通信情報保護學會誌
第 11 券 第 1 號, 2001. 2

고성능 타원곡선 암호시스템의 연산기 구현

이 병 윤*, 박 종 서**, 최 용 제***, 김 무 섭***, 김 호 원***, 정 교 일***

요 약

유한체위에서의 정의된 타원곡선의 이산대수 문제의 어려움에 기초한 타원곡선 암호시스템은 다양한 타원곡선을 사용할 수 있기 때문에 다양한 암호시스템을 구성할 수 있다. 특히 비트 당 안전도가 가장 높은 타원곡선 암호시스템은 차세대 공개키 암호시스템으로 주목을 받고 있다. 짧은 키의 사용으로 스마트 카드나 모발(Mobile) 시스템 등과 같은 제약적인 환경의 인증 및 암호화에 사용 가능하다. 본 논문에서는 고성능의 타원곡선 암호시스템을 구성하고 연산기를 VHDL 언어를 이용하여 구현하였다.

Ⅰ. 서 론

공개키 암호화 알고리즘은 1976년 Whitfield Diffie 와 Martin Hellman에 의해서 제안되었다. 공개키 암호화 시스템은 기존의 대칭키 암호 시스템에서 하나의 비밀키를 사용해 암호화와 복호화를 수행하는 것과는 달리 서로 다른 두 개의 키를 사용하여 암호화와 복호화를 수행한다. 암호화키(공개키:Publickey)는 개방된 네트워크 상에서의 공개키 목록(Directory)에 등록하고 복호화키(개인키:Private key)는 개인이 비밀리에 보관한다. 사용자 A에게 암호통신을 하려는 사용자는 공개키 디렉토리에 공개된 사용자 A의 공개키로 전달하려는 평문 M을 암호화(Encryption : C = E(M))하여 암호문 C를 생성한다. 생성된 암호문 C를 사용자 A에게 전송하면 A는 자신만의 개인키로 복호화(Decryption : C = D(c))하여 평문을 추출한다. 그림 1은 공개키 암호시스템의 개략적인 프로토콜을 설명한 것이다.

공개키 암호 시스템에서는 각각의 사용자마다 자신의 공개키 하나와 개인키 하나만을 가지고 있으면 된다. 예를 들어 네트워크 내의 사용자가 1000명이라면 전체 시스템에 필요한 키의 개수는 2,000개이다[1]. 이에 반해 대칭키 암호시스템의 경우 1000명의 사용자의 경우 교환 및 관리되어야할 키의 개수는 약 500,000개 정도이다. 인터넷과 같은 규모가 매우 큰 네트워크 상에서의 키 관리는 불가능하다.

특히 인터넷의 특성상 통신할 상대가 불특정 다수일 가능성이 높다. 이와 같은 경우에는 안전한 키 분배가 큰 문제이다. 공개키 암호시스템은 비밀키의 공유의 문제가 없고 개인이 관리해야하는 키의 개수가 매우 적다. 이와 같이 공개키 암호 시스템은 대칭키 암호 시스템에 비해서 암호의 기능적인 측면에서 우위에 있다고 볼 수 있지만, 현재 널리 사용되고 있는 암호 시스템은 대칭키 암호 시스템이다. 그 이유는 대칭키 암호 시스템이 공개키 암호 시스템에

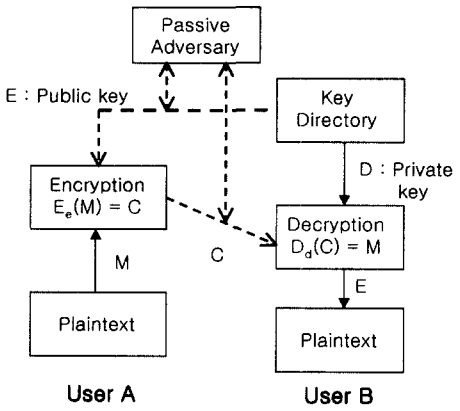


그림 1. 공개키 암호시스템

비해 데이터 처리 속도가 대략 100~1000배 빠르기 때문이다. 가장 효율적인 접근은 두 암호 시스템의 장점을 혼합한 방법이다. 이 방법은 공개키 개념을 이용하여 공통의 비밀키(Secret key)를 교환하고 그 후에는 교환된 공통의 비밀키를 사용하여 대

량의 데이터를 대칭키 암호 알고리즘을 수행하는 형태이다. 이러한 혼합시스템(Hybrid system)은 대칭키 암호 시스템의 고속의 암호화를 제공하고 공개키 암호 시스템을 사용하여 키 분배 문제점을 해결한 시스템이다[9].

공개키 암호 알고리즘은 여러 가지가 제안되었으나 오늘날 안전성과 효율성 문제 때문에 세 가지 형태만이 존재한다.

첫째, 소인수 분해의 문제(Integer Factorization Problem)의 어려움을 이용한 RSA시스템이 있다. 둘째, 이산 대수 문제(Discrete Logarithm Problem)의 어려움을 이용한 DSA와 Diffie-Hellman 그리고 ElGamal 스킴(Scheme)이 있으며 마지막으로 타원곡선의 이산 대수 문제(Elliptic Curve Discrete Logarithm Problem)의 어려움을 이용한 타원곡선 암호시스템(Elliptic Curve Cryptosystem)이 있다[2].

