$GF(p)$ 상의 Scalable한 몽고메리 곱셈기

이광진°, 장용희, 권용진
한국항공대학교
{leekj°, yhjang, yjkwon}@tikwon.hangkong.ac.kr

A Scalable Architecture of Montgomery Multiplier on $GF(p)$

Kwang-Jin Lee°, Yong-Hee Jang, Yong-Jin Kwon
Dept. of Telecom. & Inform. Eng., of Hankuk Aviation University

요 약

최근 인터넷의 발달과 함께 인터넷 상에서의 데이터 보안에 대한 요구가 매우 증가되고 있다. 그래서 공개키 또는 비밀키 알고리즘을 사용하여 데이터 보안을 해결하고 있다. 대부분의 공개키 알고리즘은 모듈러 연산들을 기반으로 하고 있으며 이 중 복잡도가 가장 높은 모듈러 멱승 연산은 모듈러 곱셈 연산을 반복 수행하여 계산된다. 그래서 모듈러 곱셈연산을 효율적으로 계산하기 위한 많은 방법들이 제안되어 왔으며 하드웨어 구현 시 속도와 효율성 문제로 몽고메리 곱셈기에 대한 연구가 주목을 받아 왔다. 현재 몽고메리 곱셈 알고리즘을 이용한 곱셈기는 대부분이 성능과 면적만을 고려한 구조로 보안성 향상을 위해 입력 데이터의 비트수 증가 시 곱셈기의 구조 변경이 요구된다. 따라서 본 논문에서는 비트수 길이가 변하더라도 곱셈기 구조는 변함이 없는 $GF(p)$상에서의 Scalable한 몽고메리 곱셈기 구조를 제안한다. Scalable한 곱셈기의 구조는 FPGA와 같이 메모리를 포함하는 하드웨어 플랫폼에 적합하다. 제안된 구조는 Xilinx FPGA를 이용하여 하드웨어로 구현하며 ModelSim Tool을 통해 기능 및 타이밍 시뮬레이션을 수행한다.

1. 서 론

최근 네트웍 기술의 발달과 더불어 음성, 이미지, 영상 등과 같은 대량의 멀티미디어 데이터들의 교환 및 저장과 전자 상거래와 같은 인터넷을 통한 정보 서비스를 사용자들이 신뢰하며 사용하기 위해서는 정보 시스템의 보안과 처리 속도가 보장되어야 한다. 데이터의 비밀 전송과 전자상거래 등에서 데이터 전송의 안전성과 신뢰성을 보장하기 위해 암호화 알고리즘을 이용하여 암호화된 정보를 전송한다. 암호화 알고리즘은 암호화키의 종류에 따라 관용키 알고리즘과 공개키 알고리즘으로 나누어진다. 관용키 알고리즘의 경우 암호화와 복호화에 사용되는 키가 같으며 암호화 속도가 빠른 반면 암호화 키를 전송해야한다. 따라서 안전성이 떨어지며 암호화 키의 보유수가 ($O(n^2)$) 많다는 단점을 가지고 있다. 공개키 알고리즘은 암호화 키와 복호화 키가 서로 다르며 암호화 키의 전송이 필요없다. 따라서 관용키 알고리즘에 비해 보다 안전하며 비밀키의 보유수가($O(n)$) 적다. 그러나 암호화의 속도가 느려서 대용량 데이터의 암호화에는 적합하지 않으며 주로 인증 및 서명, 키 분배 등에 사용된다. 이러한 암호 알고리즘들을 소프트웨어로 구현할 경우, 쉽게 구현이 가능하며 적은 양의 데이터를 암호화 할 경우에는 빠른 속도로 처리할 수 있다. 그러나 날로 증가하는 대량의 멀티미디어 데이터를 실시간 고속 전송과 인터넷을 이용한 전자상거래에 사용되기 위해서는 암호 알고리즘의 하드웨어적 구현이 필요하다. 오늘날 암호시스템을 효율적으로 수행하는 하드웨어의 개발이 관심의 대상이 되고 있고 암호시스템의 효율적인 수행을 위해서는 연산기의 효율적인 연산이 필요하다. 특히 공개키 암호 알고리즘에서 수학적인 해를 찾기 위해서는 매우 오랜 시간이 소요된다. RSA 알고리즘에서와 같이 유한체 상에서의 연산은 덧셈, 뺄셈의 경우에는 각각의 비트 논리 연산만을 사용하기 때문에 하드웨어의 효율성은 곱셈연산에 의해 좌우된다[1][2]. 따라서 곱셈연산의 효율적인 구현은 암호 시스템의 효율성에 많은 영향을 미치기 때문에 구조적으로 간단하고, 연산시간을 효율적으로 개선 되도록 하는 연구가 진행되고 있다. 모듈러 지수연산의 고속화를 위하여 모듈러 승산 횟수를 줄이는 방법과 승산부분을 고속화하는 방법이 주로 연구되어 왔고, 기존 연구의 경우 하드웨어 면적을 최소하기 위한 동일 회로의 반복적인 수행 또는 하드웨어의 수행속도를 높이는 연구를 진행하여 왔다. 그러나 암호 강도를 높이기 위해 키의 값을 증가할 경우 이러한 하드웨어 구조는 입력 값의 길이가 변함에 따라 전체적인 하드웨어 구조의 변경이 요구된다. 본 논문에서는 유한체 상에서의 연산 시 가장 많은 소비시간이 소요되는 곱셈 연산에 대해 몽고메리 곱셈 알고리즘을 이용하며 입력 값의 크기에 따라 하드웨어의 구조적 변경 없이 적용할 수 있는 scalable한 몽고메리 곱셈기의 구조를 제안한다[3].

