2001년도 대한전기학회 하계학술대회 논문집 2001.7.18-20

타원곡선 암호화 시스템을 위한 유한필드 곱셈기의 설계

이욱, 이상설
원광대학교 전기공학과

Design of Finite Field Multiplier for Elliptic Curve Cryptosystems

Wook Lee, Sang-Seol Lee
Wonkwang University

Abstract - Elliptic curve cryptosystems based on discrete logarithm problem in the group of points of an elliptic curve defined over a finite field. The discrete logarithm in an elliptic curve group appears to be more difficult than discrete logarithm problem in other groups while using the relatively small key size. An implementation of elliptic curve cryptosystems needs finite field arithmetic computation. Hence finite field arithmetic modules must require less hardware resources to archive high performance computation. In this paper, a new architecture of finite field multiplier using conversion scheme of normal basis representation into polynomial basis representation is discussed. Proposed architecture provides less resources and lower complexity than conventional bit serial multiplier using normal basis representation. This architecture has synthesized using synopsys FPGA express successfully.

1. 서 론

고도의 정보화 사회에서 공개키 암호화 시스템은 없어서는 안될 필수적인 요소이다. 현재 널리 쓰이고 있는 공개키 암호화 알고리즘으로는 RSA(Rivest-Shamir-Adleman)와 ECC(Elliptic Curve Cryptography)가 있으며, 특히 ECC는 작은 키 사이즈를 가지고도 높은 안전도를 제공할 수 있는 장점을 가지고 있다. RSA 1024 비트의 안전도를 ECC가 보유하기 위해선 160비트의 키 사이즈만으로도 충분한 것으로 알려져 있다[5]. 다른 암호화 시스템에 비해 작은 키 사이즈를 갖음으로써 데이터의 암호(Encryption)와 복호(Decryption) 과정이 빠르며 또한 하드웨어 자원과 전력 소모가 적어 스마트 카드와 같은 휴대용 보안장치의 보안 코프로세서(Security Coprocessor)로서 사용될 수 있다.

ECC를 이용하여 암호와 복호 과정을 수행하기 위해서는 선택되어진 타원곡선 상에서 스칼라 곱셈을 필요로 한다. 스칼라 곱셈은 타원곡선상의 임의의 점 P를 선택하고 점P에 대해 임의의 정수 k를 곱한 kP를 구하는 것이 된다. 이 스칼라 곱셈을 구현하기 위해서는 유한필드 상에서의 산술연산이 주된 연산이 된다. 그러므로 ECC의 효과적인 구현을 위해서는 높은 성능을 갖는 유한필드 산술연산이 요구되어진다.

유한필드 상에서의 산술연산은 F_p와 F_{2^m}에서 가능하다. 그러나 하드웨어의 구현은 이진 유한 필드 F_{2^m}이 캐리의 전파가 일어나지 않기 때문에 더욱 적합하다. 또한 이진 유한 필드는 normal basis와 polynomial basis로 표현 될 수 있다. normal basis 의 경우 덧셈 연산은 비트-XOR 연산, 제곱 연산은 순환 쉬프트 연산으로 대응될 수 있어 단순한 구조를 갖음으로 실제 구현에 있어서 polynomial basis 에 비해 유리한 점을 갖는다. 그러나 곱셈 연산에 경우 normal basis에서는 다소 복잡해진다. normal basis에서 구현된 대표적인 곱셈기는 Massey-Omura 곱셈기[7]로 대부분의 ECC를 이용한 응용에서는 Massey-Omura 또는 그것의 변형된 형태를 사용하고 있다.

본 논문에서는 normal basis에서의 곱셈 연산에 소요되는 하드웨어의 자원과 복잡도를 줄이기 위해 normal basis를 polynomial basis로 변환하여 유한필드 직렬 곱셈기를 설계하였다. 제안된 구조는 normal basis의 유리한 점을 이용하면서 필요에 따라 polynomial basis의 장점을 취하는 구조를 갖는다. 설계된 곱셈기의 구조는 normal basis에서 구현된 다른 곱셈기에 비해 적은 자원이 요구되며 그 단순한 구조로 낮은 하드웨어 복잡도를 갖는다.

