GF(2m)에서의 사칙연산을 수행하는 GFAU의 설계

준회원 김 문 경, 정회원 이 용 석

Design of a GFAU(Galois Field Arithmetic Unit) in GF(2m)

Moon-Gyung Kim Associate Member, Yong-Surk Lee Regular Member

요 약

본 논문에서는 GF(2m) 상에서의 ECC 암호화 알고리즘을 지원하기 위한 GFAU(Galois Field Arithmetic Unit)의 구조를 제안한다. GFAU는 GF(2m)상에서의 덧셈, 곱셈, 나눗셈을 수행하며 동시에 두 개의 덧셈이나 두 개의 곱셈, 또는 하나의 덧셈과 하나의 곱셈을 동시에 처리할 수 있는 능력을 가지고 있다. 기본 구조는 변형된 유클리드 알고리즘의 나눗셈기를 기반으로 제안되었으며, 이 기본구조에 곱셈기 및 덧셈기의 기능을 추가하여 제어부와 함께 구현되었다. GF(2193)을 위한 GFAU는 Verilog-HDL를 이용하여 하향식설계방식으로 구현되었고 C-언어로 작성된 사이클 단위 시뮬레이터를 이용하여 개선되고 검증되었다. 검증된 모델은 삼성 0.35um, 3.3V CMOS 표준 셀 라이브러리로 합성되었으며 최악조건 3.0V, 85℃에서 104.7㎒의 주파수에서 동작하며, 전체 게이트 수는 약 25,889이다.

ABSTRACT

This paper proposes Galois Field Arithmetic Unit(GFAU) whose structure does addition, multiplication and division in GF(2m). GFAU can execute maximum two additions, or two multiplications, or one addition and one multiplication. The base architecture of this GFAU is a divider based on modified Euclid's algorithm. The divider was modified to enable multiplication and addition, and the modified divider with the control logic became GFAU. The GFAU for GF(2193) was implemented with Verilog HDL with top-down methodology, and it was improved and verified by a cycle-based simulator written in C-language. The verified model was synthesized with Samsung 0.35um, 3.3V CMOS standard cell library, and it operates at 104.7㎒ in the worst case of 3.0V, 85℃, and it has about 25,889 gates.

Ⅰ. 서 론

네트워크 기술의 발달로 현대사회는 유무선 네트워크와 같은 거대한 공통매체로 정보를 공유하며 이런 정보기술에는 필요에 따라 정보를 암호화하여 보호하며, 개인의 정보를 인증해 줄 수 있는 IC카드와 같은 정보 보호 기술이 반드시 필요하게 되었다.[1] 이러한 정보보호기술은 암호학적인 알고리즘에 의해 복잡한 연산을 거쳐 이루어지는데, 이 암호기술은 크게 대칭키 암호 시스템과 공개키 암호 시스템이라는 두 가지로 나누어진다.[2] 대칭키 암호 시스템이란 암호화키와 복호화키가 같은 암호시스템에 사용된 암호 알고리즘을 의미하는 것으로, 구현하기가 비교적 쉽고 동작속도도 빨라 대용량의 정보를 암호화/복호화하여 교환할 수 있으나, 사전에 양자간에 비밀리에 키를 공유하고 있어야만 하는 단점이 있어 키의 안전한 보관 및 분배가 이 암호 시스템의 안전성과 밀접한 관련이 있게 된다. 공개

* 연세대학교 전기전자공학과 Processor 연구실 (bungae@yonsei.ac.kr)
논문번호 : #020454-1017, 접수일자 : 2002년 10월 18일
※ 본 연구는 2002년도 하이닉스 반도체의 연구비 보조에 의하여 연구되었음.

