휴대 단말기용 타원곡선 암호 프로세서의 설계

이완복* · 김정태**

Design of a Elliptic Curve Crypto-Processor for Hand-Held Devices

Wan-Bok Lee* · Jung-Tae Kim**

요 약

최근 인터넷 및 유무선 통신 인프라가 발전함에 따라, 개인 휴대용 단말기나 스마트 카드 등의 다양한 방면에서 개인 정보보호를 위해 고비도의 암호 시스템이 요구되고 있다. 본 논문에서는 연산력이 떨어지는 무선통신용 단말이나 내장형 시스템에서 고비도의 암호 연산력을 제공할 수 있는 타원곡선형 암호 하드웨어의 설계에 대해 소개한다. 효율적인 암호 연산기를 제작하기 위해 먼저 타원곡선 암호 시스템의 핵심 연산 계층도를 분석해보고, 직렬 셀 곱셈기와 확장유클리드 알고리즘을 수정하여 유한체 나눗셈기를 적용하여 제작하였다. 제작된 암호 시스템은 시뮬레이션 결과 올바른 동작을 보임을 확인할 수 있었으며, 초당 수천회의 서명이 가능한 수준이었다.

ABSTRACT

The more improved the Internet and the information technology, the stronger cryptographic system is required which can satisfy the information security on the platform of personal hand-held devices or smart card system. This paper introduces a case study of designing an elliptic curve cryptographic processor of a high performance that can be suitably used in a wireless communicating device or in an embedded system. To design an efficient cryptogrphic system, we first analyzed the operation hierarchy of the elliptic curve cryptographic system and then implemented the system by adopting a serial cell multiplier and modified Euclid divider. Simulation result shows that the system was correctly designed and it can compute thousands of operations per a secdond.

키워드

암호프로세서, 공개키, 확장유클리드 알고리즘, 칩설계

I. 서 론

최근 인터넷 및 무선통신의 급성장으로 정보 보호는 아주 중요한 문제로 인식되고 있으며, 암호 시스템에 기반을 둔 공개키 기반 구조 시스템이나 가상 사설망 기술 등은 이제 기업이나 개인 정보보호를 위해 보편적으로 사용되고 있다. 암호 시스템은 소프트웨어를 이용하여 쉽게 구현할 수 있지만 실시간 응용을 위해서는 적합하지 않다. 고비도의 안정성을 제공하기 위해서는 암호 시스템의 하드웨어 구현이 바람직하다. 특히, 계산력이 충분하지 못한 휴대용 단말이나, 무선 통신 기기에서 안전한 통신과 전자상거래를 보장하기 위해서는 핵심 암호 연산을 하드웨어적으로 수행하는 것이 필요하다.

본 논문에서는 이러한 용도로 사용할 수 있는 타원곡선 암호 프로세서의 설계에 대해 소개한다. 설계한 암호 프로세서는 보안용 MCU와 더불어 각종 암호 시스템의

* 공주대학교 게임디자인학과
** 목원대학교 정보전자영상공학부

접수일자 : 2006. 10. 26

고속화를 위해 연동될 수 있을 뿐만 아니라, 단독적으로 사용되어 보안토큰이나 스마트 카드에 활용 될 수 있다.

암호 프로세서의 효율적인 설계를 위해 본 연구에서는 하드웨어 소프트웨어 코디자인 기법을 도입하여, 타원 곡선형 암호 연산의 연산량을 모듈별로 측정하고 그것을 기반하여 각종 연산 모듈을 계층별로 설계하였다.

본 논문은 다음과 같이 이루어져 있다. 2장에서는 타원곡선 암호 시스템의 연산을 분석하고 연산 모듈별 소요 시간과 빈도를 분석하였다. 3장에서는 설계한 암호 시스템의 전체 구조 및 각 내부 모듈의 구체적인 설계 내용에 대해 소개한다. 4장에서는 설계한 암호 프로세서의 성능에 대해 측정하고 5장에서 결론을 맺는다.

