情報保護學會論文誌
第13券 第3號, 2003. 6

SPA에 견디는 스칼라 곱셈 방법과 하드웨어

윤 중 철*, 정 석 원*, 임 종 인*

A Scalar Multiplication Method and its Hardware with resistance to SPA(Simple Power Analysis)

YoonJoong Chul*, SeokWon Jung*, Jong-in Lim*

요 약

본 논문에서는 side-channel 공격법 중 SPA(Simple Power Analysis)에 견디면서도 효율적인 연산이 가능한 scalar multiplication 방법과 하드웨어 구조를 제시한다. 기존에 제시된 SPA에 견디는 스칼라 곱셈 방법은 연산 속도가 느린 것이 약점이다. 따라서 이를 보안하는 방법에 대한 연구는 중요한 분야이다. 본 논문에서 제시한 타원곡선암호법 전용 하드웨어는 SPA에 견디면서도 동일한 유한체 연산기(multiplier, inverter)를 사용한다는 가정 하에 Coron의 방법 보다 연산 속도가 빠른 스칼라 곱셈 방법과 구조를 제시한다. 논문에서 제시하는 하드웨어는 n비트 키를 사용할 때 연산 속도가 $2n$ · (Inversion cycle)+3(Multiplication cycle)만이 소요된다.

ABSTRACT

In this paper, we propose a scalar multiplication method and its hardware architecture which is resistant to SPA while its computation speed is faster than Coron's. There were SPA-resistant scalar multiplication method which has performance problem. Due to this reason, the research about an efficient SPA-resistant scalar multiplication is one of important topics. The proposed architecture resists to SPA and is faster than Coron's method under the assumption that Coron's and the proposed method use same finite field arithmetic units(multiplier and inverter). With n-bit scalar multiple, the computation cycle of the proposed is $2n$ · (Inversion cycle)+3(Multiplication cycle).

Keyword : *부채널 공격, 타원곡선 암호법, SPA, 스칼라 곱셈방법, 파이프라인 방법*

Ⅰ. 서 론

최근의 하드웨어 공격법 중에 가장 주목받고 있는 것이 전력 소모량을 이용한 공격법이다. 1999년에 Kocher[3] 등은 전력 소모량을 측정하여 DES의 키를 찾아내는 SPA(Simple Power Analysis)와 DPA(Differential Power Analysis)를 소개하였다. 이 후 이에 대한 많은 연구결과가 진행되었는데 타원곡선암호법도 마찬가지로 적용된다.

Coron[1]은 SPA에 견디는 스칼라 곱셈(scalar multiplication) 방법을 제안하였으나 기존 방법에 비해 느린 연산 시간을 가졌다. 이진 방법(binary method)으로 구현했을 때 스칼라(scalar multiple) d가 n비트이면 $(n-1)$번의 두배(doubling)연산과 $(n-1)$번의 덧셈(addition)연산이 필요하다. Hasan[2]은 이를 개선한 n단계의 알고리즘을 제안하였으나 이는 Koblitz curve에 한정된 것이다. Izu와 Takagi[8]는 프로세서(processor) 2개를 사용한다는 가정 하에 한번의 두배와 $(n-1)$번의 덧셈이 소요되는 스칼라 곱셈방법을 제시하였다.

* 고려대학교 정보보호대학원({jsw, jcyoon, seonognim}@cist.korea.ac.kr)

본 논문에서는 타원곡선암호법 전용 하드웨어를 구현할 때 SPA에 견디면서도 동일한 유한체 연산기(multiplier, inverter)를 사용한다는 가정 하에 Coron의 방법 보다 연산 속도가 빠른 스칼라 곱셈 방법과 구조를 제시한다. 논문에서 제시하는 하드웨어는 연산 속도가 $2n$ · (Inversion cycle)+3(Multiplication cycle)만이 소요되고 Izu와 Takagi와 같이 가정하면 n번의 덧셈이 소요된다.

2장에서는 Coron의 알고리즘을 하드웨어로 구성할 때 한계에 대해 얘기하고 3장에서는 제안하는 알고리즘을 소개하고 이를 하드웨어로 구현할 때 어떤 효과가 있는지 설명한다. 4장에서는 3장에서 제시한 알고리즘의 하드웨어 구조에 대해 다루고 5장에서는 Coron의 방법과 제시하는 방법에 대한 연산속도를 비교한다.