키 암호 시스템은 암호화키와 복호화키가 다르게 작성하고, 공개키는 누구나 접근할 수 있도록 하는 반면, 비밀키는 안전하게 보관하는 방식을 취하게 된다. 이를 통해 사전에 키를 분배할 필요가 없고, 관리할 키의 개수가 적어지며 디지털 서명이 가능하다는 장점을 가지게 된다.[3]

공개키 알고리즘은 공개키가 누구나 접근이 가능한 방식을 쓰기 때문에 비밀키의 안전도를 높이기 위해 복잡한 수학적인 연산을 필요로 하게 되는데, 1978년 소인수 분해의 어려움에 기반을 둔 RSA가 소개되어 지금까지도 널리 사용되고 있으며, 현재 타원곡선 암호화 알고리즘을 이용한 타원곡선암호화 시스템(ECC)의 실용화가 새로이 거론되고 있다.[4]

본 논문에서는 GF(2m)상에서 ECC의 구현을 위해 많이 알려진 직렬곱셈기와 나눗셈기의 구조를 분석하여, 덧셈기 기능을 포함하는 연산유닛 회로를 설계, 검증하며, 나아가 이들을 응용할 수 있는 EC C 암호화 알고리즘의 구현에 대한 base를 제안한다. 제안된 연산유닛은 GF(2m)상에서 사칙연산을 수행할 수 있기 때문에 Galois Field Arithmetic Unit(G FAU)이라고 칭하도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 GF(2 m)상의 연산에 대해, 3장에서는 본 GFAU의 구현에 대하여, 4장에서는 설계된 구조를 합성 및 비교하고 5장에서 결론을 짓는다.

Ⅱ. GF(2m)상의 연산

유한체, 또는 Galois field란 암호 이론이나 부호 이론에서 주로 사용되는 원소의 개수가 유한인 체(0에 의한 나눗셈을 제외하고는 사칙연산에 대해 닫혀있는 군)를 말한다. p를 소수라고 하면 p개의 원소로 되는 유한체가 존재한다. 이것을 GF(p)로 쓰고 소체(prime field)라 부른다. 이의 표현방식은 {0, 1, …, p-1}이 된다. 마찬가지로, GF(2)상에서의 모든 연산은 modulo 2에 기반 하여 수행되며, 이의 표현방식은 {0, 1}이 된다. 이러한 체 F를 기반으로 하는 다항식 p(x)가 기약 다항식(irreducible polyno mial)이 된다면, 이는 p(x)를 모듈러로 하는 체 F위의 다항식 환의 잉여류 환은 체가 된다는 것이 증명되어 있다.[5] 본 논문에서 다루는 체 F는 GF(2 m)으로 GF(2)를 m으로 확장시킨 형태를 가진다. 이 체 위의 소수 p(x)를 field generator로 사용하게 되며, 수식 1과 같은 꼴을 가진다.

$$P(x)=x^m+\sum_{i=0}^{m-1}p_ix^i \tag{1}$$

또한, GF(2m)상에서의 모든 원소들은 수식 2와 같은 다항식 형태를 가지게 된다.

$$A(x)=\sum_{i=0}^{m-1}a_ix^i \tag{2}$$

GF(2m)상에서의 덧셈은 각 벡터들 간의 합으로 나타나게 되는데, 모듈로 2에 대한 연산을 수행하게 되므로 1 또는 0의 값을 가지게 되어 배타적 논리합(exclusive-OR)연산으로 간단하게 대치될 수 있다.

$$\begin{aligned}A(x)+B(x) &= \sum_{i=0}^{m-1}a_ix^i+\sum_{i=0}^{m-1}b_ix^i\\ &= \sum_{i=0}^{m-1}(a_i+b_i)x^i\\ &= \sum_{i=0}^{m-1}(a_i\oplus b_i)x^i\end{aligned} \tag{3}$$

곱셈의 경우에는 다음과 같이 풀어나가게 되는데, 그 순서에 따라 구현되는 모습이 달라진다.

$$\begin{aligned}&A(\alpha)\times B(\alpha)\\ &=A(\alpha)\sum_{i=0}^{m-1}b_i\alpha^i\\ &=b_0A+b_1A\alpha+\cdots+b_{m-1}A\alpha^{m-1}\\ &=(\cdots(b_{m-1}A\alpha+b_{m-2}A)\alpha+\cdots+b_1A)\alpha+b_0A\end{aligned} \tag{4}$$

