공개키 암호화 시스템을 위한 효율적인 곱셈기 설계

김현성*, 전준철*, 이형목*, 유기영*

*경북대학교, 컴퓨터공학과

Design of an Efficient Multiplier for Public Key Cryptosystem

Hyun-Sung Kim*, Jun-Cheol Jeon*, Hyung-Mok Lee*, Kee-Young Yoo*

*Department of Computer Engineering Kyungpook National Univ.

요 약

본 논문에서는 유한체 연산을 바탕으로 하는 공개키 암호화 프로세서를 위한 효율적인 곱셈기 구조를 제안한다. 제안된 곱셈기는 다항식으로 항이 모두 1인, AOP, 기약 다항식을 사용하였다. 제안된 구조는 LFSR 구조에 기반한 곱셈기 구조이다. VHDL 코드 시뮬레이션 결과 제안된 구조가 기존의 구조에 비해서 보다 효율적인 구조 복잡도를 가짐을 알 수 있었다.

I. 서론

최근 인터넷의 급속한 확산으로 전자상거래가 활발하게 이루어지고 있다. 이러한 전자 상거래에서 보안 기술은 필수적이다. 보안을 위한 알고리즘은 키의 특성에 따라 비밀키/공개키 암호 알고리즘으로 나눌 수 있다. 특히, 키 분배에 효율적인 공개키 암호화 알고리즘을 이용한 보안 기술이 많이 사용되고 있다[1][2].

대부분의 공개키 암호화 시스템에서는 정수 혹은 유한필드 상에서의 모듈러 지수 연산을 기본으로 하고 있다[1][2]. 그러므로, 효율적인 암호화 시스템 구현을 위해서 효율적인 지수기 설계는 아주 중요하다. 특히, 지수기 설계에 있어서 모듈러 곱셈기는 가장 기본이 되는 구조이다. 다양한 곱셈기가 여러 가지 구조를 기반으로 제안되었다[4][5][6][8].특히, LFSR (Linear Feedback Shift Register) 구조를 기반으로 Fenn은 효율적인 모듈러 곱셈기를 설계하였다[6]. 전체적인 구조 복잡도로 와 시간 복잡도를 가진다. Yeh와 Wang은 시스톨릭 어레이 기반의 병렬/순차 곱셈기를 설계하였다[4][5][8].

본 논문에서는 유한체 상에서 공개키 암호화 시스템을 바탕으로 스마트 카드와 같은 IC카드의 암호화 프로세서에 효과적으로 사용 가능한 새로운 구조의 곱셈기를 제안한다. 제안된 구조는 다항식 기저를 사용하고 기약 다항식으로는 다항식의 항이 모두 1인 속성의 AOP를 사용한다. 제안된 구조는 내적 알고리즘에 기반 한 구조이다. 제안된 구조의 검증을 위해 Altera사의 MAX+PLUS를 이용하여 시뮬레이션을 수행하였다. 시뮬레이션 결과 본 논문에서 제안된 구조가 기존의 구조에 비해서 보다 효율적인 구조 복잡도를 가짐을 알 수 있었다.

II. 유한필드

유한필드는 Galois 필드(GF)로도 불린다. 비록 유한필드가 많은 소수의 지수 차수에 대해서 존재하지만, 암호학에서 주로 사용되는 필드는 소수 q에 대한 소수 유한필드 GF(q)와 양수 m에 대한 이진 유한필드 GF(2^m)이다. 유한필드 GF(2^m)은 길이가 m인 2^m개의 가능한 비트 스트링으로 구성된다[3].

$$GF(2^m)=\{(a_{m-1}\ a_{m-2}\ \cdots\ a_1\ a_0)|a_i\in GF(2),\ 0\le i\le m-1\}$$

여기서 GF(2)는 GF(2^m)의 하부필드(Sub-field)라 불리고, 유한필드 GF(2^m)은 GF(2)의 확장필드

라고 한다. 필드에서의 원소들을 표현하기 위해서는 정규기저 (Normal Basis) 표기법, 이원기저 (Dual Basis) 표기법, 다항식기저(Polynomial Basis) 표기법 등의 세 가지 표기법이 있다. 그러나 정규기저 표기법과 이원기저 표기법을 이용한 연산에서는 연산전후에 기저변환 단계를 거쳐야 하는 문제가 있다. 본 논문에서는 다항식기저 표기법으로 필드상의 원소들을 표현한다. 다항식기저 표기법에서의 GF(2^m)의 각 원소는 다음과 같이 m차수 미만의 다항식으로 표현된다.

$$a(x)=a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\cdots+a_1x+a_0,$$
$$a_i\in GF(2),\ 0\le i\le m-1.$$

GF(2^m)상에서 연산 후 연산 결과를 필드의 원소로 만들기 위해서는 차수 m의 기약 다항식(Irreducible Polynomial)이 필요하고, 이 기약 다항식을 이용한 모듈러 연산이 필요하다.

