알고리즘 레벨 유한체 연산에 대한 최적화 연구

(Optimization Techniques for Finite Field Operations at Algorithm Levels)

문상국

목원대학교 전자공학과

Sangook Moon

Mokwon University, Department of Electronic Engineering

E-mail : smoon@mokwon.ac.kr

요 약

$GF(2^m)$를 기본으로 하는 유한체 연산에서 덧셈과 뺄셈은 그 구현이 단순하지만, 곱셈, 나눗셈이나 역원을 구하는 데에는 수학적으로 복잡한 수식을 간략화하는 과정이 필수적이다. 유한체 연산은 기본적으로 normal basis와 polynomial basis 두 가지 측면에서 접근할 수 있고 이 두 방법은 각각 장단점을 가지고 있다. 본 연구에서는 두 가지 basis 중에서 수학적인 접근이 용이한 polynomial basis를 사용한 접근방식을 채택하여 수학적인 원리를 이용한 수식의 간략화를 꾀하고 최적화 하는 방법을 제시한다.

ABSTRACT

In finite field operations based on $GF(2^m)$, additions and subtractions are easily implemented. On the other hand, multiplications and divisions require mathematical elaboration of complex equations. There are two dominant way of approaching the solutions of finite filed operations, normal basis approach and polynomial basis approach, each of which has both benefits and weakness respectively. In this study, we adopted the mathematically feasible polynomial basis approach and suggest the optimization techniques of finite field operations based of mathematical principles.

키워드

GF, Finite field operations, polynomial basis, algorithm

Ⅰ. 서 론

최근 혹은 차세대를 위한 암호 시스템에서 사용되는 연산은 기존의 DES나 RSA에서 사용되었던 비선형 S-box table이나 정수에서의 소수 덧셈군을 사용하기보다는, 2의 m제곱수에 대한 residue 집합을 원소로 삼는 Galois Field 유한체를 사용하는 연산에 대한 역할이 점점 중요해지는 추세이다 [1]. 또한, 이 유한체 연산은 일반 정수 연산과는 달라서 곱셈이나 나눗셈 등의 경우 연산 과정이 매우 복잡한 단계를 여러번 반복해야 하기 때문에 일반적으로 구현 할 경우 연산 소요 시간이 매우 오래 걸리게 된다.

유한체 연산에 대한 효율성은 기존의 정보통신 코딩 이론에서 연구 되었던 바, 부호화 이론에서 어느 정도 연구의 완성도가 이루어져 있다고 알기 쉽지만, 그에 대한 구현은 큰 비트열을 가지는

연산에 대한 것들이 아니었기 때문에 주로 테이블 방식으로 구현이 되었다. 현대 암호학적인 어플리케이션이 적용되면서 초소형 정보 기기에 복잡한 인증 방식이 도입됨에 따라, 제한된 자원 하에서 수백 비트 이상의 높은 비트열에 대한 유한체 연산 처리를 이용한 방식이 새로운 관심을 끌게 되었고, 따라서 높은 비트 수에 대한 효율적인 유한체 연산 처리에 대한 연구가 요구되고 있다. 또한, 이론적인 최적화 알고리즘의 개발만으로는 시대적인 추세에 발맞추기 어렵기 때문에, 소프트웨어적인 내용을 파악하고 있는 알고리즘 설계자가 직접 이를 FPGA에 마이크로프로세서와 연동하여 SoC 형태로 구현함으로써 실제 어플리케이션에 적용이 가능한 지를 검증한다면 더욱 바람직 할 것이다. 이를 SoC 형태로 구현하기 위해서는 알고리즘 레벨에서의 단순화가 필수적으로 요구된다.