여기서, P(x)의 근이 α일 경우, P(α)=0이라는 성질을 이용하여 αA(α)의 값을 구할 수 있고 이는 수식 5와 같이 된다.

$$\begin{aligned}&P(\alpha)=\alpha^m+\sum_{i=0}^{m-1}p_i\alpha^i=0\\ &\alpha^m=\sum_{i=0}^{m-1}p_i\alpha^i\\ &\alpha A(\alpha)\\ &=a_0\alpha+\cdots+a_{m-2}\alpha^{m-1}+a_{m-1}\alpha^m\\ &=a_0\alpha+\cdots+a_{m-2}\alpha^{m-1}\\ &\quad+a_{m-1}(p_0+p_1\alpha+\cdots+p_{m-1}\alpha^{m-1})\\ &=a_{m-1}p_0+(a_0+a_{m-1}p_1)\alpha+\cdots\\ &\quad+(a_{m-2}+a_{m-1}p_{m-1})\alpha^{m-1}\end{aligned} \tag{5}$$

나눗셈의 경우에는 확장된 유클리드 알고리즘을 통하여 구하게 되는데, 이때는 표 1과 같은 순서로 구한다.

표1. 확장된 유클리드 알고리즘을 이용한 나눗셈
Table 1. Division by extended Euclidean algorithm

```
R=B(α); S=P=P(α); U=A(α), V=0;
count=0;
for i=1 to 2m do
  if r_m ==0 then
    R=αR; U=αU mod P;
    count=count+1;
  else
    if s_m==1 then
      S=S-R; V=V-U;
    end
    S=αS;
    if count==0 then
      R↔S; U↔V;
      U=αU mod P;
      count=count+1;
    else
      U=U/α mod P;
      count=count-1;
    end
  end
end
```

이 경우에는 U(α)/α의 값을 구해야 하는데, 여기서는 P(x)의 특성상 p0의 값이 항상 1이 되는 점에 착안해서 수식 6과 같은 변형을 통하여 구해준다.

$$P(\alpha)=\alpha^m+\sum_{i=1}^{m-1}p_i\alpha^i+1=0$$
$$1=\alpha^m+\sum_{i=1}^{m-1}p_i\alpha^i$$
$$\frac{1}{\alpha}=\alpha^{m-1}+\sum_{i=1}^{m-1}p_i\alpha^{i-1} \qquad (6)$$

여기서, U(α)/α의 값을 구하면 수식 7과 같이 된다.

$$\frac{U(\alpha)}{\alpha}$$
$$=u_{m-1}\alpha^{m-2}+\cdots+u_1+\frac{u_0}{\alpha}$$
$$=u_{m-1}\alpha^{m-2}+\cdots+u_1+u_0(\alpha^{m-1}+p_{m-1}\alpha^{m-2}+\cdots+p_1)$$
$$=u_0\alpha^{m-1}+(u_0p_{m-1}+u_{m-1})\alpha^{m-2}+\cdots+(u_0p_1+u_1) \qquad (7)$$

나눗셈기에서 쓰이는 연산을 정리해 보면 표2와 같으며, αA(α)나 A(α)/α 같은 연산들이 주로 쓰이고 있음을 알 수 있다.

