情報保護學會論文誌
第18卷 第3號, 2008. 6

페어링 기반 암호시스템의 효율적인 유한체 연산기*

장 남 수[1*†], 김 태 현[1*], 김 창 한[2‡], 한 동 국[3**], 김 호 원[4**]

[1]고려대학교 정보경영공학전문대학원, [2]세명대학교, [3]한국전자통신연구원,
[4]부산대학교 공과대학 정보컴퓨터공학부

Efficient Finite Field Arithmetic Architectures for Pairing Based Cryptosystems*

Nam Su Chang[1*†], Tae Hyun Kim[1*], Chang Han Kim[2‡], Dong-Guk Han[3**], Ho Won Kim[4**]

[1]Graduate School of Information Management and Security, Korea University,
[2]School of Information & Communication systems, Semyung University,
[3]Electronics and Telecommunications Research Institute,
[4]School of computer science and engineering in Pusan National University

요 약

페어링 기반의 암호시스템의 효율성은 페어링 연산의 효율성에 기반하며 페어링 연산은 유한체 $GF(3^m)$에서 많이 고려된다. 또한 페어링의 고속연산을 위하여 삼항 기약다항식을 고려하며 이를 기반으로 하는 하드웨어 설계방법에 대한 연구가 활발히 진행되고 있다. 본 논문에서는 기존의 $GF(3)$ 연산보다 효율적인 새로운 $GF(3)$ 덧셈 및 곱셈 방법을 제안하며 이를 기반으로 새로운 $GF(3^m)$ 덧셈-뺄셈 unified 연산기를 제안한다. 또한 삼항 기약다항식을 특징을 이용한 새로운 $GF((p^m)$ MSB-first 비트-직렬 곱셈기를 제안한다. 제안하는 MSB-first 비트-직렬 곱셈기는 기존의 MSB-first 비트-직렬 곱셈기보다 시간지연이 대략 30%감소하며 기존의 LSB-first 비트-직렬 곱셈기보다 절반의 레지스터를 사용하여 효율적이며, 제안하는 곱셈 방법은 삼항 기약다항식을 사용하는 모든 유한체에 적용가능하다.

ABSTRACT

The efficiency of pairing based cryptosystems depends on the computation of pairings. pairings is defined over finite fileds $GF(3^m)$ by trinomials due to efficiency. The hardware architectures for pairings have been widely studied. This paper proposes new adder and multiplier for $GF(3)$ which are more efficient than previous results. Furthermore, this paper proposes a new unified adder-subtractor for $GF(3^m)$ based on the proposed adder and multiplier. Finally, this paper proposes new multiplier for $GF(3^m)$. The proposed MSB-first bit-serial multiplier for $GF(p^m)$ reduces the time delay by approximately 30 % and the size of register by half than previous LSB-first multipliers. The proposed multiplier can be applied to all finite fields defined by trinomials..

Keywords : Pairing based cryptosystems, Finite Field Arithmetic, Bit-Serial Multiplier

Ⅰ. 서 론

최근 페어링(pairing)의 겹선형성(bilinearity)과 같은 특성들이 새로운 암호시스템에 사용되면서 Weil 페어링과 Tate 페어링 등의 페어링의 효율성에 대한 관심과 연구가 증가하고 있다. 페어링을 효율적으로 계산하기 위한 알고리즘은 Miiler에 의해서 처음 제안되었다[14]. 이후 Barreto 외 3인[2]과 Galbraith 외 2인[8]은 표수가 2인 유한체 위에서 정의된 초특이 타원 곡선(super-singula elliptic curve)에서 Tate 페어링의 계산 속도를 향상시킬 수 있는 방법을 제안하였다. Duursma와 Lee[7]는 표수가 3인 경우에 닫힌 공식(closed formula)를 제안하였고 Kwon[13]은 표수가 2인 초특이 타원 곡선으로 Duursma-Lee 방법을 확장하였다. Barre -to외 3인[1]은 초특이 abelian variety에서 페어링을 효율적으로 계산하기 위한 일반적인 접근 방법인 η_T 페어링을 제안하였다.

최근 페어링을 구현하기 위한 많은 방법이 제안되었다[3,5,9,10,11,13]. 페어링 구현은 소프트웨어뿐만 아니라 하드웨어에서 다양한 방법으로 진행되고 있지만 병렬기법과 pipelining 기법 등 속도를 향상시킬 수 있는 다양한 방법이 있기 때문에 하드웨어 구현에 대한 연구가 좀 더 활발히 진행되고 있다. 타원곡선 암호시스템에서 사용되는 유한체 $GF(p)$와 $GF(2^m)$에 대한 하드웨어 연구는 많이 진행되었다. 하지만 페어링 기반 암호시스템의 효율성을 높이기 위한 Duursma-Lee[7]알고리즘과 η_T 페어링[1]등의 사용으로 인하여 유한체 $GF(3^m)$에 대한 하드웨어 구현 방법이 필요하게 되었다. 다항식 기저(polynomial basis)를 이용한 $GF(3^m)$ 연산 방법들은 [4,12,15]에서 연구되었고 [10]에서는 정규 기저를 이용한 방법에 대하여 고려하였다. [15]에서 제안된 덧셈 방법은 $GF(3)$의 원소를 두 비트로 표현하였고 3 XOR와 4 OR 게이트를 이용하여 $GF(3)$에서 덧셈을 할 수 있는 방법을 제안하였고 곱셈의 경우 4 AND와 2 OR 게이트를 이용하여 구성하였다[10] .

본 논문에서는 [10]의 $GF(3)$ 덧셈 및 곱셈기가 아닌 새로운 $GF(3)$ 덧셈 및 곱셈기를 제안하며 이를 기반으로 $GF(3^m)$ 덧셈-뺄셈 unified 연산기를 제안한다. 또한, 페어링 기반 암호시스템의 효율성을 높이기 위하여 삼항 기약다항식이 많이 사용되는 것에 착안하여 기존의 MSB-first 및 LSB-first 비트-직렬 곱셈기보다 효율적인 새로운 $GF(p^m)$ MSB-first 비트-직렬 곱셈기를 제안한다. 제안하는 MSB-first 비트-직렬 곱셈기는 기존의 MSB-first 곱셈기에 비하여 시간지연에서 효율적이며, 기존의 LSB-first 비트-직렬 곱셈기보다 50%의 레지스터를 사용한다.

본 논문의 구성은 다음과 같다. 2절에서는 페어링 연산에 대하여 기술하며 3절에서는 기존의 $GF(3^m)$ 연산인 $GF(3)$ 연산 방법과 $GF(p^m)$ 비트-직렬 곱셈기에 대하여 설명한다. 4절에서는 새로운 $GF(3)$ 덧셈 및 곱셈기와 $GF(3^m)$ 덧셈-뺄셈 unified 연산기 및 $GF(p^m)$ MSB-first 비트-직렬 곱셈기를 제안한다. 5절에서는 제안하는 방법과 기존의 방법을 복잡도를 비교하고 결론을 내린다.

Ⅱ. 수학적 배경

유한체 $GF(q)=GF(p^m)$는 위수(characteristi c)가 p인 유한체라 하자. $GF(q)$에서 정의된 타원곡선 E의 점들의 군을 $E(GF(q))$에 의해서 표기하자. 소수 r이 q^k-1을 나누고 $0<i<k$에 대하여 q^i-1를 나누지 않는다면 위수가 r인 $E(GF(q))$의 부분군(subgroup)은 security parameter k를 갖는다. 위수가 r인 부분군은 r-torsion 군이고 $E(GF(q))[r]$로 표기한다. divisor는 $E(GF(q^k))$에 있는 점들의 formal sum이다. divisor $D=\sum n_P(P)$의 degree는 $\sum n_P$이다.

$f: E(GF(q^k))\rightarrow GF(q^k)$는 E에서 rational function이고 f의 divisor는 $(f)=\sum \mathrm{ord}_P(f)(P)$와 같이 정의된다. $\mathrm{ord}_P(f)$는 점 P에서 f의 zero 또는 pole의 order이다. 만일 f가 zero와 pole을 갖지 않는다면 $\mathrm{ord}_P(f)=0$이다. divisor D에 대하여 $D=(f)$를 만족하는 f가 존재하면 D는 principal divisor라 한다. $D=\sum n_P(P)$가 principal이기 위한 필요충분조건은 $\sum n_P=0$과 $\sum [n_P](P)=O$를 만족하는 것이다. O 는 무한원점(point at infinity)을 의미한다. 두 divisor의 차이 D_1-D_2가 principal divisor이

접수일 : 2007년12월 16일; 채택일 : 2008년 3월 5일
* "본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음" (IITA-2008-(C1090-0801-0025))
** "본 연구는 지식경제부 및 정보통신연구진흥원의 IT핵심 기술개발사업[2005-S-088-04, 안전한 RFID/USN을 위한 정보보호 기술] 사업의 일환으로 수행하였음."
† 주저자, ns-chang@korea.ac.kr
‡ 교신저자, chkim@semyung.ac.kr

Algorithm 1 표수가 3인 경우 η_T 페어링의 계산 방법

Input : $E(GF(3^m))$의 원소인 $P=(x_P,y_P)$와 $Q=(x_Q,y_Q)$.