타원곡선은 약 150년 전부터 수학적으로 광범위하게 연구가 되어왔지만 타원곡선 암호시스템은 1985년 N. Koblitz와 V. Miller에 의해서 처음 제시되었다[9]. 타원곡선을 이용한 공개키 암호시스템은 유한체(Finite fields) 위에서 정의된 타원곡선 군에서의 이산 대수 문제에 기초한다. 타원곡선 암호 시스템은 비트 당 안전도가 타 공개키 시스템보다 효율적이라고 알려져 있다[2].

II. 타원곡선 암호시스템의 특징

1. 타원곡선 이산대수 문제 (Elliptic Curve Discrete Logarithm Problem)

타원곡선의 이산 대수 문제는 유한체(Finite field)위에 정의된 타원곡선 E가 정의되어있을 때, 정수 a 배의 관계에 있는 타원곡선 위의 두 점 P, Q를 모두 안다고 하더라도 정수 a를 계산하는 것이 어렵다는 점을 이용한 것이다.

$$a*P=Q\ (\ P,Q \in E(F_q)) \qquad (1)$$

이 때, 주의해야할 점은 연산 * 는 일반적인 사칙연산이 아니라 타원곡선 상의 배수 연산이다. 또한 P와 Q는 타원곡선 위의 점을 의미하며 소문자로 표기된 a는 타원곡선상의 점이 아닌 일반적인 정수를 의미한다.

타원곡선 암호시스템의 주연산은 타원곡선 위의 점 P를 a배 하는 배수 연산이다. 이 배수계산은 서로 다른 두 점을 더하는 덧셈 연산과 한 점을 두 배 하는 더블 연산으로 구할 수 있다.

DSA(Digital Signature Algorithm)와 Diffie-Hellman 과 ElGamal 스킴의 일반적인 이산대수 문제와 비교하였을 때, 매우 유사하게 보인다. 일반적인 이산 대수 문제는 유한체 Fq 와 g, h ∈ Fq 가 주어졌을 때 $g^a = h$를 만족하는 정수 a를 구하는 문제로서 g를 a번 곱하는 곱셈의 문제이고, ECDLP는 타원곡선 상의 점 P를 a번 더하는 덧셈의 문제이다. 하지만 덧셈과 곱셈의 차이를 제외시킨다면 두 문제는 추상적으로 유사하게 보이고 ECDLP가 해결하기 더 쉬워 보이나 실질적으로는 ECDLP 문제의 해결이 더 어렵다고 알려져 있다. 일반적인 이산대수문제는 두 개의 대수적 연산인 덧셈과 곱셈을 가지고 있다 이것은 덧셈을 이용한 Indexcalculus method로서 부분지수 복잡도(Subexponential time)를 가지는 알고리즘을 찾을 수 있다. 하지만 ECDLP는 타원곡선상의 두 점에 대한 덧셈이라는 연산만을 가지고 있으므로 초특이 타원곡선(Super-singular elliptic curve)을 제외하고는 Index-calculus method를 사용할 수 없다. 그러므로 타원곡선 암호 시스템에서 초특이 타원곡선을 제외하면 완전지수 복잡도(Full-exponential time)를 가지는 시스템을 구현할 수 있다. 현재 공개키 암호시스템 중에서 가장 널리 사용되고 있는 RSA 시스템은 sub-exponential time 알고리즘이 존재한다. 이 sub-exponential time 알고리즘의 수행시간은 상수 c에 대해서 다음과 같다.

$$O(\exp((c+o(1)))(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}) \qquad (2)$$

ECDLP를 위한 최상의 알고리즘은 완전지수 복잡도 알고리즘이며, 수행시간은 $O(\sqrt{p})$ 이다. 이것은 ECDLP가 소인수문제나 이산대수문제보다 풀기 어렵다는 것을 의미한다[3].

2. 타원곡선 암호 스킴(Scheme)

타원곡선 암호시스템의 이해를 돕기 위해 다양한 공개키 암호 스킴에 적용한 예를 설명하겠다.

2.1 타원곡선을 이용한 Diffie-Hellman의 키교환

두 사용자가 안전하지 않은 네트워크 상에서 비밀키를 공유하려고 할 때 사용하는 방법이다. 두 사용자 A와 B는 우선 자기 자신의 개인키와 공개키를 만들기 위해 유한체 Fq와 그 위에 정의된 타원 곡선 E를 정하고 그 위의 점 β를 정한다. β는 일반적인 곱셈군의 Diffie-Hellman에서 생성자 g의 역할을 한다. 그러나 β는 반드시 생성자일 필요도 없고 타원 곡선의 점의 개수 N을 정확히 알 필요도 없다. β는 위수가 충분히 큰 E위의 점이면 된다. 그 다음 A와 B는 각각 임의의 정수 난수 a, b를 선택하고 자신의 개인키로 보관한다. 또한 공개키 aβ, bβ를 계산하여 서로 교환한다. 그리고 공통된 비밀키로서 P = abβ를 사용한다. A는 B의 공개키 bβ에 자신의 개인키 a를 적용시켜 P를 알 수 있고, B도 같은 방법으로 P를 알 수 있다. 그림 2는 키 분배 과정을 간단히 나타낸 것이다.

