단순 전력분석 공격에 대처하는 타원곡선 암호프로세서의 하드웨어 설계

최병윤*

Hardware Design of Elliptic Curve processor Resistant against Simple Power Analysis Attack

Byeong-yoon Choi*

이 논문은 2010년 동의대학교 교내 연구비 지원에 의한 결과임

요 약

본 논문은 스칼라 곱셈, Menezes-Vanstone 타원곡선 암호 및 복호 알고리즘, 점-덧셈, 점-2배 연산, 유한체상 곱셈, 나눗셈 등의 7가지 동작을 수행하는 GF(2^{191}) 타원곡선 암호프로세서를 하드웨어로 설계하였다. 단순 전력 분석에 대비하기 위해 타원곡선 암호프로세서는 주된 반복 동작이 키 값에 무관하게 동일한 연산 동작으로 구성되는 몽고메리 스칼라 곱셈 기법을 사용하며, GF(2^m)의 유한체에서 각각 1, (m/8), (m-1)개의 고정된 사이클에 완료되는 GF-ALU, GF-MUL, GF-DIV 연산장치가 병렬적으로 수행되는 동작 특성을 갖는다. 설계된 프로세서는 0.35um CMOS 공정에서 약 68,000개의 게이트로 구성되며, 시뮬레이션을 통한 최악 지연시간은 7.8 ns로 약 125 MHz의 동작속도를 갖는다. 설계된 프로세서는 320 kps의 암호율, 640 kbps을 복호율 갖고 7개의 유한체 연산을 지원하므로 다양한 암호와 통신 분야에 적용할 수 있다.

ABSTRACT

In this paper hardware implementation of GF(2^{191}) elliptic curve cryptographic coprocessor which supports 7 operations such as scalar multiplication(kP), Menezes-Vanstone(MV) elliptic curve cipher/decipher algorithms, point addition(P+Q), point doubling(2P), finite-field multiplication/division is described. To meet structure resistant against simple power analysis, the ECC processor adopts the Montgomery scalar multiplication scheme which main loop operation consists of the key-independent operations. It has operational characteristics that arithmetic units, such GF_ALU, GF_MUL, and GF_DIV, which have 1, (m/8), and (m-1) fixed operation cycles in GF(2^m), respectively, can be executed in parallel. The processor has about 68,000 gates and its simulated worst case delay time is about 7.8 ns under 0.35um CMOS technology. Because it has about 320 kbps cipher and 640 kbps rate and supports 7 finite-field operations, it can be efficiently applied to the various cryptographic and communication applications.

키워드

타원곡선암호, 공개키 암호, 전력분석공격, 스칼라 곱셈, SoC

Key word

ECC cryptography, Public-key cryptography, Power analysis attack, Scalar Multiplication

* 종신회원 : 동의대학교 컴퓨터공학과(교신저자, bychoi@deu.ac.kr)
접수일자 : 2011. 09. 22
심사완료일자 : 2011. 10. 27

Ⅰ. 서 론

수학적인 암호 분석 공격에 대처하도록 설계된 암호 알고리듬도 회로로 구현되었을 경우 문제가 야기될 수 있다. 공격자는 암호 알고리듬의 수학적인 측면의 안전성을 우회하기 위해 구현상의 약점을 이용할 수 있다. 이러한 문제는 알고리듬 구현시 전력 소비, 전자파, 시차정보, 오류 출력과 같은 부채널(side-channel) 정보가 누출되는 현상에 기인한다. 이러한 부채널 정보를 이용한 공격 방식 중에서 전력 분석 공격이 가장 현실성이 크고 위험성이 큰 것으로 알려지고 있다. 전력 분석 공격 중에는 한 번의 알고리듬 수행에 따른 전력 소모량을 관찰하여 키 값을 유도하는 공격을 단순 전력 분석 공격과 여러 번의 알고리듬 수행을 통해 수집된 전력 소모량 측정치들 사이에 상관관계를 통계적으로 분석하여 비밀 키 값을 유도하는 차분 전력 분석 공격이 있다[1,2]. 따라서 전력 소비 분석 공격은 암호 알고리듬을 구현하는 하드웨어 개발자에게는 이를 대처하기 위한 회로 설계에 큰 짐을 부여하고 있다. 이러한 전력 분석 공격에 대처하려면 키 값에 무관하게 연산 시간과 전력소비가 유사하도록 회로를 설계될 필요가 있다.

1985년 Miller와 Koblitz의 연구 결과에 기반을 둔 타원 곡선 공개키 암호 알고리듬은 짧은 키 길이 때문에 실용적인 공개키 응용 분야에 RSA 공개키 알고리듬을 대체하거나, RSA 알고리듬의 대안으로 사용범위가 크게 확대되고 있다[3]. 그러나 장점이었던 타원 곡선 암호 시스템의 짧은 키 길이는 전력 분석 공격에 대해 RSA 보다 상대적으로 취약할 수 있음을 보여준다. 그리고 암호 시스템이 속도에만 초점을 두고 설계되면 전력 분석 공격과 시차공격과 같은 부채널 공격에 매우 취약할 수 있다. 따라서 타원 곡선 암호 시스템의 응용을 확대하려면 이러한 공격에 대한 체계적인 분석과 대처 방안 연구가 필요하다.

본 논문의 구성은 다음과 같다. 2장에서는 타원곡선 암호알고리즘과 몽고메리 스칼라 곱셈에 대해 살펴보고, 3 장에서는 단순 전력 분석 공격에 대처할 수 있도록 수정된 몽고메리 방식을 제안하고, GF(2^{191}) 타원 곡선 암호 프로세서를 설계하고, 4장에서 검증 및 성능을 분석하였다. 마지막으로 결론을 기술하였다.