Output : $\eta_T(P,Q)$

```
01 : if b=1 then y_P←-y_P;                    // GF(3^m)에서 연산 //
02 : r_0←x_P+x_Q+b;                           // GF(3^m)에서 연산 //
03 : R_0←-y_P r_0+y_Q σ+y_P ρ;                // GF(3^6m)에서 연산 //
04 : for i=0 to (m-1)/2 do
05 :     r_0←x_P+x_Q+b;                       // GF(3^m)에서 연산 //
06 :     R_1←-r_0^2+y_P y_Q σ-r_0 ρ-ρ^2;       // GF(3^6m)에서 연산 //
07 :     R_0←R_0 R_1;                         // GF(3^6m)에서 연산 //
08 :     x_P←x_P^{1/3}; y_P←y_P^{1/3};        // GF(3^m)에서 연산 //
09 :     x_Q←x_Q^3; y_Q←y_Q^3;                // GF(3^m)에서 연산 //
10 : end for
11 : Return R_0;
```

면 D_1,D_2는 동치(equivalent)라 하고 $D_1 \sim D_2$라 표기한다. $P\in E(GF(q))[r]$이고 D가 $(P)-(O)$와 동치인 divisor라 하자. 그러면 rD는 principal이고 $(f_{r,P})=rD=r(P)-r(O)$를 만족하는 rational function $f_{r,P}$가 존재한다. [2]와 [8]로부터 위수 r의 Tate 페어링은 다음과 같은 bilinear map이다.

$$e_r(P,Q): E(GF(q))[r]\times E(GF(q^k))[r]\rightarrow GF(q^k)^*/(GF(q^k)^*)^r$$

Tate 페어링은 $e_r(P,Q)=f_{r,P}(Q)$에 의해서 계산된다. 페어링의 결과는 $GF(q^k)^*/(GF(q^k)^*)^r$의 원소이기 때문에 $GF(q^k)^*$의 원소 a,b에 대하여 $a=bc^r$인 $c\in GF(q^k)^*$가 존재한다. 페어링이 암호학적으로 사용되기 위하여 이 값은 유일해야 하기 때문에 final exponentiation이 수행되어야 한다. reduced Tate 페어링은 다음과 같이 정의된다.

$$e_r(P,Q)=e_r(P,Q)^{(q^k-1)/r}$$

위의 페어링에서 두 번째 입력값 Q는 $E(GF(q^k))[r]$의 원소이기 때문에 $E(GF(q))[r]$에서도 정의될 수 있다. 이러한 문제를 제거하기 위하여 distortion map[17]을 이용한다. 대부분의 $GF(q^k)$ 연산이 distortion map의 사용으로 $GF(q)$의 연산으로 대체되기 때문에 연산 속도가 향상된다. distortion map을 이용한 modified Tate 페어링은 다음과 같이 정의된다.

$$\hat{e}_r(P,Q)=e_r(P,\phi(Q))$$

P와 Q는 $E(GF(q))[r]$의 원소이고 ϕ는 distor tion map이다. [7]에서 distortion map과 위의 modified Tate 페어링을 결합하여 닫힌 공식(closed formula)을 만드는 방법을 제안하였다.

본 논문에서는 표수가 3인 경우와 $GF(3^m)$위에서 정의된 초특이(supersingular) 타원곡선을 다룰 것이다. $GF(3^m)$위에서 정의된 초특이(supersing ular) 타원곡선은 다음과 같다,

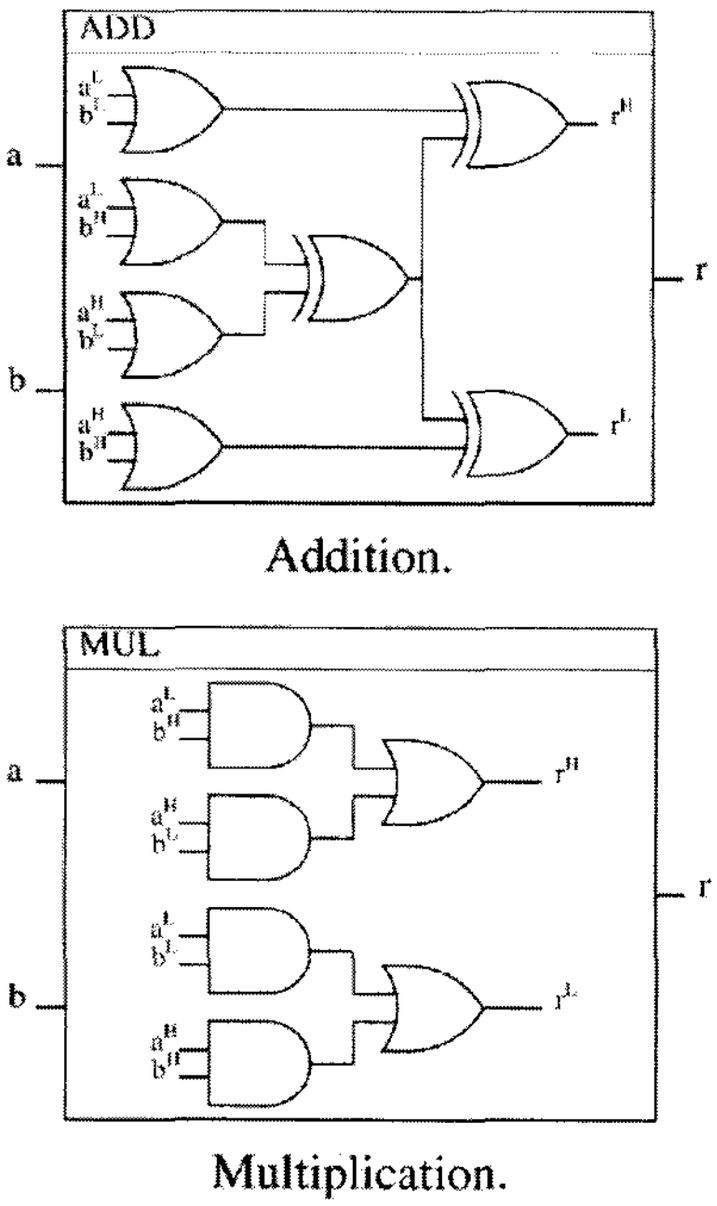
$$E^b: y^2=x^3-x+b,\ \text{where } b\in\{-1,1\}.$$

[2]에 의하면 표수가 3인 유한체 위에서 정의된 타원곡선은 security parameter가 6이기 때문에 안전성과 공간측면에서 가장 효과적인 페어링 연산을 제공한다. 표수가 3인 경우 distortion map $\phi: E(GF(3^m))\rightarrow E(GF(3^{6m}))$는 다음과 같이 정의된다.

$$\phi(P)=\phi((x,y))=(\rho-x,\sigma y)$$

σ와 ρ는 $GF(3^{6m})$의 원소이고 $\sigma^2+1=0$과 $\rho^3-\rho-1=0$을 만족한다. $GF(3^m)$위에서 $GF(3^{6m})$의 기저(basis)는 $\{1,\sigma,\rho,\sigma\rho,\rho^2,\sigma\rho^2\}$이다. $GF(3^{6m})$의 원소 A는 다음과 같이 표현된다.

$$\begin{aligned}A&=(a_0,a_1,a_2,a_3,a_4,a_5)\\&=a_0+a_1\sigma+a_2\rho+a_3\sigma\rho+a_4\rho^2+a_5\sigma\rho^2\end{aligned}$$



〔그림 1〕 기존의 *GF(3)* 연산기 : (a) 덧셈기 (b) 곱셈기

따라서 $GF(3^{6m}) = GF(3^{3m})[\sigma]/(\sigma^2+1)$은 $GF(3^{3m})$의 이차 확장체이고 $GF(3^{3m}) = GF(3^m)[\rho]/(\rho^3-\rho-1)$은 $GF(3^m)$의 3차 확장체로써 생각할 수 있기 때문에 $GF(3^{6m})$에서의 모든 연산은 $GF(3^m)$에서 연산으로 바꿀 수 있다. [1]에서 제안된 표수가 3인 경우 η_T 페어링을 계산하는 알고리즘은 Algorithm 1과 같다.

Ⅲ. 기존의 *GF(3^m)* 연산 방법

$f(x) = x^m + f_t \cdot x^t + f_0$를 $GF(3)$위에서 차수가 m인 삼항 기약다항식이라 하고 α를 $f(x)$의 해라 하자. 그러면 유한체 $GF(3^m)$는 $GF(3)[X]/(f(x))$와 동형이기 때문에 $GF(3^m)$의 원소 $a(\alpha)$는 다항식 기저를 이용해서 다음과 같이 표현할 수 있다.

$$a(\alpha) = a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0 = (a_{m-1}\ldots a_1 a_0)$$
$$, \text{where } a_i \in GF(3). \quad (1)$$

그리고 α는 $f(x)$의 해이므로 $f(\alpha)=0$에 의하여

$$\alpha^m = -f_t\alpha^t - f_0 \quad (2)$$

이다. $a(\alpha)$, $b(\alpha) \in GF(3^m)$라 하면, $c(\alpha)=a(\alpha)+b(\alpha)$는

$$c(\alpha) = a(\alpha)+b(\alpha) = \sum_{i=0}^{m-1}(a_i+b_i)\alpha^i$$

이고 a_i+b_i는 *GF(3)*의 덧셈 연산이다.

