情報保護學會論文誌
第13券 第3號, 2003. 6

$GF(2^n)$에서의 직렬-병렬 곱셈기 구조*

정 석 원**, 윤 중 철**, 이 선 옥**

Design of Serial-Parallel Multiplier for $GF(2^n)$

Seok Won Jung**, Joong Chul Yoon**, Seon-Ok Lee**

요 약

요즘 암호시스템을 효율적으로 수행하는 하드웨어의 개발이 관심의 대상이 되고 있다. 암호시스템의 효율적인 수행은 연산기의 효율적인 연산이 뒷받침되어야 한다. 특히 유한체 $GF(2^n)$에서의 곱셈기는 여러 연산 중에서 효율성이 고려되어야 할 핵심적인 연산이다. 이 논문에서는 유한체에서의 곱셈기를 시간 복잡도(time complexity)와 하드웨어 복잡도(size complexity) 사이의 교환(trade-off)을 고려하여 기존 곱셈기[5][12]의 하드웨어 복잡도인 #AND(AND gate 수) = n^2, #XOR(XOR gate 수) = n^2-1 보다 개선된 #AND = $n\lceil \frac{n}{2} \rceil$, #XOR = $n(\lceil \frac{n}{2} \rceil +1)-\delta_n$ (n이 짝수이면 $\delta_n=1$, n이 홀수이면 $\delta_n=0$)이고 두 클럭 내에 결과를 얻을 수 있는 직렬-병렬 곱셈기를 제안한다. 우리는 기존의 논문에서 제안된 곱셈기와 구조를 달리하여 공간의 제약이 있는 하드웨어에 적합한 효율적인 연산기의 구현방안을 제시한다.

ABSTRACT

Recently, an efficient hardware development for a cryptosystem is concerned. The efficiency of a multiplier for $GF(2^n)$ is directly related to the efficiency of some cryptosystem. This paper, considering the trade-off between time complexity and size complexity, proposes a new multiplier architecture having $n\lceil \frac{n}{2} \rceil$ AND gates and $n(\lceil \frac{n}{2} \rceil +1)-\delta_n$ XOR gates, where $\delta_n=1$ if n is even, $\delta_n=0$ otherwise. This size complexity is less than that of existing multipliers[5][12] which are n^2 AND gates and n^2-1 XOR gates. While a new multiplier is a serial-parallel multiplier to output a result of multiplication of two elements of $GF(2^n)$ after 2 clock cycles, the suggested multiplier is more suitable for some cryptographic device having space limitations.

Keyword : *유한체, 타원곡선 암호법, 직렬·병렬 곱셈기, 시간복잡도, 공간복잡도*

Ⅰ. 서 론

유한체는 암호학에서 실질적으로 응용되고 있을 뿐만 아니라 코드이론[11]등에도 폭넓게 적용되고 있기 때문에 많은 관심의 대상이 되고 있다. 예를 들어 RSA 암호화 과정[9], Diffe-Hellman 키 교환 프로토콜의 과정[3]을 살펴보면 유한체의 연산이 적용되고 있음을 알 수 있다. 특히 최근에 동일한 안전도를 유지하면서 기타 다른 공개키 암호 시스템보다 짧은 길이의 키를 갖는 타원곡선을 이용한 암호시스템이 암호학적으로 여러 분야의 기능적 프로토콜에 응용되고 있다. 타원곡선 암호시스템[4][7]은 제한적인 조건의 계산능력과 공간에 제약이 있는 스마트카드, 토큰, 단말기 등의 디바이스에서 적합하게 적용될 수 있음

* 본 연구는 한국전자통신연구원 위탁연구과제(2002-S-402) 지원으로 수행하였습니다.
** 고려대학교 정보보호대학원({jsw, jcyoon, seonognim}@cist.korea.ac.kr)

을 보여준다. 이러한 타원곡선 암호 기법을 사용하는 전자 서명 프로토콜을 수행하는 과정과 암호화, 키 동의 프로토콜에서도 역시 유한체의 효율적인 연산은 중요한 역할을 한다[13].

유한체 연산 중에서 덧셈, 뺄셈의 경우에는 각각의 비트 논리 연산만을 사용하기 때문에 하드웨어의 효율성은 AND와 XOR 논리 연산이 복합적으로 결합된 형태의 곱셈 연산에 의해 좌우된다. 유한체에서 곱셈 연산의 효율적인 구현은 암호 시스템의 효율성에 많은 영향을 미치기 때문에 구조적으로 간단하고, 연산하는 시간이 효율적으로 개선되도록 하는 연구가 진행되고 있다.

