제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

효율적인 몽고메리 모듈러 곱셈기의 설계

박혜영*, 유기영**
*경북대학교 정보보호학과
**경북대학교 컴퓨터공학과
e-mail : thewise@infosec.knu.ac.kr

A Design of Efficient Modular Multiplication based on Montgomery Algorithm

Hye-Young Park*, Kee-Young Yoo**
*Dept. of Information Security, Kyungpook National University
**Dept. of Computer Engineering, Kyungpook National University

요 약

본 논문에서는 몽고메리 모듈러 곱셈(Montgomery Modular Multiplication) 알고리즘을 이용하여 효율적인 모듈러 곱셈기를 제안한다. 본 논문에서 제안한 곱셈기는 프로그램 가능한 셀룰라 오토마타(Programmable Cellular Automata, PCA)를 기반의 구조로 설계되어 하드웨어 복잡도를 줄이고, 곱셈시 몽고메리 알고리즘을 이용하여 일반적인 나눗셈 없이 모듈러 연산을 수행하여 시간 복잡도를 최소화 한다. 제안된 곱셈기는 시간적, 공간적인 면에서 간단하고 효과적으로 구성되어 지수연산을 위한 하드웨어의 하부구조나 오류 수정 코드(Error Correcting Code)의 연산에서 효율적으로 이용될 수 있을 것이다.

1. 서론

오늘날 인터넷의 발달과 함께 쉽게 네트워크에 접근할 수 있게 됨에 따라 시스템에 대한 보호와 저장된 데이터 혹은 네트워크를 통해 전송되는 데이터에 대한 보안에 대한 필요성이 급증하고 있다. 이러한 필요를 충족시키기 위해 다양한 암호화 알고리즘과 빠르고 효율적인 알고리즘의 구현을 위한 많은 연구가 이루어지고 있다.

이러한 암호화 알고리즘의 구현은 대개 유한체 GF(p)상이나 GF(2^m)상에서 이루어 지고 있으며[1], 특히 복잡한 연산을 반복적으로 수행하는 공개키 암호화 시스템을 구현하기 위해서는 유한체 상에서의 효율적인 구현이 필수적이다.

공개키 암호화 시스템 중에서 Diffie-Hellman 이나 Elgamal Cryptosystem, 그리고 최근에 주목 받고 있는 타원 암호 시스템(Elliptic Curve Cryptosystem, ECC)은 유한체 상의 곱셈 연산과 지수 연산 그리고 나눗셈 연산을 필요로 한다[2]. 기본적으로 이러한 연산은 AB 혹은 AB^2 연산을 하부구조로 하여 구현이 가능하다.

A 와 B 의 곱을 계산하는 AB 연산은 다른 연산과는 달리 연산 후 늘어나는 자릿수를 제한하기 위해 모듈러 연산을 수행한다. 본 논문에서는 곱셈 연산 후에 추가적인 모듈러 연산을 피하기 위하여 몽고메리 모듈러 곱셈 알고리즘을 사용한다. 1985 년에 P. L. Montgomery 에 의하여 제안된 몽고메리 알고리즘은 $T = ABR^{-1} (\mathrm{mod}\, N)$ 을 계산하는 효율적인 알고리즘이다. 여러 연구들에서 몽고메리 알고리즘을 이용하여 효율적인 곱셈기와 지수기를 제안하고 있다. C. D. Water 는 몽고메리 모듈러 곱셈 연산을 위한 시스톨릭 어레이(systolic array)를 제안하였고[3], Koç 과 Acar 는 몽고메리 모듈러 곱셈 알고리즘을 유한체 GF(2^m)상에 적합하게 개선하였다[4]. 또한 GF(2^m)상의 몽고메리 곱셈기와 지수기를 [4], [5]에서 소프트웨어적으로 효율적으로 구현하여 제안하였다. Wu 는 Koç 과 Acar 의 알고리즘을 일반화하여 비트 병렬 구조의 곱셈기를 설계하였고, m^2-AND+(m^2-m/2)-XOR 의 하드웨어 복잡도를 가지며 T_A+(⌈$\log_2(m$-1)⌉+1)T_X 의 지연시간을 가진다[6]. 본 논문에서는 PCA 구조의 효율적인 몽고메리 모듈러 곱셈기를 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서 유한체에