II. 유한체 연산

유한체 연산에 대한 연구의 필요성을 보이는 실제적인 예로, 타원곡선의 뼈대가 되는 스칼라 점 곱셈 (scalar point multiplication) 연산이 진행되는 전형적인 과정을 예로 들어본다. 타원 곡선 스칼라 곱셈 연산은 큰 두 줄기인 점 덧셈 (point addition) 타원 곡선 연산과 두배점 (point double) 타원 곡선 연산으로 배분되어 이루어지는데, 각각의 타원 곡선 연산은 유한체 곱셈, 유한체 나눗셈, 유한체 덧/뺄셈으로 세분화 되어 연산이 수행된다. 전통적인 타원 곡선 암호 연산에서 k (> 160)비트의 키가 사용된다면 곡선 상위의 한 점에 대한 스칼라 곱셈을 수행하는 경우 점 덧셈 연산은 $\frac{1}{2}k$ 번, 두배점 연산은 k 번 수행된다. 이 때, 유한체 곱셈 계열 (곱셈과 제곱을 포함한) 연산과 유한체 나눗셈 연산은 각각 $4k$, $\frac{3}{2}k$ 번 수행되어야 한다. 즉, 수행 과정이 복잡하면서 반복 회수가 많은 유한체 곱셈, 나눗셈, 제곱 연산 등의 하위 연산을 최적화할 수 있다면, 높은 비트열에 대한 유한체 연산을 사용하는 타원 곡선이나 다른 유한체 연산을 사용하는 암호 시스템에서의 연산 처리 능력이 획기적으로 개선될 수 있는 것이다.

표 1에서와 같이, 점 덧셈 연산에서는 8번의 유한체 덧셈, 1번의 유한체 곱셈, 1번의 유한체 나눗셈, 1번의 유한체 제곱 연산이 필요하다는 것을 알 수 있고, 두배점 연산에서는 4번의 유한체 덧셈, 1번의 유한체 곱셈, 2번의 유한체 제곱 연산, 그리고 1번의 유한체 나눗셈이 필요하다.

표 1. 타원 곡선 상의 점 연산에 필요한 유한체 연산의 개수
Table 1. Number of field operations required for EC point operation

	점 덧셈 연산	두배점 연산	점 역원 연산
유한체 덧셈	8	4	1
유한체 곱셈	1	1	0
유한체나눗셈	1	1	0
유한체 제곱	1	2	0

III. 유한체 곱셈 알고리즘

기존에 연구된 유한체 곱셈 연산기에는 크게 직렬 유한체 곱셈기, 배열 유한체 곱셈기, 하이브리드 유한체 곱셈기가 존재하였다. 직렬 유한체 곱셈기는 Mastrovito에 의하여 제안되어 유한체 곱셈기의 가장 기본적인 구조로 자리 잡아 왔고 [2], 이를 병렬로 처리하기 위해 m 배의 자원을 투자하여 m 배의 속도를 얻어 낸 결과가 2차원 배열 유한체 곱셈기이다. 배열 유한체 곱셈기는 가격 대 성능 비에 있어서 효율이 떨어지는 반면, 이런 기존의 방식의 장점만을 취하여 제안된 방식이 1999년 Paar에 의해 제안된 하이브리드 곱셈기이다 [3]. 이 하이브리드 유한체 곱셈기는 기존의 Mastrovito의 직렬 곱셈기 안에 기본적인 연산기로서 2차원 배열 곱셈기를 탑재한 방식으로, 하드웨어 자원도 직렬 곱셈기와 배열 곱셈기의 중간 이하가 되고 수행 속도도 직렬 곱셈기에 비해 상당히 빠르기 때문에 유한체 곱셈기의 연구에 역사상 중요한 위치를 차지한다. 하지만, 사용 가능한 유한체로서 유한체의 차수가 합성수인 합성수 유한체를 사용하기 때문에 암호학적인 안전도가 떨어져 많은 응용 분야에의 적용에 힘들다.

III. 유한체 나눗셈 알고리즘

유한체 상에서의 나눗셈은 일반적인 응용에 필수적으로 요구되는 적용 분야가 드물었기 때문에 대부분이 소프트웨어적으로 구현되어 왔고, 하드웨어로 구현된다고 해도 전용 나눗셈기를 따로 할당하지 않고 곱셈기를 재사용 함으로써 유한체 나눗셈을 구현해 왔다. 하지만 암호학적인 적용 분야에 유한체가 도입됨으로써 유한체의 범위가 넓어져 계산이 복잡해짐으로 인해 점차 전용 하드웨어 유한체 나눗셈기의 계산 단순화에 대한

연구가 필수적이다.