하드웨어 곱셈기는 구조적인 관점에서 직렬구조, 병렬구조 그리고 직렬-병렬 구조로 나눌 수 있다. 직렬구조는 쌍대기저(dual basis) 또는 정규기저(normal basis)를 이용한 연구가 이루어져 왔으며[2][1], 병렬구조는 표준기저(standard basis), 정규기저 등 다양한 기저에 대한 연구가 이루어졌다. 그러나 최근에는 표준기저가 아닌 기저를 사용할 때, 추가적으로 기저변환 하드웨어가 필요함으로 표준기저를 이용하여 시간 복잡도와 하드웨어 복잡도의 효율성을 최적화하려는 논문들이 다수 나오고 있다[5][12]. 직렬-병렬 구조의 하드웨어는 대부분 유한체의 확장 차수가 합성수일 때에 부분체(subfield)의 원소에 대한 연산은 병렬로 처리하고 전체적으로는 직렬로 처리하는 방식의 연구가 진행되어 왔다[8]. 그러나 [10]에서 Smart가 언급했듯이 차수가 합성수인 어떤 확장체에서는 ECDLP 문제가 Weil descent 방법을 이용하여 분석되었다. 이러한 결과로 ECDLP에 안전성을 근거한 암호시스템을 구축하기 위해서는 유한체를 선택할 때 차수에 대한 조건을 확인할 필요가 있다.

본 논문에서는 부분체를 이용하는 방법이 아닌 다른 방법으로 직렬-병렬 구조가 가능한 하드웨어 구조를 소개한다. 이 때 병렬 구조는 Mastrovito 방식의 곱셈기 구조이다. 또한 본 논문에서 사용하는 기약다항식(irreducible polynomial)은 삼항 기약다항식(trinomial)을 사용하고, 곱셈 연산에서 한 원소를 두 부분으로 나누어 표현한 후 이들을 각각 다른 한 원소에 곱하는 방법을 택한다. 이 방법은 모듈러감산(modular reduction) 연산이 한 번만 발생하기 때문에 Mastrovito 행렬을 쉽게 구할 수 있으며 공간복잡도와 시간복잡도를 효율적으로 구성할 수 있다.

이 방법은 병렬 곱셈기와 비교할 때 공간복잡도가 절반 정도 줄어드는 반면, 시간복잡도가 두 배 늘어나는 교환이 있다. 앞에서도 언급했듯이 타원곡선 알고리즘은 차세대 공개키 알고리즘으로 환경이 제약적인 스마트카드, 스마트카드 리더기, 휴대폰 단말기 등에 적용이 가능하다. 또한 RSA 알고리즘과 더불어 타원곡선 알고리즘이 여러 응용 분야에 사용될 가능성이 높아짐에 따라 이들 알고리즘을 통합한 칩의 설계가 필요한 시점이다. 이런 면에서 본 논문에서 제안하는 직렬-병렬 구조는 병렬구조와 직렬구조의 중간적인 위치에서 공간복잡도와 시간복잡도의 교환을 고려해야하는 환경에 적용될 수 있는 구조를 가짐을 시사한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 유한체 및 삼항 기약다항식의 기본적인 특성을 살펴본다. 3장에서는 $GF(2^n), n=2m$인 경우에 삼항 기약다항식을 이용한 병렬 곱셈기의 하드웨어 구조에 대해 알아본다. 4장에서는 3장의 결과를 기반으로 직렬-병렬 곱셈기를 제시한다. 5장에서는 직렬-병렬 곱셈기의 복잡도에 대해 살피고 6장 결론에서는 $n=2m+1$인 경우를 포함하여 병렬구조곱셈기와 제안하는 곱셈기의 효율성을 비교 분석하여 제안한 곱셈기가 갖는 장점과 단점에 대해서 기술하고자 한다.

II. 유한체 및 삼항 기약다항식

유한체는 덧셈과 곱셈에 대해서 결합법칙과 교환법칙, 분배법칙이 성립하고, 덧셈에 대한 항등원과 역원, 곱셈에 대한 항등원과 역원을 가지는 유한개의 원소로 구성된 집합이다[6]. 유한체 위에서 두 원소의 덧셈을 하드웨어로 구성하면, 각 비트를 XOR 것으로 나타난다. 그러나 유한체 위의 곱셈 연산은 덧셈과 달리 XOR와 AND가 연관되어 나타나는 복잡한 과정으로 이루어진다.

