논문 2005-42SD-5-4

확장체 $GF(p^n)$에서 효율적인 다항식 곱셈 방법

(Efficient Polynomial Multiplication in Extension Field $GF(p^n)$)

장 남 수*, 김 창 한*

(Namsu Chang and Chang Han Kim)

요 약

확장체 $GF(p^n)$의 구성에서 차수와 다항식 곱셈 방법은 밀접한 관련을 가진다. 기존의 다항식 곱셈 방법인 KO] 및 MSK 방법은 효율적으로 계수-곱셈 연산량을 줄인다. 그러나 이들 방법을 이용하여 확장체 곱셈을 구성할 경우, 일반적으로 해당하는 분할 방법의 배수가 되도록 패딩(Padding)하여 구성하지만 이에 대한 기준이 모호하며 계수-곱셈의 연산량이 최소가 되도록 패딩하는 방법 또한 제안되지 않았다. 본 논문에서는 확장체 곱셈을 효율적으로 구성할 수 있는 기본적인 성질과 계수-곱셈의 연산량이 최소가 되는 다항식 차수를 찾는 알고리즘을 제안한다. 본 논문에서 제안하는 알고리즘을 적용하면 기존의 방법을 그대로 적용하여 구성할 때 보다 확장체의 차수가 증가할수록 더 많은 계수-곱셈 연산량을 줄일 수 있다. 따라서 본 논문의 결과는 스마트 카드 등 작은 공간 복잡도를 요구하는 병렬처리 곱셈기에 효율적으로 적용될 수 있다.

Abstract

In the construction of an extension field, there is a connection between the polynomial multiplication method and the degree of polynomial. The existing methods, KO and MSK methods, efficiently reduce the complexity of coefficient-multiplication. However, when we construct the multiplication of an extension field using KO and MSK methods, the polynomials are padded with necessary number of zero coefficients in general. In this paper, we propose basic properties of KO and MSK methods and algorithm that can reduce coefficient-multiplications. The proposed algorithm is more reducible than the original KO and MSK methods. This characteristic makes the employment of this multiplier particularly suitable for applications characterized by specific space constrains, such as those based on smart cards, token hardware, mobile phone or other devices.

Keywords : Karatsuba-Ofman Algorithm, Multi-Segment Karatsuba Algorithm, Successive Extension, Parallel Multiplier

I. 서 론

인수분해 문제, 이산대수 문제 등을 기반으로 하는 대다수의 공개키 암호 시스템은 지수 연산을 기반으로 한다. 지수 연산은 알고리즘에 따라 장단점을 가지며, 이들 알고리즘의 효율성은 곱셈 연산의 연산량을 기준으로 한다. 따라서 유한체 곱셈 연산의 효율적인 구성은 하드웨어 및 소프트웨어 분야에서 오랜 기간 동안 화제가 되어왔으며 현재까지도 활발히 연구 중이다. 알고리즘의 효율성은 자원의 사용이나 복잡도 등이 기준이 되며 이중에서 복잡도는 알고리즘의 가치를 판단하는 가장 큰 기준이다. 일반적으로 유한체의 크기를 n이라하면 대략의 복잡도를 $O(n), O(n^2)$ 등으로 표현한다.

확장체 $GF(p^n)$에서 복잡도는 계수간의 덧셈, 뺄셈, 곱셈 등의 연산량을 말한다. 이들 중에서 곱셈의 연산량은 가장 큰 비중을 차지하므로 복잡도 측정 시 많은 비중을 차지한다. 아래의 표는 Pentium IV 2.8GHz에서 VisualC++로 측정한 GF(p)의 속도이며 단위는 μsec 이다.

Prime Size	mul	sqr	add	sub
175bit	0.672	0.656	0.034	0.047
89bit	0.187	0.162	0.031	0.043
62bit	0.109	0.102	0.031	0.042
31bit	0.0156	0.0063	0.0047	0.0047

* 학생회원, 고려대학교 정보보호대학원
(Center for Information and Security Technologies (CIST), Korea Univ.)
** 정회원, 세명대학교 정보보호학과
(Dept. of Information and Security Semyumg Univ.)
※ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음
접수일자: 2005년1월5일, 수정완료일: 2005년4월18일

일반적인 확장체 p^n의 경우는 계수간의 연산량이 복잡도의 중요한 기준이 되며, $p=2$인 이진체의 경우 하드웨어 분야에서 주로 고려되므로 연산량(게이트 수)와 시간지연을 동시에 고려하여야 하며 이를 시간(Time)과 공간(space) 복잡도라 한다. 본 논문에서는 이진체가 아닌 일반적인 확장체의 경우를 기준으로 고려하며 , 또한 본 논문의 결과를 이용하면 이진체 환경에 대한 분석 또한 용이하다.

일반적인 확장체에서 다항식 곱셈 연산은 다음과 같은 기준으로 나누어 볼 수 있다.

- 기저에 따른 확장체 곱셈 구성
- 곱셈 방법에 따른 확장체 곱셈 구성
- 기약 다항식의 선택에 따른 곱셈 구성

확장체에서 기저는 다항식기저(Polynomial Basis)와 정규기저(Normal Basis)로 나누어 볼 수 있다. 다항식 기저는 소프트웨어와 하드웨어 모두에서 널리 쓰이고 있으나 정규기저의 경우는 제곱의 경우 쉬프트(Shift) 연산으로 간단하게 구성이 되지만 곱셈의 경우 행렬(Matrix) 연산을 기반으로 하므로 하드웨어 환경에는 적합하나 소프트웨어 환경에는 적합하지 않다. 또한 기약다항식의 선택에 따라 모듈러 감산(reduction) 연산시 덧셈 연산량이 결정된다. 따라서 일반적으로 기약다항식의 헤밍 웨이트가 작을수록 효율적으로 구성된다. 하지만, 예외적으로 올원 기약다항식(All-One Polynomial)의 경우 헤밍웨이트는 가장 높지만 이들보다 더욱 효율적으로 구성되는 환경이 존재한다. 현재 삼항 기약다항식, 오항 기약다항식, 올원 기약다항식 등이 주 연구대상이 되고 있다.