본 논문의 2장에서는 $GF(p)$ 상의 몽고메리 알고리즘을 소개한 후 3장에서는 scalable한 몽고메리 곱셈기의 구조를 제안한다. 4장에서는 제안한 곱셈기 구조의 하드웨어 구현 결과를 분석한다.

2. $GF(p)$ 상의 몽고메리 곱셈 알고리즘

몽고메리 알고리즘은 모듈러 연산을 나눗셈으로 처리하지 않고 쉬프트연산과 덧셈연산만으로 처리되므로 효율적인 하드웨어 구현에 적합한 알고리즘이다[4][5][6].

식(1)은 $GF(p)$상에서의 몽고메리 곱셈 알고리즘을 나타낸다.

$$C = AB2^{-n} \bmod p \qquad (1)$$

여기서 $n-bit$는 식(2)의 조건을 만족한다.

본 논문은 과학기술부·한국과학재단지정「한국항공대학교 인터넷 정보검색연구센터」의 연구비 및 IDEC의 지원으로 수행되었음

$$0 \le A,\ B < p < 2^n,\ \gcd(p,2) = 1 \quad (2)$$

AB의 최하위 n-$bits$가 '0'이면 식(1)은 모듈러 연산 없이 한 번의 곱셈과 n-$bits$ right shift로 계산되어진다.

즉, 몽고메리 곱셈 알고리즘의 중요한 점은 AB의 최하위 n-$bits$가 '0'이 되도록 적절한 p의 배수를 추가하는 것이다. 일반적으로 공개키 암호시스템에서 사용되는 n-$bits$는 수백에서 수천 $bits$이고 r-$bit \times r$-bit 곱셈기를 사용하기 위해 n-$bits$수를 $m \cdot r$-$bits$($m \cdot r = n$)로 나눌 필요가 있다.

A의 r-bit를 a_i ($0 \le i \le m-1$)로 표현하면 식(3),(4)와 같다.

$$A = a_{m-1}2^{r(m-1)} + \cdots + a_1 2^r + a_0 \quad (3)$$

$$A = (a_{m-1}, \cdots, a_1, a_0)_{2^r} \quad (4)$$

식(1)을 m번 반복 실행하면 식(5)와 같이 C의 값을 얻을 수 있다.

$$C := (C + a_i B)/2^r \bmod p \quad (5)$$

여기서 m은 식(6)과 같이 n-$bits$를 r-$bits$로 나눈 실수 값보다 크거나 같은 정수 중에서 가장 작은 정수 값이다.

$$m = \left\lceil \frac{n\text{-}bits}{r\text{-}bits} \right\rceil \quad (6)$$

C의 최하의 r-$bits$를 '0'으로 만들기 위해 식(7)과 같이 p의 배수 형태인 $t_i p$를 추가한다. 여기서 t_i는 식(8)과 같다.