유한체 $GF(2^n)$는 $GF(2)$ 위의 n차 기약 다항식 $p(x)$에 대해 $GF(2)[x]/(p(x))$로 생각할 수 있다. 그러므로, 유한체 $GF(2^n)$의 임의의 원소는 $n-1$차 다항식으로 표현된다. 즉, 유한체의 두 원소 $A(x)$, $B(x) \in GF(2^n)$ 는

$$A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0,$$
$$B(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1x + b_0$$

으로 표현된다. 여기에서 $0 \le i, j \le n-1$ 에 대해 $a_i, b_j \in GF(2)$ 이다. 또한 이를 간단히 벡터 형식으

로 $A=(a_{n-1}, a_{n-2}, \cdots, a_1, a_0)$ 로 표현한다.

일반적으로 연산의 효율성을 위해서 유한체 $GF(2^n)$을 생성하는 n차 기약 다항식 $p(x)$을 삼항 기약다항식으로 택한다.

정의1

$p(x) \in GF(2)[x]$인 n차 다항식 중 항의 개수가 세 개인 기약다항식을 $GF(2)$ 위의 삼항 기약다항식(trinomial)이라 한다.

기약다항식 $p(x)$는 $GF(2^n)$의 원소 $A(x)$, $B(x)$의 곱셈연산의 결과가 유한체 $GF(2^n)$의 원소가 되도록 모듈러 감산 연산에 쓰인다.

본 논문에서는 삼항 기약다항식 $p(x)=x^n+x^k+1$에서 $k<\frac{n}{2}$로 가정한다. 삼항 기약다항식이 존재하는 경우 이러한 다항식은 항상 존재한다[13].

Ⅲ. $2m\times m$ 병렬 곱셈기($n=2m$)

유한체의 차수가 $n=2m$ 이라 하면 $A(x)$, $B(x)$는 각각 다음과 같이 표현할 수 있다.

$$A(x)=a_{2m-1}x^{2m-1}+a_{2m-2}x^{2m-2}+\cdots+a_1x+a_0,$$
$$B(x)=b_{2m-1}x^{2m-1}+b_{2m-2}x^{2m-2}+\cdots+b_1x+b_0.$$

이 때 $A(x)B(x) \bmod p(x)$ 는

$$\begin{aligned} &A(x)B(x) \bmod p(x) \\ &=\{[A(x)B_1(x) \bmod p(x)]x^m+ \\ &\quad [A(x)B_0(x) \bmod p(x)]\} \bmod p(x), \end{aligned}$$

여기서

$$B(x)=B_1(x)x^m+B_0(x),$$
$$B_1(x)=b_{2m-1}x^{m-1}+b_{2m-2}x^{m-2}+\cdots+b_{m+1}x+b_m$$
$$B_0(x)=b_{m-1}x^{m-1}+b_{m-2}x^{m-2}+\cdots+b_1x+b_0.$$

이 때 $A(x)B_1(x) \bmod p(x)$, $A(x)B_0(x) \bmod p(x)$는 $2m$차 다항식과 m차 다항식의 모듈러 곱셈(modular multiplication)이다. 따라서 $2m\times m$ 곱셈기(multiplier)를 통해 위의 두 계산을 구현할 수 있고 $A(x)B(x) \bmod p(x)$는 $A(x)B_1(x) \bmod p(x)$, $A(x)B_0(x) \bmod p(x)$ 결과값을 모듈라 연산을 통해 구할 수 있다. 따라서 $2m\times m$ 곱셈기을 통해 다시 $2m\times 2m$ 곱셈기를 만들 수 있다.

먼저 모듈러 감산을 하기 전 $A(x)$와 $B_0(x)$의 곱셈 결과는 x^{2m}부터 x^{3m-2}항을 갖는다. 이 부분은 삼항 기약다항식 $p(x)$에 의해 모듈러 감산이 되어야 하는데 k에 대한 가정($k<\frac{n}{2}$)에 의해 차수 $2m$이상의 항은 한번의 모듈러 감산만으로 $n(=2m)$이하의 차수로 낮출 수 있다. 왜냐하면 $A(x)B_0(x)$의 최대 차수인 x^{3m-2} 는 $p(x)$에 의해 x^{m-2}와 x^{m+k-2} 로 나누어 지는데 이 때 $k<\frac{n}{2}$ 이므로 $m+k-2<2m$이다.