표2. 나눗셈에서 매 사이클마다의 연산
Table 2. Operation per cycle in division

r_m	s_m	count	R	S	U	V	count
0	X	X	αR	S	αU	V	count+1
1	0	0	αS	R	αV	U	count+1
1	0	≥1	R	αS	U/α	V	count-1
1	1	0	α(S-R)	R	α(V-U)	U	count+1
1	1	≥1	R	α(S-R)	U/α	V-U	count-1

Ⅲ. GFAU의 구현

이전 단원에서 설명된 나눗셈기의 연산에는 덧셈기와 곱셈기에 쓰이는 연산 또한 처리할 수 있는 능력을 가지고 있다. 첫째로, V와 U에 초기값을 넣고 rm, sm, count에 각각 1을 넣어주게 되면 V가 저장되는 셀에 가산결과가 넣어진다. 또한, R에 A(α)를 넣고 계속 αA(α)를 수행하게 하고, B(α)의 해당 벡터 값들에 따라 제어신호를 생성하여 R값과 더하도록 하는 회로를 추가하면 곱셈 결과가 S에 기록되게 할 수 있다. 이 경우 B(α)를 저장하는 레지스터가 추가되는데, 이를 M1, M2라고 하자.

이전 단원에 나온 표에서 나온 연산을 수정하여 곱셈 및 나눗셈 모드에서의 연산을 모두 수행할 수 있도록 회로를 구현하면 그림 1, 그림2와 같이 나오게 된다. 표 3과 표 4는 연산에 따라 그림 1과 2의 회로의 MUX에 들어갈 제어신호 리스트를 나타내준 것이다. 덧셈의 경우에는 rs_out이나 uv_out의 출력을 사용하고, 곱셈의 경우에는 rs_out과 xs_out, uv_out과 xv_out의 출력을 사용하여 M1, M2의 shift 입력에 맞추어 수행해 준다. 나눗셈의 경우에는 나눗셈에 쓰이는 각 cell마다 출력을 선별하여 그 값을 받아들이도록 한다. 단, R과 S의 경우에는 xR 또는 xS의 연산 시에 모듈로 곱셈을 사용하지 않기 때문에 단지 shift동작만 일어나는 것으로 대치된다. 또한, rm과 sm에 대한 저장 공간을 보장하여주고, 여기 저장된 값을 통하여 나눗셈의 제어신호 생성에 도움을 주도록 한다. R의 경우에는 r_out, xr_out, xs_out, rs_out 중에서 선택하고, S의 경우에는 r_out, xs_out, rs_out 셋 중에서 선택하도록 한다. U의 경우에는 u_out, xu_out, xv_out, uv_out에서 선택하고, V의 경우에는 u_out, xv_out, uv_out에서 선택하도록 구현한다.

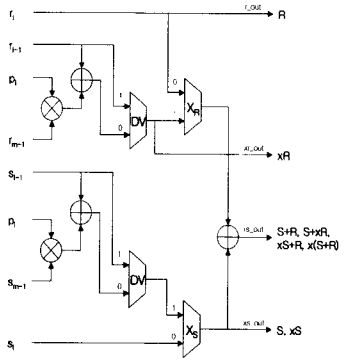


그림1. RS 연산회로
Fig. 1. RS calculation circuit

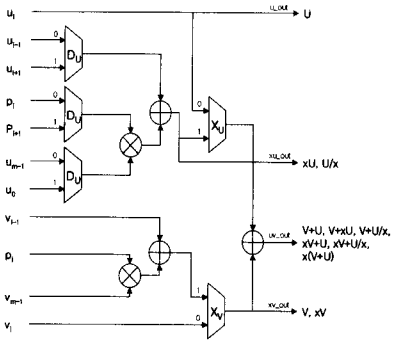


그림2. UV 연산회로
Fig. 2. UV calculation circuit

표3. 제어 신호에 따른 RS 연산회로의 출력
Table 3. RS circuit outputs by control signals

Operation	DV	XR	XS	r_out	xr_out	xs_out	rs_out
Addition	X	0	0	R	xR	S	S+R
Multiplication	0	0	0	R	xR	S	S+R
Division	1	1	0	R	xR	S	S+xR
	1	0	1	R	xR	xS	xS+R
	1	0	1	R	xR	xS	xS+R
	1	1	1	R	xR	xS	x(S+R)
	1	1	1	R	xR	xS	x(S+R)