2. 유한 필드 Representation

유한 필드 F_{2^m}은 F_2상에서 m차원의 벡터 공간으로 표현될 수 있으며 따라서 F_{2^m}의 임의의 원소 β의 polynomial representation은 다음과 같이 표현 될 수 있다.

$$\beta = \sum_{i=0}^{n} a_i x^i = a_n x^n + \ldots + a_1 x^1 + a_0 \qquad (1)$$

여기서, n < m , $a_i \in (0,1)$

이것의 normal basis는 다음과 같이 표현 될 수 있다.

$$\{\beta, \beta^2, \beta^3, \ldots, \beta^{2^{m-1}}\} \qquad (2)$$

이때 β는 normal basis에서 생성자가 되며 따라서 모든 임의의 원소 $e \in F_{2^m}$는 normal basis 형태로 표현되어질 수 있다. 즉 e는 다음과 같이 normal basis representation으로 표현되어 질 수 있다.

$$e = \sum_{i=0}^{m-1} e_i \beta^{2^i} = e_{m-1}\beta^{2^{m-1}} + \ldots + e_1 \beta^{2^1} + e_0 \beta^{2^0} \qquad (3)$$

normal basis에서 덧셈과 제곱연산은 단순한 XOR과 순환 쉬프트 연산으로 나타낼 수 있다. 덧셈의 경우 벡터 공간에서 비트-XOR 된다. 제곱연산의 경우 다음과 같이 나타낼 수 있다.

$$e^2 = \sum_{i=0}^{m-1} e_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} e_{i-1} \beta^{2^i} \qquad (4)$$

따라서 제곱연산은 벡터 공간에서 순환 쉬프트 연산으로 나타낼 수 있다.

3. Normal Basis와 Polynomial Basis의 곱셈 연산

normal basis에서 유한 필드 F_{2^m}의 임의의 원소 A와 B의 곱셈은 다음과 같이 유도 될 수 있다.

$$A=\sum a_i\beta^{2^i},\quad B=\sum b_j\beta^{2^j} \tag{5}$$

$$C=A\bullet B=\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}a_ib_j\beta^{2^i}\beta^{2^j}=\sum_{k=0}^{m-1}c_k\beta^{2^k} \tag{6}$$

식(6)에서 c_k는 각각의 cross-product 성분을 basis 성분의 합으로 맵핑할 수 있으며 이는 다음과 같다.

$$c_k=\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}\lambda_{ijk}\beta^{2^k} \tag{7}$$

여기서 λ_{ijk}는 lambda matrix 또는 multiplication table이라 불리며 곱셈 연산의 복잡도를 결정한다. normal basis중 최소의 none zero λ_{ij}를 갖는 normal basis를 optimal normal basis(ONB)라 한다. 만약 유한필드 F_{2^m}이 ONB가 되면 이때 복잡도는 2m-1이 된다. 실제 곱셈연산을 위해서 식(7)을 단지 λ_{ij0}성분만 포함하도록 유도할 수 있으며 다음과 같이 나타낼 수 있다[3].

$$c_k=\sum_{i=o}^{m-1}\sum_{j=0}^{j=m-1}a_{i+k}b_{j+k}\lambda_{ij0} \tag{8}$$

식(8)을 이용한 가장 대표적인 구현 예로는 Massey-Omura 곱셈기가 있다. Massey-Omura 곱셈기는 입력 A와 B로부터 c_0성분을 계산하고 같은 로직을 사용하여 c_k성분을 구할 수 있다. 그림 1은 optimal normal basis(ONB) F_{2^6}에서 구현된 Massey-Omura 곱셈기의 예이다. 대부분의 ECC의 응용에서 구현되어진 곱셈기 구조는 Massey-Omura 혹은 그것의 변형된 형태를 이용하고있다[1][4].

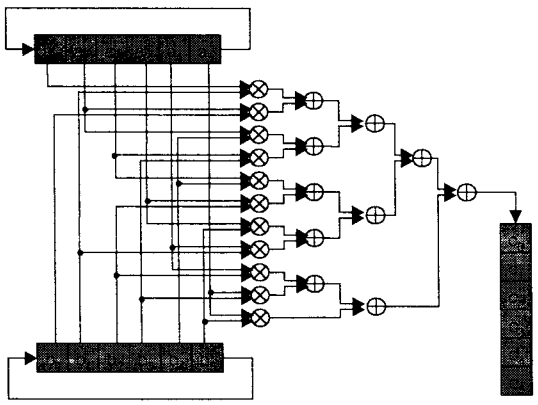

그림 1. F_{2^6} Massey-Omura 곱셈기 (Optimal Normal Basis)

polynomial basis에서 유한 필드 F_{2^m}의 임의의 원소 A(x)와 B(x)의 곱셈연산은 다음과 같이 나타낼 수 있다.