본 논문에서는 구현하려는 하드웨어가 유한체의 연산기 중 복잡도(complexity)가 높은 곱셈기(multiplier)와 역원계산기(inverter)를 각 각 1개씩만 가지고 있다고 가정한다.

II. Coron의 알고리즘

Coron[1]이 제시한 SPA에 견디는 스칼라 곱셈 알고리즘은 다음과 같고 $d=\sum_{i=0}^{n-1} d_i 2^i$ 일 때 dP를 구하는 것이다.

Algorithm 1 of Coron
(Double-and-add resistant against SPA)

1. input P
2. $Q[0] \leftarrow P$
3. for i from $n-2$ to 0 do
3.1. $Q[0] \leftarrow 2Q[0]$
3.2. $Q[1] \leftarrow Q[0]+P$
3.3. $Q[0] \leftarrow Q[d_i]$
4. output $Q[0]$

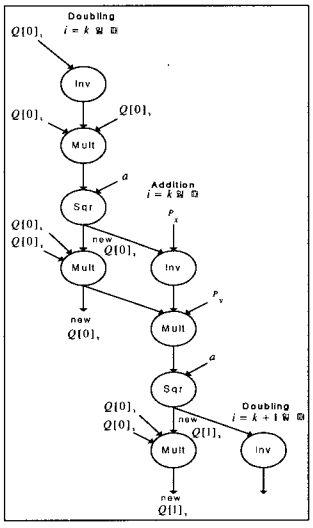
Coron의 알고리즘은 left-to-right 이진 방법의 변형이며 덧셈($Q[1] \leftarrow Q[0]+P$)을 항상 계산하여 SPA를 피한다. 이 때, $Q[0] \leftarrow 2Q[0]$단계에서 얻어진 결과값 $Q[0]$를 이용해 $Q[1] \leftarrow Q[0]+P$를 계산해야 하는데 구체적으로 $GF(2^m)$의 경우에 [그림 1]과 같은 dependence graph를 갖는다. 이 그림은 [7]에 정의된 방법에 의해 분석한 것인데 [7]에 정의된 덧셈과 두배는 $GF(2^m)$의 사칙연산으로 이루어져 있지만 실제 구현 시 복잡도가 높은 연산은 곱셈(또는 제곱)과 역원계산이다. [그림 1]은 덧셈과 두배를 곱셈(제곱)과 역원계산이 있는 것으로 간단화하여 나타낸 것이다. 예를 들어 dependence graph에 있는 두배의 첫 번째 곱셈(Mult)은 입력이 $Q[0]_x$, $Q[0]_y$와 역원계산(Inv)의 결과 값인 $\frac{1}{Q[0]_x}$이다. 이는 [7]에 두배에서 정의된 $Q[0]_x+\{Q[0]_y \times \frac{1}{Q[0]_x}\}$를 계산하는 부분인 데 데 $Q[0]_x$와 $\frac{Q[0]_y}{Q[0]_x}$의 덧셈이 단순한 비트 XOR(bit-XOR)이므로 분석에 큰 영향을 주지 않으므로 통합하여 Mult로 표현하였다.

Coron의 알고리즘을 하드웨어로 구현한다하고 이 때 타원곡선의 주요 연산인 역원, 곱셈을 전용으로 담당하는 계산기를 각 각 1개씩 가진 형태로 구현한다면 앞서 말한 [그림 1]과 같은 이유로 소프트웨어에 대한 하드웨어의 이점인 parallel 성질을 이용하거나 pipeline 연산을 효과적으로 수행할 수 없게되어 소프트웨어로 구현했을 때 필요한 $2(n-1)$번의 타원곡선 연산 단계와 비슷한 계산량이 필요하다.

구체적으로, [표 1][1]은 데이터의 dependency와 연산자가 각 1개씩이라는 가정 하에 만들어진 연산표이다. 이 때 역원 연산과 곱셈 연산은

inversion cycle ≈ 4(multiplication cycle)

이라고 가정하자[4].



[그림 1] Coron방법의 dependence graph($GF(2^m)$경우)

1) 유한체의 덧셈과 뺄셈이 연산 시간에 영향을 줄 수 있지만 미미하므로 본 표에서는 제외한다.