표4. 제어 신호에 따른 UV 연산회로의 출력
Table 4. UV circuit outputs by control signals

Operation	DU	XU	XV	u_out	xu_out	xv_out	uv_out
Addition	X	0	0	U	xU	V	V+U
Multiplication	0	0	0	U	xU	V	V+U
Division	0	1	0	U	xU	V	V+xU
	0	0	1	U	xU	xV	xV+U
	1	1	0	U	U/x	V	V+U/x
	0	1	1	U	xU	xV	x(V+U)
	1	0	0	U	U/x	V	V+U

RS 연산회로와 UV 연산회로는 서로 독립적으로 동작시킬 수 있기 때문에, 서로 다른 연산을 수행할 수 있다. 나눗셈을 수행해야 할 경우를 제외한다면, RS 연산회로는 R, S, M1에 저장된 값을 통해 덧셈이나 곱셈을 수행하고, UV 연산회로는 U, V, M2에 저장된 값을 통해 독립적인 다른 연산을 수행할 수 있다. 이 둘을 독립적으로 제어해 주게 될 경우, 동시에 2개의 덧셈이나 2개의 곱셈, 또는 하나의 곱셈을 수행하는 동안 1회 이상의 덧셈이나 load/store를 수행해 줄 수 있다. 또한, 구현된 곱셈 연산회로는 승산누적(Multiply-and-Accumulate)형태의 연산을 지원하기 때문에 덧셈을 포함하고 있다. 이와 같이 구현된 GFAU는 그림 3과 같이 나타낼 수 있다.

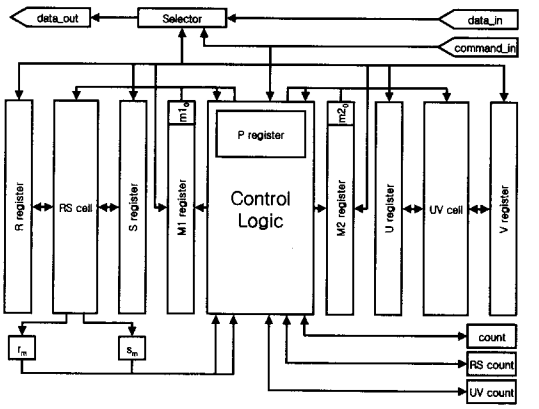


그림3. GFAU의 블록다이어그램
Fig. 3. Block diagram of GFAU

ECC에서 사용하는 Point Double의 경우[6] 기존의 방식을 사용할 경우 1회의 나눗셈과 3회의 곱셈(2회의 제곱), 5회의 덧셈, 16회의 load/store가 필요하다. 본 논문에서 제안된 GFAU를 사용할 경우에는 표 5에서 보는 바와 같이 기존의 방식보다 1회의 곱셈과 2회의 덧셈, 2회의 load/store가 줄어든 1

회의 나눗셈, 2회의 곱셈, 3회의 덧셈, 14회의 load/store로 수행이 가능하다.

표5. GFAU를 사용한 GF(2^m) ECC상에서의 Point Double
Table 5. Point Double on GF(2^m) ECC using GFAU

Load R,x_1 Load S,p(x) Load U,y_1 Load V,0		x_1	p(x)	y_1	0	?	?	4 load	
Div RSUV		?	?	y_1/x_1	?	?	?	1 div	
Load V,x_1		?	?	y_1/x_1	x_1	?	?	1 load	
Add UV		?	?	$\lambda=x_1+y_1/x_1$	x_1	?	?	1 add	
Store U,tmp_1		?	?	λ	x_1	?	?	1 store	
Load V,tmp_1 Load M2,tmp_1		?	?	λ	λ	?	λ	2 load	
Mul UVM2	Load R, x_1	x_1	?	?	$\lambda^2+\lambda$	?	0	3 load	3 load
Load U,a	Load M1,x_1	x_1	?	a	$\lambda^2+\lambda$	x_1	0	1 store	1 store
Add UV	Load S,0	x_1	0	$x_2=\lambda^2+\lambda+a$	$\lambda^2+\lambda$	x_1	0	2 mul	1 mul
Store U,tmp_2	Mul RSM1	?	x_1^2	x_2	$\lambda^2+\lambda$	0	0	1 add	
Load V,tmp_2	Store S,tmp_1	?	x_1^2	x_2	x_2	0	0		
Load M2,tmp_1				x_2	x_2		λ		
Mul UVM2				?	λx_2+x_2		0		
Load U, tmp_3		?	x_1^2	x_1^2	λx_2+x_2	0	0	1 load	
Add UV		?	x_1^2	$y_2=x_1^2+\lambda x_2+x_2$	λx_2+x_2	0	0	1 add	
Store U,tmp_4		?	x_1^2	y_2	λx_2+x_2	0	0	1 store	