$A(x)$와 $B_0(x)$의 곱셈 결과에서 x^{2m}부터 x^{3m-2} 항에 대한 모듈러감산은 구체적으로 다음과 같다.

$$\begin{aligned} x^{2m} &= 1 \quad + x^k, \\ x^{2m+1} &= x \quad + x^{k+1}, \\ &\vdots \\ x^{3m-3} &= x^{m-3} + x^{m+k-3}, \\ x^{3m-2} &= x^{m-2} + x^{m+k-2}. \end{aligned} \tag{1}$$

이제 (1)을 기반으로 $2m\times m$ 병렬 곱셈기를 구해보자.

$$\begin{aligned} &A(x)B_0(x) \bmod p(x)=C(x) \\ &=c_{2m-1}x^{2m-1}+c_{2m-2}x^{2m-2}+\cdots+c_1x+c_0 \end{aligned}$$

라 하자. 일단 $A(x)$와 $B_0(x)$의 곱셈에서 x^{2m} 이상의 항을 고려하지 않는다면

$$\begin{aligned} c_0 &= b_0a_0, \\ c_1 &= a_1b_0+a_0b_1, \\ &\vdots \\ c_{2m-1} &= a_{2m-1}b_0+a_{2m-2}b_1+ \\ &\quad \cdots+a_{m+1}b_{m-2}+a_mb_{m-1} \end{aligned}$$

이다.

이 부분은 아래 행렬식 (2)에서 T와 B의 곱으로 나타낼 수 있다.

x^{2m}이상의 항은 식 (1)에서 각각 두 항으로 나누어진다. 이 때 나누어진 첫 번째 항만을 고려하면 c_0부터 c_{m-2}은 다음과 같이 바뀐다($A(x)$와 $B_0(x)$

의 곱셈의 최고차 항이 x^{3m-2} 이고 이를 모듈라 감산한 항은 $m-2$ 차이다).

$$c_0=(b_0a_0)+(a_{2m-1}b_1+\cdots+a_{m+1}b_{m-1}),$$
$$c_1=(a_1b_0+a_0b_1)+(a_{2m-1}b_2+\cdots+a_{m+2}b_{m-1}),$$
$$\vdots$$
$$c_{m-3}=c_{m-3}+a_{2m-1}b_{m-1}.$$

이 부분은 다시 아래 행렬식 (2)에서 $(T+U)$ 와 B의 곱으로 나타낼 수 있다. 마지막으로 식 (1)에서 두 번째 항은 모듈러 감산 후에 각 계수가 c_k 부터 c_{m+k-2} 에 더해지므로 최종적으로 행렬식 (2)를 얻게 된다.

$$C=(T+U+V)B, \tag{2}$$

$$C=\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_k \\ \vdots \\ c_{m-2} \\ \vdots \\ c_{m+k-2} \\ \vdots \\ c_{2m-1} \end{pmatrix}, \quad B=\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \\ \vdots \\ b_{m-2} \\ \vdots \\ b_{m-1} \end{pmatrix},$$

$$T=\begin{pmatrix} a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & \cdots & 0 & 0 \\ & \vdots & & & \vdots & \\ a_k & a_{k-1} & a_{k-2} & \cdots & 0 & 0 \\ & \vdots & & & \vdots & \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & 0 \\ & \vdots & & & \vdots & \\ a_{m+k-2} & a_{m+k-3} & a_{m+k-4} & \cdots & a_k & a_{k-1} \\ & \vdots & & & \vdots & \\ a_{2m-1} & a_{2m-2} & a_{2m-3} & \cdots & a_{m+1} & a_m \end{pmatrix},$$

$$U=\begin{pmatrix} 0 & a_{2m-1} & a_{2m-2} & \cdots & a_{m+2} & a_{m+1} \\ 0 & 0 & a_{2m-1} & \cdots & a_{m+3} & a_{m+2} \\ & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & a_{m+k+2} & a_{m+k+1} \\ & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & a_{2m-1} \\ & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix},$$

$$V=\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ & \vdots & & & \vdots & \\ 0 & a_{2m-1} & a_{2m-2} & \cdots & a_{m+2} & a_{m+1} \\ & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & a_{k+4} & a_{k+3} \\ & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & a_{2m-1} \\ & \vdots & & & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

이 때 C는 $(2m\times1)$ 행렬이고 B는 $(m\times1)$ 행렬이고 T, U, V는 모두 $(2m\times m)$ 행렬이다. 따라서 $A(x)B_0(x) \bmod p(x)$의 계수 c_0부터 c_{2m-1}은 $T+U+V$의 각 행의 원소와 B행렬의 내적(inner product)으로 얻을 수 있다.