대한 기본적인 개념과 특징을 살펴보고, 3 장에서는 몽고메리 모듈러 곱셈 알고리즘과 알고리즘을 위한 세부 단계를 소개한다. 4 장에서는 본 논문에서 제안된 곱셈기를 구현한 구조인 셀룰라 오토마타와 프로그램 가능한 셀룰라 오토마타에 대해 살펴보고, 5 장에서는 PCA 구조의 몽고메리 모듈러 곱셈기를 제안하며, 6 장에서는 제안된 구조를 기존의 구조와 비교 분석한다. 마지막으로 7 장에서 결론을 내린다.

2. 유한체

유한필드 혹은 갈로아 체로 불리는 유한체는 교환, 결합, 분배 법칙에 닫혀 있고, 덧셈, 뺄셈, 곱셈, 나눗셈 연산이 가능한 유한한 원소를 가지는 집합이다. 유한체의 위수(order)는 유한체의 원소의 개수이고 유한체의 위수인 자연수 q 는 소수(prime)이거나 소수의 지수 승이다. 유한체를 표기하는 다양한 방법이 있는데 위수가 소수 q 와 양의 정수 n 에 대하여 q^n 이라고 하면 Fq^n 혹은 GF(q^n)로 나타낼 수 있고, q^n 개의 원소를 가진 유한체는 본질적으로 하나뿐이다[7].

암호학에서는 일반적으로 두 종류의 유한체가 사용되는데, 소수 유한체(prime finite field)와 이진 유한체(binary finite field)가 그것이다. 본 논문에서는 이 두 가지 중에서 이진 유한체를 고려한다.

이진 유한체, GF(2^m)에서 원소들을 표기하기 위해서 다항식 기저 표기법(polynomial basis representation), 정규 기저 표기법(normal basis representation), 이원 기저 표기법(dual basis representation)등이 있다. 본 논문에서는 기저 변환 단계가 필요 없는 다항식 기저 표기법으로 원소를 표시한다.

다항식 기저 표기법에 따르면, 유한체 GF(2^m)상의 한 원소 A 가 길이 m 의 다항식이라면 다음과 같이 이진 다항식으로 표현 할 수 있다.

$$A = a_0 + a_1\alpha + ... + a_{m-2}\alpha + a_{m-1}\alpha \qquad (1)$$

여기서 $a_i (0 \le i \le m-1)$ 는 GF(2)의 원소이다[4].

3. 몽고메리 알고리즘

본 논문에서는 모듈러 곱셈시 연산의 수행속도를 높이기 위해 몽고메리 알고리즘을 사용한다. P. L. Montgoemry 는 $T = ABR^{-1} (\mathrm{mod}\, N)$ 을 계산하는 효율적인 알고리즘을 제안하였다. 몽고메리 알고리즘은 다음의 식과 같은 결과값을 계산한다.

$$T = ABR^{-1} (\mathrm{mod}\, N) \qquad (2)$$

여기서 $A, B < N < R$ 이다. R 과 N 은 서로 소이며, 이러한 관계 때문에 다음의 식 (3)을 만족하는 두 개의 수 R^{-1} 와 N'가 존재한다.

$$RR^{-1} + NN' = 1 \qquad (3)$$

이 때 R^{-1} 는 모듈러 N 에 대하여 R 의 역수이고 다음의 $RR^{-1} = 1 (\mathrm{mod}\, N)$ 식이 성립한다. R^{-1} 와 N'는 확장 유클리드 알고리즘(Extended Euclidean Algorithm)을 이용하여 구할 수 있다[8].

몽고메리 알고리즘의 가장 중요한 특징은 R 을 r 의 멱수(power)로 선택함으로써 본질적으로 빠른 연산인 시프트 연산을 통해 다른 모듈러 곱셈보다 빠르게 연산이 가능하다는 것이다. 따라서, 몽고메리 모듈러 곱셈 알고리즘은 N 에 의한 나눗셈에 의해 모듈러 곱셈을 수행하는 일반적인 방법보다 R 에 의한 나눗셈과 모듈러 감소시키는 방법을 이용하여 더 빠른 연산 속도를 얻는다.