Ⅳ. 합성결과

GFAU는 Verilog-HDL언어로 하향식설계방식으로 설계되었다. 이의 검증은 먼저 각각의 목표 블록에 대한 입출력신호로 테스트 벡터를 가하여 진행되었다. 또한 C언어를 통해 GFAU와 흡사한 동작을 수행할 수 있도록 사이클 단위로 중간 수행 단계를 확인할 수 있는 시뮬레이터가 설계되었고 전체 프로세서의 동작을 검증하기 위해 이 시뮬레이터와의 중간 출력 및 최종 출력 비교가 이루어졌다. 각 명령어에 대한 전체 프로세서의 동작을 검증하였다.

본 논문에서 제안된 유닛의 구성은 다음과 같다. 메모리를 제외한 논리연산 유닛만의 크기로 볼 때, 전체적인 연산 유닛의 크기는 배타적 논리합 게이트 6m개, 논리곱 게이트 4m개, 2입력 MUX 9m개가 된다. R, S, U, V값을 저장하는데 4m+2개의 D-flipflop이 쓰이게 되고, 선별적으로 입력을 받게 되므로 이에 필요한 3입력 MUX 2m개, 4입력 MUX 2m개가 추가된다. 추가로, 곱셈 수행 시에 B(x)의 계수를 저장하고 매 사이클마다 shift를 수행해 주어 곱셈제어용 신호를 생성시켜주기 위한 M1, M2를 위해 2m개의 D-flipflop을 필요로 하게 된다.

메모리에서 레지스터로 값을 읽고 쓰는데 걸리는 시간을 s라고 가정할 경우, 덧셈의 경우에는 3s+1, 곱셈의 경우 m+4s, 나눗셈의 경우 2m+5s 사이클이 소요된다. 허나, 실제적인 ECC연산을 수행할 경우 이미 처리한 결과를 다시 수행하기 때문에 실제적인 데이터 이동 횟수가 줄어들게 된다. 작성된 프로그램을 통해 계산해 본 결과, ECC에서 쓰이는 Point Addition[6]의 경우에는 4m+20s+5 사이클, Point Double의 경우에는 4m+14s+3 사이클이 소요된다. 이 결과를 다른 논문의 구현 결과와 비교하면 표 6과 같이 나온다.

표6. ECC 연산 속도 비교
Table 6. Comparison of ECC Calculation Speed

	GFAU GF(2^m)	Rosner[7] GF((2^n)^m)	Hasan[8] GF(2^m)
Basis	Polynomial	Polynomial	Dual
Point Addition	4m+20s+5	12m+21	10m
Point Multiplication	4m+14s+3	14m+33	11m

제안된 회로는 최종적으로 193bit를 기준으로 제어부와 함께 구현되었으며, m≤193을 만족하는 GF(2m) 상의 연산을 수행할 수 있도록 설계되었다. 이의 합성결과는 표 7과 같다. 참고로, 비슷한 성능을 가지는 직렬곱셈기의 경우 p(x)=x193+x15+1만을 사용하도록 특화시켜 구현하면 약 7000 게이트의 크기를 가지며[9] 나눗셈기의 경우 본 논문에서 설계한 회로에서 곱셈에서의 B(x) 벡터 값을 저장하기 위한 M1 및 M2 메모리 영역이 없다는 점을 제외하면 거의 동일한 회로구성을 가지기 때문에 약 21000~22000 게이트를 필요로 한다. 본 회로는 위에 언급한 직렬곱셈기와 나눗셈기를 따로 구현하여 사용하는 경우보다 면적이 약 3천 게이트 이상 작고, 더 효율적으로 연산을 수행할 수 있는 구조를 가진다.