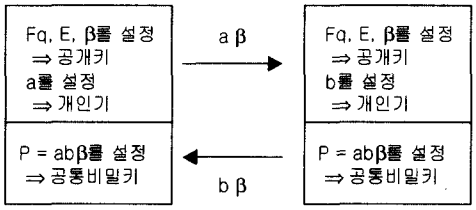


그림 2. Diffie-Hellman ECC

2.2 ElGamal 타원곡선 암호 시스템

ElGamal ECC는 사용자가 다른 사람에게 메시지를 안전하게 보내려고 할 때 사용하는 방법이다. 사용자 B가 사용자 A에게 메시지 M=(M1, M2)를 보내려고 할 경우 두 사용자는 유한체 Fq와 타원곡선 E 위의 점 β를 선택한다. 사용자 A는 임의의 정수 a를 선택하여 aβ를 계산하여 공개한다.

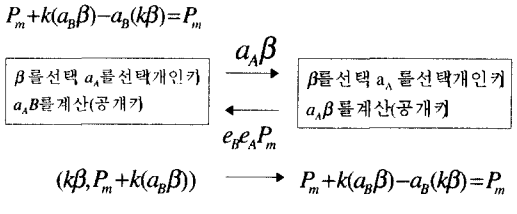


그림 3. ElGamal ECC

사용자 B는 사용자 A의 공개키 aβ를 갖고 온 후 정수 난수 k 를 선택하여 ($k\beta$, $M1*k(a\beta)$, $M2*k(a\beta)$)를 보낸다. 메시지를 읽기 위해 A는 전송 받은 kβ에 자신의 비밀키 a 를 곱하여 a(kβ)를 계산한다. 전송 받은 메시지 $M1*k(a\beta)$, $M2*k(a\beta)$에 계산한 a(kβ)를 나누어 M1 과 M2를 얻을 수 있다. 그림 3는 ElGamal ECC 과정을 간단히 나타낸 것이다.

2.3 ECDSA(Elliptic Curve Digital Signature Algorithm)

ECDSA는 타원곡선을 이용한 전자 서명 알고리즘이다. ECDSA가 160 비트 q와 1024비트 p를 가진 DSA와 비슷한 안전도를 갖기 위해서는 매개변수 n 이 약 160비트이면 된다. 이 경우 DSA와 ECDSA는 같은 서명 길이(320비트)를 갖는다.

ECDSA에서 키 생성은 다음과 같은 순서로 생성할 수 있다[4].

① 유한체 위에 정의된 타원곡선 E를 선택한다. 이때 곡선 위의 점의 수는 큰 수 n에 의해 나누어져야 한다.
② 위수(order)가 n인 타원곡선 위의 한 점 P를 선택한다.
③ 구간 [2, n-2]에서 유일하고 예측할 수 없는 정수 d를 선택한다.
④ Q = dP를 계산한다.
⑤ A의 공개키는 (E, P, n, Q) 이고 비밀키는 d 이다.

메시지 M에 대해서 서명을 한다면 다음과 같은 순서로 서명을 할 수 있다.

① 구간 [2, n-2]에서 통계적으로 예측하기 힘든 정수 k를 선택한다.
② kP = (x1, y1) 과 r = x1 mod n을 계산한다.
③ r = 0 이면, 다시 ① 단계부터 다시 수행한다.
④ k-1 mod n을 계산한다.
⑤ s = k-1{h(m) + dr} mod n을 계산한다. (h : SHA-1)
⑥ s = 0 이면 다시 ① 단계로 다시 수행한다..
⑦ 메시지 m 에 대한 서명은 (r, s) 이다.

ECDSA에서 서명에 대한 검증은 다음의 과정을

거치면 된다.

① 인증된 공개키 (E, P, n, Q)를 얻는다.
② r과 s가 구간 [1, n-1]에 있는지 확인한다.
③ w = s-1 mod n과 h(m)을 계산한다.
④ u1 = h(m) w mod n 과 u2rw mod n을 계산한다.
⑤ u1P + u2Q = (x0, y0) 와 v = x0 mod n을 계산한다.
⑥ v = r를 확인한다.

3. 타원곡선 암호의 연산

타원곡선 암호에서 사용되는 기본 연산은 타원곡선 위의 점 P의 정수 a배의 곱셈연산이다. 곱셈연산을 다음 식과 같이 두 점의 덧셈과 한 점의 두 배 연산으로 분해할 수 있다.

$$100\ P = 2(2(P+2(2(2(P+2P)))))) \quad (3)$$

타원곡선이 F_{2^m} 의 유한체 위에 다음과 같이 정의되어 있을 경우 타원곡선 위의 서로 다른 두 점 P(x1, y1) 와 Q(x2, y2)의 덧셈을 구하는 공식은 다음과 같다.

$$E:\ y^2+xy=x^3+ax^2+b \quad (4)$$

$$x_3=\lambda^2+\lambda+x_1+x_2+a \quad (5)$$

$$y_3=\lambda(x_1+x_3)+x_3+y_1 \quad (6)$$

$$\lambda=\frac{y_2+y_1}{x_2+x_1} \quad (7)$$

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

$$x_3=\lambda^2+\lambda+a \quad (8)$$

$$y_3=x_1^2+(\lambda+1)x_3 \quad (9)$$

$$\lambda=x_1+\frac{y_1}{x_1} \quad (10)$$

위의 두 점의 덧셈 공식과 한 점의 두 배 공식에 사용된 덧셈과 곱셈과 나눗셈은 모두 일반적인 실수체의 연산들이 아니라 유한체 위에 정의된 덧셈과 곱셈과 나눗셈이다.