3.1 기존의 *GF(3)* 연산기

본 절에서는 기존의 *GF(3)* 연산에 대하여 기술한다 [10,15]. 식 (1)에서 a_i는 $GF(3)$의 원소이기 때문에 a_i를 표현하기 위해서는 2비트가 필요하게 된다. [15]에서는 $GF(3)$의 원소를 표현하기 위하여 다음과 같은 표기법을 사용하였다.

$$a_i = (a_i^H, a_i^L), \text{ where } a_i^H = a_i \div 2$$
$$\text{and } a_i^L = a_i \bmod 2 \quad (3)$$

식 (3)의 표현법을 이용하여 $GF(3)$의 원소 a_i와 b_i의 덧셈 $r_i = a_i + b_i$은 3 XOR 게이트와 4 OR 게이트를 이용하여 다음과 같이 계산된다.

$$r_i^H = (a_i^L \vee b_i^L) \oplus t \text{ and } r_i^L = (a_i^H \vee b_i^H) \oplus t,)$$
$$, \ where \ t = (a_i^L \vee b_i^H) \oplus (a_i^H \vee b_i)$$

특히 $2(a_i^H, a_i^L) = -(a_i^H, a_i^L) = (a_i^L, a_i^H)$이므로 뺄셈과 2에 의한 곱셈은 덧셈을 이용하여 쉽게 계산할 수 있다. 또한, 곱셈 연산은

$$r_i^H = (a_i^L \wedge b_i^H) \vee (a_i^H \wedge b_i^L),$$
$$r_i^L = (a_i^L \wedge b_i^L) \vee (a_i^H \wedge b_i^H),$$

이고 [10]에서 소개된 *GF(3)*의 덧셈기 및 곱셈기는 [그림 1]과 같다.

3.2 기존의 *GF(p^m)* 비트-직렬 곱셈기

본 절에서는 기존의 비트-직렬 곱셈기에 대하여 기술한다[4,6]. *GF(p^m)*의 곱셈은 두 원소의 케리(Carry) 전파가 없는 다항식 곱셈과 주어진 *f(x)*를 이용한 모듈러 연산으로 구성된다. 즉, *r'(α)=a(α)·b(α)*의 곱셈 연산과 *r(α)= r'(α) mod f(α)*의 모듈러 연산으로 구분된다. 일반적으로 모듈러 연산은 기약다항식 *f(x)*를 이용한 나

늣셈으로 구성되나 하드웨어에서는 많은 하드웨어 자원을 사용하는 단점으로 interleaved 연산으로 나눗셈 없이 처리한다. 따라서 shift-and-add 방법으로 $GF(p^m)$의 곱셈 연산을 구성한다. 임의의 $GF(p^m)$의 원소 $a(\alpha)$, $b(\alpha)$의 곱 $r(\alpha)=a(\alpha)\cdot b(\alpha) \bmod f(x)$은 $a(\alpha)$의 계수처리 방향에 따라 MSB-first (Most-Significant Bit)와 LSB-first(Least- Significant Bit)로 나뉜다. 본 논문에서는 일반적인 페어링 기반의 암호 시스템에서 사용하는 삼항 기약다항식 $f(x)=x^m+f_tx^t+f_0$만을 고려한다.

3.2.1 기존의 MSB-first 비트-직렬 곱셈기

MSB-first 비트-직렬 곱셈의 기본적인 구조는 $r(\alpha)=a(\alpha)\cdot b(\alpha) \bmod f(x)$의 계산을 위하여 m번의 반복 연산을 수행한다. 즉,

$$r'(\alpha) = a(\alpha)\cdot b(\alpha)$$
$$=((\cdots(a_{m-1}b(\alpha)\cdot\alpha + a_{m-2}b(\alpha))\cdot\alpha +\cdots)\cdot\alpha$$
$$+ a_1b(\alpha))\cdot\alpha + a_0b(\alpha).$$

이며 반복 구조로 쉽게 설계된다. 반복적인 결과물 저장을 위한 레지스터(register) r이 하나 필요하며, $r(\alpha)=r(\alpha)\cdot\alpha+a_ib(\alpha)$ 연산을 m번 반복한다. 이때 i는 $m-1$부터 0까지 수행하며 $a_ib(\alpha)$는 $m-1$차 다항식이고 $r(\alpha)\cdot\alpha$는 m차 다항식이므로 한 번의 뺄셈으로 $r(\alpha)$에 대한 모듈러 감산 연산을 구성할 수 있다. MSB-first 비트-직렬 곱셈은 Algorithm 2와 같다.

자세한 MSB-first 비트-직렬 곱셈의 하드웨어 구조는 [그림 2(a)]와 같다. $\times\alpha$는 $r(\alpha)\cdot\alpha$ 연산부로 비용이 들지 않으며 MOD의 세부연산의 구성은 [그림 2(b)]와 같

Algorithm 2 MSB-first 비트-직렬 곱셈

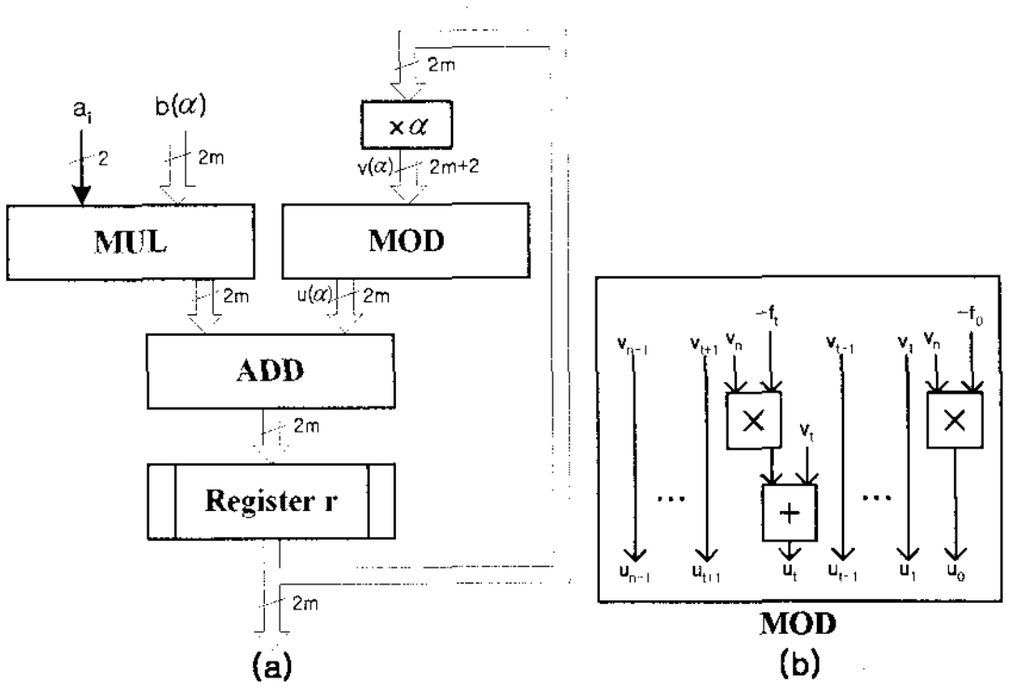
Input :

$$a(\alpha)=\sum_{i=0}^{m-1} a_i\alpha^i, \quad b(\alpha)=\sum_{i=0}^{m-1} b_i\alpha^i, \quad a_i, b_i \in GF(p)$$

Output :

$$r(\alpha)=a(\alpha)\cdot b(\alpha)=\sum_{i=0}^{m-1} r_i\alpha^i, \quad r_i \in GF(p)$$

$r(\alpha) \leftarrow 0$
for i = $m-1$ to 0 do
  $r(\alpha) \leftarrow r(\alpha)\cdot\alpha + a_i\cdot b(\alpha) \bmod f(x)$
end for
Return ($r(\alpha)$)



〔그림 2〕 (a) GF(3^m) MSB-first 비트-직렬 곱셈기
(b) MOD의 세부구조

다. 본 논문의 경우 삼항 기약다항식만을 고려하므로 [그림 2(b)]와 같이 MOD 연산부는 매우 간단하게 구성된다. $v(\alpha)=r_{m-1}\alpha^m+\ldots+r_0\alpha=v_m\alpha^m+\cdots+v_1\alpha_1$,이므로 삼항 기약다항식 $f(x)=x^m+f_tx^t+f_0$에 의하여

$$u(\alpha)=v(\alpha) \bmod f(x)$$
$$=\left(\sum_{i=1}^{m-1} v_i\alpha^i\right)+((-f_t)\cdot v_m)\alpha^t+((-f_0)\cdot v_m)$$

이므로, $((-f_t)\cdot v_m)$와 $((-f_0\cdot v_m)$에서 두 번의 $GF(3)$ 곱셈 연산, $r_{t-1}\alpha^t+((-f_t)\cdot v_m)\alpha^t$에서 한 번의 $GF(p)$ 덧셈 연산만으로 수행된다. 이때 $-f_t$, $-f_0$는 $f(x)$가 정의되면 고정값이므로 복잡도에 영향을 주지 않는다.

전체 Critical Path는 MOD연산부와 ADD 연산부에 의존하므로, $GF(p)$ 단위 연산을 각각 mul과 add로 정의하면 1·mul+2·add이다. 따라서 전체 시간지연은 m·mul+$2m$·add이다. 공간복잡도를 살펴보면 MUL에서 m개의 $GF(p)$ 곱셈기, ADD에서 m개의 GF(p) 덧셈기, MOD에서 각각 2개와 1개의 GF(p) 곱셈기 및 덧셈기가 필요하므로 전체 공간복잡도는 $(m+2)$·mul+ $(m+1)$·add이다.