〔표 1〕 Coron 알고리즘 1의 연산표

Iter-ation	cycle	Doubling $Q[0] \leftarrow 2Q[0]$	$Q[0]$	Addition $Q[1] \leftarrow Q[0]+P$	$Q[1]$
$i=n-2$	1c	Inversion			
	2c				
	3c				
	4c				
	5c	Multiplication			
	6c	Squaring	new $Q[0]_x$ of $Q[0]$		
	7c	Multiplication	new $Q[0]_y$ of $Q[0]$	Inversion	
	8c				
	9c				
	10c				
	11c			Multiplication	
	12c			Squaring	new $Q[1]_x$ of $Q[1]$
$i=n-3$	13c	Inversion		Multiplication	new $Q[1]_y$ of $Q[1]$
	14c				
	15c				
	16c				
	17c	Multiplication			
	18c	Squaring	new $Q[0]_x$ of $Q[0]$		
	19c	Multiplication	new $Q[0]_y$ of $Q[0]$	Inversion	

Coron이 제시한 방법은 하드웨어 구현에 효과적인 right-to-left 이진 방법으로 간단히 변형될 수 있고 다음과 같다.

Algorithm 2 of Coron
(Double-and-add resistant against SPA)

1. input P
2. $Q[0] \leftarrow P$, $Q[1] \leftarrow O$, $Q[2] \leftarrow O$
3. for i from 0 to $n-1$ do
3.1. $Q[2] \leftarrow Q[1]+Q[0]$
3.2. $Q[0] \leftarrow 2Q[0]$
3.3. $Q[1] \leftarrow Q[1+d_i]$
4. output $Q[1]$

알고리즘 2는 각 단계에서 알고리즘 1이 가지고 있는 데이터 dependency가 존재하지 않아 본 논문의 가정에 의해 분석해 볼 때 알고리즘 1에서는 두 번의 곱셈(한번의 곱셈과 한번의 제곱)이 끝난 후에 두 배에서 역원을 계산했던 것과는 달리 덧셈에서 역원계산을 끝내고 나면 바로 두배에서 역원계산을 수행할 수 있다.

그러나 [표 2]에서 설명한 것과 같이 알고리즘 2에서 이전 반복(iteration)에서 두배 연산이 끝나기 전에 덧셈에서 역원계산을 할 수가 없다.

〔표 2〕 Coron 알고리즘 2의 연산표

Iter-ation	cycle	Addition $Q[2] \leftarrow Q[1]+Q[0]$	$Q[2]$	Doubling $Q[0] \leftarrow 2Q[0]$	$Q[0]$
$i=0$	1c	Inversion			
	2c				
	3c				
	4c				
	5c	Multiplication		Inversion	
	6c	Squaring	new $Q[2]_x$ of $Q[2]$		
	7c	Multiplication	new $Q[2]_y$ of $Q[2]$		
	8c				
	9c			Multiplication	
	10c			Squaring	new $Q[0]_x$ of $Q[0]$
$i=1$	11c	Inversion		Multiplication	new $Q[0]_y$ of $Q[0]$
	12c				
	13c				
	14c				
	15c	Multiplication		Inversion	
	16c	Squaring	new $Q[2]_x$ of $Q[2]$		
	17c	Multiplication	new $Q[2]_y$ of $Q[2]$		
	18c				
	19c			Multiplication	

Ⅲ. 제안 알고리즘

여기서 제시하는 알고리즘은 기존에 right-to-left 이진방법으로 알려진 알고리즘을 변형한 것으로 다음과 같은 특징이 있다.

1. Coron과 같이 덧셈 연산을 각 단계에서 d_i 값에 관계없이 수행하여 SPA에 견딘다.

2. Coron의 알고리즘에서 발생하는 문제를 해결하여 동일한 하드웨어(곱셈기(multiplier) 1개, 역원기(inverter) 1개)로 속도 향상을 볼 수 있다.

[제안 algorithm]

1. input P
2. $Q[0] \leftarrow O$, $Q[1] \leftarrow O$, $R[0] \leftarrow O$, $R[1] \leftarrow P$
3. for i from 1 to n do
3.1. $R[0] \leftarrow R[1]$
3.2. $R[1] \leftarrow 2R[1]$