몽고메리가 제안한 알고리즘은 다음과 같다.

Input : A, B, R, N'

Output: $T = ABR^{-1} (\mathrm{mod}\, N)$

Step 1 : $temp = A * B$

Step 2 : $U = temp * N' (\mathrm{mod}\, R)$

Step 3 : $T = (temp + U * N) / R$

Step 4 : If $T \ge N$ Then return $T - N$

Else return T

4. 셀룰라 오토마타

CA 는 규칙적으로 상호 연결된 많은 셀들로 구성되어 있는 유한 머신이다[9][10]. 각각의 셀들은 적용된 법칙과 자신과 연결된 이웃의 현재 상태 값에 따라 새로운 값으로 갱신된다. 이 때 이웃이란 자기 자신을 포함하여 셀의 상태 갱신에 직접적으로 영향을 주는 셀을 의미한다. CA 를 구성하는 중요한 요소는 각 셀의 상태 갱신에 적용되는 법칙과 여기에 직접적으로 관여하여 셀의 갱신에 직접적으로 영향을 미칠 수 있는 이웃 셀의 개수이다.

셀룰라 오토마타의 법칙은 셀의 이웃과 시간을 사용하여 불린 함수(Boolean function)로 표현할 수 있다. 시간 n 에서 m 번째 셀의 상태 값을 $S_n(m)$ 이라고 했을 때, 법칙 90 은 다음과 같은 수식으로 표현 될 수 있다.

$$S_{n+1}(m) = S_n(m-1) \oplus S_n(m+1) \qquad (4)$$

여기서 $\oplus$ 는 XOR 연산이다[11]. 이와는 다르게 $S_+ = S^- \oplus S^+$ 로 표시할 수도 있다. 법칙 150 은 $S_+ = S^- \oplus S \oplus S^+$ 로 표현이 가능하고, 법칙 170 은 $S_+ = S^+$, 법칙 240 은 $S_+ = S^-$ 로 표현 할 수 있다.

표. 1 다양한 법칙에 따른 CA 의 상태변화

	000	001	010	011	100	101	110	111
90	0	1	0	1	1	0	1	0
150	0	1	1	0	1	0	0	1
170	0	1	0	1	0	1	0	1
240	0	0	0	0	1	1	1	1

CA 에 적용된 법칙이 XOR 연산만으로 이루어진 것을 선형(linear) CA, XOR 연산 이외의 연산으로 이루어진 것을 비선형(non-linear) CA 라고 한다. 이 가운데 XOR 연산과 XNOR 연산이 사용되는 CA 를 추가적(additive) CA 라고 한다. 그리고 CA 구조에서는 셀의

가장 오른쪽 셀의 오른쪽 이웃과 가장 왼쪽 셀의 왼쪽 이웃이 존재하지 않으므로 이를 고려하는 경계조건을 결정해야 한다. 가장 왼쪽 셀의 왼쪽 이웃과 가장 오른쪽 셀의 오른쪽 이웃이 서로 이웃 한 것으로 간주하는 PBCA(Periodic Boundary CA), 가장 왼쪽 셀의 왼쪽 이웃과 가장 오른쪽 셀의 오른쪽 이웃을 모두 '0'으로 간주하는 NBCA(Null Boundary CA) 등이 있다.

본 논문에서는 NBCA 의 경계조건을 가진 1-차원, 2-상태, 3-이웃 셀룰라 오토마타를 고려한다.

PCA 는 셀마다 입력 값을 컨트롤하여 동일한 셀에 클럭마다 다른 법칙을 적용 할 수 있는 CA 를 말한다.

