

이진체와 확장체에 기반한 타원곡선 암호시스템의 하드웨어 모듈 개발

전향남, 정필규, 김동규
부산대학교 컴퓨터공학과

Development of Hardware Modules for Elliptic Curve Cryptosystems based on Binary Field and Optimal Extension Field

Hyang-Nam Jun, Peel-Kyue Jung, Dong Kyue Kim
Dept. of Computer Engineering Pusan National University

요약

1985년 N. Koblitz와 V. Miller가 각각 독립적으로 제안한 타원곡선 암호시스템(ECC : Elliptic Curve Cryptosystems)은 유한체 위에서 정의된 타원곡선 군에서의 이산대수 어려움에 기초한다. 타원곡선 암호시스템은 다른 공개키 시스템에 비해 보다 짧은 길이의 키만으로도 동일한 수준의 안전도를 유지할 수 있다는 장점으로 인하여, 스마트카드나 모바일 시스템 등에서와 같이 메모리와 처리능력이 제한된 하드웨어에도 이식 가능한 장점이 있다. 본 논문에서는 타원곡선 암호시스템에 필요한 유한체 연산을 이진체(Binary Finite Field)인 GF(2^{193})과 OEF(Optimal Extension Field) 상에서 VHDL 언어를 사용하여 구현을 하였고 각 연산의 성능을 비교하였다.

1. 서론

최근 전자상거래가 활성화됨에 따라서 전자서명, 인증 및 정보의 암호화에 대한 중요성이 대두되고 있다. 초기에 암호 기술은 간단한 전치 및 대치방식을 사용했으나, 근래 컴퓨터의 계산능력이 비약적으로 증대되고 암호를 해독할 수 있는 새로운 알고리즘들의 등장으로 인해 보안을 제공하기 위한 암호 방식은 점차 복잡도가 높아지고 있다. 현재 가장 많이 사용되고 있는 공개키 암호 알고리즘으로는 RSA(Rivest-Shimir-Adleman)와 ECC(Elliptic Curve Cryptosystem)가 있다.

이중 ECC는 1985년 N. Koblitz와 V. Miller가 각각 독립적으로 제안한 알고리즘으로 유한체(Finite Field) 위에서 정의된 타원곡선 군에서의 이산 대수 문제에 기초한다.[1] 이 암호 시스템은 보다 짧은 길이의 키만으로 다른 공개키 시스템과 동일한 수준의 안전도를 유지할 수 있다. 예를 들어, 163비트 키를 갖는 ECC 알고리즘은 1024비트의 키를 갖는 RSA와 동등한 수준의 안전성을 가지는 것으로 알려져 있다.

이런 ECC의 장점으로 인해 메모리와 처리능력이 제한된 하드웨어에도 이식이 가능하다. 또한 동일한

유한체 연산을 사용하면서도 다양한 타원곡선을 선택할 수 있어서 고수준의 안전도를 제공하므로 차세대 공개키 암호시스템으로 주목을 받고 있다.[2~3]

현재 타원곡선 암호는 차세대 암호 기법으로 인정 받아 새 표준 암호 기술이 개발 중이며 외국에서는 타원곡선 암호와 관련하여 소프트웨어 및 하드웨어 구현이 활발하게 이루어지고 있다. 반면 국내에서는 소프트웨어를 이용한 타원곡선 암호시스템 구현의 사례가 있을 뿐 하드웨어 구현은 아직 미비한 상태이다. 특히 GF(2^n)상에서 타원곡선 암호시스템을 하드웨어로 구현한 사례만이 극소수 있을 뿐 (p^n)상에서의 타원곡선 암호시스템의 하드웨어 구현 사례는 거의 없다.

따라서 본 연구에서는 GF(p^n)중의 하나인 OEF (Optimal Extension Field)라는 유한체 상에서 타원곡선 암호시스템에 필요한 유한체 연산을 VHDL을 사용하여 하드웨어로 직접 구현한다. 또한 현재 활발히 구현되고 있는 GF(2^n)상에서 유한체 연산을 구현하여 OEF상의 유한체 연산과 그 성능을 비교하며, 유한체 연산을 기본으로 구현되는 타원곡선 상의 덧셈, 더블링 연산에 어떤 영향을 미치는지에 대해 구현을 통해

알아본다..