II. 타원곡선 암호 시스템의 구성

RSA[1], ElGamal[2], 등을 비롯한 대부분의 공개키 암호 알고리즘들은 모두 가환군 내에서의 이산대수나 소인수분해 문제에 기반하고 있다. 이 중 RSA 알고리즘은 현재까지 가장 널리 알려졌으며 또한 사용되어지고 있는데, 최근 576 비트 길이의 키는 소인수 분해 공격에 의해 그 보안성이 깨어질 수 있다는 것이 보고 된 바 있다. 그렇기 때문에 RSA 알고리즘이 실제적인 암호 공격을 견디기 위해서 576 비트 이상의 키 길이를 가져야 하는 것이 타당하다. 그러나 키 길이가 길어지면, 통신하는 양단간에 주고받는 메시지의 정보량이 늘어나야 함과 동시에, 계산량이 많아지기 때문에 연산능력이 제한된 이동 통신 단말기에서는 그 사용이 제한될 수 있다. 반면에 1995년 Koblitz와 Miller에 의해 제안된 타원곡선을 이용한 공개키 암호 시스템은 비트당 안전도가 타 공개키 시스템보다 효율적이며 160 비트의 키 길이로서 1024 비트 RSA 암호 알고리즘과 대등한 암호학적 비도를 가지는 것으로 알려져 있다[2]. 키 길이가 짧아질 경우, 암호 연산에 요구되는 대역폭과 메모리가 작아질 수 있으며, 이로 인해 메모리와 처리능력이 제한된 스마트카드, 이동통신에서의 보안성 제공 같은 응용에서 중요한 기반 암호 기술로서 활용 될 수 있다.

타원 곡선 암호에서 가장 중요한 연산 과정은 k 상수배 연산(스칼라 곱셈) 이다. 이 연산은 RSA 알고리즘에서 유한체의 지수승 연산에 해당하는 부분으로서 그 역계산이 매우 어려운 특성이 있다. 이 스칼라 곱셈을 수행하기 위해서는 크게 두 부분으로 나누어지는 연산 계층의 단위 연산들을 구현하여야 한다. 그림 1은 타원 곡선 암호화에서 사용되는 연산의 계층을 도식적으로 나타내었다. 점 덧셈 연산과 두배점 연산은 스칼라 곱셈 아래에 위치하는데, 이들 연산은 최하위 계층인 GF(2m) 상의 두 원소에 대한 유한체 곱셈, 유한체 제곱, 유한체 나눗셈, 유한체 덧셈 연산을 기본으로 구성되고 있다.

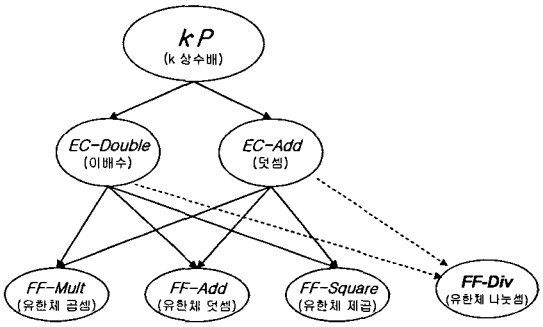


그림 1. 타원 곡선 암호의 연산 계층도
Fig. 1. Operation Block Diagram of Elliptic Curve Encryption

그림 1의 연산 계층도에서 상위 계층 연산을 1회 수행하기 위해서는 하위 계층 연산을 다수 회 수행하여야 한다. 따라서, 최하위 계층의 연산에 소용되는 시간은 전체 연산 속도와 밀접한 관계가 있으며. 빠른 암호화 성능을 보장하기 위해서는 말단의 연산들을 하드웨어로 구현하여 그 계산 시간을 가속화할 수 있다.

비록 타원 곡선 암호가 비교적 적은 계산량으로 높은 수준의 암호학적 비도를 가진다고 하지만, 현재까지 개발된 Java Card나 Smart Card에서 구현되어 상용화 되기는 어렵다[3]. 아래 표 1은 [3]에서 ECC2-109 알고리즘에 대하여 3개의 상용 Java Card 상에서 소프트웨어적으로 구현하여 타원 곡선상의 각 연산들을 수행하였을 때 소요되는 시간을 정리한 것이다. 실험 결과, 타원 곡선 암호의 기반이 되는 점 덧셈과 이배수 연산과정에서 모두 9분 이상의 시간이 소요되는 것을 확인할 수 있다. 이러한 수준의 계산력은 상용의 타원곡선 암호 시스템을 구현하기에 부족하다. 타원 곡선 암호의 핵심이 되는 스칼라 상수배 연산을 1회 수행하기 위해서는 수백회의 점 덧셈이나 이배수 연산이 소요되는 점을 고려해 볼 때[4], 단 한번의 암호화 과정에 소요되는 시간은 수십 시간에 이를 전망이다.