다양한 곱셈 방법이 존재하지만 이들 모두는 일반적인 곱셈 방법, 카라슈바(Karatsuba-Ofman, KO) 방법 혹은 이들을 변형한 방법으로 구성되어 있다. 이들 두 가지 방법의 가장 큰 차이점은 계수간의 곱셈 구성 방법의 차이이다. 이들 두 곱셈 방법의 가장 큰 차이점은 계수간의 곱셈을 변형하여 계수간의 곱셈 수는 줄이고 덧셈 수는 늘리는 것이다. 일반적인 확장체 $GF(p^n)$의 경우 계수간의 곱셈이 계수간의 덧셈 연산 보다 복잡도가 훨씬 크므로 계수간의 곱셈 연산량이 줄어들면 효율성이 증가하나 $p=2$인 이진체의 경우 계수간의 곱셈 연산은 덧셈 연산과 같이 한 개의 게이트(Gate)로 구성되므로 곱셈 연산량이 줄어든다고 하여 꼭 효율성이 증가한다고는 말할 수 없다. 본 논문에서는 다항식 곱셈 방법에 대한 분석 결과를 제시하는바 일반적인 다항식 기저를 사용할 경우의 기약 다항식의 선택문제와 본 논문의 결과와는 무관하다.

확장체 곱셉 연산량을 효율적으로 줄이는 방법으로 카라슈바(Karatsuba-Ofman)방법이 있다. 카라슈바 방법의 효율성을 높이는 방법으로 Leone[10]은 최적 반복 횟수를 제안하였다. 그러나 카라슈바 방법은 확장체의 차수가 2의 배수일 때만 적용되는 잠재적인 문제점이 있다. 이러한 문제점은 Ernst[5]가 제안한 다중분할 카라슈바 방법으로 상당부분 보완되었다. 그러나 [5]에서는 다중분할 카라슈바 방법의 복잡도가 제시되지 않아 실질적인 적용에 어려움이 있었다. 그러나 [1]에서 Ernst가 제안한 방법보다 효율적인 다중분할 카라슈바 방법과 복잡도를 제안하였다.

본 논문에서는 [1]와 [10]에서 제안된 다항식 곱셈 방법을 효율적으로 구성하는 알고리즘을 제안한다. 일반적으로 확장체의 차수가 작으면 큰 차수에 비하여 계수-곱셈 및 덧셈 등의 연산량이 작다고 알려져 있으나 본 논몬의 결과에 의하면 확장체의 차수에 따라 다른 결과를 보인다.

Ⅱ. 기존의 확장체 곱셈 구성 방법

1. 다항식 곱셈 방법에 따른 분류

본 절에서 $GF(p)$에서의 n차 기약다항식 $f(x)$에 의하여 생성된 유한체 $GF(p^n)$의 원소간의 곱셈에 관하여 살펴보자. 다항식 기저에 의해 $GF(p^n)$의 원소 $a(x)$와 $b(x)$는

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

와 같이 표현되며 이때 $a_i,\ b_i \in GF(p)$이다. 확장체위의 곱셈은 확장체의 두 원소 $a(x)$와 $b(x)$를 $n-1$차 다항식으로 보고 곱하는 다항식 곱셈 과정과 곱셈의 결과인 $2n-2$차 다항식

$$c(x)=c_0+c_1x+\cdots+c_{2n-3}x^{2n-3}+c_{2n-2}x^{2n-2},\ c_i \in GF(p) \quad (1)$$

를 n차 기약 다항식 $f(x)$로 나눈 나머지를 구하는 과정인 모듈로 축소(Modulo Reduction)로 나뉜다. 효율적인 곱셈을 위해서는 다항식 곱셈 과정과 모듈로 축소과정을 최적화해야 한다. 본 절에서는 기존의 SchoolBook (SB) 곱셈 방법, 카라슈바(Karatsuba-Ofman) 곱셈 방법[10], 다중분할 카라슈바(Multi-Segment Karatsuba)

방법[1]에 대하여 살펴보도록 한다.

본 논문에서는 계수간의 덧셈, 뺄셈, 곱셈 연산량을 #ADD, #SUB, #MUL로 표기하도록 한다. 또한 계수간의 덧셈과 뺄셈 연산의 시간지연의 경우 T_X(XOR 게이트의 시간지연)로 표기하며, 곱셈 연산의 경우는 T_A(AND 게이트의 시간지연)로 표기하며, 전체 시간지연의 경우 T_{TOT}로 표기한다.

가. SB 곱셈 방법

SchoolBook(SB)방법을 이용하여 유한체 원소의 다항식 곱셈을 수행하면 식(1)에서 c_i는 $\sum_{j=0}^{i} a_j \cdot b_{i-j}$이다. 따라서 곱셈의 연산량는 $a(x)$의 전체 계수의 수와 $b(x)$의 전체 계수의 수를 곱한 만큼 필요하므로 $n \times n = n^2$ 번의 계수-곱셈 연산이 필요하며 덧셈의 연산량은 $c(x)$의 $n-1$차를 기준으로 대칭이므로

$$2 \cdot \left(\sum_{i=1}^{n-2} i \right) + (n-1) = n^2 - 2n + 1 = (n-1)^2$$

번의 계수-덧셈 연산이 필요하다. 따라서 연산량은 다음과 같다.

$$\#\mathrm{ADD} = (n-1)^2,$$
$$\#\mathrm{MUL} = n^2 .$$

나. KO 곱셈 방법[10]

$GF(p^n)$에서 기본적인 KO방법은 유한체의 원소를 2중-분할하여 divide-and-conquer방법을 적용한다. $n/2$이 짝수라면 KO는 반복 적용될 수 있다. 따라서 확장체 $GF(p^n)$에서 $n = 2^m \cdot t (m \neq 0)$의 형태일 때 연산이 가능하다. $n-1$차의 다항식에 KO를 직접 적용하면 $\log_2 n$번 반복 수행이 가능하다.