III. 타원곡선 암호시스템의 설계

타원곡선 암호시스템을 구현하기 위해서는 다양한 선택적 요소들이 있다. 타원곡선 암호시스템을 적용한 응용에 따라 이들 요소들을 적합하게 조합하는 것이 필요하다. 이들 파라미터에 따라서 안전성에 중점을 둘 것인지 성능에 중점을 둘 것인지를 결정할 수 있다. 파라미터들에는 유한체, 기저, 타원곡선, 좌표 계 등이 있다. 본 고에서는 고속의 타원곡선 연산을 수행할 수 있는 파라미터들을 선택하였다.

1. 타원곡선 암호시스템의 구현 파라미터

유한체에 대해서 살펴보면 실제 구현에 사용되는 유한체에는 표수(characteristic)가 3보다 큰 소수인 F_p 가 있으며, 표수가 2 또는 짝수인 F_{2^m}형태가 있다. 수학적인 연구에 의하면 2^m과 p가 비슷한 크기라면 그 위에 정의된 ECDLP는 대략 같은 시간 복잡도를 가지고 있다고 한다[2]. 구현에 있어서는 하드웨어의 경우에는 유한체 F_{2^m} 가 F_p 에 비해서 효율적이다. 하지만 소프트웨어의 경우에는 F_p 사용이 효과적일 수도 있다. 보안 측면에서 볼 때 유한체 F_{2^m}는 ECDLP의 복잡도에 악영향을 준다. 그러므로 특별한 보안을 요구하는 응용에 있어서는 유한체 F_p 사용이 효과적일 수도 있다. 실수에서 정의된 곡선을 사용하는 것이 아니라 유한체 위에 타원곡선을 사용하는 것은 유한개의 원소로 이루어지고 실수에서 정의된 것과는 달리 round off

표 1. 타원곡선의 덧셈과 두 배 연산 공식의 체 연산 횟수

operation	Affine	Projective(Z1≠1)	Projective(Z1=1)
Doubling	1I +2M + 2S + 8A	4M +4S +8A	4M + 4S +8A
Addition	1I +2M + S + 6A	12M + 4S +9A	8M + 5S + 9A

error 가 발생하지 않기 때문에 암호학적 목적에 적합하기 때문이다[5].

특히 유한체 F_{2^m} 의 경우에는 세 가지의 기저 중에 한가지를 선택해야 한다. Polynomial basis 나 Normal basis 나 Optimal normal basis 세 가지가 있는데 Normal basis 보다는 Optimal normal basis 가 더 효율적이기 때문에 Polynomial basis 나 Optimal normal basis 중에서 한가지 기저를 선택하여 유한체를 구성한다. 타원곡선 암호시스템이 적용될 응용에 따라서 기저는 선택되어질 수 있을 것이다. Polynomial basis는 속도가 빠르지만 공간을 많이 차지하고, normal basis 는 적은 공간을 사용하지만 역원을 구하는 속도가 느리다[6].

타원곡선에 관해서 살펴보면, 타원 곡선은 표수에 따라서 형태가 조금씩 차이가 있다. 체의 표수가 2와 3이 아닌 경우에는 타원 방정식은 식 (11) 과 같다. 체의 표수가 2인 경우에는 식 (12) 또는 식 (13) 의 타원곡선을 사용하며 체의 표수가 3인 경우에는 식 (14)를 사용한다.

$$y^2 = x^3 + ax + b \quad (\text{단, } 4a^3 + 27b^2 \neq 0) \tag{11}$$

$$y^2 + xy = x^3 + ax^2 + b \tag{12}$$

$$y^2 + cy = x^3 + ax + b \tag{13}$$

$$y^2 = x^3 + ax^2 + bx + c \tag{14}$$

좌표 계는 Affine coordinates 와 Projective coordinates가 있는데 사용하는 좌표 계에 따라서 타원곡선 위의 두 점의 덧셈 공식과 두 배 공식의 차이가 있다. 다음 표 1은 두 좌표 계에 따라 덧셈 공식과 두 배 공식의 체의 연산 횟수를 나타낸 것이다.

본 논문에서 구현하는 타원곡선 암호시스템은 고속의 암·복호화를 수행할 수 있도록 파라미터를 구성하였다.

유한체는 F_{2^n} 를 사용하였다. 이는 하드웨어 구현에 있어서 20~30% 정도의 성능향상이 있다. 유한체의 사이즈는 향후의 안전도 요구량을 고려하여 193비트를 선택하였다. 193비트의 타원곡선 암호시스템은 1536비트 RSA 시스템과 유사한 안전도를 나타낸다[7].

유한체의 기저는 공간 사용은 많지만 고속의 타원곡선 암호시스템을 위해 Polynomial basis를 선택하였다. 타원곡선 방정식은 Certicom사의 "SEC 2"에서 제안하는 $y^2 + xy = x^3 + ax + b$를 사용하였다. 모듈라 연산을 수행하기 위한 기약다항식(Irreducible polynomial)은 $x^{193} + x^{15} + 1$을 선택하였다[7].