GF(2)의 원소를 계수로 갖는 m차의 기약 다항식을 $f(x)$ 할 때, 다항식의 계수가 모두 '1' 인 다항식 $f(x)=x^m+x^{m-1}+x^{m-2}+\cdots+x+1$을 AOP(All One Polynomial)라 한다. 이 방정식의 근을 α라고 두면, AOP는 $\alpha^{m+1}+1=0$, (m+1은 소수) 의 속성을 가진다[6]. 100보다 작은 m에 대해서 m이 2, 4, 10, 12, 18, 28, 36, 52, 58, 60, 66, 82 일 때 기약 다항식으로서의 AOP를 만족한다.

즉, 본 논문에서는 AOP의 속성을 이용하여 GF(2^m)보다 하나 확장된 GF(2^{m+1})상에서 곱셈 연산이 수행된다. GF(2^{m+1})상의 한 원소 A는 $A=A_m\alpha^m+A_{m-1}\alpha^{m-1}+A_{m-2}\alpha^{m-2}+\cdots+A_1\alpha+A_0$, ($A_m$=0)으로 표현된다. 여기서, $A_i=a_i+A_m$, $0\le i\le m-1$ 이다. 또한, 기저 { 1, α, α^2 ,…, α^{m-1}, α^m }은 GF(2^m)상의 표준기저에서 하나 확장된 기저이다. 이러한 속성은 곱셈연산을 수행하는데 있어서 효율적인 모듈러 감소를 제공할 수 있다.

III. 제안된 곱셈기

1. 곱셈 알고리즘

유한필드 상의 모듈러 곱셈을 위해서 AOP 속성에 기반 한 다음과 같은 속성들이 필요하다.

정의 1 : $A^{(1)}$과 $A^{(-1)}$을 각각 원소 A=(A_m, A_{m-1}, A_{m-2}, A_{m-3}, … , A_1, A_0)의 한 비트 오른쪽 시프트와 왼쪽 시프트라고 두자. 즉,

$A^{(1)}$=(A_{m-1}, A_{m-2}, A_{m-3}, … , A_1, A_0, A_m)

$A^{(-1)}$=(A_0, A_m, A_{m-1}, … , A_3, A_2, A_1).

그러면, $A\alpha \bmod p(x)$ 와 $A\alpha^{-1} \bmod p(x)$ (여기서 $p(x)=x^{m+1}+1$) 는 각각 다음과 같이 나타낼 수 있다.

$$A\alpha \bmod p(x) = A^{(1)}$$
$$A\alpha^{-1} \bmod p(x) = A^{(-1)}$$

위의 두 식에서 내적은 다음과 같이 정의된다.

정의 2 [7] : A와 B를 GF(2^m)상에 원소라고 두면, 두수의 내적은 다음과 같이 정의 된다.

$$A\cdot B=\left(\sum_{j=0}^{m}A_j\alpha^j\right)\cdot\left(\sum_{j=0}^{m}B_j\alpha^j\right)$$
$$=\sum_{j=0}^{m}A_jB_j\alpha^{2j}$$

정의 2에 의해서 $A^{(i)}$와 $B^{(-i)}$의 내적은 다음과 같다.

$$A^{(i)}\cdot B^{(-i)}=\left(\sum_{j=0}^{m}A_{\langle j-i\rangle}\alpha^j\right)\left(\sum_{j=0}^{m}B_{\langle j+i\rangle}\alpha^j\right)$$
$$=\sum_{j=0}^{m}A_{\langle j-i\rangle}B_{\langle j+i\rangle}\alpha^{2j}$$

여기서 $\langle x\rangle$는 $x \bmod m+1$의 결과를 의미한다. 즉,

$$A\cdot B=A^{(0)}B^{(0)}+A^{(1)}B^{(-1)}+A^{(2)}B^{(-2)}+\ldots+A^{(m)}B^{(-m)}$$

여기서 $A^{(0)}=A^{(-0)}=A$ 이다. 즉 내적을 이용한 모듈러 곱셈 알고리즘은 다음 그림 1과 같다.