표7. 제안된 회로의 합성 결과
Table 7. Synthesis result of the proposed circuit

사용된 라이브러리	삼성 0.35um 공정
게이트 수	약 25,889 (Interconnection area포함)
임계경로	최악조건에서 9.55ns (3.0V 85℃)
동작 주파수	104.712 ㎒
연산별 동작 속도 (load시간 제외)	- 덧셈 1 사이클 * 2개 - 곱셈 m=193 사이클 * 2개 - 나눗셈 2m=386 사이클

V. 결 론

본 논문에서는 GF(2m)에서의 ECC 암호화 알고리즘을 지원하기 위한 연산유닛인 GFAU를 구현하였다. 구현된 GFAU는 GF(2m)상에서의 기본 연산인 덧셈, 곱셈, 나눗셈을 어떠한 인수를 가지는 기약 다항식에 대해서라도 수행할 수 있으며, 동시에 2개의 덧셈, 2개의 곱셈 또는 1개의 덧셈과 1개의 곱셈을 수행할 수 있는 능력을 가진다. 최종적으로 삼성 0.35um 공정을 사용하여 합성한 결과로 동작주파수는 약 104.7㎒가 나왔고 전체 회로의 크기는 제어부를 포함하여 25,889게이트가 된다.

참고문헌

[1] 김철, 암호학의 이해i, 영풍문고, 1996

[2] B. Schneier, Applied Cryptography second edition, John Wiley & Sons, Inc., 1998

[3] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, Handbook of A pplied Cryptography, CRC press, pp. 15-45, 1997

[4] J. Guajardo and C. Paar, "Efficient Algorithms for Elliptic Curve Crypto systems," Advances in Cryptology - CRYPTO 97, B.S Kaliski, ed., pp.342 -356, 1997

[5] 전자통신연구원, 암호학의 기초, 경문사, pp. 25-28, 1999년 3월

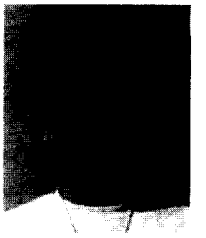
[6] CERTICOM Homepage, ECC Tutori al, http://www.certicom.com/resource s/ecc_tutorial/ecc_tutorial.html, 2001

[7] M. Rosner, Elliptic Curve Cryptosys tems on reconfigurable hardware, M aster's Thesis, Worcester Polytechni c Institute, Worcester, USA, 1998

[8] M.A. Hasan and A.G. Wassal, "VL SI Algorithms, and Implementation of a Versatile GF(2m) Processor," I EEE Trans. Computers, vol. 49, no.1 0, pp.1064-1073, Oct. 1997

[9] 이광엽, "ECC알고리즘에 기반한 공개키 암호화 프로세서 설계", 산업기술연구소 논문집, 서경대학교, pp.235-242, 2000년 6월

김 문 경(Moon-Gyung Kim) 준회원



1997년 2월 : 연세대학교 전자공학과 졸업

1999년 2월 : 연세대학교 전자공학과 석사

1999년 3월~현재 : 연세대학교 전기전자공학과 박사과정

<관심분야> 마이크로프로세서 설계, SMT, 영상처리, 암호화

이 용 석(Yong-Surk Lee) 정회원

1973년 2월 : 연세대학교 전기공학과 졸업

1977년 2월 : University of Michigan Ann Arbor 석사

1981년 3월 : University of Michigan Ann Arbor 박사

1993년~현재 : 연세대학교 전기전자공학과 교수 재임 중

<관심분야> 마이크로프로세서 설계