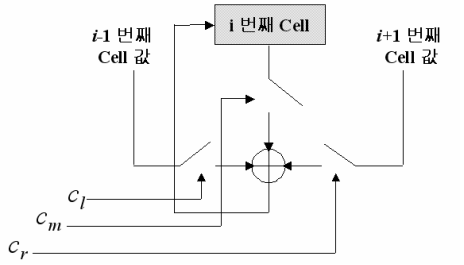

그림 1. 3-이웃 PCA 구조

위의 **그림 1** 은 3-이웃 PCA 의 기본적인 구조이다. 그림에서 C_l 은 i-1 번째 셀을 제어하는 컨트롤 값, C_m 은 i 번째 셀을 제어하는 컨트롤 값, C_r 은 i+1 번째 셀을 제어하는 컨트롤 값이다. 예를 들어 표 1 에서 설명된 법칙 170 은 i+1 번째 셀의 값을 가져와 그 다음 클럭에 i 번째 셀을 갱신한다. 이를 구현하기 위하여 C_l=0, C_m=0, C_r=1 의 컨트롤 값을 사용하면 된다. 이처럼 PCA 구조를 사용하면 컨트롤 값을 조절하여 필요한 법칙을 필요에 따라 적용할 수 있다.

5. 제안된 몽고메리 모듈러 곱셈기

본 장에서는 PCA 구조를 기반으로 몽고메리 모듈러 곱셈 알고리즘을 이용한 곱셈기를 제안한다.

[알고리즘 1] 개선된 몽고메리 모듈러 곱셈기

$$\begin{aligned}
\text{Input} \quad & : A=(a_0,a_1,\ \dots\ ,a_{m-2},a_{m-1}),\\
& \quad B=(b_0,b_1,\ \dots\ ,b_{m-2},b_{m-1}),\\
& \quad N=(n_1,\ \dots\ ,n_{m-3},n_{m-2})\\
\text{Output:}\ & T=(t_0,t_1,\ \dots\ ,t_{m-2},t_{m-1})\\
& \quad =ABR^{-1}(\bmod N)\\
\text{Step1} \quad & : T=0\\
\text{Step2} \quad & : \text{For } i=m-1 \text{ DownTo } 0\\
\text{Step3} \quad & : \quad T=T+A*b_i\\
\text{Step4} \quad & : \quad T=T+t_0*2*N\\
\text{Step5} \quad & : \quad T=T/2
\end{aligned}$$

여기서 R 은 2^m 이고 A 와 B 는 N 보다 작은 수이다. 개선된 몽고메리 모듈러 곱셈 알고리즘에서는 바이너리 값으로 A, B, N 을 입력 받아 T 의 값을 출력한다.

Step 3 를 연산한 후에 T 의 값이 짝수이면 t_0 가 0 이고, T 의 값이 홀수이면 t_0 가 1 이 된다. Step 4 에서 이 t_0 의 값에 따라 N 이 더해지는데, N 이 더해지는 것에 상관없이 결과값인 T 는 항상 짝수가 되고, 다음 단계인 Step 5 에서 2 로 나누기 연산이 가능하다. Step 4 와 Step 5 에서 가장 하위 비트가 필요 없기 때문에 N 의 최하위 비트인 n_0 는 XOR 연산에서도 제외되고, 따라서 입력하지 않는다.

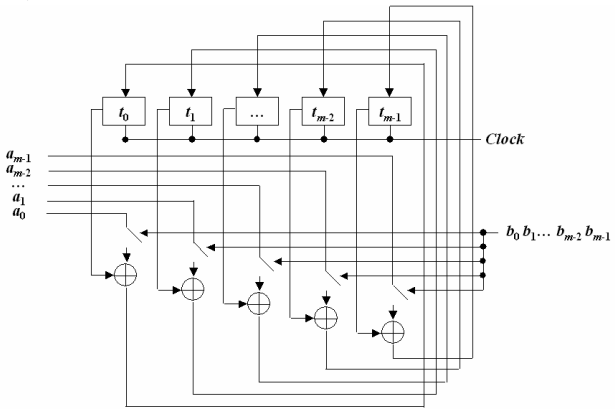

그림 2. $T=T+A*b_i$ 를 연산하는 PCA 구조

위의 그림 2 에서는 Step 3 의 연산을 수행하게 되고, m – bit 의 레지스터와 m 개의 XOR 게이트, m 개의 스위치가 필요하다.