2. 타원곡선 암호 알고리즘

1985년 Neal Koblitz와 Victor Miller에 의해서 개발된 타원곡선 암호시스템은 타원곡선 이산로그문제(ECDLP: Elliptic Curve Discrete Logarithm)에 근거를 두고 있다. 타원곡선 이산로그문제는 타원곡선 상의 임의의 한 점 P에 정수 k를 곱한 것이 Q=kP일 때 k와 P를 알 때 Q를 구하기는 쉬우나 Q와 P를 알 때는 정수 k를 계산하기가 어렵다는 것을 나타낸다.[4] 타원곡선 암호에서 기본이 되고 가장 중요한 연산인 스칼라 곱셈 연산은 타원곡선 상의 동일한 두 점의 합(doubling)과 서로 다른 두 점의 합(adding)을 반복적으로 이용하여 구할 수 있다. 스칼라 곱셈 연산은 소수체(Prime Field)인 GF(p)를 확장한 GF(pⁿ)과 이진체 GF(2ⁿ)상에서 모두 구현이 가능하지만, 확장체와 이진체에서 각각 구현될 때 체(field) 상의 연산인 덧셈, 곱셈, 역원에 따라 타원곡선 상의 스칼라 곱셈 연산의 성능은 달라진다. 특히 소수체에서 p를 일반적으로 사용되는 CPU의 처리단위에 가까운 32비트 또는 64비트로 설정한 것을 OEF(Optimal Extension Field)라고 하는데 OEF 상에서 소프트웨어 구현은 효율적인 것으로 알려져 있다. [5]

먼저, 타원곡선(E)이 GF(2ⁿ)상에서 다음과 같이 정의되었을 때 타원곡선 위의 서로 다른 두 점 P(x1, y1)와 Q(x2, y2)의 덧셈을 구하는 공식은 다음과 같다.

$$E : y^2 = x^3 + ax^2 + b, \quad a, b \in GF(2^n) \quad (1)$$

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad (2)$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \quad (3)$$

$$\lambda = \frac{y_2 + y_1}{x_2 + x_1} \quad (4)$$

타원곡선 위의 한 점 P(x1, y1)의 두 배 연산을 구하는 공식은 아래와 같다.

$$x_3 = \lambda^2 + \lambda + a \quad (5)$$

$$y_3 = x_1^2 + (\lambda + 1)x_3 \quad (6)$$

$$\lambda = x_1 + \frac{y_1}{x_1} \quad (7)$$

타원곡선이 GF(pⁿ)상에서는 다음과 같이 정의되며 타원곡선 위의 서로 다른 두 점 P(x1, y1)와 Q(x2, y2)의 덧셈을 구하는 공식은 다음과 같다.

$$E : y^2 = x^3 + ax + b, \quad (8)$$

$$\text{단, } p > 3, \quad a, b \in GF(p^n), \quad 4a^3 + 27b^2 \neq 0$$

$$x_3 = \lambda^2 - x_1 - x_2 \quad (9)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \quad (10)$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad (11)$$

아래의 식은 타원곡선 위의 한 점 P(x1,y1)의 두 배 연산을 구하는 공식이다.

$$x_3 = \lambda^2 - 2x_1 \quad (12)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \quad (13)$$

$$\lambda = \frac{3x_1^2 + a}{2y_1} \quad (14)$$

위의 공식에서 사용된 덧셈, 곱셈, 나눗셈은 모두 일반적인 실수의 연산이 아니라 유한체 상에서의 연산들이다.

3. 타원곡선 암호시스템의 설계

타원곡선 암호시스템을 구현하기 위해서 선택할 수 있는 다양한 요소들이 있는데 그 중에서 대표적인 요소가 유한체, 기저, 타원곡선, 좌표계 등이다.