[10]에서 제안된 방법에서 계수-덧셈 연산은 두 과정으로 나누어진다. 첫째는 $n/2$ 비트 다항식 곱셈기의 입력 값의 연산이고, 두 번째는 $n/2$ 비트 다항식 곱셈기의 출력 값의 연산이다. 만약 KO 방법을 1번 반복 수행한다면 첫 번째 과정 $n/2$ 개의 비트 덧셈 $(A_0 + A_1)$, $(B_0 + B_1)$에서 $2(n/2)$개의 XOR 게이트가 필요하고 두 번째 과정 $n/2$차 곱셈기의 출력 값 A_0B_0, A_1B_1, $(A_0 + A_1)(B_0 + B_1)$과 계수간의 덧셈연산에서 $3n-4$개의 XOR 게이트가 필요하다. 따라서 KOA가 m번의 반복 연산을 수행하며 최하위 연산기를 SB 곱셈기를 사용한다고 가정할 때 전체 연산량은 다음과 같다.

$$\#\mathrm{ADD} = 3^m \cdot \mathrm{ADD}_{n/2^m} + (4n/2^m - 1)3^m - (4n-1),$$
$$\#\mathrm{SUB} = 3^m \cdot \mathrm{SUB}_{n/2^m} + (4n/2^m - 1)3^m - (4n-1),$$
$$\#\mathrm{MUL} = 3^m \cdot \#\mathrm{MUL}_{n/2^m} .$$

위의 결과에 의하여 KO 방법을 이용한 병렬 처리 곱셈기는 SB 방법을 이용한 병렬 처리 곱셈기보다 공간 복잡도가 낮다. 부가적으로 [10]에서는 최적화된 KOA의 반복 횟수를 $n/2^k = 4$일 때의 k로 제안하였다.

다. MSK 곱셈 방법[1]

MSK 곱셈방법은 KOA방법을 확장하여 유한체의 원소를 다중으로 분할하여 곱한다. 유한체 $GF(2^n)$에서 $n \pmod k = 0$ 이라 가정하면 k-다중 분할 카라슈바 (MSK_k) 방법으로 두 원소의 곱셈을 수행할 수 있다. 만약 $n \pmod k \neq 0$ 이라면 필요한 만큼의 계수를 0으로 채운다. 본 절에서는 다중분할을 이용한 병렬 처리 곱셈기에 관하여 설명한다. 이는 간단하게 5중 분할(5-Segment)을 예로 설명한다. $GF(2^n)$에서 $n = 5^m t$ $(m \neq 0)$라 가정하면 $a(x)$, $b(x) \in GF(2^n)$는 5중 분할로 세분되며 이를 표현하면 다음과 같다.

$$A_i = a_{\in/5} + \cdots + a_{(i+1)n/5-1} x^{\frac{n}{5}-1},$$
$$B_i = b_{\in/5} + \cdots + b_{(i+1)n/5-1} x^{\frac{n}{5}-1},$$

이때 $0 \leq i \leq 4$이다.

따라서 n비트의 곱셈 $c(x)$는 [1]의 3.2절에 표기된 식과 같이 계산되며 그 결과를 이용하여 m번의 반복 후 하위-곱셈 방법을 SB 방법으로 구성하였을 경우 연산량은 다음과 같음을 알 수 있다.

$$\#\mathrm{ADD} = 14^m (n/5^m - 1)^2 + 16n/3 \cdot ((14/5)^m - 1) - 19/13(14^m - 1),$$
$$\#\mathrm{SUB} = 8n/3 \cdot ((14/5)^m - 1) - 12/13(14^m - 1),$$
$$\#\mathrm{MUL} = (14/25) \cdot n^2,$$

Divide-and-conquer방법은 시간과 공간 복잡도에서 교환이 일어난다. 그러므로 반복횟수가 늘어남에 따라 시간 복잡도는 증가하지만 효과적으로 공간복잡도가 감소됨을 알 수 있다. $n = p^m t$ 라 가정하면 최대 m번의 반복 수행이 가능하다. 하지만 m은 최적화된 반복횟수가 아니다. 왜냐하면 $GF(2^p)$에서 SB 곱셈기가 MSK_p보다 낮은 공간 복잡도를 가지기 때문이다. 그러므로 [5]의 아이디어를 적용하면 $GF(2^{p^m t})$에서 $m \neq 0$이라 가정하면 MSK_p의 최적화된 반복횟수는 $n/p^k \neq p$를 만족하는

표 1. 환경에 따른 곱셈, 제곱, 역원 계산의 연산량
Table 1. Comparison the Complexity of Field Arithmetics with MUL, SQR and INV.

환 경		MUL		SQR		INV		
기약다항식	기 저	ADD	MUL	ADD	MUL	ADD	MUL	INV
x^2+1	다항식기저	5	3	3	2	2	4	1
x^2-2	다항식기저	6	3	2	3	3	4	1
x^2+x+1	ONB TYPE I	4	3	4	2	2	4	1
x^2-x-1	정규기저	4	3	3	3	2	4	1
x^2-x+1	정규기저	4	3	4	2	2	4	1

표 2. 기존의 방법과 SE방법의 장단점 비교
Table 2. Comparing Extension Field Arithmetic between Direct Extension Method and Successive Extension Method.