3.2.2 기존의 LSB-first 비트-직렬 곱셈기

LSB-first 비트-직렬 곱셈기 또한 기본적인 구조는 $r(\alpha)=a(\alpha)\cdot b(\alpha) \bmod f(\alpha)$의 계산을 위하여 m번의 반복 연산을 수행하며,

$$r'(\alpha) = a(\alpha)\cdot b(\alpha)$$
$$=((\cdots(a_0b(\alpha)+a_1b(\alpha)\cdot\alpha)+\cdots)+a_{m-2}b(\alpha)\cdot\alpha^{m-2})$$

$$+ a_{m-1}b(\alpha)\cdot \alpha^{m-1}$$

이다. 따라서 LSB-first 연산은 두 개의 반복 연산으로 구성된다. 첫 번째는 매번 반복 시 $b(\alpha)\cdot\alpha^i$에 α를 곱하여 저장하는 연산부이며, 두 번째는 i번째에 저장된 $b(\alpha)\cdot\alpha^i$에 a_i를 곱한 후 저장값 $r(\alpha)$와 더하는 연산부이다. 즉, MSB-first와 달리 모듈러 연산부가 병렬로 구성되므로 시간복잡도는 감소하나 레지스터 등의 추가 자원이 필요하게 된다. LSB-first 비트-직렬 곱셈은 Algorithm 3과 같다.

자세한 LSB-first 비트-직렬 곱셈의 하드웨어 구조는 [그림 3]와 같다. LSB-first 비트-직렬 곱셈기와 같이 $\times\alpha$는 $s(\alpha)\cdot\alpha$ 연산부로 비용이 들지 않으며 MOD의 세부연산의 구성은 [그림 2(b)]와 같다. 전체 Critical Path는 MOD연산부가 병렬화되어 MUL과 ADD 연산부에 의존하므로, *GF(p)* 단위 연산을 각각 mul과 add로 정의하면 1·mul+ 1·add이다. 따라서 전체 시간지연은 m·mul+ m·add이다. 공간복잡도를 살펴보면 MUL에서 m개의 *GF(p)* 곱셈기, ADD에서 m개의 *GF(p)* 덧셈기, MOD에서 각각 2개와 1개의 *GF(p)* 곱셈기 및 덧셈기가 필요하므로 전체 공간복잡도는 $(m+2)\cdot$ mul+$(m+1)\cdot$ add이며 추가적으로 레지스터 s가 요구된다.

Algorithm 3 LSB-first 비트-직렬 곱셈

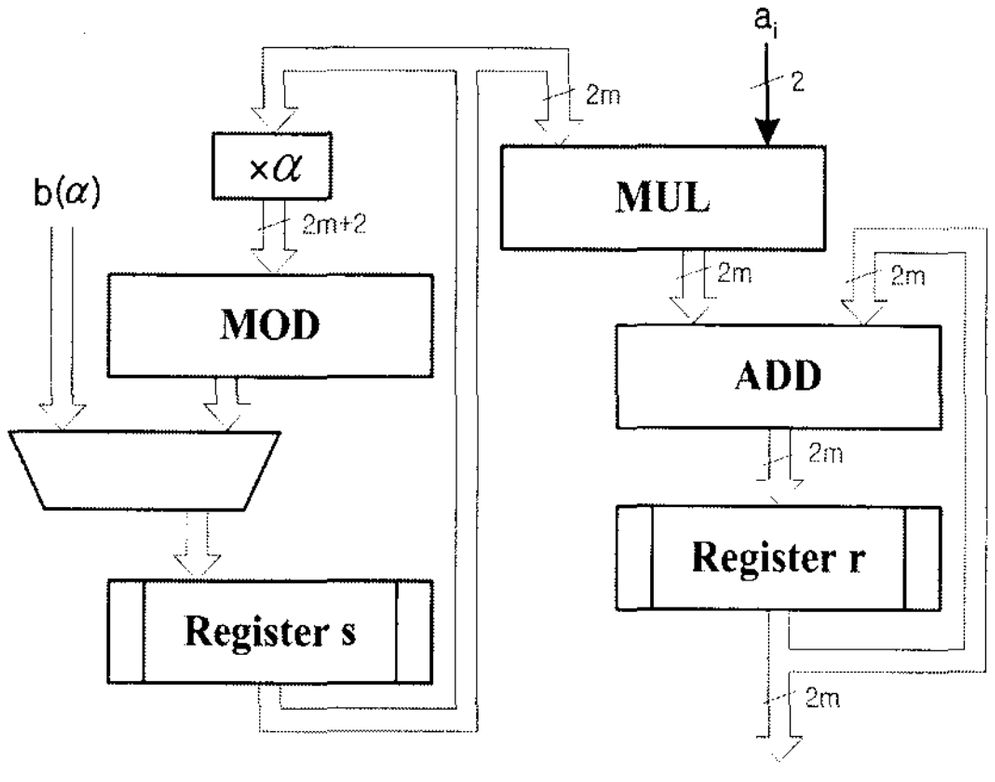
Input :

$a(\alpha)=\sum_{i=0}^{m-1} a_i\alpha^i, \quad b(\alpha)=\sum_{i=0}^{m-1} b_i\alpha^i, \quad a_i, b_i \in GF(p)$

Output :

$r(\alpha)=a(\alpha)\cdot b(\alpha)=\sum_{i=0}^{m-1} r_i\alpha^i, \quad r_i \in GF(p)$

$r(\alpha) \leftarrow 0$
$s(\alpha) \leftarrow b(\alpha)$
for $i = 0$ to $m-1$ do
  $r(\alpha) \leftarrow r(\alpha) + a_i s(\alpha)$
  $s(\alpha) \leftarrow s(\alpha)\cdot\alpha \ mod\ f(x)$
end for
Return $(r(\alpha))$



〔그림 3〕 GF(3^m) LSB-first 비트-직렬 곱셈기

Ⅳ. 제안하는 페어링 기반 암호시스템의 유한체 연산기

본 절에서는 새로운 *GF(3)* 연산방법과 *GF(3^m)* 덧셈-뺄셈 unified 연산기 MSB-first 타입의 새로운 *GF(p^m)* 비트-직렬 곱셈 방법을 제안한다. 제안하는 비트-직렬 곱셈 방법은 페어링 기반의 암호 시스템에서 효율성을 위하여 삼항 기약다항식을 사용하는데 착안하여 MOD 연산부를 효율적으로 병렬화한다.

4.1 제안하는 *GF(3)* 연산기

본 절에서는 새로운 *GF(3)* 연산방법을 제안한다. 제안하는 *GF(3)* 연산기는 *GF(3^m)*의 곱셈기 구성에 효율적인 방법을 고려하여 [15]와 달리 $a_i=(a_i^H, a_i^L)$에서 a_i^H를 부호 비트로 a_i^L를 데이터 비트로 사용한다. 즉, {0, 1, 2}는 각각 {(0,0), (0,1), (1,1)}로 표현된다. 이와 같은 표현 방법으로 *GF(3)*의 임의의 두 원소 a_i, b_i의 덧셈 $r_i=a_i+b_i$를 비트 단위 연산으로 표현하면

$$r_i^H = (a_i^H \oplus b_i^L) \wedge t,$$

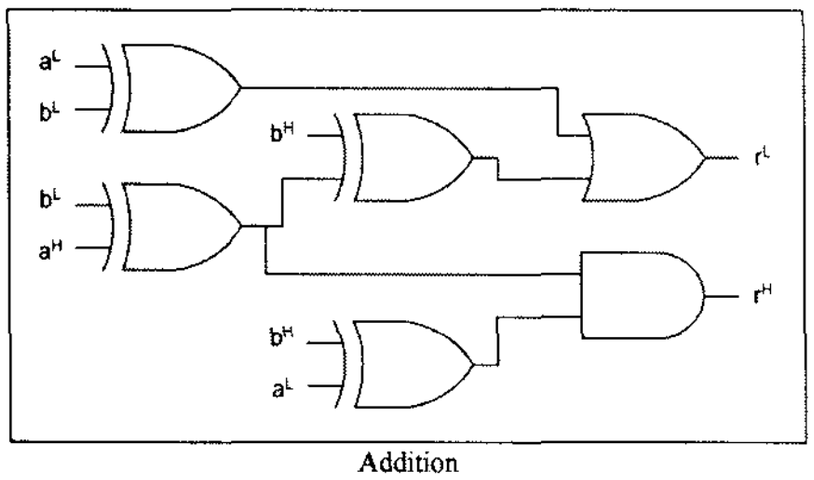
$$r_i^L = (a_i^H \oplus t) \vee (a_i^L \oplus b_i^L),$$