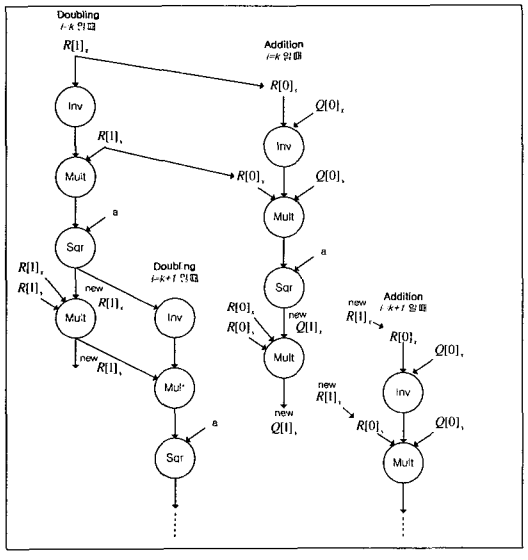
3.3. $Q[1] \leftarrow Q[0] + R[0]$

3.4. $Q[0] \leftarrow Q[d_{i-1}]$

4 output $Q[0]$

제안 알고리즘은 두배($R[1] \leftarrow 2R[1]$)와 덧셈($Q[1] \leftarrow Q[0] + R[0]$)의 데이터 dependency를 없애 Coron 알고리즘이 덧셈이 계산되기 위해서 두배의 값이 끝나기를 기다려야 하는 문제를 해결한 것이다. 이 때 for-loop이 1부터 시작되므로 단계 3.4.는 $i-1$값에 근거한 d_{i-1}에 의해 갱신된다.

Coron의 방법에 적용한 것과 동일한 유한체 연산 하드웨어(역원기 1개, 곱셈기 1개)를 사용할 때 Coron의 알고리즘에 비해 loop의 각 반복에서 4번의 곱셈 시간을 줄일 수 있다. 이는 추가 저장공간 $R[0]$를 두어 덧셈과 두배에서 각 각 다른 것(덧셈에는 $R[0]$, 두배에는 $R[1]$)을 사용하여 Coron 알고리즘 2에서 발생하는 두배의 제곱 연산이 끝난 후에 덧셈의 역원 계산을 해야하는 문제를 해결하였다. 이 때 알고리즘 입장에서는 각 반복의 첫 단계에서 $R[0]$의 갱신이 일어나야 하지만 [표 3]에서 보는 바와 같이 덧셈이 시작되기 전에 갱신한다.



〔그림 2〕 제안 알고리즘의 dependency graph

유한체 연산에서 역원 연산은 가장 많은 시간을 요하는 연산이다. [4]에서 구현된 역원 아키텍처는 곱셈이 $l$cycle에 이루어진다고 가정할 때 역원은 최소 $4l$ cycle에서 $6.2l$cycle이 소요된다. [4]에서 제시된 역원기를 사용한다면 역원이 진행될 때 최소 4번에서 6번의 곱셈을 할 수 있다. 따라서 두배 부분에서 역원이 실행되고 있을 때 덧셈에서 필요한 3번의 곱셈 연산(제곱 연산 포함)을 수행할 수 있다. [표 3]는 이런 효과를 이용했을 때 제안 알고리즘의 연산표이다. 이 표에 의하면 전체 스칼라 곱셈 알고리즘에서 3번의 곱셈 시간을 없앨 수 있다. [표 3]는 역원이 곱셈의 4배라고 가정했을 때의 연산단계이다.

〔표 3〕 제안 알고리즘의 연산표

loop	cycle	Doubling $R[1] \leftarrow 2R[1]$	$R[1]$	Addition $Q[1] \leftarrow Q[0] + R[0]$	$Q[1]$	$R[0]$
$i=1$	1c	Inversion				
	2c					
	3c					
	4c					$R[0] \leftarrow R[1]$ $(R[0] = P)$
	5c	Multiplication		Inversion		
	6c	Squaring	new $R[1]_x$ of $R[1]$			
	7c	Multiplication	new $R[1]_y$ of $R[1]$			
	8c					
$i=2$	9c	Inversion		Multiplication		
	10c			Squaring	new $Q[1]_x$ of $Q[1]$	
	11c			Multiplication	new $Q[1]_y$ of $Q[1]$	
	12c					$R[0] \leftarrow R[1]$ $(R[0] = 2P)$
	13c	Multiplication		Inversion		
	14c	Squaring	new $Q[0]_x$ of $Q[0]$			
	15c	Multiplication	new $Q[0]_y$ of $Q[0]$			
	16c					
$i=3$	17c	Inversion		Multiplication		
	18c			Squaring	new $Q[1]_x$ of $Q[1]$	