Ⅱ. 타원곡선 공개키 암호 알고리즘과 몽고메리 스칼라 곱셈

2.1. 타원곡선 암호알고리즘

유한체 GF(2^m)상의 타원곡선 $E(x,y)$은 식(1)와 같이 표현된다. 암호 알고리즘에 사용되는 타원곡선은 GP(p), GP(2^m)상에서 구현되는데, 본 논문에서는 안전성과 구현의 용이성을 고려하여 다항식 기저를 사용하는 유한체 GF(2^m)을 사용하는 타원곡선을 사용하였다.

$$E(x,y): y^2+xy=x^3+ax+b, \quad (1)$$
여기서, $b\neq 0,\ a,b\in GF(2^m)$,
$(x,y)\in GF(2^m)\times GF(2^m)$

타원곡선상의 두 점 $P=(x_1,y_1)$와 $Q=(x_2,y_2)$에 대하여, 점 P의 역원 $-P=(x_1,x_1+y_1)$이며, $Q=-P$인 경우 $P+Q=O$이다. 타원곡선 암호알고리즘의 핵심 연산은 스칼라 곱셈으로 한 점 P에 임의의 정수 k를 곱하는 $Q=kP$ 연산인데, 이러한 연산은 두 점의 덧셈($P+Q$)와 동일한 점을 더하는 점 2배 연산($2P$)의 조합으로 수행된다. 점 덧셈과 점 2배 연산은 식(2), (3)으로 정의된다.

○ P ≠ Q인 경우 R(x3, y3)=P+Q연산

$$x_3=\lambda^2+\lambda+x_1+x_2+a, \quad (2)$$
$$y_3=\lambda(x_1+x_3)+x_3+y_1,$$
여기서, $\lambda=(y_2+y_1)/(x_2+x_1)$

○ P=Q인 경우 R=2P 연산

$$x_3=x_1^2+b/x_1^2=\lambda^2+\lambda+a \quad (3)$$
$$y_3=x_1^2+(x_1+y_1/x_1)x_3+x_3=x_1^2+(\lambda+1)x_3$$
여기서, $\lambda=x_1+y_1/x_1$

2.2. 이진 스칼라 곱셈 알고리즘

일반적인 그림 1과 같은 스칼라 곱셈 알고리즘은 키 $k=(k_{n-1},\cdots,k_1,k_0)$ 값을 상위 비트에서 하위 비트 혹은 하위 비트에서 상위 비트로 검사해서, 점 덧셈과 점 2배 연산으로 구현된다. 이진 스칼라 곱셈이나 부호화 리코딩을 하는 방식은 키 값에 따라 내부 루프의 연산시간이 다르므로, 고속 연산이 가능하다는 장점이 있지만, 키 값에 따라 연산 시간이 다르다는 것은 보안 측면에 문제

가 있다.

```
Input :  an integer k >0 and point P
Output : Q = kP
1. if k_{n-1} == 1  then
       Q[0] = P
   else Q[0] = O
2. for i=n-2  downto 0
   2-1.  Q[0] = 2Q[0]
   2-2.  if k_i ==1  then
   2-3.  Q[0] = Q[0] + P
3. return (Q[0])
```

그림 1. 이진 스칼라 곱셈 알고리즘
Fig. 1 Binary scalar multiplication algorithm

2.3. Montgomery-Ladder 기반 스칼라 곱셈 알고리즘

몽고메리가 제안한 그림 2의 스칼라 곱셈 알고리듬의 특성은[4] 첫째, 초기에 P_1과 P_2사이에 Montgomery Ladder조건, $P_2-P_1=P$를 유지하도록 [단계 2]와 같이 설정한다. 둘째, 매 루프 입력과 출력, P_1과 P_2사이에 $P_2-P_1=P$의 관계가 유지된다. 셋째, 매 루프의 수행 동작이 키(k_i) 값과 무관하게 항상 하나의 점 덧셈과 점 2배 연산으로 구성된다.

```
Input :  an integer k >0 and point P
Output : Q = kP
1. set  k ←(k_{n-1},⋯ k_1,k_0)_2   // k_{n-1} = 1
2. set   P_1←P,  P_2 ← 2P
3. for i=n−2 downto 0  do
       if k_i ==1  then
            set  P_1←P_1+P_2,  P_2 ← 2P_2
       else
            set  P_2←P_1+P_2,  P_1 ← 2P_1
4. return (Q= P_1 )
```