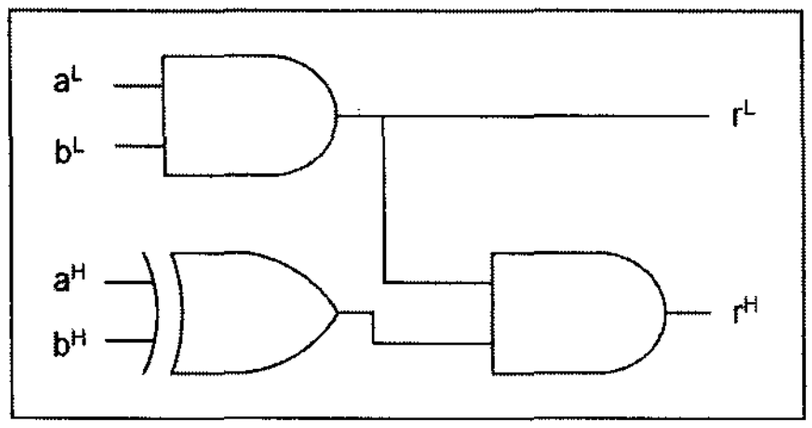
이고, 이때 $t = a_i^L \oplus b_i^H$이다. *GF(3)* 뺄셈의 경우, $r_i=a_i-b_i$를 $r_i=a_i+(-b_i)$으로 처리하며, $(-b_i)$연산은 부호 비트를 $-b_i = (b_i^H \oplus b_i^L, b_i^L)$와 같이 한번의 XOR연산으로 계산할 수 있다. 또한, *GF(3)*의 임의의 두 원소 a_i, b_i의 곱셈 $r_i=a_i\cdot b_i$를 비트 단위 연산으로 표현하면

$$r_i^H = r_i^L \wedge (a_i^H \oplus b_i^H),$$

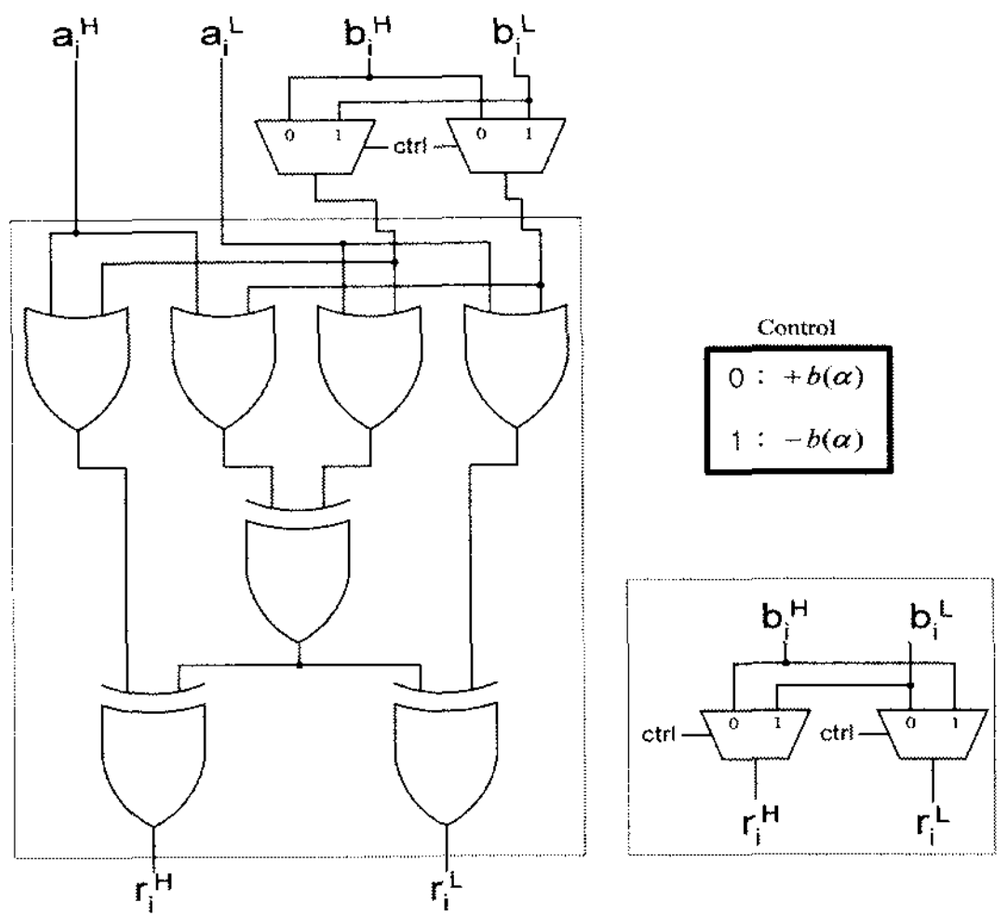
$$r_i^L = a_i^L \wedge b_i^L,$$

이고, *GF(3)*의 덧셈기 및 곱셈기의 구조는 [그림 4]와 같다.





〔그림 4〕 제안하는 *GF(3)* 연산기 : (a) 덧셈기
(b) 곱셈기(a)

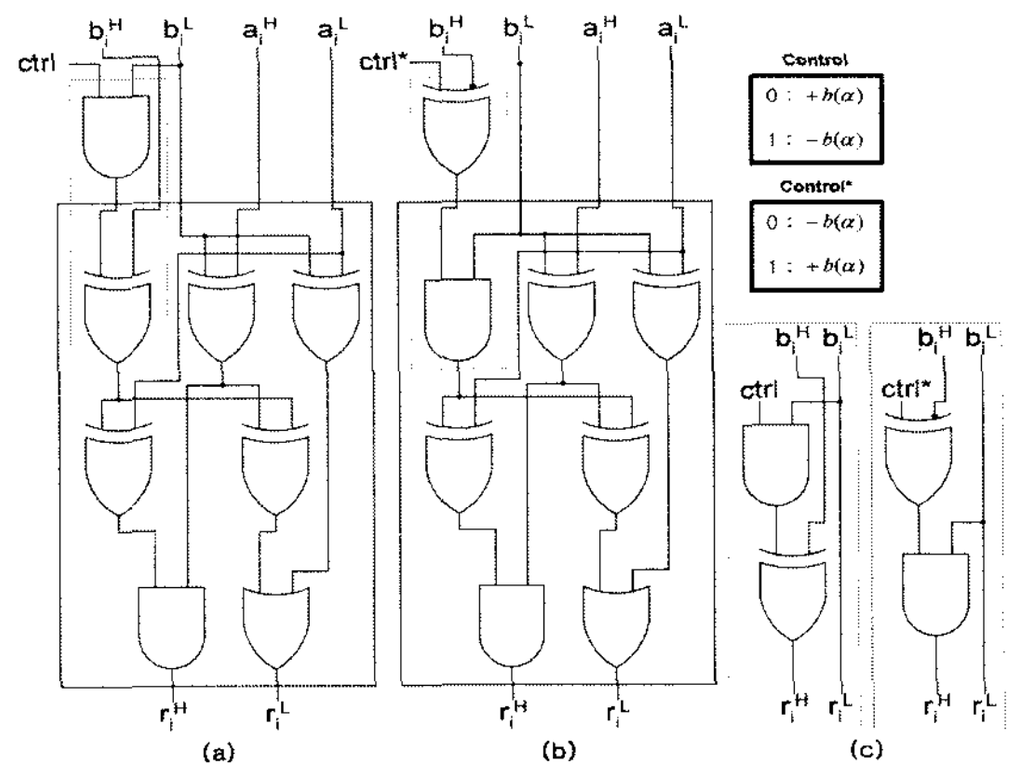


〔그림 5〕 기존의 $GF(3^m)$ 연산기 :
(a) 덧셈-뺄셈 unified 연산기 (b) $r_i=-b_i$

4.2 제안하는 $GF(3^m)$ 덧셈-뺄셈 Unified 연산기

본 절에서는 제안하는 *GF(3)* 덧셈기를 이용하여 $GF(3^m)$ 덧셈-뺄셈 unified 연산기를 제안한다. [5,6,10] 등에서와 같이 대부분의 논문에서 GF(3^m) 덧셈과 뺄셈 연산은 GF(3)의 원소 특징을 이용하여 unified 형태로 구성된다. $GF(3^m)$의 원소 $a(\alpha), b(\alpha)$에 대하여 기존의 *GF(3)* 연산기는 $-(a_i^H, a_i^L) = (a_i^L, a_i^H)$으로 쉽게 설계된다. 따라서 unified 구조를 고려하는 경우 $r(\alpha) = a(\alpha) \pm b(\alpha)$의 설계를 위하여 2개의 2-input MUX를 사용하여 구성한다. 이를 설계하면 [그림 5(b)]와 같으며 이를 이용한 덧셈-뺄셈 unified 연산기는 [그림 5(a)]와 같다.

기존의 *GF(3)* 덧셈기는 $-a_i$를 연산할 때 두 비트를 바꿔야하지만 본 논문에서 제안하는 방법은 부호 비트만 바꾸면 되므로 간단하며, $r(\alpha) = a(\alpha) \pm b(\alpha)$인 경우 $-b(\alpha)$의 구성 시 [그림 6(c)]과 같이 효율적으로 적용된다. 제안하는 $-b_i$ 연산기는 [그림 6(c)]와 같으며 이를 이용한 덧셈-뺄셈 unified 연산기는 [그림 6(a)](또는 [그림 6(b)])와 같다. 제안하는 $GF(3^m)$ 덧셈-뺄셈 unified 연산기는 [그림 6(a)]와 같이 1AND+ 1XOR 게이트를 추가하여 구성 가능하며 시간지연의 경우 1AND 게이트의 시간지연만 추가된다. FPGA 환경에서 *GF(3)* 덧셈기는 2개의 4-input LUT로 구성된다. 따라서 이를 고려한 $GF(3^{97})$인 경우 unified 연산기에서는 2-input MUX가 194개 추가적으로 요구되는 반면에



〔그림 6〕 제안하는 $GF(3^m)$ 연산기 :
(a) TYPE I 덧셈-뺄셈 unified 연산기
(b) TYPE II 덧셈-뺄셈 unified 연산기
(c) $r_i=-b_i$

제안하는 unified 연산기의 경우 97개의 and(또는 XOR) 게이트만으로 구성 가능하다.

4.3 제안하는 $GF(p^m)$ 비트-직렬 곱셈기

본 절에서는 새로운 $GF(p^m)$ 비트-직렬 연산 방법을 제안하며, 이는 *f(x)*가 삼항 기약다항식이라는 특징을 기반으로 하므로 유한체의 표수에 의존하지 않는다. 이전 절에서 표수가 3인 경우에 대하여 논리를 전개하였으므로 본 절에서도 p가 3인 경우를 예로 기술한다.

