ 
       $carry(MSB) \leftarrow 0$ 
       $m_{i,j} \leftarrow ((P_{i,j} \text{ mod } r) + M(a_i \times b_{i-1,j})) \text{ mod } r$ 
    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

```

(6)

그림 4는 MMA를 이용하여 설계된 몽고메리 승산부에 관한 회로합성 그림이다. 연산속도를 증가시키기 위하여 룩업 테이블을 사용하였으며 입력되는 데이터의 스트링 값에 따라 어레이구조를 사용할 수 있도록 가변적으로 설계하였다.



그림 4. 몽고메리 매트릭스 승산부
Fig 4. Montgomery matrix multiplicator

그림 5는 MMA를 이용하여 전체 RSA를 설계한 회로이다. 그림 5에서 RSA의 구성은 몽고메리 메트릭스 승산부와 외부 메모리를 이용하여 처리속도를 향상시킬 수 있도록 하였으며 이에 관한 제어는 별도의 제어블록을 이용하여 제어된다.

III. 결 론

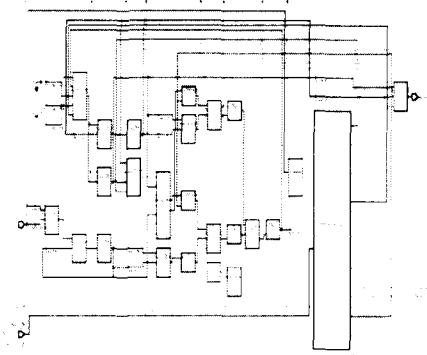


그림 5. MMA를 사용한 RSA 전체 회로
Fig. 5 RSA system using MMA

그림 6은 MMA를 사용한 RSA 시스템에 대한 모의실험 결과파형이다. @50MHz에 대하여 입력 데이터에 대한 출력파형이 기존 RSA 시스템에 비하여 약 30%의 응답시간이 빠르게 출력됨을 보이고 있다.

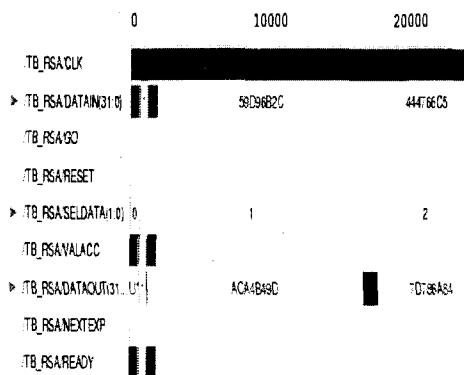


그림 6. MMA를 사용한 RSA 모의실험
Fig. 6. RSA simulation for MMA

정보통신의 발달에 힘입어 네트워크의 발전은 IT산업의 주요한 분야로 자리매김하고 있으며 이러한 정보통신과 네트워크는 투명한 조건을 전제로 한다. 이러한 시점에서 공개키에 대한 사용량이 기하급수적으로 증대되고 있지만 공개키의 처리속도의 한계성으로 인하여 비약적인 발전은 기대하게 어렵다. 그러므로 본 논문에서는 RSA와 같은 공개키에서 성능을 좌우하게 되는 모듈러 승산기와 지수 승산기에 대하여 메트릭스를 이용한 MMA를 제안하였으며 MMA를 이용하여 RSA를 설계하였다. 설계된 MMA는 기존의 RSA 몽고메리 승산기에 비하여 어레이 배열로 인한 면적과 외부 메모리 사용으로 인한 손해를 본다. 그러나 처리속도 면에서 기존의 RSA에 비하여 MMA를 사용한 RSA는 응답시간이 30% 빠르다는 것을 확인하였다. 이는 보편화 되어가는 네트워크상의 보안문제를 다소 해결할 수 있을 것으로 사료된다.

참고문헌

- [1] T. ElGamal, "A Public Key Cryptosystem Signature Scheme Based on Disc Logarithms", IEEE Trans. Inform Theory, Vol. 31, 1985, pp. 469-472
- [2] O. Goldreich, "Two Remarks Concerning Goldwasser-Micali-Rivest Signature Scheme In Proceeding CRYPTO'86, Lecture Notes in Computer Science No. 263, Springer-Verlag 1987, pp. 104-110
- [3] O. Goldreich, H. Krawczyk, M. Luby, "On the existence of pseudorandom generators", SIAM Computing, Vol. 22(6), 1993, pp. 1163-

- [4] O. Goldreich, L. A. Levin, "A Hard-core Predicate for All One-Way Functions", Proceedings of the 21st ACM Symposium on Theory of Computing, 1989, pp. 25-32
- [5] S. Goldwasser, S. Micali, "Probabilistic Encryption", Journal of Computer and Science, Vol. 28, 1984, pp. 270-299
- [6] S. Goldwasser, S. Micali, R. L. Rivest, "Digital Signature Scheme Secure Against Adaptive Chosen Message Attack", SIAM On Computing, Vol. 17, No. 2, 1988, 281-308
- [7] R. Impagliazzo, M. Naor, "Efficient crypto schemes probably as secure as subset sum", 30th Annual Symposium on Foundations of Computer Science, IEEE, 1989, pp. 236-241
- [8] R. J. McEliece, "A Public-key Cryptosystem based on Algebraic Coding Theory", D. Progress Report 42-44, Jet Propulsion Laboratory
- [9] M. Naor, M. Yung, "Universal One-Way Functions and their Cryptographic Applications", In Proceeding 21st ACM Symposium on Theory of Computing, 1989, pp. 33-43
- [10] M. O. Rabin, "Digital Signatures and Probabilistic Functions as Intractable as Factorization", Technical Report MIT/LCS/ TR-212, MIT, 1978.

저자소개



정우열

1982년 원광대학교 전자공학과
공학사

1984년 경희대학교 대학원
전자공학과 공학석사

1999년 원광대학교 대학원
전자공학과 공학박사

1995년~현재 한려대학교
정보통신학과 교수