그림 2. 몽고메리 스칼라 곱셈 알고리듬
Fig. 2 Montgomery scalar multiplication algorithm

몽고메리 방식은 [단계 3]의 매 루프마다 수행되는 동작이 동일하므로 시차 공격과 단순 전력 분석 공격에 대처하는 특징을 갖고 있다. 그러나 기존 스칼라 곱셈 알고리듬의 경우 점 덧셈($P+Q$)이 $k_i=1$인 경우에만 필요한데 비해, 몽고메리 방식의 경우 매 루프마다 필요하므로 연산 속도 측면에서 비효율적인 적으로 판단되어 그동안 널리 사용되지 않았다. 기존 몽고메리 스칼라 곱셈 방식의 속도 문제를 개선한 연구 결과가 그림 4와 같이 Lopez 등에 의해 제안되었다[5]. Lopez는 매 루프 동작시 입력 P_1, P_2사이에 몽고메리 Ladder 조건($P=P_2-P_1$)이 유지되면, 식(2), (3)의 P_1+P_2 연산과 $2P_1$연산의 결과 점의 x좌표, 즉, x_3을 그림 3의 [단계 4]와 같이 $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, $P(x,y)$ 의 x 좌표만으로 표현할 수 있다는 것을 입증하였다. 단, x-좌표 로 반복 루프 동작을 할 경우 속도 문제를 해결할 수 있지만, 최종적인 결과 $Q=kP$의 y 좌표 값을 생성하는 동작(단계[6, 7])이 마지막에 필요하다.

```
Input : An integer k ≥ 0 and
        a point P=(x,y)∈E
Output : Q=kP
1. if (k=0) or (x=0) then
     Q←O=(0, 0)  and stop
2. set k←(k_{n-1},⋯,k_1,k_0)_2
3. set x_1←x, x_2←x^2+b/x^2 // k_{n-1}=1
4. for i = (n−2)  to 0
     set  t←x_1/(x_1+x_2)
     if (k_i=1) then
       set  x_1←x+t^2+t,   x_2←x_2^2+b/x_2^2
     else
       set  x_1←x_1^2+b/x_1^2 , x_2←x+t^2+t,
5. set   r_1←x_1+x,  r_2←x_2+x
6. set  y_1←r_1(r_1r_2+x^2+y)/x+y
7. return (Q=(x_1,y_1))
```

그림 3. 수정된 몽고메리 스칼라 곱셈 알고리듬 I
Fig. 3 Modified Montgomery scalar multiplication I

Ⅲ. 전력 분석 공격에 대처하는 타원곡선 암호프로세서 설계

3.1. 설계 사양

본 논문에서 설계한 타원 곡선 공개키 암호 프로세서는 단순 전력 분석 공격에 대처하는 수정된 몽고메

리 스칼라 알고리듬을 지원하며, 내부 동작 모드에 따라 암호 및 복호 기능, 유한체 연산 기능 등 7가지 알고리즘을 수행하도록 하였다. 그리고 설계한 타원 곡선 암호 프로세서는 다양한 m값이 아닌 ANSI X9.62 표준[6]을 지원하도록 m=191을 갖는다. 또한 목표 암・복호율은 200 kbps이상으로 하고, 호스트 프로세서에 대한 보조프로세서로 동작하는 것을 목표로 하였다. 표 1은 타원 곡선 암호 프로세서에서 사용한 타원곡선을 나타낸다.

표 1. 타원 곡선
Table. 1 Elliptic curve

유한체	GF(2^{191})
원시 기약 다항식	$f(x)=x^{191}+x^9+1$
타원 곡선	$y^2+xy=x^3+ax^2+b$
a	2866537b676752636a68f56554e12640276b649ef7526267
b	2e45ef571f00786f67b0081b9495a3d95462f5de0aa185ec

3.2. MV 타원 곡선 암호 알고리듬

대표적인 타원 곡선 암・복호 방식은 Elgamal 과 캐나다 워털루 대학 교수인 Menezes와 Vanstone이 제안한 방식(MV 방식)이 있다[3]. Elgamal 방식은 메시지가 타원 곡선상의 점이어야 하는 제약조건이 필요하므로, 전송할 ASCII 코드 형태의 메시지 값을 타원 곡선상의 점으로 대응시키는 사전 처리 과정이 필요하다. 본 논문에서는 메시지가 타원 곡선상의 점이어야 하는 제약조건을 제거한 MV 알고리즘을 사용하였다. 본 논문에서 MV 타원 곡선 암호 및 복호 동작을 구현하는 단계는 다음과 같다. G, aG, E, $GF(2^m)$는 통신 당사 간에게 공개된 정보이고, a, b는 송수신 당사자의 비밀키 정보이다.

○ 암호 동작

[단계 1] 스칼라 곱셈 동작을 통해 $C_1=kG$를 구한다.

$$C_1=kG=(C_1[x],\ C_1[y]) \tag{4}$$

[단계 2] 난수 k와 공개키 정보 aG를 사용하여, $k(aG)$를 계산한다.

$$(x,y)=k(aG)\quad //aG=\text{공개 정보} \tag{5}$$

단 x=0이거나 y=0이면 k값이 적절하지 않은 값이므로 조기에 종료하고 호스트 프로세서에게 오류 신호를 전달한다. [단계 3] 메시지 $M=(M_1,M_2)$에 대해 유한체 곱셈을 사용하여 C_2을 구한다.

$$C_2=(M_1 \bullet x,\ M_2 \bullet y) \tag{6}$$

$C=(C_1,C_2)$를 암호 결과로 수신부에 전송한다.

○ 복호 동작

[단계 1] 수신된 C_1 정보에 스칼라 곱셈 동작을 사용하여 (x,y)를 구한다.

$$(x,y)=a(C_1) \tag{7}$$

[단계 2] 유한체 나눗셈 동작을 사용하여 메시지 M을 유도한다.

$$M_1=C_2[x]/x,\qquad M_2=C_2[y]/y \tag{8}$$

3.3. 구현에 맞게 수정한 몽고메리 스칼라 곱셈 알고리즘

본 논문에서는 Lopez가 제안한 그림 3의 수정된 몽고메리 스칼라 곱셈 알고리듬을 구현에 맞게 특이 조건을 처리할 수 있도록 수정하였다. 암호 동작이 데이터에 무관하게 안전하게 동작하려면 특이 조건에 대한 처리가 필요하다. 기존 알고리듬은 몇 가지 구현상 문제점을 갖고 있다.