2. 타원곡선 암호시스템의 구조

본 논문에서 구성하는 타원곡선 암호시스템은 하드웨어와 소프트웨어가 연동하여 동작하는 시스템이다. 타원곡선 암·복호화에 주연산인 타원곡선 위의 점의 배수연산의 기본 연산이 되는 유한체의 덧셈과 곱셈과 역원을 구하는 연산기를 하드웨어로 구현하고 하드웨어 연산기로 계산된 결과 값들을 프로그램이 처리하는 방식의 타원곡선 암호시스템이다.

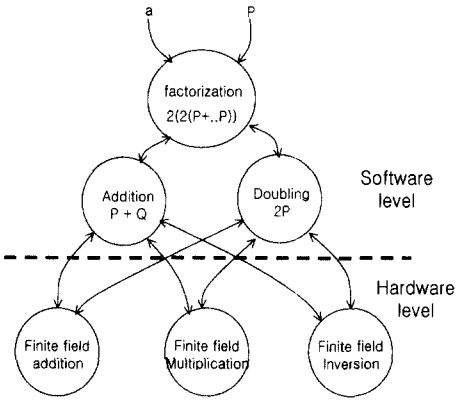


그림 4. 타원곡선 암호시스템의 데이터 처리 계층

타원곡선 암호시스템의 데이터 처리 과정을 살펴보면, 프로그램이 계산할 타원곡선 위의 점의 배수 값을 입력받아 이 값을 가장 효율적으로 계산할 수 있도록 덧셈과 두 배 연산으로 인수 분해한다. 프로그램은 인수 분해된 결과를 가지고 타원곡선의 덧셈 연산과 두 배 연산에 해당하는 공식을 적용한다. 공식을 수행하는 과정의 모든 덧셈과 곱셈 그리고 역원연산은 하드웨어로 구현된 연산기에 의해서 계산된 결과 값을 가지고 수행한다.

핵심 연산기는 두 개 혹은 한 개의 유한체 위의 값을 입력받아 두 값을 더하거나 곱하거나 역원을 계산하여 출력한다.

Ⅳ. 타원곡선 암호 핵심 연산기 설계

본 고에서 구현하는 연산기는 유한체위의 정의된

이진 값들의 덧셈과 곱셈 그리고 역원을 계산한다. 핵심 연산기의 회로 설계는 하드웨어 기술 언어인 VHDL을 이용해서 설계하였다. 설계한 회로의 기능 검증은 MaxPlusII를 이용하여 검증하였다.

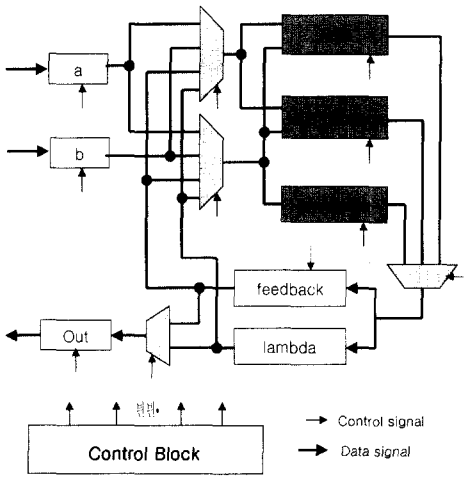


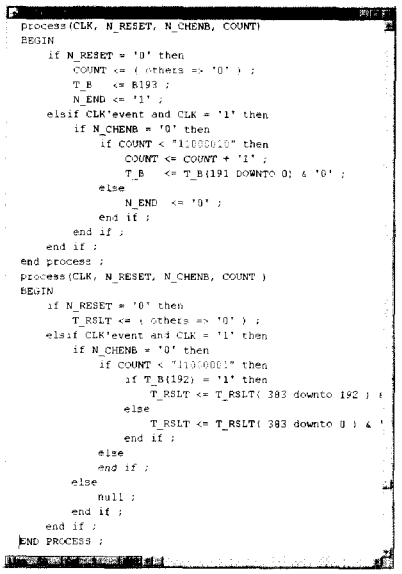
그림 5. 타원곡선 암호시스템의 핵심 연산기 블록도

1. 타원곡선 암호 연산기 구조와 기능

연산기는 서로 다른 두 점의 덧셈과 한 점의 두 배 연산을 구하는 공식을 수행하는데 필요한 유한체 덧셈과 곱셈 그리고 역원을 수행한다. 연산기 구조는 그림 5와 같다. 입력 값을 저장하는 a, b 두 개의 레지스터가 있으며 덧셈기와 곱셈기와 역원기가 있다. 결과 값은 feedback 레지스터와 lambda레지스터에 저장되며 연산기 외부로 출력될 값을 저장하는 Out레지스터가 존재한다. 공식을 수행하는 과정의 중간 결과는 feedback레지스터나 lambda레지스터에 저장하여 입출력 오버헤드를 줄일 수 있다.