Algorithm 2에서 $r(\alpha)$는 $r(\alpha)=r_{m-1}\alpha^{i-1}+\cdots+r_0$이다. 따라서 $r(\alpha)\cdot\alpha$는 $r(\alpha)=r_{i-1}\alpha^m+\cdots+r_0\alpha$이고 식 (2)에 의하여

$$\begin{aligned} r(\alpha)\cdot\alpha \bmod f(x) &= \sum_{i=0}^{m-1} r_i\alpha^{i+1} \bmod f(x) \\ &= r_{m-1}\alpha^m + \sum_{i=0}^{m-2} r_i\alpha^{i+1} \bmod f(x) \\ &\equiv (r_{t-1}-r_{m-1}\cdot f_t)\alpha^t-(r_{m-1}\cdot f_0) \\ &\quad + \sum_{i=0,\ i\neq t-1}^{m-2} r_i\alpha^{i+1} \end{aligned} \tag{4}$$

이다. 따라서 $r(\alpha)\cdot\alpha$ 연산은 $(r_{t-1}-r_{m-1}\cdot f_t)\alpha^t$에서 각각 한번의 *GF(3)* 곱셈 및 뺄셈으로 연산된다. 이를 이용하면 Algorithm 2의 복 연산문인 $r(\alpha) \equiv r(\alpha)\cdot\alpha + a_i\cdot b(\alpha) \bmod f(x)$은 i가 $m-1$에서 0까지 감소하는 것을 고려할 때, 다음과 같이 표현된다.

$$r(\alpha)^{(i)} = \sum_{j=0}^{m-1} r_j^{(i)}\alpha^j = r(\alpha)^{(i+1)}\cdot\alpha + a_i\cdot b(\alpha).$$

따라서 위의 식에 식 (4)를 적용하면,

$$\begin{aligned} r(\alpha)^{(i)} &\equiv r(\alpha)^{(i+1)}\cdot\alpha + a_i\cdot b(\alpha) \bmod f(x) \\ &= (r_{t-1}-r_{m-1}\cdot f_t + a_i b_t)\alpha^t + (a_i b_0 - r_{m-1}\cdot f_0) \\ &\quad + \sum_{i=0,\ i\neq t-1}^{m-2} (r_i + b_{i+1})\alpha^{i+1} \end{aligned} \tag{5}$$

이고, $r(\alpha)^{(m)}=0$이다.

Definition 1. (α)가 $GF(3^m)$의 원소라 하고 기약 다항식을 $f(x)$에 대하여 $\tilde{f}(x)=f(x)\cdot x$라 하면,

$$\begin{aligned} \bar{a}(\alpha) &= a(\alpha) - a_{t+1}\cdot f_t\cdot\tilde{f}(\alpha) \\ &= a(\alpha) - a_{t+1}\cdot f_t\cdot f(\alpha)\alpha \end{aligned}$$

이다.

Theorem 1. $a(\alpha), b(\alpha)\in GF(3^m)$이면, Definition 1에 의하여 두 원소의 곱은

$$\begin{aligned} a(\alpha)b(\alpha) \bmod f(x) &= (a(\alpha)b(\alpha)\alpha \bmod f(x)x)/\alpha \\ &= \{a(\alpha)b(\alpha)\alpha \bmod \tilde{f}(x)\}/\alpha \\ &= \{\tilde{a}(\alpha)\cdot b(\alpha) \bmod \tilde{f}(x)\}/\alpha \\ &= \{\tilde{a}(\alpha)\cdot\bar{b}(\alpha) \bmod \tilde{f}(x)\}/\alpha \end{aligned}$$

Algorithm 4 제안하는 MSB-first 비트-직렬 곱셈

Input : $\tilde{a}(\alpha)=a(\alpha)\alpha=\sum_{i=0}^{m+2}\tilde{a_i}\alpha^i=(0,0,a_{m-1},\cdots,a_0,0)$,

$b(\alpha)=\sum_{i=0}^{m-1} b_i\alpha^i=(b_{m-1},\cdots,b_0)$, $a_i,b_i\in GF(3)$,

$-f_t, -f_0, -f_tf_0\in GF(3)$

Output : $r(\alpha)=a(\alpha)\cdot b(\alpha) \bmod f(x)=\sum_{i=0}^{m-1} r_i\alpha^i$, $r_i\in GF(3)$

$\delta \leftarrow 0$, $\lambda \leftarrow 0$

$\overline{b_{m+1}} \leftarrow 0$, $\overline{b_0} \leftarrow 0$

for $i = m+2$ to 0 do

1. Multiplication

(a) ($\bar{b}(\alpha)$ Initialization) $\overline{b_{m+1}} \leftarrow b_{t+1}\cdot(-f_t)$, $temp1 \leftarrow b_{t+1}\cdot(-f_tf_0)$

(b) $v(\alpha) \leftarrow \sum_{j=0,\ j\neq t+1,1}^{m-1} \tilde{a_i}b_j\alpha^j$, $v_{t+1} \leftarrow \delta\cdot(-f_t)$, $v_1 \leftarrow \delta\cdot(-f_0)$

(c) (Precomputation) $temp2 \leftarrow \overline{b_{m+1}}\cdot\widetilde{a_{i-1}}$, $temp3 \leftarrow \overline{b_1}\cdot\widetilde{a_{i-1}}$

2. Addition

(a) ($\bar{b}(\alpha)$ Initialization) $\overline{b_1} \leftarrow temp1+b_1$

(b) $r(\alpha) \leftarrow \sum_{j=2}^{m-1}(v_j+c_{j-1})\alpha^j$, $r_m \leftarrow r_{m-1}$, $r_1 \leftarrow v_1+\lambda$, $r_0 \leftarrow v_0$

(c) (Precomputation) $\delta \leftarrow temp2+r_{m-1}$, $\lambda \leftarrow temp3+v_0$

end for

Return $(r(\alpha)/\alpha)$

이고, 이때 $a(\alpha)\alpha = \tilde{a}(\alpha) = \sum_{i=0}^{m} \tilde{a}_i \alpha^i$이다. 그리고 f(x)가 삼항 기약다항식이면, $b(\alpha) \equiv \bar{b}(\alpha) \bmod f(x)$이고 $\bar{b}(\alpha)$의 t+1항의 계수는 0이다.(단, t<m-1이다.)

*GF(3)*의 0이 아닌 임의의 원소 a_i는 자신의 역원이므로 $a_i^2 = 1$이다. 따라서 $\bar{b}(\alpha)$의 $t+1$항은

$$b_{t+1}\alpha^{t+1} - b_{t+1} \cdot f_t \cdot f_t \alpha^{t+1} = (b_{t+1} - b_{t+1})\alpha^{t+1}$$

이므로, $t+1$항의 계수는 0이다. 따라서 Algorithm 2의 $r(\alpha) \equiv r(\alpha) \cdot \alpha + a_i \cdot b(\alpha) \ mod \ f(x)$는 Theorem 1에 의하여 $r(\alpha) \equiv r(\alpha) \cdot \alpha + \tilde{a}_i \cdot \bar{b}(\alpha) \bmod \tilde{f}(x)$이며 반복 연산을 고려하여 표현하면 다음과 같으며 $\tilde{f}(x)$가 $m+1$차이므로 $\tilde{r}(\alpha)^{(i+1)}$은 $m+1$차이다.

$$\begin{aligned} r(\alpha)^{(i)} &\equiv r(\alpha)^{(i+1)} \cdot \alpha + \tilde{a}_i \cdot \bar{b}(x) \bmod \tilde{f}(x) \\ &= \tilde{r}(\alpha)^{(i+1)} + \tilde{a}_i \cdot \bar{b}(x) \bmod \tilde{f}(x) \\ &= \left(\widetilde{r_{m+1}^{(i+1)}} + \widetilde{a_i \bar{b}_{m+1}}\right)\alpha^{(m+1)} \\ &\quad + \sum_{j=0}^{m} \left(\widetilde{r_j^{(i+1)}} + \widetilde{a_i \bar{b}_j}\right)\alpha^j \bmod \tilde{f}(x) \end{aligned} \quad (6)$$

식 (6)에서 $\widetilde{r_{m+1}^{(i+1)}} + \widetilde{a_i \bar{b}_{m+1}}$를 $\delta^{(i)}$라 하면