3.1. 타원곡선 암호시스템의 구현 파라미터 설정

실제 구현에 사용되는 유한체는 표수(characteristic)가 3보다 큰 소수(GF(p)) 또는 GF(pⁿ), 표수가 짝수인 GF(2ⁿ) 그리고 p가 32비트 또는 64비트와 비슷한 소수인 OEF의 형태가 있다. 실험에서는 유한체상의 연산 구현 측면에서 효과적인 GF(2ⁿ)와 보안 측면에서 강도가 높은 OEF를 선택하였다.

현재 사용되고 있는 1024비트 RSA 공개키 알고리즘보다 높은 안전도를 나타내는 193비트~217비트 크기의 유한체를 사용하였다. 각 유한체에서 사용 가능한 기저에는 Polynomial basis, Normal basis, Optimal Normal Basis 세 가지가 있는데 본 실험에서는 Polynomial basis로 구현을 하였다. 그리고 좌표계에는 Affine coordinates와 Projective coordinates가 있는데 각 좌표계마다 사용되는 연산의 개수가 다른데, 구현에는 Affine coordinates를 사용하였다.

타원곡선의 안전성은 기본적으로 타원곡선을 정의하고 있는 유한체와 타원곡선의 위수 등에 의존하는데 안전하고 효율적인 연산을 위해서 일반적으로 권장 타원곡선으로 많이 사용되고 있는 SEC그룹의 선전기준을 참고하여 타원곡선을 선택하였다. [6]

◆ 유한체 GF(2^n)에서 사용하는 파라미터

- (1) $p=2$
- (2) Polynomial basis
- (3) $n=193$
- (4) 타원방정식은 Certicom사의 SEC2에서 제안하는 $y^2 + xy = x^3 + ax + b$ 를 사용
- (5) 기약다항식(irreducible polynomial)은 $f(x) = x^{193} + x^{15} + 1$ 을 사용
- (6) Affine coordinates

◆ 유한체 OEF에서 사용하는 파라미터

- (1) $p=2^{31}-1$
- (2) Polynomial basis
- (3) $n=7$
- (4) 타원방정식은 Certicom사의 SEC2에서 제안하는 $y^2 = x^3 + ax + b$ 를 사용
- (5) 기약다항식(irreducible polynomial)은 $f(x) = x^7 - 1$ 을 사용
- (6) Affine coordinates

3.2. 이진체에서의 유한체 연산 구현 방법

타원곡선 위에서 같은 한 점을 두 배하거나 다른 두 점을 더하는 스칼라 곱셈 과정에서 필요한 연산은 유한체 상의 덧셈, 곱셈, 나눗셈 연산이다. 이 연산들은 구현되어지는 유한체에 따라 구현 방법이 다르다..

우선, 이진체 연산의 구현에 대해 살펴보면, 이진체의 덧셈은 polynomial basis로 구현이 되었으므로 캐리가 발생하지 않아 간단한 XOR연산으로 구현 가능하다. 이진체의 모듈라 곱셈기는 2개의 193비트 입력을 곱하는 곱셈 모듈과 곱한 결과를 모듈러 연산하는 reduction 모듈로 구성된다. 곱셈 모듈은 LSB(Least Significant Bit)부터 한 비트씩 순차적으로 곱셈을 수행하는 Bit-serial 방법을 사용하여 구현하였다. 그리고 reduction 모듈은 기약다항식을 이용하여 곱한 결과가 193비트를 넘을 때마다 reduction하는 방법을 사용하였다.[7]

마지막으로 가장 중요한 연산인 역원을 구하는 역승산기는 크게 페르마 이론(Fermat Theorem)과 확장 유클리드 알고리즘(Extended Euclid Algorithm)을 사용하여 구현하는데, 다항식 기저에 더 적합한 방법인 확장 유클리드 알고리즘을 사용하였다. 또한 수정된 확장 유클리드 알고리즘의 사용으로 더 빠른 시간 안에 적은 면적으로 구현하였다.[8] 아래의 그림1은 이진체의 역원 모듈로, 몫을 계산하는 부분(D)과 나머지를 계산하는 부분(M)으로 구성이 된다. 그리고 이진체 상의 타원곡선 위의 연산 중 다른 두 점의 합을

계산하는 adding의 구현 모듈은 그림2와 같다.