	$GF(p)\rightarrow GF(p^n)$	Successive Extension
장점	- n번의 모듈러 감산 연산을 수행 - **확장체 차수에 의존하지 않는다.**	- 덧셈(뺄셈) 연산을 최적화 - 다양한 기저와 기약 다항식을 선택.
단점	- 기저 선택에 제약을 받는다. - 기약 다항식의 선택에 제약을 받는다.	- **확장체의 차수가 작은 소수의 곱으로 구성** - 연속적으로 확장할 때마다 모듈러 감산 연산을 수행

최대의 k 값이다.

2. Successive Extension

Successive Extension(SE) 방법은 확장체 $GF(p^n)$에서 n이 작은 소수(2,3,...) 로 구성되었을 경우 장점을 가진다. $GF(p^{2^3})$의 곱셈을 고려할 때,

$$GF(p)\rightarrow GF(p^2)\rightarrow GF((p^2)^2)\rightarrow GF(((p^2)^2)^2)$$

과 같이 연속적으로 계수-곱셈을 수행한다. 따라서 $GF(p^{2^3})$의 경우 일반적으로 구성하면 All-One-Polynomial을 사용할 수 없으나 SE의 경우 사용이 가능하다. 이차 기약다항식에 따른 효율성을 비교하면 표 1과 같다.

SE를 사용한 구성과 DE(Direct Extension)을 사용한 환경을 고려하면 표 2와 같은 장단점을 가진다.

표 2에 비교된 바와 같이 일반적인 확장체에서 SE방법은 DE방법에 비해 비효율적이다. 따라서 본 논문에서는 DE를 고려한 구성만 고려한다. 따라서 SE에 대한 세부적인 결과는 [16]을 참고하도록 한다.

III. 효율적인 다항식 곱셈 구성 방법 제안

본 절에서는 임의의 확장체 $GF(p^n)$에서 다항식 곱셈을 효율적으로 구성하는 방법에 대하여 제안한다. 본 논문은 일반적인 확장체 환경을 고려하므로 계수-곱셈 연산량을 최소화하는 방법을 기술하도록 한다. KO방법을 적용하면 확장체의 차수가 $n=2^m t$ 일 때 t 가 작은 값일수록 효율성이 좋아진다. 즉, 확장체의 차수가 소수이거나 비교적 큰 소수들의 곱으로 구성되어 있으면 실질적으로 효율성이 없다. 하지만 표준문서 등에서 제안하는 확장체의 차수는 안전성을 고려하여 대부분 소수이므로 소수 차수를 가지는 확장체에 대한 대책이 필요하다.

1. KO 방법을 사용한 최적 확장체 차수 선택

KO 방법을 사용한 다항식 곱셈 연산은 다음과 같은 두 가지 성질을 만족한다. 확장체의 차수가 홀수일 때는 KO방법이 적용되지 않으므로 SB 방법을 사용하는 것으로 가정한다.

정리 1. (KO 방법을 다항식 곱셈에 적용할 경우)
기약다항식의 차수를 n라 하고 #MUL을 다항식 곱셈의 계수-곱셈 연산량이라 하자. 만약 n이 짝수이고 $n > 8$ 이면,

$$\#\mathrm{MUL}_{KO}(n) < \#\mathrm{MUL}_{SB}(n-1)$$

이다.

n이 짝수이므로 $n=2t$ 라 하고 이때 t 는 1보다 큰 정수이다. $n-1$차 확장체의 경우는 홀수 차수이므로 SB 방법이 적용되어 계수-곱셈의 연산량은 $\#\mathrm{MUL}_{SB}(n-1)=(2t-1)^2$이다. 그리고 확장체의 차수가 n인 경우는 한번 KO 방법을 적용 가능하므로 $\#\mathrm{MUL}_{KO}(n)=3t^2$이다. 따라서 $\#\mathrm{MUL}_{SB}(n-1) > \#\mathrm{MUL}_{KO}(n)$인 경우를 고려해보자.

$$\begin{aligned}&\#\mathrm{MUL}_{SB}(n-1) > \#\mathrm{MUL}_{KO}(n),\\ &\rightarrow (4t^2-4t+1)-3t^2>0,\\ &\rightarrow t^2-t4+1=(t-2)^2-3>0,\end{aligned}$$

이고 이때 t는 양의 정수이므로 부등식을 만족하기 위해서는 $t > 4$ 이어야 한다. 즉 n이 8보다 큰 경우 $\#\mathrm{MUL}_{SB}(n-1) > \#\mathrm{MUL}_{KO}(n)$ 이다. 정리 1에 의하여 만약 n이 홀수이면 n 번째 계수로 0을 추가하여 n+1차에 KO방법을 적용하는 것이 더 효율적임을 알 수 있다. 또한, n이 짝수일때 $\#\mathrm{MUL}_{SB}(n-1) > \#\mathrm{MUL}_{KO}(n)$이면 $\#\mathrm{MUL}_{KO}(2(n-1)) > \#\mathrm{MUL}_{KO}(2n)$ 또한 성립함을 알 수 있다. ($\#\mathrm{MUL}_{KO}(2n)=3\cdot\#\mathrm{MUL}_{SB}(n)$이므로) 이를 정리하면 다음과 같은 정리 2를 얻을 수 있다.

정리 2. (KO 방법을 다항식 곱셈에 적용할 경우)
기약다항식의 차수를 n 라 하고 #MUL을 다항식 곱셈의 계수-곱셈 연산량이라 하자. 만약 $n=2^k t$ 이고 정리 1을 만족한다고 하자. k 가 양의 정수이고 $t > 4$ 이면

$$\#\mathrm{MUL}_{KO}(2^k t) < \#\mathrm{MUL}_{KO}(2^{k-1}(2t-1))$$

이다.