$$C := (C + a_i B + t_i p)/2^r \quad (7)$$

$$t_i := (c_0 + a_i b_0) q \bmod 2^r \quad (8)$$

q는 몽고메리 곱셈기에서 처음 한번만 계산하여 사용한다.

$$q = -p^{-1} \bmod 2^r = -p_0^{-1} \bmod 2^r \quad (9)$$

식(7)의 $(C + a_i B + t_i p)$의 최하위 r-$bits$는 '0'이 된다.

$$\begin{aligned} & c_0 + a_i b_0 + t_i p \bmod 2^r \\ &= c_0 + a_i b_0 + (c_0 + a_i b_0)(-p^{-1})p \bmod 2^r \\ &= c_0 + a_i b_0 - (c_0 + a_i b_0) \bmod 2^r \\ &= 0 \end{aligned} \quad (10)$$

C의 값은 항상 n보다 작으나 p보다 항상 작지는 않다. 그러므로 몽고메리 곱셈기의 마지막 단계에서 C에서 p를 빼 주어야 한다.

$$c := \begin{cases} C & (C < p) \\ C - p & (C \ge p) \end{cases} \quad (11)$$

r-$bits$ 곱셈기를 적용한 $GF(p)$상에서의 몽고메리 알고리즘은 다음과 같다[4]. 입력 값 A, B, p의 n-$bits$를 r-bit로 나누고 식(6)의 m번 반복수행 후 식(11)을 적용한다.

Input	$A = (a_{m-1}, \cdots, a_1, a_0)_2^r$
	$B = (b_{m-1}, \cdots, b_1, b_0)_2^r$
	$P = (p_{m-1}, \cdots, p_1, p_0)_2^r,\ (0 \le A < p)$
	$q = -p^{-1} \bmod 2^r$
Output	$C = AB2^{-n} \bmod p$

$C := 0$
for i = 0 to $m-1$
  $z := 0$
  $t_i := (c_0 + a_i b_0) q \bmod 2^r$
  for j = 0 to $m-1$
    $S := c_j + a_i b_j + t_i p_j + z$
    if $(j \ne 0)$ then $c_{j-1} := S \bmod 2^r$
  $z := S/2^r$
$c_{m-1} := z$
if $(C > p)$ then $C := C - p$

3. $GF(p)$상의 scalable한 몽고메리 곱셈기의 구조

scalable한 구조는 입력 비트의 길이가 변하여도 unit을 재사용 하거나 반복적으로 사용될 수 있다.[4] 이 때 제어부와 메모리의 크기는 변경이 허용되나 데이터 패스는 변함이 없어야 한다. 그림1에서 scalable한 몽고메리 곱셈기의 구조를 나타내었으며 입력 비트수의 변함에 따른 Processing Unit은 변함이 없으며, 단지 메모리 사이즈의 변경만 발생한다. 본 논문에서 제시한 구조에서의 메모리 사이즈는 $m \times r$-$bits$ 이다. 그러므로 입력 비트의 길이에 따라 m의 길이만 변경된다. 본 논문에서 제안한 곱셈기 구조는 MUX, RAM, F/F, Arithmetic Core로 이루어지며, Arithmetic Core는 식(12),(13),(14)과 같이 몽고메리 곱셈 알고리즘에서의 곱셈, 덧셈, 뺄셈 연산을 수행한다.

$$S := c_j + a_j + z \quad (12)$$

$$S := c_j + t_i p_i + z \quad (13)$$

$$S := c_j - p_j \quad (14)$$

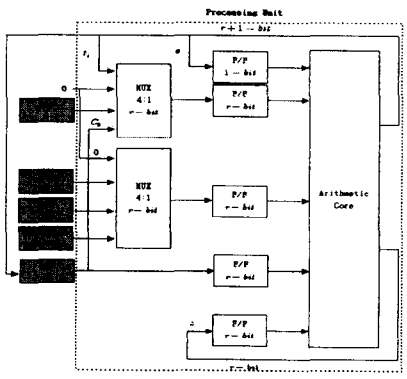

그림 1 Scalable한 몽고메리 곱셈기의 구조

그림 2 타이밍 시뮬레이션 결과

그림1과 같이 제안한 몽고메리 곱셈기에서 RAM A, B, p, q는 Write없이 read만을 수행하므로 Single Port RAM을 사용하며, RAM C는 read와 write 모두 사용하므로 DPRAM(Dual Port RAM)을 사용하여 같은 시간에 상이한 어드레스로 read, write 모두 가능하므로 지연 요소를 제거했다. processor unit-q는 식(9)와 같이 p의 역원을 계산하는 unit으로 확장 유클리드 알고리즘을 이용하여 q의 값을 구한다[7].