$$A(x)=\sum_{i=0}^{m-1}a_ix_i,\quad B(x)=\sum_{i=0}^{m-1}b_ix_i$$

$$P(x)=x^m+\sum_{i=0}^{m-1}p_ix_i \tag{9}$$

$$C(x)=A(x)B(x)\ \bmod\ P(x) \tag{10}$$

여기서, $a_i,\ b_i,\ p_i\in(0,1)$

식(9)에서 $P(x)$는 차수가 k인 유한필드 원시다항식이 된다. 식(10)을 구현하는 방법은 적용 분야에 따라 다양하며 목적에 가장 적합한 방법을 선택하는 것이 바람직 할 것이다. VLSI 디자인에서 time과 area는 서로 trade-off 관계에 있다. 본 논문에서는 보다 area에 효율적인 디자인을 목적으로 하며 따라서 그에 적합한 방법으로는 유한필드 직렬 곱셈기가 가장 효율적일 것 이다[6]. polynomial basis에서 직렬 곱셈기 또한 그 목적에 맞게 수정된 디자인들이 많은 분야에서 연구되었다[2]. 그러나 그 기본적인 원리는 같으며 다음과 같다.

$$C'^{(i)}=xA'^{(i-1)}\bmod P(x)+a_{k-i}B(x) \tag{11}$$

$$\text{for } i=1,2,\ldots,m;\ C'^{(0)}=0;\ C(x)=C'^{(m)}$$

식(11)에서 $xA'^{(i-1)}$은 차수가 m인 polynomial이며 원시다항식 $P(x)$와 모듈러 연산이 취해진다. $C'^{(i)}$는 i번째 부분 결과가 된다. 그림 2는 유한필드 F_{2^m}에서 식(11)의 직렬 곱셈기를 나타내었다.

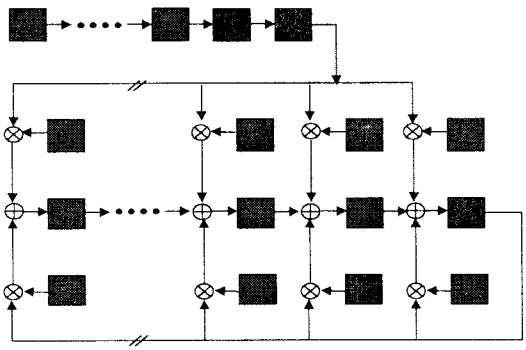

그림 2. F_{2^m} 직렬 곱셈기(Polynomial Basis)

그림 2에서 보이는 바와 같이 입력 A(x)가 MSB부터 차례로 직렬로 입력되어 B(x)와 곱해지고 P(x)와 모듈러 연산을 취한후 C(x)에 보내진다.

4. Normal Basis에서 Polynomial Basis로의 변환과 역 변환

벡터 공간에서 normal basis와 polynomial basis는 order와 1을 제외하고 그 표현 방식은 같다. 즉 ONB의 경우 x^{m-1}을 포함하고 polynomial basis의 경우 1을 포함하는 점을 제외하고 나머지 부분은 표현 방식에 있어서 동일하다. ONB에서 각각의 항을 생각해 보자. 식(3)의 i번째 항은 식(12) 같이 쓸 수 있다.

$$a_ix^{2^i} \tag{12}$$

$$a_ix^k \tag{13}$$

여기서 2는 Z_{m+1}의 정수집합 에서 primitive이며 따

라서 2는 Z_{m+1} 에서 생성자가 된다. 그러므로 $2^i \bmod m+1$은 i가 0에서 m-1사이의 값을 갖을때 1과 m 사이의 모든 정수를 표현할 수 있다. 따라서 식(12)는 식(13)과 같이 바꾸어 쓸 수 있다.
식(12)와 식(13)으로부터 F_{2^m}의 임의의 원소 A의 normal basis representation을 다음의 polynomial representation으로 변환 할 수 있다.

$$A=\sum_{i=0}^{m-1} a_i x^{2^i}=\sum_{i=0}^{m-1} a_i x^k \tag{14}$$

$$P(x)=x^m+\sum_{i=0}^{m-1} x^i \tag{15}$$

여기서, $a_i \in (0,1)$, $k=2^i \bmod m+1$

polynomial basis로 변환할 경우 m+1비트로의 확장이 필요하며 식(15)는 주어진 ONB에서의 원시다항식이 된다. 식(15)로부터 다음식을 유도 할 수 있다.

$$x^{m+1}=x^m x=1 \tag{16}$$

$$x^{m+1}+x^m=1+x^m=\sum_{i=1}^{m-1} x^i \tag{17}$$

5. 제안된 곱셈기의 구조

제안된 곱셈기의 구조는 그림 2의 직렬 곱셈기의 변형된 형태를 갖게된다. ONB의 입력 A와 B가 곱셈기의 입력 레지스터로 입력되어지며 이때 ONB에서 polynomial basis로 변환이 이루어진다. 따라서 변환 과정에서는 단지 상호 레지스터간의 연결만 필요하며 시간지연이나 게이트의 소모는 발생하지 않으며 출력 역시 같은 방법으로 이루어진다. 제안된 직렬 곱셈기의 구조는 그림 3과 같으며 m+1비트 확장으로 인한 x^{m+1}과 x^m+x^{m+1}에서 전파되는 캐리를 고려해주어야 한다. 이는 식(16)과 식(17)로부터 구할 수 있다.