첫째, [단계 1]에서 P의 x-좌표가 x=0조건을 배제하여 단계 6에서 x=0인 나눗셈이 발생하는 것을 방지하고, x=0인 경우 항상 무한 원점(O) 결과를 반환하였다. 그러나 식 (1)의 GF(2^m)상의 타원 곡선 은 x=0인 조건에서 y≠0인 점이 존재할 수 있으므로, 항상 무한 원점(point at infinity)으로 결과를 반환한 것은 문제가 있다. 단, 식 (1)의 경우 (0, 0)는 타당한 근이 아니므로, 하드웨어 구현시 무한원점으로 $O=(0,0)$을 정의해서 사용할 수 있다.

둘째, [단계 3]의 경우 항상 키의 최상위 비트(k_{n-1})을 1로 가정하여, $P_1=P$, $P_2=2P$에 해당하는 값을 설정하고 있다.

셋째, [단계 4]에서 $x_1 + x_2 = 0$ 조건과 k_i 값에 따라 $x_1 = 0, x_2 = 0$조건이 0으로 나누는 문제가 발생할 수 있다. 즉, 중간 연산 과정에 무한 원점(O) 혹은 $x_1 = 0, x_2 = 0$조건이 발생할 수 있다.

그러나 기존 논문에서는 이에 대한 자세한 해결 방안이 제시되고 있지 않다. 이러한 문제는 구현시 사용하는 무한 원점($O=(x,y)=(0,0)$) 과 무한원점이 아닌 점, ($x=0, y\neq 0$)에 대한 구별이 단계 4의 연산 시 반영되지 않았기 때문에 발생한 것이다. 위의 문제에 대한 해결하기 위해 다음 방식을 적용하였다.

첫째, 스칼라 곱셈 곱셈의 점 2배 연산과 점 덧셈으로 구성되는데, P점의 x 좌표 값이 0이면 타원곡선 연산 정의[3]에 따라 $2P=O$이 된다. 그러면 3P의 경우 $3P=2P+P=P$가 된다. 즉 $x=0$인 경우 kP연산에서 k가 홀수 이면 결과는 P가 되고, k가 짝수이면 결과는 무한원점 $O=(0,0)$가 된다. $x=0$인 경우는 위의 연산 특성에 따라 조기 처리되어 종료되며, 이러한 사전 처리 과정에 의해 [단계 4]로 진입한 P점의 값은 항상 $x\neq 0$을 보장받을 수 있다.

둘째, $k_{n-1}=1$의 제약조건을 없애기 위해, $P_1=O=(0,1)$과 $P_2=(x,0)=P$로 설정하여. 몽고메리 Ladder조건 $P_2-P_1=P$을 만족시킨다. 좌표의 두 번째 값은 플래그로 $x=0$인 무한원점(O)과 $x=0$이지만 무한원점이 아닌 경우를 구별한다.

셋째, 기존 방식의 단계 4의 문제점을 해결하기 위해 기존 x-좌표만을 사용하는 연산 특성을 수정하여, 각각의 좌표 점을 x좌표와 플래그의 조합으로, 즉 (x, flag)로 표현하였다. 즉 flag=1, 즉 (0, 1)이면 무한 원점(O)을 나타내고, flag=0, 즉 (x, 0)이면 x값에 관계없이 무한원점이 아닌 점을 나타낸다. 좌표점이 무한원점인 경우 단계 4의 P+Q, 2P 연산에서 $x_1+x_2=0$, $x_1=0$, $x_2=0$의 특수한 경우를 처리할 수 있다. 몽고메리 방식의 경우 반복 루프 동작에서 몽고메리 ladder 조건에 따라 $P=P_2-P_1$이므로($P=O$ 조건은 사전 제외됨), 단계4에서 $(x_1=x_2)\cdot(y_1=y_2)=1$인 경우와 $x_1=x_2=0$이며 $(y_1\neq y_2)$인 조건은 존재하지 않는다.

넷째, $(x_1=x_2)\cdot(y_1\neq y_2)$인 경우는 연산 규칙에 따라 $R=P_1+P_2=O$이다.

```
Input :k ≥ 0,   P=(x,y)∈E
Output :  Q=kP
1. if (x=0) {
  if (k=even)   Q←O=(0, 0)  and stop
     else   Q←P=(x, y)  and stop }
2. if (P=O)  Q←O=(0,0)   and stop
3. k←(k_{n-1},…,k_1,k_0)_2
4. x_1f←(0, x_1f[flag])=(0,1),
   x_2f←(x, x_2f[flag])=(x, 0)
   //  P_1=O, P_2=P, P_2−P_1=P
5. for i = (n−1) to 0
  if (x_1f[x] ≠ x_2f[x])  {    // [5-1]
    if (k_i=1) {    // P_1←P_1+P_2
      if (x_1f[flag]=1)
          x_1f←x_2f=(x_2, x_2f[flag])
      else if (x_2[flag]=1)
              x_1f←x_1f=(x_1,x_1f[flag])
      else  {
          t←x_1/(x_1+x_2), x_1f←x+t^2+t,
        if (x_2 ≠ 0)
          x_2f←(x_2^2+b/x_2^2, 0)  // P_2←2P_2
        else   x_2←(0, 1) = O  } }
    else   // k_i=0
    { // P_2←P_1+P_2
    if (x_1f[flag]=1)
        x_2f←x_2f=(x_2, x_2f[flag])
      else if (x_2[flag]=1)
              x_2f←x_1f=(x_1,x_1f[flag])
      else  {
          t←x_1/(x_1+x_2), x_2f←x+t^2+t,
          if (x_1 ≠ 0),
            x_1f←(x_1^2+b/x_1^2, 0)  // P_1←2P_1
          else   x_1←(0, 1) = O  } }}
  else {  //  [5-2]: (x_1=x_2) & (y_1≠y_2)
    if (k_i=1)  {
// P_1←P_1+P_2=O,  P_2←2P_2=P
          x_1f←(0, 1),  x_2f←(x, 0)  }
   else { x_2f←(0, 1), x_1f←(x, 0) }
6. if ((x_1=0) • (x_1f[flag]=1)),
       y_1←0 //P_1=O
  else if ((x_2=0) • (x_2f[flag]=1)) //  P_2=O
        y_1←(x+y)
 // P_2−P_1=P,  P_1=−P=(x,x+y)
  else {  r_1←x_1+x,   r_2←x_2+x,
        y_1←r_1(r_1r_2+x^2+y)/x+y   }
7. return (Q=(x_1,y_1))
```