정리 2는 정리 1에 의하여 쉽게 증명된다. 정리 1과 정리 2의 결과에 의하면 확장체의 차수가 크더라도 계수-곱셈 연산량은 작음을 알 수 있다. 예를 들어 $GF(p^{258})$에서 다항식곱셈을 수행하는 경우 계수-곱셈 연산량은 49,923이지만 $GF(p^{320})$에서 다항식 곱셈을 수행하는 경우 계수-곱셈 연산량은 18,225로 대략 2.5배 정도 작음을 알 수 있다. 따라서 임의의 확장체에서 계수-곱셈의 복잡도가 최적화되도록 구성할 필요가 있다.

Algorithm 1
KO 방법을 이용한 효율적인 다항식 곱셈의 구성

INPUT : 확장체의 차수 n
OUTPUT : 효율성이 높은 차수 n'

1. $2^k - 2^{k-2} < n \le 2^{k+1} - 2^{k-1}$인 k를 찾는다.
2. k<3이면 일반적인 KO 방법을 적용한다.
3. 다음 세가지 경우중 해당하는 경우를 찾는다.
   3.1 $3 \cdot 2^{k-2} < n \le 2^k$이면 State=4.
   3.2 $2^k < n \le 5 \cdot 2^{k-2}$이면 State=5.
   3.3 $5 \cdot 2^{k-2} < n \le 6 \cdot 2^{k-2}$이면 State=6.
4. $n' = \mathrm{State} \cdot 2^{k-1}$를 반환한다.

입력 값으로 들어간 확장체의 차수 n이 아닌 n' 에서 구성하는 경우 더 작은 계수-곱셈의 복잡도를 가진다. 따라서 알고리즘 1에서 $n < n'$ 인 경우 n'-n 개의 0을 패딩(Padding)하여 다항식 곱셈 연산을 구성한다. 제안하는 알고리즘 1을 이용하면 임의의 차수를 가지는 확장체에 대하여 KO 방법을 이용하여 다항식 곱셈 연산 과정을 수행시 가장 작은계수-곱셈을 가지는 차수를 구할 수 있게 된다. 위의 알고리즘을 사용한 결과와 기존의 방법을 사용한 결과를 비교하면 표 3과 같다.

표 3에서와 같이 알고리즘 1을 적용하면 기존의 KO 방법에서 가장 효율적인 차수를 찾아 구성할 수 있게 된다. 표 3에서 보인 것과 같이 확장체의 차수가 증가할수록 수십 배 이상의 계수-곱셈 연산량을 줄일 수 있다.

2. MSK3 방법을 사용한 최적 확장체 차수 선택

MSK_3방법을 사용한 다항식 곱셈 연산은 다음과 같은 성질을 만족한다. 3-중분할의 경우 확장체의 차수가 3의 배수여야 적용되므로 n이 3의 배수이면 n-1 과 n-2 는 3의 배수가 아니므로 다항식 곱셈을 수행할 경우 SB 방법을 적용한다.

표 3. 기존의 KO 방법과 알고리즘 1을 이용한 방법의 계수-곱셈 연산량 비교
Table 3. Comparing the Number of Coefficient Multiplications between KO Method and KO Method with Algo 1.

구성방법	n=7	n=22	n=60	n=76	n=92	n=122
KO방법	49	363	2,025	3,249	4,761	11,163
Algo 1	27	243	729	2,025	2,187	2,187

구성방법	n=146	n=180	n=234	n=298	n=370	n=394
KO방법	15,987	18,225	41,067	66,603	102,675	116,427
Algo 1	6,075	6,561	6,561	18,225	19,683	19,683

정리 3. (MSK3 방법을 다항식 곱셈에 적용할 경우)
기약다항식의 차수를 n 라 하고 #MUL을 다항식 곱셈의 계수-곱셈 연산량이라 하자. 만약 n = 3t 이면,

$t \ge 2$인 경우 : $\#\mathrm{MUL}_{MSK_3}(n) < \#\mathrm{MUL}_{SB}(n-1)$

$t \ge 3$인 경우 : $\#\mathrm{MUL}_{MSK_3}(n) < \#\mathrm{MUL}_{SB(n-1)} < \#\mathrm{MUL}_{SB}(n-2)$

이다.

정리 3에 의하여 n = 3t일 때 t 가 2보다 큰 경우 3t' - 2, 3t' - 1, 3t' 중에서 3t' 의 경우가 가장 계수-곱셈량이 작다. 따라서 확장체의 차수가 9보다 큰 경우 확장체의 차수가 3의 배수인 것만을 고려한다.

정리 4. (MSK3 방법의 다항식 곱셈을 적용 경우)
기약다항식의 차수를 $n(3^{k-1} < n < 3^k)$ 이라 하고 #MUL을 다항식 곱셈의 계수-곱셈 연산량이라 하면

1. $\#\mathrm{MUL}_{MSK_3}(3^{k-1}) < \#\mathrm{MUL}_{MSK_3}(4 \cdot 3^{k-2}) < \#\mathrm{MUL}_{MSK_3}(2 \cdot 3^{k-1}) < \#\mathrm{MUL}_{MSK_3}(3^k)$
2. $3^{k-1} < n < 4 \cdot 3^{k-2}$라 하면 $\#\mathrm{MUL}_{MSK_3}(n) > \#\mathrm{MUL}_{MSK_3}(4 \cdot 3^{k-2})$
3. $4 \cdot 3^{k-2} < n < 2 \cdot 3^{k-1}$라 하면 $\#\mathrm{MUL}_{MSK_3}(n) > \#\mathrm{MUL}_{MSK_3}(2 \cdot 3^{k-1})$
4. $2 \cdot 3^{k-1} < n < 3^k$라 하면 $\#\mathrm{MUL}_{MSK_3}(n) > \#\mathrm{MUL}_{MSK_3}(3^k)$.

이고 이때 $k > 0$ 이다.