유한체 $GF(2^m)$ 상에서 나눗셈은 유한체 곱셈에서와 마찬가지로 소수 다항식에 의존하여 수행되는데, 유한체 내의 임의의 두 원소가 $A(x), B(x)$라고 하고 나눗셈 결과 다항식이 $Z(x)$라고 한다면 다음 형태를 가진다.

$$Z(x) = \frac{A(x)}{B(x)} \bmod P(x) \qquad (1)$$

유한체 곱셈에서는 다항식 연산에 따라 곱한 다음 주어진 소수 다항식을 이용하여 차수를 줄임으로써 곱셈 결과를 얻는다. 하지만 유한체 나눗셈에서는 잉여 관계 때문에 단순한 차수 줄임으로써는 올바른 결과를 얻을 수 없다. 그림 1의 예를 보자. 그림에서와 같이 소수 다항식과 유한체의 원소가 주어진다고 가정하자. 두 원소의 곱한 결과는 다음과 같다.

$$x^{192} \cdot (x^2+1) = x^{192} + x^{16} + x \qquad (2)$$

이제 곱셈 결과를 두 곱하는 수로 각각 나누어 보자. ①에서는 우연히 $x^{16}+x$가 $(x^{15}+1)\cdot x$ 인 점에 착안하여 나눗셈 결과가 x^2+1로 바로 나오게 되지만 일반적인 ② 같은 경우에는 처음 나눗셈이 수행될 때 어떤 부분이 지수 감소가 되었는지 모르면 계산을 끝까지 수행할 수가 없게 되고 만일 끝까지 수행이 된다고 해도 올바른 결과를 장담할 수가 없으며 시간도 오래 걸리게 된다. 따라서 유한체 나눗셈은 반드시 다음과 같은 순서로 행해져야 한다.

순서 1 : 먼저 $B(x)$의 곱셈에 대한 역원을 찾는다.

순서2:

$$Z(x) = \frac{A(x)}{B(x)} \bmod P(x) = A(x) \cdot B^{-1}(x) \bmod P(x) \qquad (3)$$

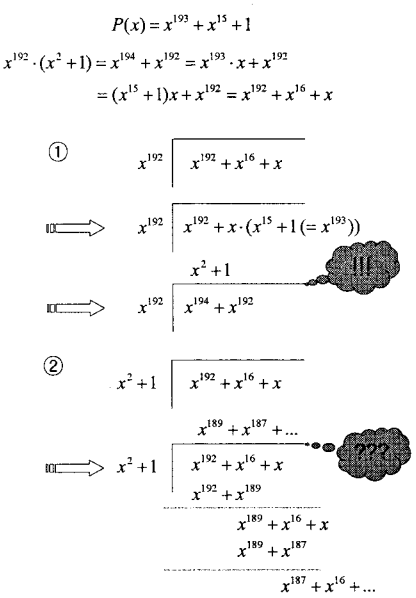

그림 1. 유한체 나눗셈의 잉여 관계의 예
Fig. 1. An example of occurring redundancy in GF division

유한체 나눗셈 연산에서 가장 먼저 해야 할 일은 제수의 곱셈에 대한 역원을 찾는 일이다. 가장 간단한 역원을 찾는 방법은, 지정된 유한체 내의 원소에 대한 역원을 모두 미리 찾아 테이블 안에 저장시켜 놓고 필요할 때 찾는 방법이다. 이 방법은 m이 매우 작은 응용 분야 $(m<8)$에서는 효율적이지만 m이 커질수록 저장 장소는 m^2에 비례하여 커져야 하기 때문에 큰 m에 대해서는 적절한 방법이 못된다고 할 수 있다. 따라서 유한체 나눗셈기를 VLSI로 구현하는 경우에 특정한 알고리즘을 사용한 방식이 사용되는 것이다. 유한체 나눗셈의 구현하는 방법 중 가장 많이 사용되는 역원을 구하는 알고리즘은 크게 다음 두 가지이다