그림 4. 제안한 스칼라곱셈 알고리즘
Fig. 4 Proposed Scalar Multiplication algorithm

따라서 $P_1 = -P_2$, $P_2 = P_1 + P = -P_2 + P$에서 결과 값은 $P_1 + P_2 = O$, $2P_2 = P$, $2P_1 = -P$가 된다. 위와 같은 방식으로 특별한 입력 조건에 대해서는, 구현에 맞게 특이 조건에 구별된 처리를 하여 결과 값을 할당하였다.

다섯째, [그림 3]의 단계 6과 단계 7에서 y값을 구하는 과정에도 별도의 처리가 필요하다. 결과 값이 P_1=(0, 1)인 무한 원점(O)인 경우 다른 좌표, P_2 값과 무관하게 $y_1 = 0$로 무한 원점 결과 처리가 필요하다. 반대로 P_2가 무한원점인 경우 몽고메리 조건에 의해 $P_2 - P_1 = P$, $O - P_1 = P$이다. 따라서 $P_1(x_1, y_1) = -P = (x, x+y)$에 의해서 $y_1 = x + y$가 된다. 이러한 입력 조합이 아닌 경우 기존 단계 6과 단계 7을 통해 최종 y-좌표를 구한다. 이러한 분석 결과를 바탕으로 본 논문에서 특이조건(special condition)을 처리하도록 본 연구에서 제안한 몽고메리 스칼라 곱셈 알고리듬은 그림 4와 같다. 각 단계에서 특수 입력 조건을 검사함과 동시에 조기 종료 환경에서도 전력 분석 공격에 대처하도록 동일 연산을 잉여로 하되 결과는 저장하지 않고 미리 정해진 값을 갖도록 하였다.

3.4. 타원곡선 암호프로세서 구조

타원 곡선 암호 프로세서는 구현하는 7가지 동작을 타원 곡선 암호 프로세서 내부에 존재하는 8 비트 모드 레지스터 값에 의해 결정된다. 이러한 동작 모드 레지스터는 암호 시스템 사용 초기에 외부에서 데이터 명령 저장 과정으로 통해 초기화가 가능하므로 설계한 프로세서를 적절히 프로그래밍 함에 의해서 다양한 응용이 가능하다. 그림 5와 그림 6은 타원 곡선 암호 프로세서의 외부 인터페이스와 내부 구성도를 나타낸다.

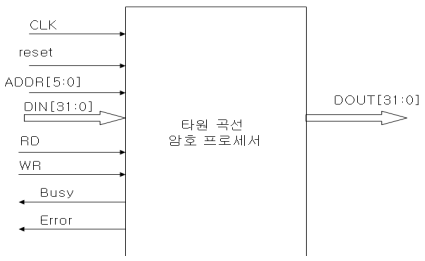

그림 5. 프로세서 외부 인터페이스
Fig. 5 Input and Output interface of processor

타원 곡선 암호 프로세서는 그림 6과 같이 입출력 데이터를 저장하는 레지스터와 스칼라 곱셈 동작을 수행하는 프로세서 코어, 제어 회로로 구성된다. 각각의 레지스터는 32-비트 단위로 외부 주소에 대응되므로 외부에서 주소 값을 통해 효율적으로 인터페이스가 가능하다.

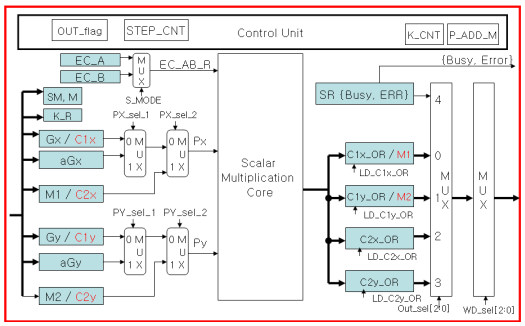

그림 6. 프로세서의 내부 구성도
Fig. 6 Internal structure of processor

그림 7의 스칼라 곱셈 코어는 크게 레지스터 파일(GF_RF), 유한체 덧셈과 플래그 검사 기능을 내장한 산술 회로(GF_ALU), 유한체 곱셈을 담당하는 GF_MUL, 그리고 유한체 나눗셈과 역원 계산 기능을 수행하는 GF_DIV 블록으로 구성된다. 단 GF_MUL과 GF_DIV의 출력은 항상 GF_ALU를 거치도록 함에 의해 출력 배선 경로를 줄임과 동시에 곱셈과 나눗셈 결과가 레지스터 파일에 저장되기 전에 GF_ALU에서 추가의 유한체 덧셈 동작을 할 수 있도록 하였다.