	α^8	α^6	α^4	α^2	α^0
$A^{(0)}B^{(0)}$	A_4B_4	A_3B_3	A_2B_2	A_1B_1	A_0B_0
$A^{(1)}B^{(-1)}$	A_3B_0	A_2B_4	A_1B_3	A_0B_2	A_4B_1
$A^{(2)}B^{(-2)}$	A_2B_1	A_1B_0	A_0B_4	A_4B_3	A_3B_2
$A^{(3)}B^{(-3)}$	A_1B_2	A_0B_1	A_4B_0	A_3B_4	A_2B_3
$A^{(4)}B^{(-4)}$	A_0B_3	A_4B_2	A_3B_1	A_2B_0	A_1B_4
	M_4	M_3	M_2	M_1	M_0

그림 1: GF(2^4)상의 곱셈을 위한 내적알고리즘.

그림 1에서 결과 값 M은 곱셈 수행 후 재배치가 필요하다. 그러나 재배치 문제는 아주 쉽게 해결할 수 있으므로 본 논문에서는 고려하지 않기로 한다. 다음절에서는 내적 연산에 기본 한 곱셈구조를 제시한다.

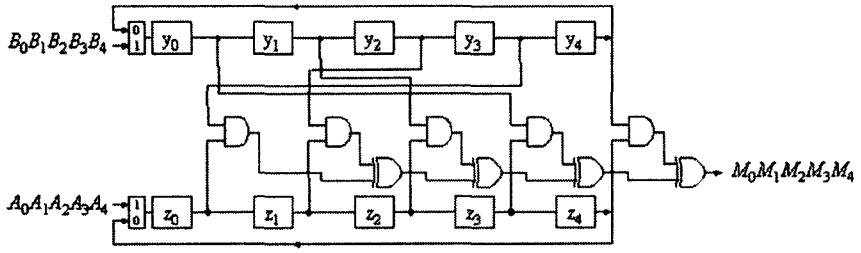

그림 2: GF(2^4)상의 내적곱셈기.

2. 곱셈기 디자인

본 논문에서 제안한 구조는 LFSR 구조에 기반한 구조이다. 또한 기본 알고리즘으로 그림 1에서 제시한 내적 곱셈 알고리즘을 사용한다. 효율적인 곱셈기 구조 이해를 위한 알고리즘은 다음과 같다.

[알고리즘 1] 내적곱셈기 알고리즘
입력 : A, B
출력 : $M=AB \bmod x^{m+1}+1$
단계1 for $i=m$ to 0
단계2 Right_Shift(y, B_i)
단계3 Right_Shift(z, A_i)
단계4 $M_i=0$
단계5 for $j=m$ to 0
단계6 for $k=0$ to m
단계7 if($k==m$) $M_j=M_j+y_k\times z_k$
단계8 else $M_j=M_j+y_k\times z_{m-k}$
단계9 Circular_Right_Shift(z)
단계10 Circular_Right_Shift(y)

알고리즘 1에서 Right_Shift(y, B_i)는 레지스터 y의 입력으로 B_i를 가지면서 오른쪽으로 1비트 시프트 연산을 의미하고, Circular_Right_Shift(z)는 z 레지스터의 오른쪽 1비트 순환 시프트를 의미한다.

알고리즘 1은 레지스터에 입력을 위한 부분과 곱셈 연산 처리를 위한 부분의 두 부분으로 나누어진다. 먼저, 레지스터 초기화를 위한 부분은 단계 1에서부터 단계 4까지의 연산이다. 매 순간마다 레지스터 y와 z에 B_i와 A_i값들이 한 비트씩 입력된다. 레지스터 초기화를 위해서는 전체 $m+1$ 클럭이 필요하다. 레지스터의 초기화가 끝나는 시점부터 매순간 한 비트의 결과 값이 출력된다. 즉 곱셈 연산을 위해서 단계 6의 인덱스 k를 갖는 for 루프가 병렬로 1 클럭만에 수행된다. 수행되는 연산은 그림 1에서 하나의 행에 해당된다. 연산이 끝나면 다음 연산을 위해서 레지스터 z와 y를 각각 오른쪽으로 1비트 씩 순환 시프트한다.