Step 3 을 연산한 후, 결과 값인 T 의 값을 이용하여 Step 4 의 연산을 수행한다. 이때 사용되는 중요한 값이 t_0 이다. t_0 는 T 의 최하위 비트로써 T 의 값이 홀수인지 혹은 짝수인지를 판별하는 역할을 한다. 만약 T 의 값이 홀수이면 N 의 값을 더해 주고 짝수이면 N 의 값을 더하지 않는다. 이 연산을 위해서는 m-1 개의 XOR 게이트와 m-1 개의 스위치가 필요하다. Step 4 를 PCA 구조로 구현하면 다음과 같다.

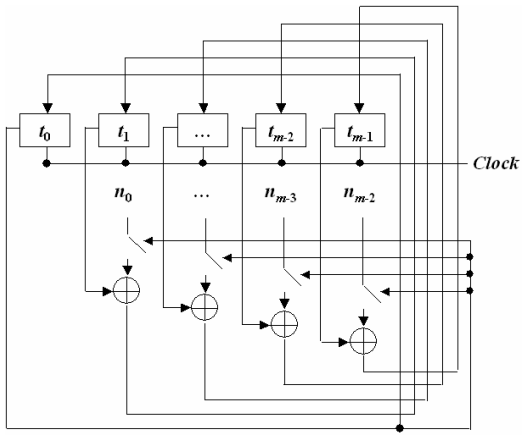

그림 3. $T=T+t_0*2*N$ 을 연산하는 PCA 구조

마지막의 단계 5 의 연산을 수행하기 위해서 NBCA 의 경계조건과 법칙 170 을 적용한 PCA 를 구현한다. 그러므로 각 셀들은 자신의 오른쪽 셀의 값을 이용하여 자신의 셀 값을 갱신하고, 왼쪽 셀의 값은 이용하지 않는다. 따라서 오른쪽 셀을 제어하는 컨트롤 값인

C_r 은 항상 1 이 되고, 왼쪽 셀을 제어하는 컨트롤 값인 C_l 은 항상 0 이 된다. 이와 같은 PCA 구조는 T 의 값을 2 로 나누는 연산을 수행하게 된다.

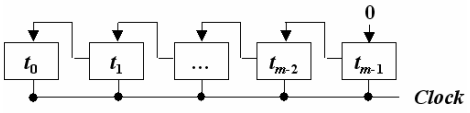

그림 4. $T = T / 2$ 를 연산하는 PBCA 구조

앞에서 구현한 그림 2, 3, 4 의 구조를 병합하여 아래와 같은 몽고메리 모듈러 곱셈기를 얻을 수 있다.

그림 5 는 앞의 그림에서 보여준 PCA 구조의 효율적인 특성을 모두 가지며, 결과적으로 별도의 모듈러 리덕션 과정 없이 곱셈 연산을 수행한다.

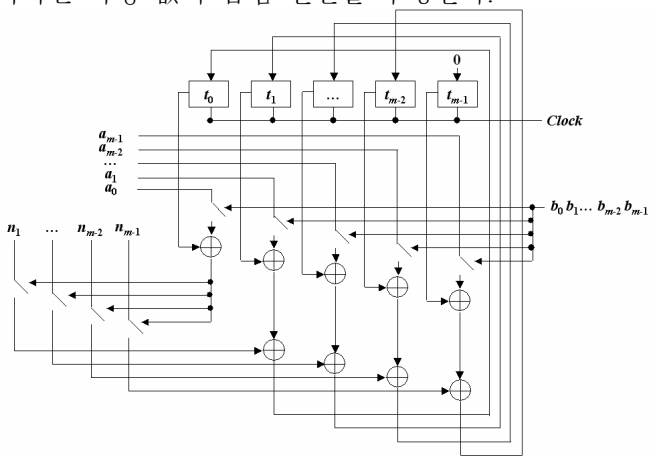

그림 5. 개선된 몽고메리 모듈러 곱셈기

위의 곱셈기는 m 클럭 사이클이 지난 후, 몽고메리 모듈러 곱셈 값을 연산하고 m 의 레지스터와 $(2m\text{-}1)$개의 XOR 게이트, $(2m\text{-}1)$개의 스위치를 필요로 한다.