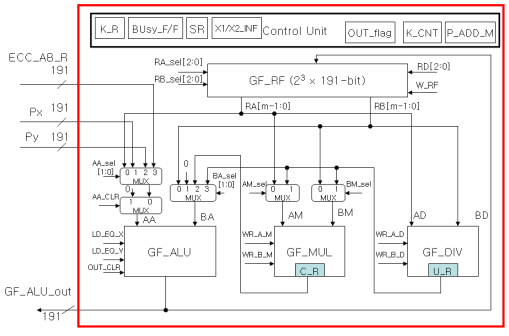

그림 7. 스칼라 곱셈 코어
Fig. 7 Scalar multiplication core

GF_ALU 블록은 유한체 덧셈 동작과 함께 입력 값의 동일여부, 특수 입력을 검사하는 기능과 무한 원점의 결과 값을 생성하는 역할을 한다. GF_RF는 타원 곡선 암호 프로세서 동작 과정에서 중간 결과를 저장하는 용도로 사용된다. GF(2^m)상의 승산기는 병렬과 직렬 연산 방식이 있다. 타원 곡선 암호 알고리듬의 스칼라 곱셈 동작의 P+Q와 2P 연산에서 유한체 곱셈 수는 2~3개 정도이고 연속적으로 사용되지 않기 때문에 면적을 고려한 곱셈기 방식이 필요하다. 본 논문에서는 직렬과 병렬 방식의 장단점을 고려하여, 매 클록 당 승수 비트의 8-비트 이상을 처리하는 그림 8의 radix-256 MSR(modified shift register)[7] 유한체 승산기를 설계하였다. radix-256 MSR 곱셈기의 경우 매 사이클마다 8개의 부분 곱을 생성하여 누적시키는 방식이다. 이 경우 곱셈 연산에 필요한 클록 사이클 수가 $\lceil m/8 \rceil$ 이 된다.

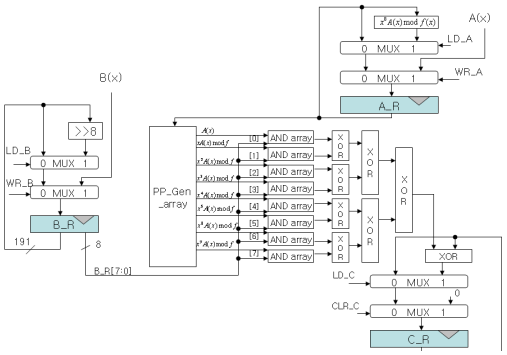

그림 8. radix-256 GF(2^m) MSR 유한체 곱셈기
Fig. 8 Radix-256 GF(2^m) MSR finite field multiplier

GF(2^m) 상의 유한체 나눗셈은 $A(x)/B(x) \bmod f(x)$로 표현되며, 일반적으로 확장된 유클리드 알고리듬으로 구현된다. 기존 확장 유클리드 알고리듬은 데이터 값에 따라 가변적인 연산 사이클을 갖는다. 가변적인 연산 시간 특성은 본 논문에서 설계사양으로 정한 전력 분석 공격에 대처하는 구조에는 적합지 않다. 따라서 2m과 (2m-2)개의 고정된 연산 사이클을 갖는 BCH 방식[8]과 (2m-2) 사이클의 연산 사이클을 갖는 Guo_Wang방식[9]을 유한체 나눗셈기로 고려하였다. 본 논문에서는 전력 분석 공격에 대비하고 radix-256 곱셈기 속도를 고려하여 기존 radix-1 방식의 Guo_Wang을 확장해서(반복 연산 동작을 그림 9과 같이 중복 배치), (m-1) 사이클에 유한체 나눗셈을 하는 radix-2 방식의 나눗셈기를 구현하여 내장하였다.