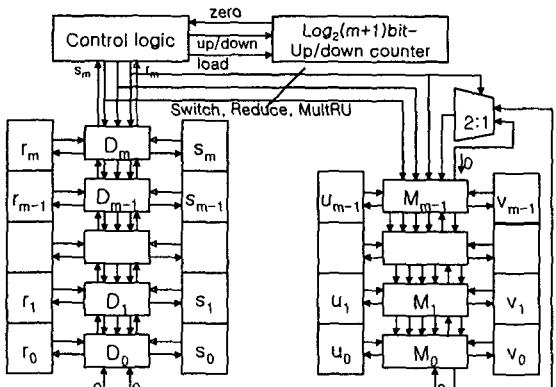


그림1. 이진체의 역원 모듈

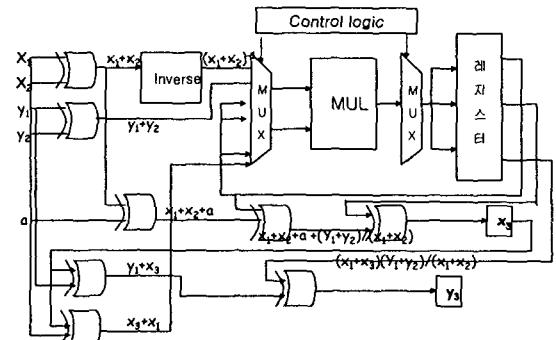


그림2. 이진체상에서 타원곡선 위의 addition 연산 모듈

3.2. 확장체에서의 유한체 연산 구현 방법

확장체의 연산은 이진체와 달리 다항식에서 $\{0, 1, 2^{31}-2\}$ 범위 내의 계수가 존재하므로 계수를 처리하는 부분이 부가적으로 필요하다. 우선, 덧셈은 입력인 두 다항식을 더하는 모듈과 더한 결과의 계수를 $2^{31}-1$ 로 reduction하는 모듈로 구성된다. 이진체와 마찬가지로 순차적인 방법을 사용하여 덧셈을 구현하였다. 곱셈기는 (31*31)비트 곱셈기를 한 비트씩 처리하는 방법을 사용하여 곱셈을 수행하는 모듈을 구현하여 (217*217)비트 곱셈기로 확장을 하였다. 곱셈모듈, 계수 모듈러 모듈, 434비트를 217비트로 reduction하는 모듈로 구성이 되는데 이진체와는 달리 reduction은 확장체의 특성을 살려 마지막 단계에서 한번만 re 수행된다. 마지막으로 역승산기는 이진체와 마찬가지로 확장 유클리드 알고리즘을 사용했는데, 계수에 대한 역원을 구하는 횟수를 줄이기 위해 확장 유클리드 알고리즘을 수정한 방법을 사용하였다. 이 방법은 계수에 대한 역원을 구하기 위한 나눗셈을 곱셈으로 대체신한다. 그래서, 계수에 대한 역원은 마지막 단계에

서 1번만 구하므로 시간 복잡도가 많이 줄어든다.[9]

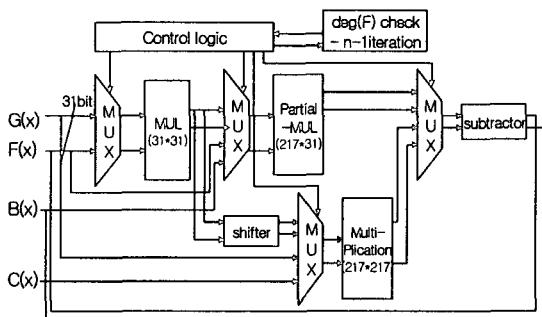


그림3. 확장체의 역원 모듈

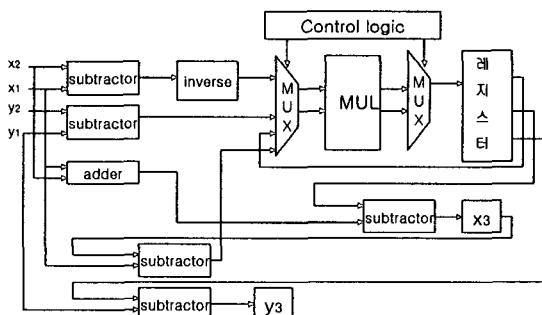


그림4. 확장체상에서 타원곡선 위의 addition 연산 모듈

그림1은 OEF상에서 구현된 역원 연산의 모듈을 나타낸 것인데 이진체에서는 덧셈 연산이 XOR로 간단하게 구현되었지만 OEF상에서는 실제 빼셈기가 추가된 것을 알 수 있다. 그리고, 그림2는 타원곡선 위의 adding 모듈을 표현한 것인데 이진체 상에서 구현된 타원곡선 위의 adding과 큰 차이점은 없다.