곱셈 연산을 수행하는데 전체 $2m+1$ 클럭이 필요하다. 원래, 레지스터 초기화와 곱셈 연산을 위해서 각각 $m+1$ 클럭이 필요하지만, 레지스터 초기화의 마지막 시점에 첫 번째 곱셈 연산을 수행하고 그 결과 값을 출력할 수 있다. 즉, 레지스터 초기화와 곱셈을 위한 첫 연산이 중복될 수 있다. 그러므로 전체 $2m+1$ 클럭이 필요하게 된다.

유한필드 GF(2^4) 상에서 구현된 곱셈기 구조는 그림 2와 같다. 제안된 곱셈기는 임의의 유한필드 GF(2^m)에 대해서 쉽게 일반화 될 수 있다.

IV. 회로 검증 및 성능평가

제안된 곱셈기는 하드웨어 기술언어인 VHDL로 코딩 되었다. 코딩된 회로의 시뮬레이션을 위해서 Altera사의 MAX+PLUS를 이용하여 시뮬레이션을 수행하였다. 시뮬레이션 툴은 회로의 복잡도 예측뿐만 아니라 제안된 곱셈기의 정상적인 기능을 검증하는 수단으로 이용되었다.

표 1은 제안된 곱셈기와 기존의 Fenn et al. [6] 구조와의 회로 복잡도 및 시간 복잡도의 비교를 보여준다.

표 1: 모듈러 곱셈기 비교

곱셈기 / 항목	Fenn's	제안된 구조
Function	$AB \bmod p(x)$	$AB \bmod p(x)$
Registers	$2m+2$	$2m+2$
AND gates	$m+1$	$m+1$
XOR gates	m	m
MUXes	$m+2$	2
Clock cycles	$2m+1$	$2m+1$

표 1에서 보여준 바와 같이 제안된 구조와 Fenn의 구조는 시간 복잡도 면에서는 동일하다. 그러나 제안된 구조는 구조 복잡도 면에서 Fenn의 구조에 비해서 훨씬 효율적임을 알 수 있다.

V. 결론

본 논문에서는 다항식 기저에 기반하는 유한체 상에서 효율적인 곱셈기를 설계하였다. 제안된 곱셈기는 AOP의 특성이 적용된 내적 곱셈 알고리즘을 기반으로 구현되었다.

표 1에서 보여준 바와 같이, 제안된 구조는 기존의 구조에 비해서 효율적인 하드웨어 복잡도를 가짐을 확인할 수 있었다. 또한 제안된 곱셈기는 상수 곱셈을 요구하는 구조에도 적용할 수 있다.

본 논문에서 제안한 곱셈기를 기반으로 스마트 카드 내에 탑재되는 효율적인 암호화 프로세서를 구현할 수 있을 것으로 기대된다.

참고문헌

[1] W.Diffie and M.E.Hellman, "New directions in cryptography," *IEEE Trans. on Info. Theory*, vol. 22, pp. 644-654, Nov. 1976

[2] T.ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. on Info. Theory*, vol. 31(4). pp. 469-472, July 1985

[3] R. Lidl, H.Niederreiter, and P.M.Cohn, *Finite Fields* (*Encyclopedia of Mathematics and Its Applications*), Cambridge University Press, 1997.

[4] C.S.Yeh, I.S.Reed, and T.K.Truong, "Systolic Multipliers for Finite Fields GF(2^m)," *IEEE Trans. Computers*, vol. C-33, pp. 357-360, 1984

[5] C.L.Wang and J.L.Lin, "Systolic Array Implementation of Multiplier for Finite Fields GF(2^m)," *IEEE Trans. on Circuits and Systems*, vol. 38, pp. 796-800, July 1991

[6] S.T.J.Fenn, M.G.Parker, M.Benaissa, and D.Taylor, "Bit-serial multiplication in GF(2^m) using irreducible all-one polynomials," *IEE Proc.-Comput. Digit. Tech.*, vol. 144, no. 6, pp. 391-393, Nov. 1997

[7] C.H.Liu, N.F.Huang, and C.Y.Lee, "Computation of AB^2 Multiplier in GF(2^m) Using and Efficient Low-Complexity Cellular Architecture," *IEICE Trans. Fundamentals*, vol E83-A, no. 12, pp. 2657-2663, Dec. 2000.

[8] 김현성, 유기영, "유한필드상에서의 곱셈기 설계 방법," 정보보호 연구회지, pp. 12-19, 2001년 4월.