이 때 $T+U+V$의 각 행은 규칙적인 성질을 가지고 있는데 구체적으로 다음과 같다.

$$A^*=(a_{2m-1}, a_{2m-2}, \cdots, a_{m+1}, a_m, \cdots, a_1, a_0)$$

는 다항식 $A(x)$를 벡터 형식으로 나타낸 것이라 하고

$$Z=(a_{k-1}, \cdots, a_0, a_{2m-1}, \cdots, a_{m+k+1}, \cdots, a_k)+(a_{2m-1}, a_{2m-2}, \cdots, a_{m+2}, a_{m+1}, 0, \cdots, 0)$$

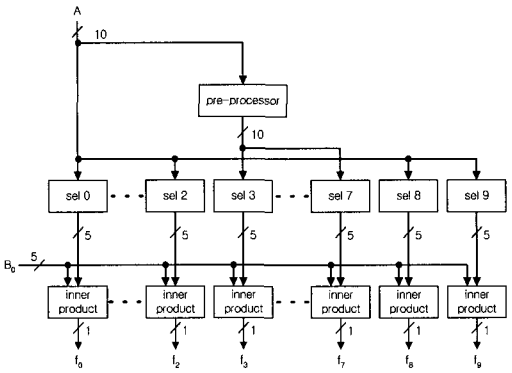
라 하자. 또한 A_i^*는 A^*를 $i+1$ 번 오른쪽으로 순환이동한 벡터라 하고 Z_{i-k+1} 은 Z를 $i-k+1$ 번 오른쪽으로 순환이동한 벡터를 나타낸 것이라 하자. 그러면 행렬 $T+U+V$의 각 행은 다음과 같이 분류 될 수 있다.

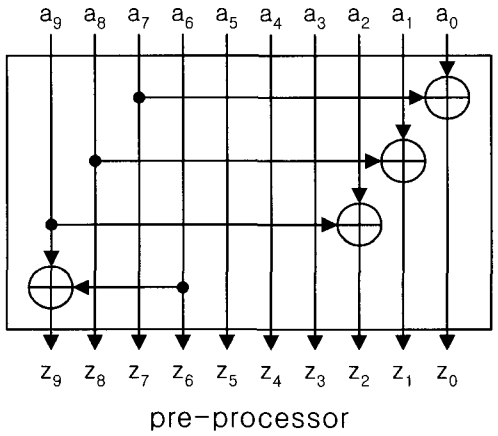
① $0\le i\le k-1$, $m+k\le i\le 2m-1$인 행은 각 각 A_i^* 벡터의 앞에서 m개의 원소와 같다.

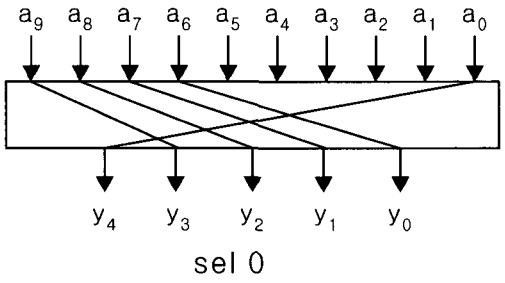
② $k\le i\le m+k-1$인 행은 각각 Z_{i-k+1} 벡터의 앞에서 m개의 원소에서 얻어진다.

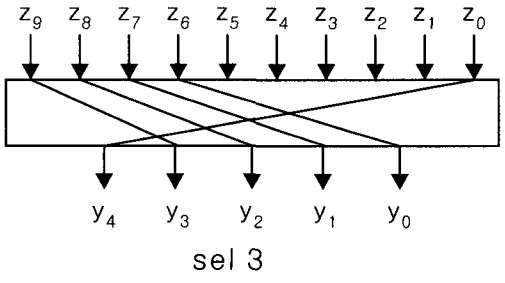
따라서 $A(x)B_0(x) \bmod p(x)$을 계산하기 위한 $2m\times m$ 병렬 곱셈기는 A^*와 Z의 재배치와 내적에 의해 얻어질 수 있다.