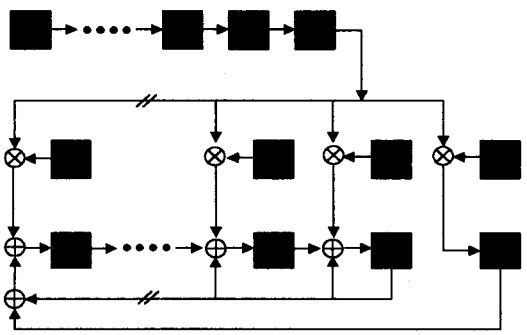

그림 3. 제안된 F_{2^m} 직렬 곱셈기 (ONB - Poly)

6. 구현 및 성능비교

ONB에서 구현되어진 그림 1 Massey-Omura 곱셈기와 본 논문에서 제안한 그림 3의 직렬 곱셈기를 비교해 보면 Massey-Omura의 경우 각각2m-1, 2m-2개의 AND와 (2-input)XOR 게이트가 필요하다. 반면 본 논문의 경우 각각 m개의 AND와 (3-input)XOR 게이트가 요구되어진다. flip-flop(FF)의 경우 Massey-Omura의 경우 3m개의 FF이 필요하며 입력 A와 B의 순환 쉬프트 연산을 필요로 한다. 본 논문의 경우 3m개의 FF가 요구되어 진다.

	본 논문	ONB
Time	m+1 clock	m clock
Complexity	m-1	2m-1
FF	3m	3m
XOR	(3-input)m	2m-2
AND	m	2m-1

표 1. 성능 비교

	FPGA	CLB(%)	Freq.(MHz)	delay(ns)
본 논문	XC4013	29.8	50	12.83
ONB	XC4000	68.7	50	85.37

표 2. 구현 결과

표 1은 두 곱셈기를 요약한 것으로 계산 소요 시간은 본 논문의 경우 m+1 클럭이 소요되며 하드웨어 자원에서는 FF은 같지만 XOR와 AND 게이트에서는 Massey-Omura 곱셈기에 비해 우수한 면을 보여준다. 표 2에는 $F_{2^{182}}$에서 구현결과를 요약하였다. Xilinx FPGA를 이용하여 합성한 결과로 delay는 설계된 디자인에서 최대 지연을 갖는 경로의 지연을 나타낸다. 두 곱셈기는 각각 50MHz에서 최대지연의 경우 현저한 차이를 보인다. 따라서 고속의 응용에서도 본 논문에서 제안된 구조가 적합하다 할 수 있다.

7. 결 론

본 논문에서는 ONB에서 polynomial basis 로의 변환 방법을 통해 유한필드 곱셈기를 설계 및 구현하였다. 구현된 곱셈기는 ONB에서 구현된 다른 곱셈기에 비해 적은 하드웨어 자원과 복잡도를 갖는다. 현재 진행중인 연구는 제안된 이론을 ECC의 스칼라 곱셈 전체로 적용범위를 확장하여 연구중이며 특히 가장 많은 자원과 시간을 요구하는 역수를 구하는 과정을 ONB와 polynomial basis를 혼합하여 구현할 수 있는 방법을 연구중이다.

[참 고 문 헌]

[1] S. Sutikno, A. Surya, "An Architecture of F_{2^m} Multiplier for Elliptic Curves Crypto-system," *IEEE ISCAS*, pp. 279-282, 2000.

[2] G. Orlando, C. Paar, "A Super-Serial Galois Fields Multiplier for FPGAs and its Application to Public-Key Algorithms," *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 232-139, 1999.

[3] M. Rosing, *Implementing Elliptic Curve Cryptography*, Manning Pub, 1999.

[4] G. B. Agnew, R. C. Mullin, S. A. Vanstone, "An Implementation of Elliptic Curve Cryptosystems Over $F_{2^{155}}$," *IEEE Journal on Selected Areas in Communications*, 11, pp. 804-813, 1993.

[5] A. Menezes, S. Vanstones, "Elliptic Curves Cryptosystems and Their Implementation," *Journal of Cryptology*, 6, pp. 209-223, 1993.

[6] E. Mastrovito, "VLSI Architecture for Computation in Galois Fields," PhD thesis, Linkoping University, Dept, Electr. Eng., Linkoping, Sweden, 1991.

[7] C.Wang, T. Truong, H. Shao, L.Deutsch, J. Omura, I. Reed, "VLSI Architectures for Computing Multiplications Invers in $GF(2^m)^n$," *IEEE Transactions on Computers*, 34, 1985.