2. 유한체 모듈라 곱셈기 설계

곱셈기 구조는 배열 구조(array structure), 비트/디지트(bit/digit) 병렬 구조,비트/디지트 순차



```
process(CLK, N_RESET, N_CHENB, COUNT)
BEGIN
    if N_RESET = '0' then
        COUNT <= ( others => '0' ) ;
        T_B   <= B193 ;
        N_END <= '1' ;
    elsif CLK'event and CLK = '1' then
        if N_CHENB = '0' then
            if COUNT < "11000010" then
                COUNT <= COUNT + '1' ;
                T_B   <= T_B(191 DOWNTO 0) & '0' ;
            else
                N_END  <= '0' ;
            end if ;
        end if ;
    end if ;
end process ;
process(CLK, N_RESET, N_CHENB, COUNT )
BEGIN
    if N_RESET = '0' then
        T_RSLT <= ( others => '0' ) ;
    elsif CLK'event and CLK = '1' then
        if N_CHENB = '0' then
            if COUNT < "11000001" then
                if T_B(192) = '1' then
                    T_RSLT <= T_RSLT( 383 downto 192 ) +
                else
                    T_RSLT <= T_RSLT( 383 downto 0 ) & '
                end if ;
            else
            end if ;
        else
            null ;
        end if ;
    end if ;
END PROCESS ;
```

그림 6. 순차적 모듈라 곱셈기 VHDL코드

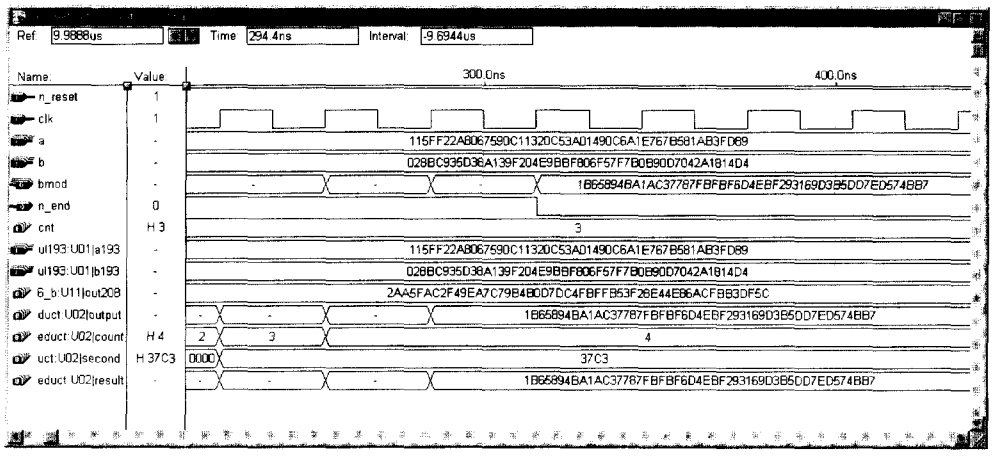


그림 7. 16비트 병렬 Hybrid 모듈라 곱셈기 기능 검증

구조 등 매우 다양한 구조를 가질 수 있다. 각각의 구조들은 유한체 기저에 따라 다른 성능은 나타낸다.

다항식 기저에서는 두 다항식의 덧셈 연산에 캐리가 발생하지 않기 때문에 병렬 처리에 매우 유리하다. 처리 속도를 높이기 위해서 배열 구조나 비트 병렬 구조 등을 취할 수 있다. 하지만 면적의 사용이 $O(k^2)$가 늘기 때문에[10] 응용에 따라 성능과 비용을 고려하여 병렬 처리 크기를 조절하여 구현할 수 있다. 스마트 카드와 같은 면적사용의 제약이 있는 응용에서는 한 비트씩 순차적으로 곱셈을 수행하는 비트 순차 형태가 적합하다.

유한체의 모듈라 곱셈기는 두 개의 입력 값을 곱하는 MUL193 모듈과 곱셈을 수행한 후의 결과 값을 모듈라 연산을 하는 REDUCT 모듈로 구성하였다. MUL193 모듈은 193비트의 두 개의 입력 값을 받아 두 값을 곱한 385비트의 결과 값을 출력한다.

REDUCT 모듈은 MUL193 모듈의 385비트의 결과 값을 입력으로 받아 모듈라 연산을 수행하여 193비트의 결과 값을 출력한다. 이 결과 값이 모듈라 곱셈의 최종 결과 값이다.

그림 6은 한 비트씩 순차적으로 곱셈을 수행하는 형태로 구현한 MUL193 모듈의 VHDL 코드이다. 이 순차적인 구조의 모듈라 곱셈기의 수행시간은 약 200클럭 정도가 소모된다. 그림 7은 MaxPlusII를 이용하여 순차적 모듈라 곱셈기의 기능을 검증한 결과 화면이다. 그림 8은 곱셈기의 합성 화면을 나타낸 것이다.

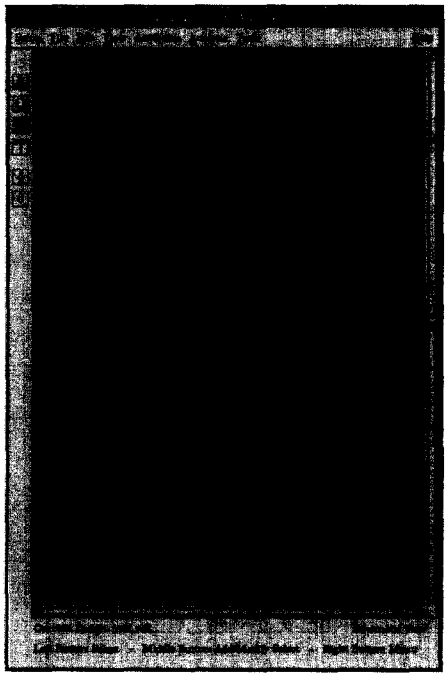


그림 8. 모듈라 곱셈기 Synopsys에서 합성 결과 화면

3. 유한체 역승산기 설계

역승산기는 193비트의 입력을 받아 193비트의 모듈라 곱셈의 역원 값을 출력한다. 본 역승산기는 F_{2^m}체에서 최상의 성능을 나타내는 almost inverse algorithm으로 구현하였다[8].