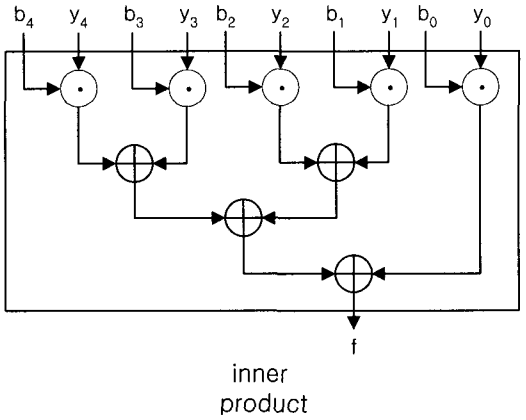
[그림 1]은 위에서 제시한 방법을 이용하여 $GF(2^{10})$에서 10×5 병렬 곱셈기를 나타낸 것이다.











〔그림 1〕 $p(x)=x^{10}+x^3+1$를 기약 다항식으로 갖는 $GF(2^{10})$의 10×5 병렬곱셈기

Ⅳ. $2m\times2m$ 곱셈기

이 장에서는 앞에서 제시한 $2m\times m$ 병렬 곱셈기를 바탕으로 $2m\times2m$ 곱셈기 구조를 제시한다.

$$A(x)B_1(x) \bmod p(x) = f_{2m-1}x^{2m-1}+\cdots+f_1x+f_0,$$
$$A(x)B_0(x) \bmod p(x) = g_{2m-1}x^{2m-1}+\cdots+g_1x+g_0.$$

라 하자. 이 때 $A(x)B(x) \bmod p(x)$는

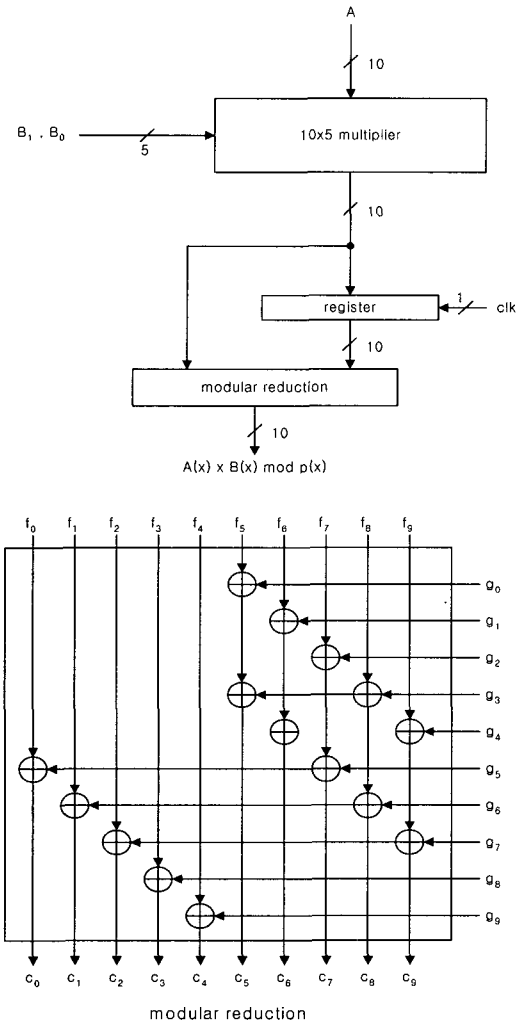
$$A(x)B_1(x)x^m + A(x)B_0(x) \bmod p(x)$$

이므로 $A(x)B_1(x)x^m$ 에 대한 모듈러감산이 필요하다. 구체적으로

① $f_{2m-1}, f_{2m-2}, \cdots, f_{m+1}, f_m$은 x^m이 곱해져 차수가 $(2m-1)$보다 커진 항의 계수이므로 $p(x)$에 의해 모듈라 감소되고 각각 $g_{m-1}, g_{m-2}, \cdots, g_1, g_0$와 $g_{m+k-1}, g_{m+k-2}, \cdots, g_{k+1}, g_k$ 에 XOR 된다.

② $f_{m-1}, f_{m-2}, \cdots, f_1, f_0$은 $g_{2m-1}, g_{2m-2}, \cdots, g_{m+1}, g_m$와 XOR 된다.