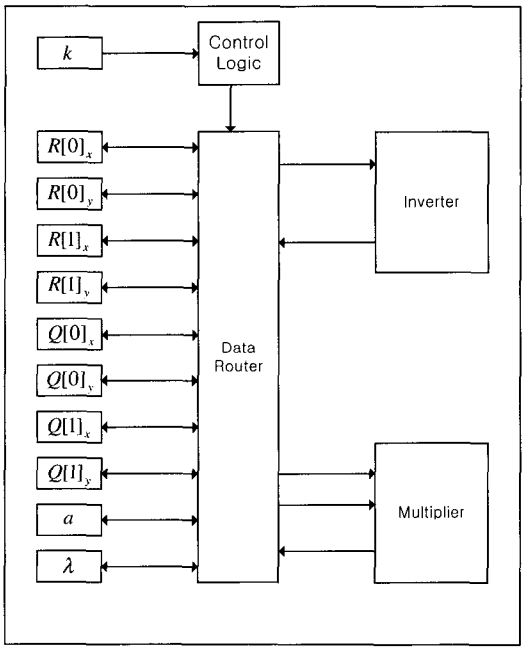
Ⅳ. 하드웨어 구조

본 논문에서 제안하는 하드웨어 구조는 크게 4가지로 분류된다. [그림 3]에서 보이는 것은 역원기(inverter), 곱셈기(multiplier), 데이터 라우터(data router) 그리고 데이터 메모리이다. 역원기, 곱셈기는 제시하는 알고리즘의 효과를 볼 수 있는, 즉 역원계산 동안 3번의 곱셈을 할 수 있는, 것은 어느 것이든 선택하여 사용할 수 있으며 각자 독자적으로 연산이 될 수 있도록 내부에 각 연산에 필요한 메모리를 가지고 있다[2].

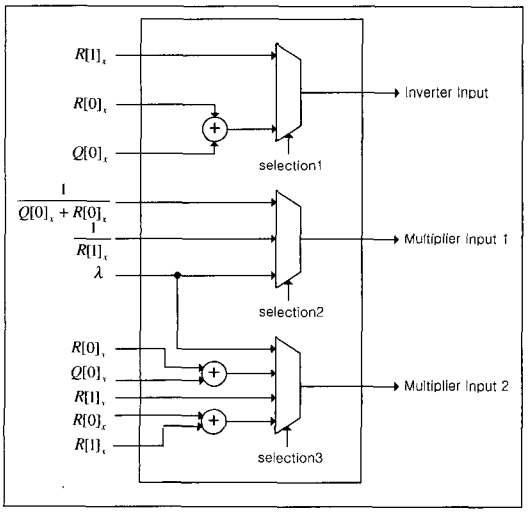
데이터 라우터는 [7]에서 정의된 덧셈과 두배에서 필요한 유한체의 XOR연산(field addition)과 역원기 및 곱셈기에 데이터를 전달하기 위한 조합논리(com-

2) 대표적으로 [4]과 [5]에서 제시한 역원기와 곱셈기를 사용할 수 있다.

binational logic)으로 이루어져 있다. [7]에 정의된 것처럼 역원기와 곱셈기에 입력되기 전에 XOR이 필요한 경우도 있고 그대로 데이터를 전달해야 하는 경우도 있으며 역원기와 곱셈기의 output을 데이터 메모리에 그대로 저장하거나 기존 값과 XOR 연산을 처리하여 메모리에 저장해야 할 때도 있다. [그림 4]은 데이터 라우터의 일부분으로 역원기와 곱셈기에 입력되는 값에 대한 로직이다.



〔그림 3〕 제안 알고리즘의 하드웨어 구조



〔그림 4〕 데이터 라우터 중에서 역원기 및 곱셈기 입력 부분

Ⅴ. 효율성 분석

먼저 곱셈은 c 클럭 사이클(clock cycle)이 소요되고 역원은 kc($k \geq 4$)이 필요하다고 가정하면 덧셈과 두배는 [7]에 의해 각 각 3번의 곱셈과 1번의 역원이 필요하므로 다음과 같다. 유한체의 덧셈이나 뺄셈 역시 타원곡선 덧셈과 두배 연산 시간에 영향을 주지만 유한체의 곱셈과 역원 계산에 비해 미미하므로 효율성 분석에서는 제외한다.

addition cycle	$3c+kc$
doubling cycle	$3c+kc$

이 때 Coron의 방법 중 효율적인 알고리즘 2를 적용한 것을 분석하면 [표 2]에 의해 덧셈에서 3번의 곱셈과 두배에서 한번의 곱셈 시간이 없어지므로

total computing cycles of Coron algorithm 2
$= n \cdot$ (addition cycle $- 3c$)
$+ n \cdot$ (doubling cycle $- c$)
$= n(2c+2kc)$ cycles

이다.