I) 페르마 (Fermat)의 정리 : $GF(2^m)$ 상의 임의의 원소 B에 대해서 $B^{2^m} = B$ 라는 관계가 성립한다. 이 정리를 이용하여 양변을 B^2으로 나누면 곱셈에 대한 역원을 식 4와 같이 반복적인 제곱과 곱셈으로 구할 수 있고 또한 이는 식 5와 같이 변형되어 표현될 수도 있다.

$$B^{-1} = B^2 \cdot B^{2^2} \cdot B^{2^3} \cdot \ldots \cdot B^{2^{m-1}} \quad (4)$$

$$B^{-1} = (\ldots(((B)^2B)^2B)^2\ldots B)^2 \quad (5)$$

II) 유클리드 (Euclid)의 알고리즘 : $GF(2^m)$ 상의 임의의 두 원소에 대하여 반복적으로 서로 나누면 최종적으로 최대 공약 다항식 (Greatest Common Polynomial, GCP)를 구할 수 있다. 이 방법은 표준 연산을 사용하므로 특히 $GF(2^m)$ 상의 두 원소가 표준 기저 방식으로 표현되었을 때 더 효율적이다.

유클리드 알고리즘 기반의 유한체 나눗셈기에 대한 가장 기틀이 되는 연구는 Brunner에 의해서 수행되었다 [4]. Brunner는 그의 논문에서 유클리드 알고리즘을 변형하여 유한체 역원을 구하는 알고리즘을 제안하였고 후에 이는 Guo의 연구에서 유한체 나눗셈 알고리즘으로 변형되었는데, 이에 대한 몇 가지 변형된 나눗셈기 구조가 몇몇 연구에 의해 제안되었다. 그중 하나가 규칙적인 구조를 이용하여 파이프라인 기술을 적용시킨 시스톨릭 배열 구조인데, 이 구조는 회로의 복잡도가 너무 큰 단점이 있다.

III. 결론

본 논문에서는 타원 곡선 암호 시스템에서 빠르고 안전하게 정보 교환을 수행할 수 있는 연산 알고리즘, 즉 유한체 $GF(2^m)$ 상에서의 곱셈, 나눗셈에 대한 기존 결과에 대해서 정리하고 개선해야 할 곱셈기와 나눗셈기의 알고리즘 레벨에서의 이슈에 대하여 정리하고 비교하였다. 본 논문에서 살펴본 바와 같이, 곱셈 알고리즘을 구현할 때의 이슈와 유한체 나눗셈 알고리즘을 구현할 때의 이슈가 매우 다양하다. 타원 곡선 암호 응용 어플리케이션을 설계하는 경우 이와 같이 다양한 관점에서의 유한체 알고리즘 레벨에서 설계 시 고려해야 할 이슈사항들을 잘 고려한다면 복잡한 타원 곡선 알고리즘을 효율적으로 개선할 수 있는 문제 해결에 도움이 될 것이다.

IV. 참고문헌

[1] B. Schneier, Applied Cryptography, second edition, John Wiley & Sons, Inc., 1996.

[2] E. D. Mastrovito, "VLSI Architectures for Computations in Galois Fields," Linkoping Studies in Science and Technology Dissertations, No. 242, 1991.

[3] C. Paar and N. Lange, "A Comparative VLSI Synthesis of Finite Field Multiplier," Proc. Third Intl Symp. Comm. Theory and Its Applications, Lake District, U. K. July 1995.

[4] H. Brunner, A. Cruiger, and M. Hosfstetter, "On Computing Multiplicative Inverses in GF(2m)," IEEE Transactions on Computers, Vol. 42, No. 8, pp. 1010-1015, Aug. 1993.