6. 비교 및 분석

본 장에서는 5 장에서 제안한 곱셈기 구조의 성능을 분석하고 다른 관련된 구조와 비교한다. 본 논문의 결과를 Wu 의 연구 결과와 비교해 보면 표 2 와 같다.

표. 2 비교 결과

	Wu[6]	제안된 곱셈기
수행연산	ABR^{-1}	ABR^{-1}
레지스터	m	m
하드웨어 복잡도	m^2-AND + $(m^2\text{-}m/2)$-XOR	$(2m\text{-}1)$-XOR + $(2m\text{-}1)$-SWITCH
지연시간	$T_A + (\lceil \log_2(m-1)+1 \rceil) T_X$	$2T_S + 2T_X$
사용된 기약다항식	irreducible trinomial	all
수행 알고리즘	몽고메리 곱셈 알고리즘	몽고메리 곱셈 알고리즘
구현구조	비트 병렬 (bit-parallel)	PCA

Wu[6]가 제안한 비트 병렬 구조의 곱셈기는 기약 다항식으로 삼항식의 구조를 가질 때, m^2 개의 AND 게이트와 $(m^2\text{-}m/2)$개의 XOR 게이트를 가지며 m 개의 레지스터가 필요하다. 이에 비해 제안된 PCA 구조의 곱셈기는 $(2m\text{-}1)$개의 XOR 게이트와 $(2m\text{-}1)$개의 SWITCH 를 필요로 한다.

본 논문에서 제안된 곱셈기는 Wu 의 비트 병렬 곱셈기보다 클럭 사이클이 더 필요하지만 하드웨어 복잡도가 훨씬 적고, 제한된 기항 다항식을 사용하지 않으므로 더 광범위하게 사용될 수 있을 것이다.

7. 결론

본 논문에서는 개선된 몽고메리 모듈러 곱셈 알고리즘을 이용하여 PCA 구조의 효율적인 곱셈기를 제안하였다. 제안된 구조는 NBCA 의 특성을 이용하여 $GF(2^m)$상에서 추가적인 모듈러 연산 없이 결과값을 계산한다. Wu 의 연구 결과에 비해 일반적인 기약 다항식에 모두 사용이 가능하고, 하드웨어 복잡도도 눈에 띄게 줄일 수 있었다.

또한 제시된 곱셈기는 간단하고 시간과 공간적인 면에서 효율적이므로 모듈러 지수연산이나 오류 수정 코드 연산의 하부구조로 사용이 가능하다.

참고문헌

[1] E. R. Berlekamp, Bit-serial Reed-Solomon encoders, IEEE Trans. IT-28, vol. 6, pp. 869~874, 1982.

[2] T. R. N. Rao and E. Fujiwara, Error-Control Coding for Computer Systems, Engle-wood Cliff, NJ: Parentice-Hall, 1989.

[3] C. D. Walter, "Systolic modular multiplication," IEEE Trans. on Computers, vol. 42, pp. 376-378, 1993.

[4] Ç. K. Koç and T. Acar, "Fast Software Exponentiation in $GF(2^k)$," Proc. 13th Symp. Computer Arithmetic, pp. 279-287, July 1997.

[5] Ç. K. Koç and T. Acar, "Montgomery Multiplication in GF(2k)," Designs, Codes, and Cryptography, vol. 14, pp. 57-69, 1998.

[6] H. Wu, "Montgomery Multiplier and Squarer for a Class of Finite Fields," IEEE Trans. on Computers, vol. 51, pp. 521-529, 2002.

[7] IEEE P1363/Draft 13, Standard Specifications for Public-Key Cryptography, In http://grouper.ieee.org/groups/1363

[8] Ç. K. Koç and T. Acar, Burton S. Kaliski, Jr, "Analyzing and comparing Montgomery Multiplication algorithms," IEEE Micro, vol. 16, no. 3, pp. 26~33, 1996.

[9] M. Delorme and J. Mazoyer, Cellular Automata, Kluwer Academic Publishers, 1999.

[10] S. Wolfram, Cellular Automata and Complexity, Addison-Wesly Publishing Company, 1994.

[11] S. Wolfram, "Statistical Mechanics of Cellular Automata," Rev. Modern Physics, vol. 55, no. 3, pp. 601~644, 1983.