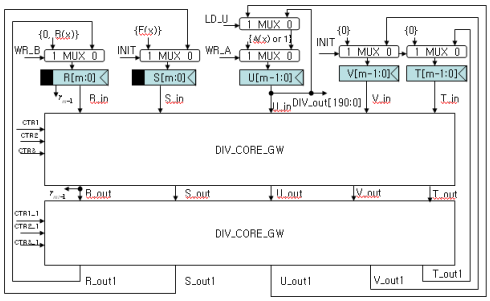

그림 9. radix-2 유한체 나눗셈기
Fig. 9 Radix-2 finite-field divider

3.5. 암·복호 및 스칼라곱셈 알고리듬의 스케줄링

유한체 연산 모듈 GF_MUL, GF_DIV, GF_RF, GF_ALU 모듈을 적절하게 시간적으로 스케줄링하여 몽고메리 타원곡선 암호 및 복호 알고리듬을 구현한다. radix-256 GF_MUL과 radix-2 GF_DIV 회로는 각각 고정된 m/8과 (m-1) 클록 사이클이 소요되므로, 데이터 종속성을 최소화하도록 연산 순서를 조정하여 병렬 연산이 가능하다. 단, GF_DIV와 GF_MUL을 중첩시킬 경우, 연산 시간이 많이 걸리는 GF_DIV를 먼저 시작 시킨 후에 GF_MUL을 수행시키고, 종료 검사는 연산 사이클 수가 작은 GF_MUL부터 검사하는 방식을 통해 구현하였다. 그림 10는 몽고메리 방식의 반복 루프 동작중 $k_i = 0$인 경우 동작(P+Q, 2P)의 x-좌표를 계산하는 타이밍 도를 나타낸다. 그림에서 횡축은 시간을 나타내므로, 수직 방향이 시간대별로 수행되는 연산과 결과가 담기는 레지스터를 나타낸다. 수직방향으로 겹치는 동작은 2개 이상의 동작이 시간적으로 중첩되어 수행됨을 나타낸다. 횡축은 연산에 필요한 사이클 수를 나타낸다. 단, GF_MUL과 GF_DIV 연산을 위해 입력 데이터를 해당 모듈 내부 레지스터로 가져오는 동작이 필요하므로, 연산 시간 계산에 추가로 1 사이클의 시간이 필요하다. 기존 스칼라 곱셈의 흐름도와 달리 연산 순서 조정으로 2개의 나눗셈 동작이 모두 유한체 곱셈 동작과 시간적으로 중첩시켜 수행될 수 있는 있어서, $O(m)$ 사이클에 고속 동작이 가능하다. $k_i = 0$인 경우 R_0에는 최종 결과 값의 x_1이, R_1에는 x_2가 저장된다. $k_i = 1$인 경우도

유사하게 처리된다. 일반적인 스칼라 곱셈과 달리 P+Q, 2P가 분리되어 필요한 것이 아니라, 몽고메리방식의 경우 통합되어 수행되고, 연산을 공유할 수 있어 고속 처리가 가능하다. 타원곡선 암호프로세서는 연산 모듈에 대한 스케줄링과 7가지 구분된 동작을 제어하기 위해 계층적인 제어 구조를 갖는다.

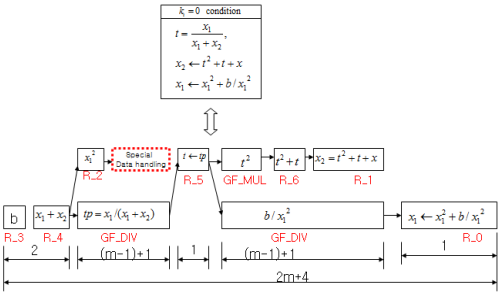

그림 10. 몽고메리 스칼라 곱셈의 루프 동작의 타이밍도
Fig. 10 Timing diagram of loop operation for Montgomery scalar multiplication

Ⅳ. 검증 및 성능 분석

본 논문에서는 타원 곡선 암호 프로세서를 검증하기 위해, 먼저 Visual C++ 언어로 유한체 관련 산술 연산과 P+Q, 2P, 암호 및 복호 동작을 멤버 함수로 하는 클래스를 정의하였다. 이를 사용하여 타원 곡선 암호 프로세서를 검증하기 위한 테스트 벡터를 추출하였다. 이를 바탕으로 유한체 연산 모듈과 타원곡선 암호프로세서를 Verilog HDL 언어로 모델링한 후, Cadence NC-Verilog 시뮬레이터로 테스트 벡터가 올바로 동작하는지 확인하였다. 스칼라 곱셈(kP) 결과와 X9.62 표준안에 정의된 테스트 데이터를 통해 설계된 회로의 올바른 동작을 확인하였다. 설계된 회로를 0.35μm CMOS 표준 셀 라이브러리[10]로 합성한 결과, 최장 지연시간이 약 7.8 ns로 최대 동작 주파수가 125 Mhz이다. 그림 11은 타원곡선 암호프로세서에 대한 FPGA 검증 환경을 나타낸다.

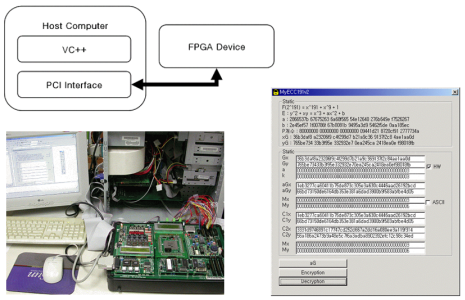

그림 11. 프로세서의 FPGA 검증 환경
Fig. 11 FPGA verification for processor

설계한 프로세서의 성능을 정량화시키기 위해, 식 (9)~식(12)의 암호 및 복호 관련 성능, 스칼라 곱셈 성능과 스칼라 곱셈의 연산 시간을 정의하였다.

$$\text{암호율}(bps) \approx \frac{2m}{2\times SMUL_t + 2\times GFMUL_t} \quad (9)$$

여기서, $SMUL_t$ = 스칼라 곱셈 시간
= 스칼라곱셈 사이클 수×클록 주기
$GFMUL_t$ = 유한체곱셈시간
= 유한체곱셈 사이클 수×클록 주기

$$\text{복호율}(bps) \approx \frac{2m}{SMUL_t + 2\times GFDIV_t} \quad (10)$$

여기서, $SMUL_t$ = 스칼라 곱셈시간
= 스칼라 곱셈 사이클 수×클록 주기
$GFDIV_t$ = 유한체 나눗셈시간
= 유한체 나눗셈 사이클 수×클록 주기

$$\text{스칼라 곱셈 성능}(bps) = \frac{2m}{SMUL_t} \quad (11)$$

여기서, $SMUL_t$ = 스칼라 곱셈시간
= 스칼라 곱셈 사이클 수×클록 주기
$= ((2m+4)+(m+3m/8)+8)\times T$
$= (3m+(3m/8)+8)\times T$

$$SMUL_t \approx ((2m+4)\times m + (m+\frac{3m}{8}+8))\times T \quad (12)$$

여기서, T= 클록 주기

본 논문의 타원곡선 암호프로세서의 경우 m=191이므로 몽고메리 스칼라 곱셈의 경우 약 (2m+4)*m+(m+3m/8+8)≒74,000 사이클이 소요되며, 스칼라 곱셈, 암호 및 복호율은 각각 640kbps, 320kbps, 640kbps의 성

능을 갖는다. 표 2는 설계한 타원 곡선 암호 프로세서의 전기적 특성을 나타낸다. 타원 곡선 암호 프로세서는 7가지 기능을 갖고 있으며, 각각 약 320kbps와 640kbps의 암호 및 복호율 특성을 갖고 있다. 표 3은 기존에 발표된 프로세서와 성능을 비교한 결과이다.