그림 9는 역승산기의 블록도 이다. 블록도에 나타난 레지스터 F, G는 기약 다항식의 크기가 194비트이기 때문에 레지스터의 크기는 194비트이다. Find_deg module은 193비트의 입력에 대한 최고차수를 구하는 모듈이다. Div module은 다항식 B를 k차수로 나누는 모듈이다. Find_deg module과 Div module은 클럭을 소모하지 않는 논리 조합회로이다. Control Block은 FSM(Finite State Machine)으로 되어 있으며 각 상태에 따라서 모든 모듈들의 제어 신호를 발생한다.

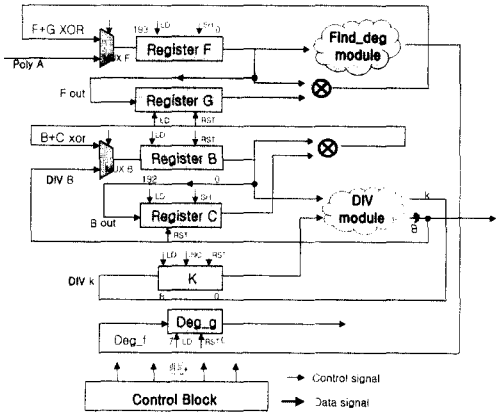


그림 9. 역승산기 블록도

AIA알고리즘의 대부분의 작업을 제어블록에서 수행을 한다. 제어 블록에는 Find_deg 모듈을 통과한 레지스터 F의 차수와 레지스터 Deg_G에 저장된 레지스터 G의 차수와 레지스터 F의 LSB 값이 입력으로 들어간다. 이들 입력 값에 의해 상태가

전이된다. 그림 10은 역승산기 모듈의 상태전이도이다.

본 논문에서 구현한 역승산기는 스마트 카드나

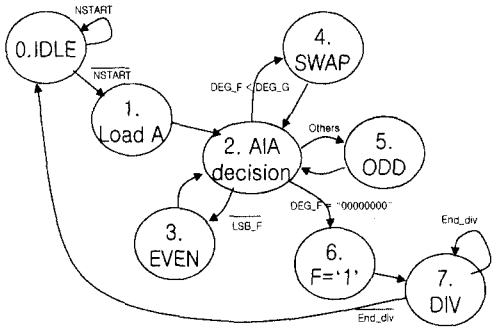


그림 10. 역승산기 상태 천이도

모빌 시스템에 적용할 수 있도록 공간 사용을 최소화하였다. Synopsys에서 현대 0.5㎛공정 라이브러리를 사용하여 합성하였을 경우 게이트 수는 약 12,600 게이트 정도이며 수행시간은 약 500클럭 정도 소모되었다.

V. 결 론

본 논문에서는 차세대 공개키 암호화 알고리즘인 타원곡선 암호시스템의 특징과 안전도와 성능에 영향을 미치는 파라미터들에 대하여 살펴보았으며, 고성능의 타원곡선 암호시스템 구현을 위한 파라미터들을 구성하였다. 유한체는 F_{2^m}에서 기저는 Polynomial basis를 선택하였으며 타원곡선은 $y^2+xy=x^3+ax+b$를 선택하였다. 모듈라 연산을 위한 기약다항식은 $x^{193}+x^{15}+1$를 선택하였다. 그리고 구성한 타원곡선 암호시스템의 연산을 효율적으로 처리할 수 있는 핵심 연산기를 하드웨어로 설계·검증하였다. 핵심 연산기내의 곱셈기는 비트 순차(bit serial)구조로 구현하였으며 역승산기는 almost inverse algorithm으로 구현하였다. 향후 연구 과제로 다양한 응용분야에 적용하기 위한 여러 가지 구현 방식에 대한 연구가 진행되어야 한다.

참 고 문 헌

[1] Certicom whitepaper, "Introduction To Information Security", March 1997.

[2] Certicom whitepaper, "Remarks On The Security of The Elliptic Curve Cryptosystem" , September 1997.

[3] Certicom whitepaper, "Current Public-Key Cryptographic system", April 1997.