정리 3은 쉽게 증명이 가능하므로 생략한다. 정리 4에서 1의 경우, [1]에서 제안된 방법에 의하여 3중분할

방법을 1번 반복하는 경우 6번의 하위-곱셈이 구성되므로 $\#MUL_{MSK_3}(3^k)=6^k$이다. 정리 4에서 2의 경우, $n=t\cdot 3^{k-2}$ 라 하면 t 는 $3<t<4$ 의 범위를 가져야하므로 만족하는 경우가 없으며, $n=t\cdot 3^{k-3}$ 라 하면 t 는 $3^2<t<4\cdot 3$ 의 범위를 가지므로 $\#MUL_{MSK_3}(t\cdot 3^{k-3}) > \#MUL_{MSK_3}(4\cdot 3^{k-2})$이다. 따라서 정리 3에 의하여 정리 4에서 2의 경우는 만족한다. 정리 4에서 3의 경우, $n=t\cdot 3^{k-2}$ 라 하면 $4<t<2\cdot 3$ 이므로 $\#MUL_{MSK_3}(t\cdot 3^{k-2}) > \#MUL_{MSK_3}(2\cdot 3^{k-1})$이므로 정리 3에 의하여 만족한다. 정리 4에서 4의 경우, $n=t\cdot 3^{k-2}$ 라 하면 $2\cdot 3<t<3^2$ 이므로 $\#MUL_{MSK_3}(t\cdot 3^{k-2}) > \#MUL_{MSK_3}(3^k)$ 이고 정리 3에 의하여 4번째 성질도 만족한다.

따라서 정리 3과 정리 4의 성질을 정리하면 알고리즘 2와 같다.

3. MSK5 방법을 사용한 최적 확장체 차수 선택

Algorithm 2
MSK3 방법을 이용한 효율적인 다항식 곱셈의 구성

INPUT : 확장체의 차수 n
OUTPUT : 효율성이 높은 차수 n'

1. n<3보다 작은 경우 n을 반환한다.
2. $3^{k-1}<n\le 3^k$을 만족하는 k를 찾는다.
3. 다음 중 해당하는 경우를 찾는다.
   3.1 $3^{k-1}<n\le 4\cdot 3^{k-2}$이면 State=4.
   3.2 $4\cdot 3^{k-2}<n\le 2\cdot 3^{k-1}$이면 State=6.
   3.3 $2\cdot 3^{k-1}<n<3^k$이면 State=9.
4. $n'=\text{State}\cdot 3^{k-2}$를 반환한다.

MSK5방법을 사용한 다항식 곱셈 연산은 다음과 같은 성질을 만족한다. 5-중분할의 경우 확장체의 차수가 5의 배수여야 적용되므로 n 이 5의 배수이면 n-1,..., n-4 는 5의 배수가 아니므로 다항식 곱셈을 수행할 경우 SB 방법을 적용한다.

정리 5. (MSK_5 방법의 다항식 곱셈을 적용 경우)
기약다항식의 차수를 n 라 하고 #MUL을 다항식 곱셈의 계수-곱셈 연산량이라 하자. 만약 n = 5t 이면,

$t\ge 1$인 경우 : $\#MUL_{MSK_5}(n)<\#MUL_{SB}(n-1)$
$t\ge 2$인 경우 : $\#MUL_{MSK_5}(n)<\#MUL_{SB}(n-1)<\#MUL_{SB}(n-2)$
$t\ge 3$인 경우 : $\#MUL_{MSK_5}(n)<\#MUL_{SB(n-1)}<\#MUL_{SB}(n-2)<\#MUL_{SB}(n-3)$
$t\ge 4$인 경우 : $\#MUL_{MSK_5}(n)<\#MUL_{SB}(n-1)<\#MUL_{SB}(n-2)<\#MUL_{SB}(n-3)<\#MUL_{SB}(n-4)$

이다.

정리 5에 의하여 n = 5t일 때 t 가 4보다 큰 경우 5t'-4, 5t'-3, 5t'-2, 5t'-1, 5t' 중에서 5t' 의 경우가 가장 계수-곱셈 연산량이 작다. 따라서 확장체의 차수가 20 보다 큰 경우 확장체의 차수가 5의 배수인 것만을 고려한다.

정리 6. (MSK_5 방법의 다항식 곱셈을 적용 경우)
기약다항식의 차수를 n ($5^{k-1} < n < 5^k$)이라 하고 #MUL을 다항식 곱셈의 계수-곱셈 연산량이라 하면

1. $\#MUL_{MSK_5}(5^{k-1})<\#MUL_{MSK_5}(5^k)$.
2. $5^{k-1}<n<6\cdot 5^{k-2}$라 하면 $\#MUL_{MSK_5}(n) > \#MUL_{MSK_5}(6\cdot 5^{k-2})$.
3. $6\cdot 5^{k-2}<n<7\cdot 5^{k-2}$라 하면 $\#MUL_{MSK_5}(n) > \#MUL_{MSK_5}(7\cdot 5^{k-2})$.
4. $7\cdot 5^{k-2}<n<2\cdot 5^{k-1}$라 하면 $\#MUL_{MSK_5}(n) > \#MUL_{MSK_5}(2\cdot 5^{k-1})$.
5. $2\cdot 5^{k-1}<n<11\cdot 5^{k-2}$라 하면 $\#MUL_{MSK_5}(n) > \#MUL_{MSK_5}(11\cdot 5^{k-2})$.
6. $11\cdot 5^{k-2}<n<3\cdot 5^{k-1}$라 하면 $\#MUL_{MSK_5}(n) > \#MUL_{MSK_5}(3\cdot 5^{k-1})$.
7. $3\cdot 5^{k-1}<n<5^k$라 하면 $\#MUL_{MSK_5}(n) > \#MUL_{MSK_5}(5^k)$.

이고 이때 k > 1 이다.