4. 성능비교

유한체 상의 연산기 구현 결과와 타원곡선 위의 스칼라 곱셈 연산에 대한 성능 비교는 표1과 같다.

| | 이진체 | 확장체 |
|-------------|-----|-----|
| 곱셈 | | |
| 역원 | | |
| EC doubling | | |
| EC adding | | |

표1. 이진체와 확장체의 연산 성능 비교

실험환경을 설명하면 다음과 같다. 이진체와 확장체의 덧셈기, 곱셈기, 역승산기는 하드웨어 기술언어인 VHDL(Very high speed integrated circuit Hardware

Description Language)로 코딩되었다. MODEL TECHNOLOGY 社의 modelsim을 사용하여 시뮬레이션을 수행하였고, Xilinx Foundation Series 3.1i 환경에서 VIRTEX 1000E에 targeting하여 에뮬레이션을 수행하였다.

5. 결론

본 논문에서는 차세대 공개키 암호화 알고리즘인 타원곡선 암호시스템을 이진체와 확장체에서 각각 하드웨어로 구현하여 그 성능을 비교, 검증하였다.

실험결과에서 보았듯이 하드웨어로 구현을 하는 경우 이진체가 확장체보다 더 효율적이었다. 그 이유는 이진체가 계수를 처리해주는 부분이 없기 때문에 유한체 연산이 확장체보다 더 간단하게 구현 가능하기 때문이다. 하지만 구현의 성능보다 보안의 측면에서 보면 확장체가 더 높은 강도를 제공하여 더 안전하다고 볼 수 있다. 타원곡선 암호시스템이 메모리 용량이 작은 환경에 적합하므로 현재로서는 이진체의 하드웨어 구현이 더 적합하나, 확장체 상에서의 연산을 보다 효율적으로 구현하여 게이트 수를 줄인다면 확장체 상의 타원곡선 암호시스템 구현이 이진체 상에서 타원곡선 암호시스템을 구현하는 것보다 훨씬 강력한 보안을 제공할 것이다.

[참고문헌]

- W. Diffie and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory.
- Certicom whitepaper, "Introduction To Information Security", March 1997
- Daavid Naccache, David M'Raihi, "Cryptographic Smart Cards", IEEE MICRO Vol. 16, No. 5, June 1993
- T. Satoh, K. Araki and S. Miura, "Overview of elliptic curve cryptography", Proc of PKC'98, Springer-Verlag, LNCS 1431, pp.29-49, 1998.
- Daniel V. Bailey and Christof Paar, "Efficient Arithmetic in Finite Field Extensions with Application in Elliptic Curve Cryptography", Jounal of Cryptology:the journal of the International Association for Cryptologic Research, 2000
- TTA.KO-12.0015 부가형 전자서명 방식 표준 - 제3부: 타원곡선을 이용한 인증서 기반 전자서명 알고리즘
- Christof Paar, "Implementation Options for Finite Field Arithmetic for Elliptic Curve Cryptosystems", Proc of 3rd Workshop on Elliptic Curve Cryptosystems, ECC'99, Waterloo, Ontario, Canada, November, 1999
- Hannes Brunner, Andreas Curiger, and Max Hofstetter, "On Computing Multiplicative Inverses in $GF(2^n)$ ", IEEE TRANSACTIONS ON COMPUTERS, Vol. 42, No. 8, August 1993
- Chae Hoon Lim and Hyo Sun Hwang, "Fast Implementation of Elliptic Curve Arithmetic in $GF(pn)$ ", Springer-Verlag, LNCS 1751, pp.405-421, 2000.