표 2. 전기적 특성
Table. 2 Electrical characteristics

항목	특성
지원 연산	스칼라 곱셈(kP), 몽고메리 스칼라곱셈 암호/ 복호, P+Q/2P GF_MUL/GF_DIV
곱셈기/나눗셈기	radix-256 GF_MUL / radix-2 GF_DIV
동작 주파수	125 Mhz
설계 공정	0.35um CMOS
게이트 수 (2-input NAND)	68,000 (스칼라 곱셈부: 58,000)
암호율/복호율	320kbps /640 kbps

표 3. 스칼라 곱셈의 성능비교
Table. 3 Performance comparison

구현	필드	디바이스 (공정)	게이트 수	성능 / 주파수
참고문헌 [11]	GF(2^{163})	Xilinx FPGA	60k	- / 30Mhz
참고문헌 [12]	GF(2^{163})	0.18um CMOS	60k	248kbps / 250Mhz
본 연구	GF(2^{191})	0.35um CMOS	58k	640 kbps /125Mhz

V. 결 론

본 논문에서는 단순 전력 분석과 시차 분석 공격에 대처하기 위해, 특이조건을 처리할 수 있도록 수정된 몽고메리 스칼라 곱셈 알고리즘을 제시하고, 이를 바탕으로 7가지 다양한 기능을 갖는 타원 곡선 암호 프로세서를 설계하고 성능을 분석하였다. 타원 곡선 암호 프로세서를 고속화하기 위해 radix-256 유한체 곱셈기와 (m-1) 사이클에 완료되는 radix-2 유한체 나눗셈기를 설계하였으며, 암호 및 복호, 스칼라 곱셈 연산을 수행하는 동안 하드웨어 병렬성을 극대화하여 암호 및 복호율이 0.35um CMOS 공정 하에서 320kbps과 640 kbps를 갖는다. 최신 공정 기술을 사용할 경우 수 Mbps, 혹은 수십 Mbps의 성능도 가능하다고 판단된다. 그리고 설계한 타원 곡선 암호 프로세서는 다양한 알고리즘 지원으로 디지털 서명 등과 함께 GF(2191)상의 유한체 연산이 필요한 암호 및 통신 분야에 적용이 가능하다.

감사의 글

반도체 설계 교육센터(IDEC)의 CAD 소프트웨어 지원에 감사드립니다.

참고문헌

[1] Thomas S. Messerges, “Power Analysis Attacks and Countermeasures for Cryptographic Algorithms”, Ph.D Thesis, University of Illinois at Chicago, 2000.

[2] 박 영호, Side Channel Attack을 고려한 알고리즘 연구, ETRI 최종 연구보고서, 2003. 11.

[3] A.J. Menezes, *Elliptic Curve Public Key Cryptosystems,* Kluwer Academic Publishers, 1993.

[4] P. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization", Mathematics of Computation, vol 48, pp.243-264, 1987.

[5] J. Lopez and R. Dahab, "Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation", CHES '99, LNCS 1717, pp.316-327, 1999.

[6] American Bankers Association, "Working Draft: American National Standard X9.62-1998 Public Key Cryptography for the Financial Services Industry," September 20, 1998.

[7] Edoardo D. Mastrovito, *VLSI Architectures for Computations in Galois Fields,* Linkoping University, Ph.D Thesis, 1991.

[8] H. Brunner, A. Curiger, and M. Hosfstetter, “On Computing Multiplicative Inverses in GF(2^m),” IEEE Transaction on Computers, Vol. 42, No. 8, pp.1010-1015, Aug., 1993.

[9] J. Guo, C. Wang, ″Systolic Array Implementation of Euclid’s Algorithm for Inversion and Division in GF(2^m),″ IEEE Transaction on Computers, Vol.47, No.10, pp.1161-1167, Oct., 1998.

[10] Samsung Electronics, STD90 /MDL90 0.35um 3.3V CMOS standard cell library for pure logic/ MDL Products, 2000.

[11] G. B. Agnew, R. C. Mullin and S. A. Vanstone, ″Implementation of Cryptosystems over GF(2155),″ IEEE Journal of Selected Area in Communication, Vol. 11, No.5, pp.804-813, 1993.

[12] 김의석, 정용진, “새로운 유한체 나눗셈기를 이용한 타원 곡선 암호(ECC) 스칼라 곱셈기의 설계,” 한국 통신 학회 논문지, Vol.29, No.5C, pp.726-736, 2004. 5.

저자소개

최병윤(Byeong-Yoon Choi)

1985년 2월 : 연세대학교 전자공학과 졸업
1992년 8월 : 연세대학교 전자공학과 공학 박사
2006년 1월 ~ 2006년12월: 오클랜드대학 방문 연구교수
1993년 3월 ~ 현재 : 동의대학교 교수
※관심분야 : RISC 마이크로프로세서 설계, 그래픽 및 암호 알고리즘의 SoC 설계