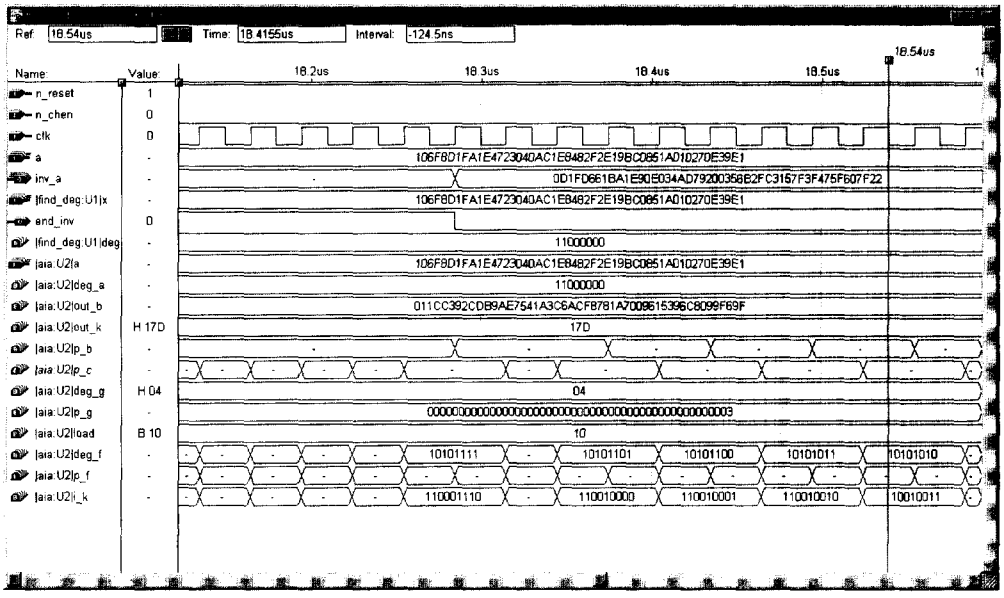
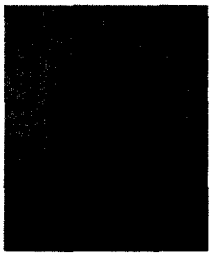


그림 11. 역승산기 기능 검증

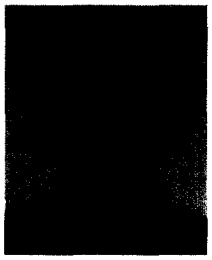
[4] Aleksandar Jurisic and Alfred J. Menezes, "Elliptic Curves and Cryptography", 1997.
[5] 박왕석, "타원곡선 암호시스템에 관한 연구", pp. 12-21.
[6] Michael Rosing, "Implementing Elliptic Curve Cryptography", 1999.
[7] Certicom research, "SEC 2 : Recommended Elliptic Curve Domain Parameters", October 1999
[8] A.Schroeppel, H.Orman, S.O'Malley and O.Spatschek, "Fast key Exchange with Elliptic Curve System", Advance in Cryptology-CRYPTO'95, 1995.
[9] W. Diffie and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, pp 644-654, 1976
[10] C. Paar, P. Fleischman and P.Soria-Rodriquez, "Fast Arithmetic for Public-Key Algorithms in Galois Field with Composite Exponents", IEEE Transactions on Computers, pp. 1025-1034, 1999
[11] J. Lopez and R. Dahab, "Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation.", CHES'99, pp. 316-327, 1999

〈著者紹介〉



이 병 윤 (Byoung-Yun Yi) 학생회원
1999년 2월 : 한국항공대학교 컴퓨터공학과 졸업
1999년 3월~현재 : 한국항공대학교 컴퓨터공학과 석사과정
관심분야 : Network Security, 암호 알고리즘 하드웨어 설계, 디지털 논리회로설계, ECC

박 종 서 (Jong-sou Park) 정회원
1983년 2월 : 한국항공대학교 통신공학과 졸업
1987년 12월 : North Carolina State University 컴퓨터공학과 석사
1994년 8월 : Penn State University 컴퓨터공학과 박사
1996년 2월 : Penn State University 컴퓨터공학과 조교수
1996년 3월~현재 : 한국항공대학교 컴퓨터공학과 조교수
관심분야 : Network Security, 항공우주용 제어기설계, VLSI



최 용 제 (Yong-Je Choi) 정회원
1997년 : 전남대학교 전자공학과 공학사
1999년 : 전남대학교 전자공학과 공학석사
1999년~현재 : 한국전자통신연구원 정보보호기술연구본부 IC카드구조연구팀, 연구원
관심분야 : 타원곡선 암호모듈 설계, 암호프로세서 설계, IC카드 설계, 정보보호

김 무 섭 (Moo-Seop Kim) 정회원
1995년 : 금오공과대학교 전자공학과 공학사
1998년 : 경북대학교 전자전기공학과 공학석사
1998년~1999년 : LG종합기술원
1999년~현재 : 한국전자통신연구원 정보보호기술연구본부 IC카드구조연구팀, 연구원
관심분야 : 암호모듈 설계, 암호프로세서 설계, IC카드 설계, 정보보호

김 호 원 (Ho-Won Kim) 정회원
1993년 : 경북대학교 전자공학과 공학사
1995년 : 포항공과대학교 전자전기공학과 공학석사
1999년 : 포항공과대학교 전자전기공학과 공학박사
1998년~현재 : 한국전자통신연구원 정보보호기술연구본부 IC카드구조연구팀, 선임연구원
관심분야 : 타원곡선 암호모듈 설계, 암호프로세서 설계, IC카드 설계, 정보보호

정 교 일 (Kyo-Il Jung) 정회원
1981년 : 한양대학교 전자공학과 공학사
1983년 : 한양대학교 산업대학원 전자계산학과 공학석사
1997년 : 한양대학교 대학원 전자공학과 공학박사
1995년~1997년 : 한국정보통신기술협회 전파통신분과위원회 연구위원
1981년~현재 : 한국전자통신연구원 정보보호기술연구본부 IC카드구조연구팀 팀장/책임연구원
관심분야 : IC카드 설계, 정보보호 시스템