4. 하드웨어 구현결과

3장에서의 그림1에서 나타낸 Scalable한 몽고메리 곱셈기의 구조와 같이 MUX, F/F, RAM(Single/Dual Port), 곱셈 및 덧셈 연산을 하는 Arithmetic Core로 이루어지는 단순한 구조를 가진다. 하드웨어 면적은 표1과 같다.

표 1 하드웨어 면적

Device		개수
MUX	4:1 MUX($r-bits$)	2
F/F	$r-bits$	4
	$1-bit$	1
RAM	Single Port($r-bits \times m$)	4
	Dual Port($r-bits \times m+1$)	1
Arithmetic core	Multiplier($r-bits \times r-bits$)	1
	Adder($r-bits$)	3
	Subtracter($r-bits$)	1

Xilinx FPGA의 하드웨어 구현으로 제안한 구조의 기능 시뮬레이션과 타이밍 시뮬레이션을 수행하였으며 그림2는 타이밍 시뮬레이션의 결과로 그 결과가 이론치와 일치함을 나타낸다.

5. 결 론

고성능의 시스템이 개발됨에 따라 해킹 시간은 점점 줄어들고 있으며 이는 보안성 유지에 장애 요소이다. 일반적으로 키의 길이를 증가시킴으로써 암호강도 문제를 해결할 수 있다. 그러나 일반적인 곱셈기의 구조는 입력 데이터 비트수의 변경 시에 곱셈기의 구조 변경이 요구됨으로 기존의 곱셈기 구조는 사용할 수 없다. 따라서 입력 데이터 비트수의 변경에도 하드웨어 구조의 변경이 없는 scalable한 하드웨어 구조가 요구되고 있다. 이에 본 논문에서는 $GF(p)$ 상의 scalable한 몽고메리 곱셈기의 구조를 제안함으로써 어떠한 키 길이에서도 하드웨어 구조의 변경 없이 재사용 가능한 구조를 제안한다. 제안된 곱셈기는 메모리를 포함하는 FPGA의 원 칩 구현에 용이하며 입력되는 값의 크기에 독립적으로 수행되므로 여러 시스템에 쉽게 사용될 수 있다.

참고문헌

[1] 김무섭, 최용제, 김호원, 정교일, 개선된 몽고메리 알고리즘을 이용한 저면적용 RSA 암호 회로 설계, 정보보호학회 논문집 제12권 제5호, 2002년 10월

[2] Ming-Cheng sun, Chih-Pin Su, Chih-Tsun Huang and Cheng-Wen Wu, Design of Scalable RSA and ECC Crypto-Processor, IEEE, 2003

[3] Alexandre F. Tenca and Cetin K.Koc, A Scalable Architecture For Modular Multiplication Based on Montgomery's Algorithm, IEEE TRANSACTION ON COMPUTERS, VOL.52, No.9 SEPTEMBER 2003

[4] Akashi Satoh and Kohji Takano, A Scalable Dual-Field Elliptic Curve Cryptographic Processor, IEEE TRANSACTIONS N COMPUTERS, VOL.52, NO.4, APRIL 2003

[5] O. Nibouche, A. Bouridance and M. Bibouche, Architectures for Montgomery's multiplication, IEEE Proc-Comput. Digit. Tech., Vol. 150, No. 6, November 2003

[6] 이선근, 김환용, 공개키 암호시스템의 처리속도향상을 위한 모듈러 승산기 설계에 관한 연구, 전자공학회논문집 제40권 제4호, 200년 4월

[7] Tao Zhou, Xingjun Wu, Guoqiang Bai, Hongyi, Chen New Algorithm and Fast VLSI Implementation for Modular Inversion in Galois Field GF(p), IEEE 2002