따라서 $2m\times m$ 병렬 곱셈기를 기반으로 한 $2m\times2m$ 곱셈기는 단지 추가로 XOR gate 만 필요로 한다. 그림 2는 그림 1에서 제시한 10×5 병렬 곱셈기를 기반으로 한 10×10 곱셈기이다.



〔그림 2〕 $GF(2^{10})$에서 10×5 병렬곱셈기를 기반으로 한 10×10 곱셈기

V. 복잡도 분석

먼저 $2m\times m$ 병렬 곱셈기는 실제 계산 전에 Z를 만들어야 하므로 $m-1$개의 2-input XOR 게이트가 필요하다. $T+U+V$와 B의 행렬 곱셈에서 $2m^2$ AND 게이트와 $2m(m-1)$ XOR 게이트가 필요하므로 $2m\times m$ 병렬 곱셈기의 하드웨어복잡도는

$$\begin{aligned}\#AND &= 2m^2=2\left(\frac{n}{2}\right)^2 ,\\ \#XOR &= 2m(m-1)+m-1=2m^2-m-1\\ &= 2\left(\frac{n}{2}\right)^2-\frac{n}{2}-1\end{aligned}$$

이다. T_X, T_A를 각 각 XOR 게이트와 AND 게이트의 시간지연이라 하면 Z를 만들 때 XOR 게이트의 시간지연 T_X가 소요되고 행렬 계산에서 T_A와 $\lceil \log_2 m \rceil T_X$가 필요하다. 따라서 $2m\times m$ 병렬 곱셈기의 시간복잡도는

$$\begin{aligned}&T_A+T_X+\lceil \log_2 m \rceil T_X\\ &= T_A+(1+\lceil \log_2 m \rceil)T_X\\ &= T_A+\left(1+\lceil \log_2 \frac{n}{2} \rceil\right)T_X\end{aligned}$$

이다.

$2m\times2m$ 곱셈기는 마지막에 모듈러감산 연산이 필요하였다. 이 때 [그림 2]에서 보듯이 일부 g_i는 두 번 XOR되므로 $2T_X$의 시간지연과 $3m$개의 XOR 게이트가 필요하다. 따라서 $2m\times2m$곱셈기의 하드웨어복잡도는

$$\begin{aligned}\#AND &= 2m^2=2\left(\frac{n}{2}\right)^2 ,\\ \#XOR &= 2m^2-m-1+3m\\ &= 2\left(\frac{n}{2}\right)^2+n-1\end{aligned}$$

이다.

그러나 두 클럭 후에 곱셈 결과를 얻는 구조이므로 시간복잡도는 한번 계산될 때의 시간 복잡도의 두 배가 된다. 따라서

$$2\left(T_A+\left(3+\lceil \log_2 \frac{n}{2} \rceil\right)T_X\right)$$

이다.

VI. 결 론

본 논문에서는 $n=2m$ 경우에 $GF(2^n)$에 대한 새로운 곱셈기를 제시하였다. 이러한 구조는 $n=2m+1$인 경우에도 쉽게 적용될 수 있는데 $B(x)$를 다음과 같이 나누고

$$B(x)=B_1(x)x^{m+1}+B_0(x),$$

$$B_0(x) = b_m x^m + b_{m-1}x^{m-1} + \cdots + b_1 x + b_0$$
$$B_1(x) = 0x^m + b_{2m}x^{m-1} + \cdots + b_{m+2}x + b_{m+1}$$

$(2m+1)\times(m+1)$ 병렬 곱셈기를 사용하여 구할 수 있다.

$n=2m+1$인 경우의 복잡도 분석은 아래 [표 1, 2]에 $n=2m$인 경우와 함께 제시하였다. 지금까지 본 논문에서 제시한 것과 같은 곱셈기 구조는 없어 [5][12]에서 제시한 병렬 곱셈기와 비교하였다.