Algorithm 3
MSK_5 방법을 이용한 효율적인 다항식 곱셈의 구성

INPUT : 확장체의 차수 n
OUTPUT : 효율성이 높은 차수 n'

1. n < 4 보다 작은 경우 n을 반환한다.
2. n = 4, 5 인 경우 작은 경우 n=5을 반환한다.
3. $5^{k-1} < n \le 5^k$을 만족하는 k를 찾는다.
4. 다음 중 해당하는 경우를 찾는다.
   4.1 $5^{k-1} < n \le 6\cdot 5^{k-2}$이면 State=6.
   4.2 $6\cdot 5^{k-2} < n \le 7\cdot 5^{k-2}$이면 State=7.
   4.3 $7\cdot 5^{k-2} < n \le 2\cdot 5^{k-1}$이면 State=10.
   4.4 $2\cdot 5^{k-1} < n \le 11\cdot 5^{k-2}$이면 State=11.
   4.5 $11\cdot 5^{k-2} < n \le 3\cdot 5^{k-1}$이면 State=15.
   4.6 $3\cdot 5^{k-1} < n \le 5^k$이면 State=25.
5. $n'=\text{State}\cdot 5^{k-2}$를 반환한다.

정리 6의 2의 경우를 예로 살펴보자. $5^{k-1}<n\le 6\cdot 5^{k-2}$

범위에 존재하는 n에 대하여 $n=t\cdot 5^{k-3}$ 인 경우 만족하는 t가 존재하지 않으면, 정리 5에 의하여 $n=t\cdot 5^{r}$ 에서 $r<k-3$이면 $\#\mathrm{MUL}_{MSK_5}(n)<\mathrm{MUL}_{MSK_5}(6\cdot 5^{k-2})$인 경우는 존재하지 않는다. 따라서 MSK_3의 경우와 같이 증명이 유도된다.

정리 5와 정리 6을 이용하여 알고리즘을 구성하면 다음과 같다.

Ⅳ. 비 교

본 논문에서는 KO 방법, MSK_3 방법, MSK_5 방법을 효율적으로 구성하는 알고리즘 1, 2, 3을 제안하였다. 본 논문에서 제안하는 알고리즘을 적용하면 기존의 방법을 그대로 사용한 경우 보다 수십 배 이상의 계수-곱셈 연산량을 줄일 수 있다. 즉, 본 논문의 알고리즘들은 각각의 다항식 곱셈 방법의 설계 기준이 된다. 그림 1은 기존의 방법 KO방법과 알고리즘 1을 적용한 KO방법의 계수-곱셈 연산량을 비교한 결과이다.

그림 1에서 보인바와 같이 기존의 KO 방법보다 알고리즘1을 KO 방법에 적용하여 사용할 경우 더욱 효율적으로 구성됨을 알 수 있다.

표 4, 5은 [1]에서 제안된 방법을 이용한 방법의 계수-곱셈량과 본 논문에서 제안한 알고리즘을 적용한 경우의 계수-곱셈 연산량 비교이다.

표 4, 5에서 보인 바와 같이 확장체의 차수가 증가함에 따라 본 논문의 알고리즘을 적용한 결과가 더욱 효율적임을 알 수 있다.

그림 2는 기존의 결과에[1] 본 논문의 알고리즘을 적용한 결과이다. 그림에서 알 수 있듯이 기존의 MSK 방법이 확장체 차수에 따라 불규칙적이었던 계수-곱셈 연산량을 가장 효율적인 차수로 제한한 형태이다. 이때 알고리즘에서 반환하는 확장체의 차수를 사용하여 구성하면 계수-덧셈(뺄셈) 연산량 또한 효율적임을 [1]에서 제안된 복잡도를 기준으로 쉽게 검증할 수 있다. 따라서 다양한 차수에 적용이 불가능한 Successive Extension 방법보다 효율적이며 특히 소수 차수를 가지는 확장체에 적용할 때 효과적이다. 본 논문의 결과를 이용하면 p=2인 이진체의 경우도 쉽게 확장이 가능하며 이는 작은 복잡도를 가지는 하드웨어를 설계하는 경우 유용하게 사용될 수 있다.

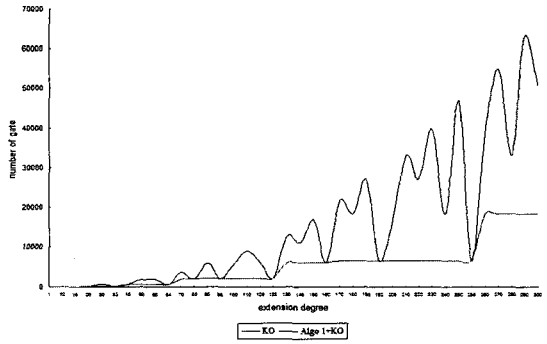

그림 1. 기존의 방법과 알고리즘 1을 적용한 경우의 다항식 곱셈의 계수-곱셈 연산량 비교

Fig. 1. Comparing the Number of Coefficient Multiplications between KO Method and KO Method with Algo 1.

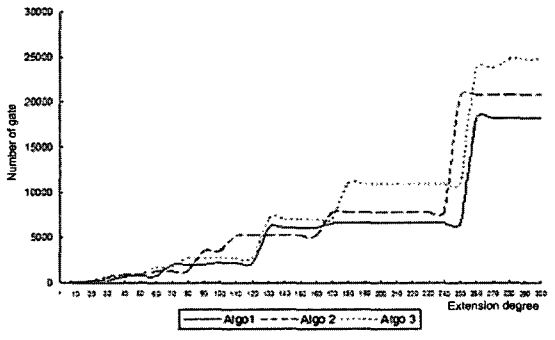

그림 2 [1]에서 제안된 방법에 본 논문에서 제안된 알고리즘을 적용한 경우 계수-곱셈 연산량 비교

Fig. 2. Comparing the Number of Coefficient Multiplications MSK Methods[1] with the Proposed Algorithm.