표 1. 세 종류의 Smart Card 비교 (ECC2-109 기준)
Table. 1 Comparison of Smart Card with Three Items

Card	Odyssey	Cyberflex	Simphonic
A + B	0:01.1	0:02.8	0:02.8
A * B	0:48.2	3:05.2	3:37.5
A / B	0:02.0	0:06.5	0:06.6
A^{-1}	8:11.2	29:15.0	27:19.2
P + P	9:32.1	33:15.0	N.A.
P + Q	9:57.3	36:45.1	N.A.

그러므로, 실용적인 타원 곡선 암호 시스템을 휴대용 단말기나 임베디드 시스템에 장착하려면, Java Card 수준의 계산력을 가진 프로세서로는 구현하기가 곤란하며, 타원 곡선 암호 전용의 암호 IP를 개발하여 실장해야 할 필요가 있다.

본 연구는 타원곡선 암호 시스템을 위한 유한체 연산기의 FPGA 설계 및 구현에 관한 것이다. 본 논문에서 고속 연산을 수행하기 위하여 기존의 Cell array 곱셈기를 개선한 Serial-Cell_array 곱셈기를 이용한 설계를 하였으며, 유한체 나눗셈기는 Modified 확장 유클리드 알고리즘을 기본으로 하여 설계하였다.

Ⅲ. 고성능 타원곡선 암호 시스템의 설계

3.1. 전체 암호 시스템 구성

본 논문에서는 타원곡선을 정의하는 파라메터는 SEC2[5] 에서 권장하는 값을 사용하였고, 기약다항식 f(x)의 차수는 m = 163을 선택하였다. f(x)는 표준 기저 방식으로 표현되며, $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ 로 정하였다. 고속연산을 위해서 유한체 곱셈은 Serial Cell_array Multiplication 구조로 설계하였고, 유한체 역원 연산을 하기위해 확장 유클리드 알고리즘을 이용하였는데, 설계된 알고리즘은 나눗셈 연산시 역원 연산기를 그대로 이용함으로써 연산속도를 빠르게 할 수 있는 구조이다. 그리고 이렇게 설계된 유한체 연산기를 사용하여 타원곡선 point 연산기(addition 과 doubling)를 설계하였다. 그리고 point 연산기를 이용해 point multiplication을 구현 하였다.

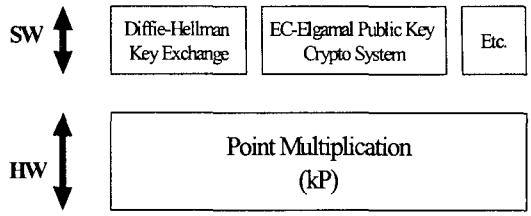


그림 2. 전체 암호 시스템의 구성도
Fig. 2. Configuration of Total Crypto System

전체 암호 시스템의 구성도는 그림 2에 보이는 바와 같다. 타원 곡선 암호 시스템을 소프트웨어적으로 구현하였을 경우, 연산시간의 대부분은 k상수배 연산에 소요된다. 그러므로, k상수배 연산은 하드웨어적으로 구현하고, 상위 레벨의 암호시스템은 소프트웨어적으로 구현할 경우, 유연한 시스템 설계가 가능해지는 장점이 있다[4].

하드웨어 모듈에서는 k상수배 연산을 가속화하는 역할을 담당하도록 한다, 내부적으로는 위 그림 1의 연산 계층도의 내부 연산들을 호출하면서 동작하며, 그 제어 구조는 다음 그림 3에서 보이는 바와 같이 타원곡선상의 덧셈과 이배수 유닛을 State Machine이 제어하면서 동작하는 구조이다.

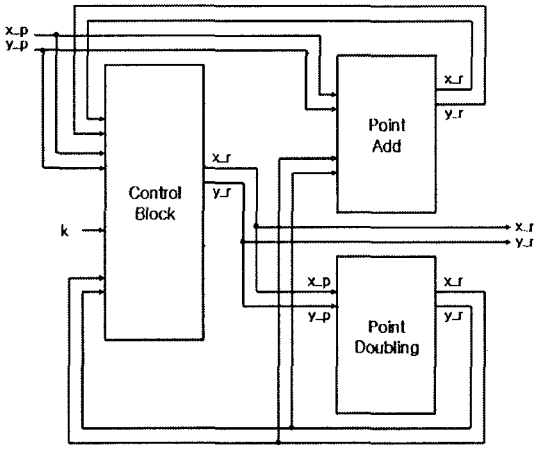


그림 3. 스칼라 곱셈기 구조도
Fig. 3. Architecture of Scalar Multiplication

여기서 k는 비밀키 값인 큰 정수이고 P는 타원곡선상의 한 점이다. kP를 연산하기 위해 덧셈 연산을 k번 반복할 수 있겠지만, 일반적으로는 두 배 연산과 덧셈 연산의 반복된 과정을 통하여 전체 계산시간을 단축하려는