제안된 방법은 [표 3]에 의해 loop의 각 단계마다 오직 2번의 inversion 시간만 있고 마지막에 3번의 곱셈이 필요하므로

total computing cycles of the proposed
$= n(2kc)+3c$ cycles

이므로 Coron의 방법에 비해 $(2n-3)c$를 줄일 수 있다.

Ⅵ. 결 론

본 논문에서는 스칼라 곱셈 방법을 하드웨어로 구현할 때 SPA에 견디면서도 연산 속도의 문제를 해결한 알고리즘과 구조를 제시하였다.

제안된 알고리즘을 하드웨어를 구현할 때 SPA에 견디면서도 동일한 유한체 연산기(multiplier, inverter)를 사용한다는 가정 하에 Coron의 방법 보다 연산 속도가 빠른 스칼라 곱셈 방법과 구조를 가졌다. 논문에서 제시하는 하드웨어의 연산 속도는 $2n \cdot$ (In-

version cycle)+3(Multiplication cycle)만이 소요된다.

참 고 문 헌

[1] J. -S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," in *Cryptographic Hardware and Embedded Systems-CHES99*, Springer-Verlag, 1999.

[2] M. A. Hasan, "Power Analysis Attacks and Algorithmic Approaches to their countermeasure for Koblitz Curve Crypto-system", in *Cryptographic Hardware and Embedded Systems-CHES2000*, Springer-Verlag, 2000.

[3] Paul Kocher, Joshua Jaffe, and Benjamin Jun, "Differential Power Analysis", *Advances in Cryptology-Crypto'99*, Springer-Verlag, 1999.

[4] E. Savas and C. K. Koc, "Architectures for Unified Field Inversion with Applications in Elliptic Curve Cryptography", in *The 9th IEEE International Conference on Electronics, Circuits and Systems-ICECS 2002.*

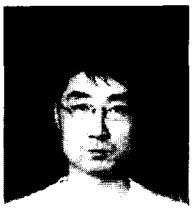
[5] E. Savas, A. F. Tenca, and C. K. Koc, "A scalable and unified multiplier architecture for finite fields $GF(p)$ and $GF(2^m)$", in *Cryptographic Hardware and Embedded Systems-CHES 2000,* Springer-Verlag, 2000.

[6] J. Solinas, "An Imporved Algorithm for Arithmetic on a Family of Elliptic Curves", *Advances in Cryptology-Crypto'97*, Springer-Verlag, 1997.

[7] IEEE P1363, "Annex A Number-Theoretic Background", in *Standard Specifications for Public Key Cryptography/D13,* IEEE, 2000.

[8] T. Izu and T. Takagi, "A fast parallel elliptic curve multiplication resistant against side channel attacks", in *Public Key Cryptography-PKC2002,* Springer-Verlag, 2002.

〈著者紹介〉



윤 중 철 (Yoon Joong Chul) 정회원
1993년 2월 : 고려대학교 수학과 졸업
1995년 8월 : 고려대학교 수학과 석사
1999년 2월 : 고려대학교 수학과 박사
1999년 3월~2002년 8월 : (주)시큐리티 테크놀로지스 선임연구원
2002년 3월~현재 : 고려대학교 정보보호대학원 조교수
<관심 분야> 부채널공격 방법론, 암호칩 설계

정 석 원 (Seok Won Jung) 정회원
1991년 2월 : 고려대학교 수학과 졸업
1993년 2월 : 고려대학교 수학과 석사
1997년 2월 : 고려대학교 수학과 박사
2002년 3월~현재 : 고려대학교 정보보호대학원 조교수
<관심 분야> 암호칩 설계, 부채널공격 방법론, 공개키 암호알고리즘, 디지털 방송 보안

임 종 인 (Jong-in Lim) 정회원
1980년 2월 : 고려대학교 수학과 학사
1982년 2월 : 고려대학교 수학과 석사
1986년 2월 : 고려대학교 수학과 박사
1999년 2월~현재 : 고려대학교 자연
과학대학 정교수, 한국통신정보보호학회 편집위원장
고려대학교 정보보호대학원 원장, 고려대학교 정보보호기술연구센터 센터장
<관심분야> 블록 암호 및 스트림 암호의 분석 및 설계. 암호 프로토콜, 공개키 암호 분석, 정보보호정책.