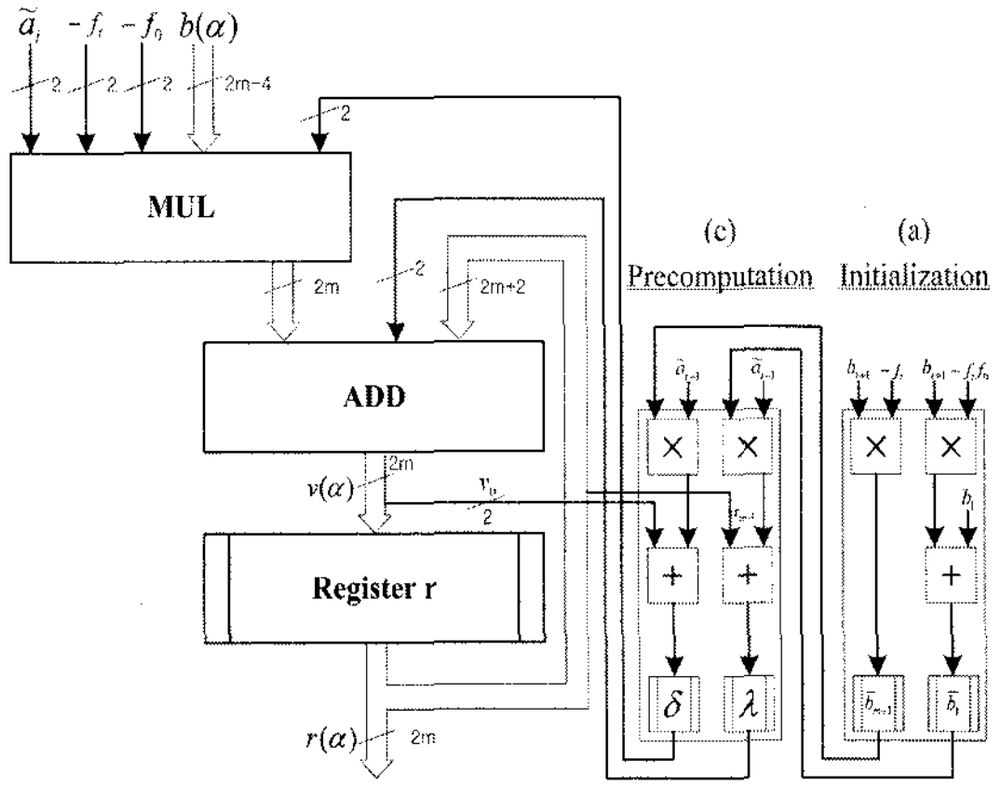
$$\begin{aligned} &\delta^{(i)} \cdot \alpha^{(m+1)} + \sum_{j=0}^{m} \left(\widetilde{r_j^{(i+1)}} + \widetilde{a_i \bar{b}_j}\right)\alpha^j \bmod \tilde{f}(x) \\ &= \left(-\delta^{(i)}\widetilde{f_{t+1}}\alpha^{t+1} - \delta^{(i)}\tilde{f}_1\alpha\right) \\ &\quad + \sum_{j=0}^{m} \left(\widetilde{r_j^{(i+1)}} + \widetilde{a_i \bar{b}_j}\right)\alpha^j \bmod \tilde{f}(x) \end{aligned}$$

이고, 이를 식 (6)에 적용하여 정리하면 $r(\alpha)^{(i)}$의 계수는 다음과 같다.

$$r_j^{(i)} \equiv \begin{cases} \widetilde{r_j^{(i+1)}} + \widetilde{a_i \bar{b}_j} & ,m \geq j \geq 0,\ j \neq t+1, 1 \\ \widetilde{r_{t+1}^{(i+1)}} + \widetilde{a_i \bar{b}_{t+1}} - \delta^{(i)}\widetilde{f_{t+1}} & ,j = t+1, \\ \widetilde{r_1^{(i+1)}} + \widetilde{a_i \bar{b}_1} - \delta^{(i)}\tilde{f}_1 & ,j = 1 \end{cases} \quad (7)$$

Theorem 2. $\widetilde{r_{m+1}^{(i+1)}} + \widetilde{a_i \bar{b}_{m+1}}$를 $\delta^{(i)}$라 하고, $\widetilde{a_{i+1}\bar{b}_0} + \widetilde{a_i \bar{b}_1}$를 $\lambda^{(i)}$이라 하면,

$$r_j^{(i)} \equiv \begin{cases} \widetilde{r_j^{(i+1)}} + \widetilde{a_i \bar{b}_j} & ,m \geq j \geq 0,\ j \neq t+1, 1 \\ \widetilde{r_{t+1}^{(i+1)}} - \delta^{(i)}\widetilde{f_{t+1}} & ,j = t+1, \\ \lambda^{(i)} - \delta^{(i)}\tilde{f}_1 & ,j = 1 \end{cases}$$



〔그림 7〕 제안하는 GF(3^m) MSB-first 비트-직렬 곱셈기

이고, $i+1$번째 반복문에서 $\delta^{(i)}$와 $\lambda^{(i)}$를 계산할 수 있으면 i번째 반복문에서 $r(\alpha)^{(i)}$의 모든 계수는 각각 한번의 *GF(3)* 곱셈 및 덧셈 연산으로 계산된다.

$\bar{b}(\alpha)$의 $t+1$항은 0이므로 $r_{t+1}^{(i)} = \widetilde{r_{t+1}^{(i+1)}} + \widetilde{a_i \bar{b}_{t+1}} - \delta^{(1)}\widetilde{f_{t+1}} = \widetilde{\boldsymbol{r_{t+1}^{(i+1)}}} - \widetilde{\boldsymbol{\delta^{(i)} f_{t+1}}}$이고, $\widetilde{r_1^{(i+1)}}$는 $r_0^{(i+1)}$이므로 $\lambda^{(i)}$를 $\widetilde{a_{i+1}\bar{b}_0} + \widetilde{a_i \bar{b}_1}$이라 하면 $r_1^{(i)} = \widetilde{a_{i+1}\bar{b}_0} + \widetilde{a_i \bar{b}_1} - \delta^{(i)}\tilde{f}_1 = \boldsymbol{\lambda - \delta^{(i)} f_{t+1}}$이다. 따라서 $\lambda^{(i)}$는 $i+1$번째 반복에서 계산 가능하다. 또한, $r_{m+1}^{(i+1)} = r_m^{(i+1)} = \widetilde{r_m^{(i+2)}} + \widetilde{a_{i+1}\bar{b}_m}$이고 Definition 1에 의하여 $\bar{b}_m$은 0이므로 $\delta^{(i)} = \widetilde{r_m^{(i+2)}} + \widetilde{a_i \bar{b}_{m+1}}$는 $i+1$번째 반복에서 계산 가능하다. 따라서 $r(\alpha)^{(i)}$의 모든 계수는 각각 한번의 *GF(3)* 곱셈 및 덧셈 연산으로 계산된다.

제안하는 비트-직렬 곱셈 방법을 정리하면 Algorithm 4와 같으며 이는 다음과 같다. 제안하는 알고리즘은 한번의 덧셈과 곱셈으로 $m+3$번 반복하며 각각 연산에서 (a), (b), (c)는 병렬로 동작한다. (a)는 $\bar{b}(\alpha)$를 초기화하는 과정으로 실제값은 $\bar{b}(\alpha) = b(\alpha) - b_{t+1} \cdot f_t \cdot f(\alpha)\alpha$의 계산 값이므로 $\bar{b}_{m+1}, \bar{b}_1$은 $f(x)$의 계수에 의하여 갱신되며 $\bar{b}_m, \bar{b}_{t+1}$은 0 그리고 나머지 $\bar{b}_i$는 b_i와 같다. 따라서 $\bar{b}_{m+1}, \bar{b}_1$만 계산하면 된다. (b)는 Theorem 2의 $r_j^{(i)}$를 계산하는 부분이다. (c)는 Theorem 2의 $\delta^{(i)}, \lambda^{(i)}$를 계산하는 부분이므로 i번째의 $\delta^{(i)}, \lambda^{(i)}$는 $i+1$번째에서 계산되어야 한다. 따라서 알고리즘은 i가 $m+2$일때 $\bar{b}(\alpha)$를 초기화하고 i가 $m+1$일때 다음 사용할 $\delta^{(m)}, \lambda^{(m)}$를 계산해야 하므로 $\tilde{a}_{m+2}, \tilde{a}_{m+1}$을 0으로 할당하여 (a)와 (c)를 준비한다. 제안하는 MSB-first 비트-

직렬 곱셈기의 하드웨어 구조는 [그림 7]과 같다.

레지스터 r에 중간값이 저장되며 $2m+2$비트의 정보를 저장하고 레지스터 δ, λ, $\bar{b}_{m+1}$, $\overline{b_1}$은 각각 2비트의 정보를 저장한다. (a)와 (c)의 구성에 *GF(3)* 곱셈기 4개와 덧셈기 3개가 소요되며 이 부분은 주 연산부와 병렬로 동작한다.

V. 비교 및 결론

본 논문에서는 새로운 *GF(3)* 덧셈기 및 곱셈기와 $GF(3^m)$ 덧셈-뺄셈 unified 연산기 및 MSB-first 비트-직렬 곱셈기를 제안하였다. 기존의 *GF(3)* 연산기 및 $GF(3^m)$ 덧셈-뺄셈 unified 연산기와 제안하는 연산기를 비교하면 [표 1]과 같다.

[표 1]의 결과와 같이 제안하는 *GF(3)* 연산기는 기존 방법에 비하여 작은 게이트로 구성되며 $GF(3^m)$ 덧셈-뺄셈 unified 연산기의 경우 제안하는 방법은 $-b_i$연산기가 *GF(3)* 덧셈기에 효율적으로 적용되어 기존의 결과에 비하여 시간-공간 복잡도면에서 효율적이다.

제안하는 $GF(p^m)$ MSB-first 비트-직렬 곱셈기는 삼항 기약다항식의 특징을 이용한다. [표 2]의 결과에서와 같이 제안하는 연산기는 기존의 MSB- first 비트-직렬 곱셈기에 비하여 1ADD, 2MUL, 8-bit 레지스터로 약간의 공간복잡도가 증가하지만 시간지연이 mMUL+$2m$ADD에서 $(m+3)$MUL +$(m+3)$ADD으로 감소하며 ADD와 MUL의 시간지연이 같다면 시간지연이 약 30%의 감소한다. 또한, 기존의 LSB-first 비트-직렬 곱셈기와 시간지연이 거의 같으나 레지스터가 반으로 감소하여 공간 복잡도면에서 효율적이다. 따라서 제안하는 방법을 사용하면 기존의 페어링 기반의 암호시스템을 더욱 효율적으로 설계할 수 있으며 제안하는 MSB-first 비트-직렬 곱셈 방법의 경우 삼항 기약다항식을 사용하는 모든 유한체에 효율적으로 적용된다.