Double-and-Add 방식이 일반적으로 많이 사용되며, 성능을 더욱 개선하기 위해 Quadruple- and-Add 알고리즘도 사용되고 있다. Double- and-And 방식을 적용하여 1회의 k배 연산을 수행할 경우에 m-1번의 동일한 두 점의 합과 최대 m-1번의 서로 다른 두 점의 합 계산만으로 Q를 계산할 수 있다. 보통 상용화되는 타원곡선 암호 시스템의 경우 m의 값이 163 또는 192 인 점을 고려하면 1회의 k 상수배 연산에서 백회 이상의 이배수 연산과 덧셈 연산이 소요되는 것을 예측할 수 있다.

그러므로, 효율적인 타원 곡선 암호 시스템을 구성하려면 하위의 점 덧셈 연산과 이배수 연산 유닛이 고속으로 수행되어야 할 필요가 있다. 우리는 고속의 타원곡선 암호 시스템 구현을 위해 점 덧셈 연산과 이배수 연산에 핵심적으로 사용되는 곱셈기와 나눗셈기를 각각 Serial Cell Array 곱셈기와 Modified Euclid Algorithm을 이용한 나눗셈기를 도입하여 설계하였다.

3.2. Point Add와 Point Doubling 구조기.

타원 곡선상의 점 덧셈과 이배수 연산은 앞의 그림 1에 보이는 바와 같이 하위 레벨에 존재하는 유한체 덧셈, 곱셈 및 나눗셈 연산으로 구성된다. 특히, 유한체 덧셈 연산은 하드웨어적으로 구현할 경우 단순히 XOR 논리로서 쉽게 구현이 되지만, 곱셈이나 나눗셈 연산은 비교적 복잡하고, 계산 시간의 대부분을 소요하기 때문에 빠른 암호 시스템을 위해서는 효율적인 설계가 요구되는 것들이다.

다음 그림 4와 그림 5는 점 덧셈 연산과 이배수 연산기의 내부 구조를 보이고 있다. 점 덧셈 연산기 내부의 Control Block은 다음 식(1)의 계산 과정을 유도하도록 제어를 하며, 이배수 연산기의 Control Block은 식(2)의 계산 과정을 유도한다. 덧셈 연산의 입력으로는 (xp,yp), (xq,yq)의 두 좌표가 사용되며, 결과값은 (xr,yr)에 저장된다. 마찬가지로 곱셈 연산의 입력으로는 (xp,yp)가 사용되며 결과값은 (xr,yr)에 저장된다.

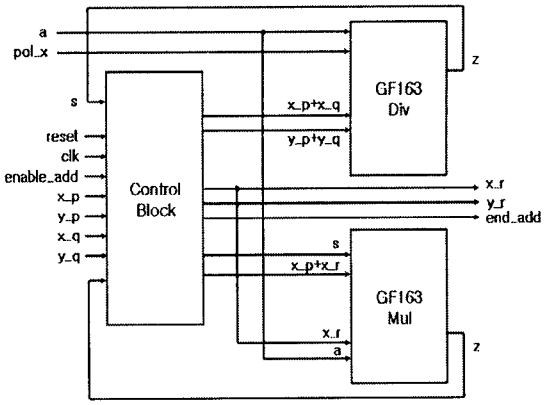


그림 4. Point Add 연산기 구조도
Fig. 4. Architecture of Point Add Arithmetic

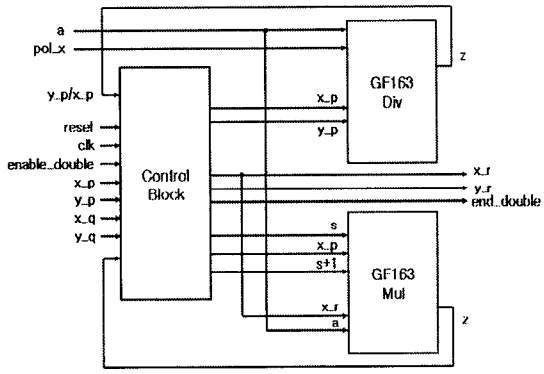


그림 5. Point Doubling 연산기 구조도
Fig. 5. Architecture of Point Doubling Arithmetic

점 덧셈연산에서는 유한체 제곱연산 1번, 유한체 곱셈연산 1번, 이배수 연산에서는 유한체 제곱연산 2번, 유한체 곱셈연산이 1번 필요로 하는 반면 본 논문에서 제안한 구조에서는 그림 4, 그림 5와 같이 순차적인 연산 구조가 이루어짐을 이용하여 Control Block 단에서 곱셈기 하나로 연산이 가능하게 설계하여 제한된 하드웨어 면적에서 게이트 수를 줄일 수가 있었다.