표 4. 기존의 MSK3 방법과 알고리즘 2을 이용한 방법의 계수-곱셈 연산량 비교

Table 4. Comparing the Number of Coefficient Multiplications between MSK3 Method and MSK3 Method with Algo 2.

	n=15	n=30	n=52	n=79
MSK_3	150	600	2,704	6,241
Algo 2	**144**	**576**	**864**	**1,296**

	n=120	n=159	n=241	n=319
MSK_3	9,600	16,854	58,081	101,761
Algo 2	**5,184**	**5,184**	**7,776**	**20,736**

표 5. 기존의 MSK5 방법과 알고리즘 3을 이용한 방법의 계수-곱셈 연산량 비교

Table 5. Comparing the Number of Coefficient Multiplications between MSK5 Method and MSK5 Method with Algo 3.

	n=10	n=40	n=60	n=80
MSK_5	56	896	2,016	3,584
Algo 2	**56**	**784**	**1,764**	**2,744**

	n=160	n=230	n=260	n=380
MSK_5	14,336	29,624	37,856	80,864
Algo 2	**9,604**	**10,976**	**23,716**	**38,416**

V. 결 론

본 논문에서는 효율적인 확장체 곱셈연산을 구성할 수 있는 효율적인 알고리즘을 제안하였다. 본 논문에서 제안한 알고리즘을 적용한 곱셈방법을 사용하여 병렬구조 곱셈 연산기를 구성하면 기존의 방법으로 구성한 것보다 차수가 증가함에 따라 수십 배 이상의 공간 복잡도를 줄일 수 있다. 따라서 논문의 결과는 낮은 공간 복잡도가 요구되는 환경에 효과적으로 적용된다.

참 고 문 헌

[1] 장남수, 한동국, 정석원, 김창한, "유한체 GF(2n)에서 낮은 공간 복잡도를 가지는 새로운 다중 분할 카라슈바 방법의 병렬처리 곱셈기", 대한전자공학회 논문지(SC), 41. 1, pp.33-40, 2004.

[2] ANSI X9.62, "Public key cryptography for the financial services industry : The Elliptic Curve Digital Signature Algorithm (ECDSA)", (available from the ANSI X9 catalog), 1999.

[3] H. Cohen, "A Course in Computational Algebric Number Theory", Springer-Verlag, Berlin, Heidelberg, 1993.

[4] G. Drolet, "A New Representation of Elements of Finite Fields $GF(2^m)$ Yielding Small Complexity Arithmetic circuit}, IEEE Trans. on Computers, vol 47, 1998, 353-356.

[5] M. Ernst, M. Jung, F. Madlener, S. Huss, and R. Blümel, "A Reconfigurable System on Chip Implementation for Elliptic Curve Ctryptography over $GF(2^n)$", In Work shop on Cryptographic Hardware and Embedded Systems (CHES'02), LNCS2523, (2002), 381-399.

[6] IEEE 1363, "Standard Specifications For Public Key Cryptography", http://grouper.ieee.org/groups/1363/,2000. 381-399.

[7] K.O. Geddes, S.R. Czapor, and G. Labahn, "Algorithms for Computer Algebra, Kluwer Academic Publishers", 1992.

[8] C. K. Koc, and B. Sunar, "Low- Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields", Proceeding of 1998 IEEE International Symposium on Information Theory, MIT, Cambridge, Massachusetts, August 16-21, 1998.

[9] N. Koblitz, "Elliptic Curve Ctyptosystems", Mathematics of Computation, vol. 48, 1987, 203-209

[10] M. Leone, "A New Low Complexity Parallel Multiplier for a Class of Finite Fields", In Work shop on Cryptographic Hardware and Embedded Systems (CHES'01), LNCS2162, (2001), 160-170.

[11] V. Miller, "Use of Elliptic Curve Cryptosys tems", Advances in Cryptology, CRYPTO'85, LNCS 218, H. C. Williams,Ed., Springer-Verlag, 1986, 417-426.

[12] C. Paar, "Efficient VLSI Architecture for Bit-Parallel Computation in Galois Fields", PhD thesis, (Engl. transl.), Institute for Experimental Mathematics, University of Essen, Essen, Germany, June 1994.

[13] C. Paar, "Low complexity parallel Multipliers for Galois fields GF((2n)4) based on special types of primitive polynomials, In 1994 IEEE International Symposium on Information Theory, Trondheim, Norway, June 27-July 1 1994.

[14] Paar C., "A new architecture for a parallel finite fields multiplier with Low Complexity Based on Composite Fields", IEEE Trans. on Computers, vol45, no. 7, ,July 1996, 846-861.

[15] C. Paar, P. Fleischmann, P. Roelse, "Efficient Multiplier Architectures for Galois Fields $GF((2^n)^4)$, IEEE Transactions on Computers, February 1998, vol. 47, no. 2, 162-170.

[16] T. Kobayashi, K. Aoki, and F. Hoshino, "OEF Using a Successive Extension," Proc. The 2000 Symposium on Cryptography and Information Security, no.B02(2000).

저 자 소 개

장 남 수(학생회원)
2002년 2월 서울시립대학교 수학과 학사.
2005년 고려대학교 정보보호 대학원 석사.
2005년~현재 고려대학교 정보보호대학원 박사과정.
<주관심분야 : 공개키 암호, 암호칩 설계 기술, 부채널 공격 방법론>

김 창 한(정회원)
1985년 2월 고려대학교 수학과 학사.
1987년 2월 고려대학교 수학과 석사.
1992년 2월 고려대학교 수학과 박사.
2002년 2월~현재 세명대학교 정보보호학과 부교수.
<주관심분야 : 정수론, 공개키 암호, 암호 프로토콜>