참고문헌

[1] P.S.L.M. Barreto, S. Galbraith, C. Ó hÉigeartaigh and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties," *Designs, Codes and Cryptography*, Vol.42, No.3, pp.239-271, 2007.

[2] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott, "Efficient algo rithms for pairing-based crypto systems," *CRYPTO 2002*, LNCS 2442, pp.354-368, Springer-Verlag, 2002.

[3] G. Bertoni, L. Breveglieri, P. Fragneto, and G.

〔표 1〕 기존의 방법과 제안하는 *GF(3)* 연산기 및 *GF(3m)* 덧셈-뺄셈 unified 연산기 복잡도 비교

GF(3) 연산기		Space Complexity	Critical Path Delay
Adder	기존의 방법 [10]	4OR + 3XOR	1OR + 2XOR
	제안하는 방법	1AND + 1OR + 4XOR	1OR + 2XOR
Multiplier	기존의 방법 [10]	2OR + 4AND	1OR + 1AND
	제안하는 방법	1XOR + 2AND	1XOR + 1AND

덧셈-뺄셈 unified 연산기	Space Complexity	Critical Path Delay
기존의 방법 [5]	2MUX + 4OR + 3XOR	1MUX + 1OR + 2XOR
제안하는 방법	2AND + 1OR + 5XOR	1AND + 1OR + 2XOR

〔표 2〕 기존의 방법과 제안하는 *GF(3m)* MSB-first 비트-직렬 곱셈기의 복잡도 비교

덧셈-뺄셈 unified 연산기	Space Complexity			Critical Path Delay	Latency (# clocks)
	ADD	MUL	Register		
기존의 MSB-firsrt 방법 [5]	m + 1	m + 2	2m	1MUL + 2ADD	m
기존의 LSB-firsrt 방법 [5]	m + 1	m + 2	4m	1MUL + 1ADD	m
제안하는 MSB-first 방법	m + 2	m + 4	2m + 8	1MUL + 1ADD	m+3

※ ADD : GF(3) 덧셈기
※ MUL : GF(3) 곱셈기

Pelosi, "Parallel Hardware Architectures for the Cryptographic Tate Pairing," *Proce edings of the Third International Conferenceon Informa- tion Technology : New Generations (ITNG'06)*, pp.186-191, 2006.

[4] G. Bertoni, J. Guajardo, S. Kumar, G. Orlando C. Paar and T. Wollinger. "Efficient $GF(p^m)$ Arith metic Architectures for Cryptogra phic Applications," *CT-RSA 2003*, LNCS 2612, pp.158-175. Springer-Verlag, 2003.

[5] J. Beuchat, M. Shirase, T. Takagi, E. Oka- moto, "An Algorithm for the Eta_T Pairing Calculation in Characteristic Three and its Hardware Implemen tation", *18th IEEE International Symposium on Computer Arithmetic, ARIT H-18*, pp.97-104, 2007.

[6] J. Beuchat, T. Miyoshi, Y. Oyama, E. Oka- moto, "Multiplication over F_{p^m} on FPGA : A Survey", *ARC-2007*, LNCS 4419, pp.214-225, Springer-Verlag, 2007.

[7] I. Duursma and H.-S. Lee, "Tate pairing im- plementation for hyper elliptic curves $y^2 = x^p - x + d$," *Asiacrypt 2003*, LNCS 2894, pp.111-123, Spring er-Verlag, 2003.

[8] S.D. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate pairing," *ANTS V*, LNCS 236 9, pp.324-337, Springer-Verlag, 2002.

[9] P. Grabher and D. Page, "Hardware Accele- ra- tion of the Tate Pairing in Characteristic Three," *CHES 2005*, LNCS 3659, pp.398- 411, Springer-Verlag, 2005.

[10] R. Granger, D. Page, and M. Stam, "Hard- ware and software normal basis arithmetic for pairing based cryptography in character istic three," *IEEE Transactions on Computers*, Vol.54, No.7, pp.852- 860, July 2005.

[11] T. Kerins, W. Marnane, E. Popovici, P. S. L. M. Barreto "Efficient Hardware for the Tate Pairing Calculation in Character istic Three," *CHES 2005*, LNCS 3659, pp.398-411, Springer- Verlag, 2005.

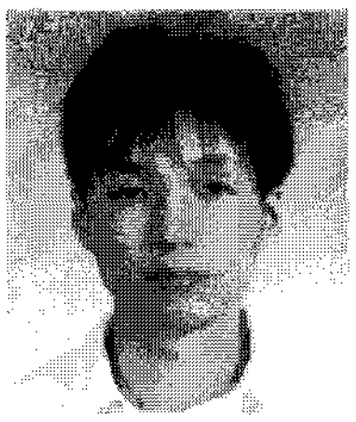
[12] T. Kerins, E. M. Popovici and W. P. Mar- nane. "Algorithms and Archi tectures for use in FPGA impleme ntations of Identity Based Encry ption Schemes," *FPL 2004*, LNCS 3203, pp.74-83, Springer-Verlag, 2004.

[13] S. Kwon, "Efficient Tate pairing computation for elliptic curves over binary fields," *ACISP 2005*, LNCS 3574, pp.134-145, Springer- Verlag, 2005

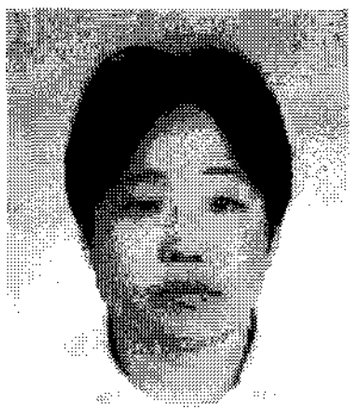
[14] V. S.Miller. Short programs for functions on curves. Unpublished manuscript, 1986. http://crypto.sta nford.edu/miller/miller.pdf.

[15] D. Page and N. Smart "Hardware Imple- men- tation of Finite Fields of Characteristic Three," *CHES 2002*, LNCS 2523, pp.529 -539, Springer- Verlag, 2003.

〈著者紹介〉

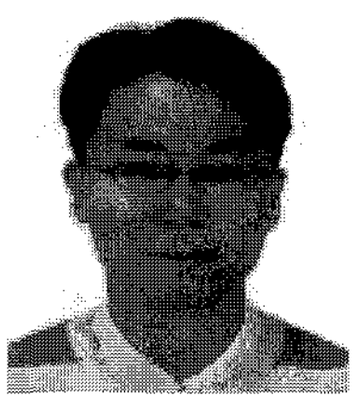


장 남 수 (Nam Su Chang) 학생회원
2002년 2월 : 서울 시립대학교 수학과 이학사
2004년 8월 : 고려대학교 정보보호 대학원 공학석사
2005년 2월~현재 : 고려대학교 정보경영공학전문대학원 박사과정
<관심분야> 암호칩 설계 기술, 부채널 공격, 공개키 암호 알고리즘, 공개키 암호 암호분석

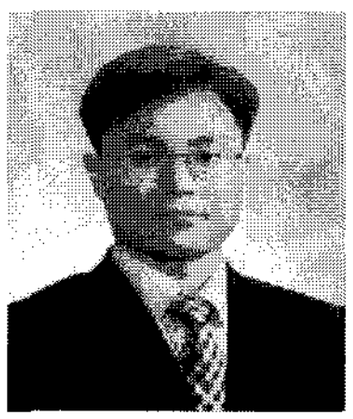


김 태 현 (Tae Hyun KIM) 학생회원
2002년 2월 : 서울 시립대학교 수학과 이학사
2004년 8월 : 고려대학교 정보보호 대학원 공학석사
2005년 2월~현재 : 고려대학교 정보경영공학전문대학원 박사과정
<관심분야> 부채널 공격, 공개키 암호 알고리즘, 암호칩 설계 기술

김 창 한 (Chang Han Kim) 종신회원
1985년 2월 : 고려대학교 수학과 학사
1987년 2월 : 고려대학교 수학과 석사
1992년 2월 : 고려대학교 수학과 박사
<관심분야> 정수론, 공개키암호, 암호프로토콜



한 동 국 (Dong-Guk Han) 정회원
1999년 : 고려대학교 수학과 졸업(학사)
2002년 : 고려대학교 수학과 석사 (이학석사)
2005년 : 고려대학교 정보보호대학원 박사 (공학박사)
2004년 4월~2005년 4월 : 일본 Kyushu Univ., 방문연구원
2005년 4월~2006년 4월 : 일본 Future Univ.-Hakodate, Post.Doc.
2006년 6월~현재 : 한국전자통신연구원 정보보호연구단 선임연구원
<관심분야> 공개키 암호시스템 안전성 분석 및 고속 구현, 부채널 분석, RFID/USN 정보보호 기술



김 호 원 (Ho Won Kim) 종신회원
1993년 2월 : 경북대학교 전자공학과 졸업(학사)
1995년 2월 : 포항공과대학교 전자전기공학과 석사(공학석사)
1999년 2월 : 포항공과대학교 전자전기공학과 박사(공학박사)
1998년 12월~2008년 2월 : 한국전자통신연구원(ETRI) 정보보호연구단 선임연구원/팀장
2008년 3월~현재 : 부산대학교 정보컴퓨터공학부 조교수
<관심분야> RFID/USN 정보보호 기술, 타원곡선 및 초타원곡선 암호 이론, VLSI 설계, embedded system