〔표 1〕 하드웨어 복잡도 비교

	하드웨어 복잡도
제안 (홀수)	#AND $= n\lceil \frac{n}{2} \rceil$ #XOR $= n\left(\lceil \frac{n}{2} \rceil + 1\right)$
제안 (짝수)	#AND $= n\frac{n}{2}$ #XOR $= n\left(\frac{n}{2}-1\right)-1$
Koc[5]	#AND $= n^2$ #XOR $= n^2-1$
Wu[12]	#AND $= n^2$ #XOR $= n^2-1$

〔표 2〕 시간 복잡도 비교

	시간 복잡도
제안 (홀수)	$2\left(T_A+\left(3+\lceil \log_2 \lceil \frac{n}{2} \rceil \rceil\right)T_X\right)$
제안 (짝수)	$2\left(T_A+\left(3+\lceil \log_2 \frac{n}{2} \rceil\right)T_X\right)$
Koc[5]	$T_A+(2+\lceil \log_2 n \rceil)T_X$
Wu[12]	$T_A+\left(2+\lceil \log_2 \lceil \frac{n+k-1}{2} \rceil \rceil\right)T_X$

[표 1, 2]에서 보듯이 제안된 곱셈기 구조는 기존의 구조에 비해 2배의 연산 시간이 소요되지만 면적이 1/2로 줄어드는 장점이 있다. 그런데 타원곡선 알고리즘은 차세대 공개키 알고리즘으로 환경이 제약적인 스마트카드, 스마트카드 리더기, 휴대폰 단말기 등에 적용이 가능하다. 또한 RSA 알고리즘과 더불어 타원곡선 알고리즘이 여러 응용 분야에 사용될 가능성이 높아짐에 따라 이들 알고리즘을 통합한 칩의 설계가 필요한 시점이다. 이런 면에서 본 논문에서 제안하는 직렬-병렬 구조는 병렬구조와 직렬구조의 중간적인 위치에서 공간복잡도와 시간복잡도의 교환을 고려해야하는 환경에 적용될 수 있는 구조를 가짐을 시사한다.

참 고 문 헌

[1] G. B. Agnew, R. C. Mullin, I. Onyszchuk and A. Vanstone, "An implementation for a fast public key cryptosystem" J. Cryptology, 3, pp.63~79, 1991.

[2] E. R. Berlekamp, "Bit-serial Reed- Solomon encoders", IEEE Trans. IT, 28, pp.869~974, 1982.

[3] W. Diffie and M. Hellman, "New direc- tion in cryptography", IEEE Trans. Inform. Theory , vol. IT-22, pp.644~654, 1976.

[4] N. Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 48(1987).

[5] B. Sunar and C. K. Koc, "Mastrovito Multiplier for all trinomials." IEEE Trans., Compt., 48(5), pp.522~527, 1999.

[6] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. New York: Cambridge Univ. Press, 1994.

[7] V. Miller, Uses of elliptic curves in cryp tography. LNCS, 218(1986).

[8] C. Paar, P. Fleischmann and P. Soria-Rodriguez, "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents", IEEE Trans. on Compu, vol. 48, no.10, pp.1025~1034, 1999.

[9] R. L. Rivest, A. Shamir and L. Adelman, "A method for obtaining digital signatures and publickey cryptosystems", Comm. ACM. vol. 21, pp.120~126, 1978.

[10] N. P. Smart, "How Secure Are Elliptic Curves over Composite Extension Fields?", EUROCRYPT 2001, LNCS 2045, pp.30~39, 2001.

[11] S. A. Vanston and P. C. V. Oorschot, *An introduction to error correcting codes with applications*, Boston: Kluwer Academic, 1999.

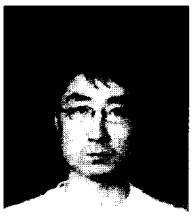
[12] H. Wu, "Low-complexity Arithmetic in finite field using polynomial basis", CHES'99, pp.357~371, 1999.

[13] IEEE P1363. *Standard for Public-Key Cryptosystem*. 1999, Draft Version 13.

〈著者紹介〉

정 석 원 (Seok Won Jung) 정회원

1991년 2월 : 고려대학교 수학과 졸업
1993년 2월 : 고려대학교 수학과 석사
1997년 2월 : 고려대학교 수학과 박사
1997년 5월~1997년 11월: 한국전자통신연구원 박사후연구원
1999년 2월~2001년 2월 : (주)텔리맨 책임연구원
2002년 3월~현재 : 고려대학교 정보보호대학원 조교수
<관심 분야> 암호칩 설계, 부채널공격 방법론, 공개키 암호알고리즘, 디지털 방송 보안



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

이 선 옥 (Seon-Ok Lee) 학생회원

2002년 2월 : 서울시립대 이학사
2002년 3월~현재 : 고려대학교 정보보호대학원 석상과정
<관심 분야> 공개키암호, 유한체연산,부채널공격 방법론,