$$x_r = s^2 + s + x_p + x_q + a,$$
$$y_r = s(x_p + x_r) + x_r + y_p,$$
$$s = \left(\frac{y_p + y_q}{x_p + x_q} \right) \quad (1)$$

$$x_r = s^2 + s + a,$$
$$y_r = x_p^2 + (s+1)x_r,$$
$$s = \left(x_p + \frac{y_p}{x_p} \right) \quad (2)$$

3.3. 유한체 모듈의 설계

타원곡선을 이용한 암호화, 복호화의 핵심 연산은 스칼라 곱셈 연산이다. 스칼라 곱셈 연산은 점 덧셈 연산과 두배점 연산의 연속으로 수행되기 때문에 점 덧셈 연산과 두배점 연산의 설계 및 구현 방법이 중요하다. 유한체 $GF(2^m)$ 위의 타원곡선 점 덧셈 연산과 두배점 연산은 유한체 덧셈, 유한체 곱셈, 유한체 나눗셈, 유한체 제곱을 통해 이루어진다. 따라서 유한체 $GF(2^m)$의 산술 연산이 타원곡선 암호 시스템의 전체적인 성능을 좌우한다.

여기서는 점 덧셈 연산과 두배점 연산을 하기 위한 유한체 덧셈기, 유한체 곱셈기, 유한체 나눗셈기의 세부 설계 사항에 대해 설명한다.

3.3.1. 곱셈기

유한체 곱셈기는 유한체 $GF(2^m)$상의 임의의 두 원소의 곱셈을 수행한다. 크게 Serial 유한체 곱셈기와 Cell array 유한체 곱셈기가 존재한다. Serial 유한체 곱셈기는 유한체 곱셈기의 가장 기본적인 구조로 자리 잡아 왔고, 이를 병렬로 처리하기 위해 m배의 자원을 투자하여 m배의 속도를 얻어낸 결과가 Cell array 유한체 곱셈기이다.

그러나 Cell array 곱셈기는 m*m개의 cell을 가져야 하므로, m이 커질수록 실제로 구현하기에는 부담이 크다.

본 논문에서 제안한 고속 연산을 위해 설계한 곱셈기는 기존의 Serial 곱셈기보다 빠르고 회로의 복잡도는 Cell array 곱셈기의 경우인 $O(m^2)$ 보다 매우 낮은 유한체 곱셈기로 serial 곱셈기의 4배정도의 빠른 처리 속도로 작동하는 기능을 가지며 Cell array 곱셈기의 면적 문제를 해결하였다. 다시 말해서 Serial 곱셈기의 작고 느린 구조와 Cell array 유한체 곱셈기의 크고 빠른 구조, 각각 너무나도 극단적인 성능의 이면을 보이는 데서 이를 개선하여 다양한 응용 분야에 따라서 비용이나 성능을 최적화 시켰다. GF 곱셈식은 식 (3)과 같이 정리할 수 있으며, 곱셈은 MSB ➔ LSB 로 수행된다.

$$\begin{aligned} Z(a) &= A(a)\cdot B(a) = A(a)\sum_{i=0}^{m-1} b_i a^i \; Mod(m) \\ &= \sum_{i=0}^{m-1} b_i (a^i A(a)) \; Mod(m) \\ &= (\ldots(A(a)b_{m-1})a + A(a)b_{m-2})a + \ldots) \\ &\quad a \; Mod(m) + A(a)b_0 \; Mod(m) \end{aligned} \quad (3)$$

기존의 Serial 곱셈기에서는 1bit씩 순차적으로 계산되던 것과 달리 제안된 방법에서는 m·n 개의 cell 을 가지고 1 clock 에서 n차수의 연산을 수행하게 된다. 기약다항식 f(x)의 계수는 0인 경우와 1 인 경우의 2 개의 형태만 존재한다. 본 논문에서는 전자의 경우를 Type-Zero cell, 후자의 경우를 Type-One cell이라 정의한다. 그림 6은 차수 n=8인 제안된 Serial-Cell_array 곱셈기의 구조를 나타내며, 그림 7은 Type-Zero와 Type-One 셀의 구조를 나타낸다.

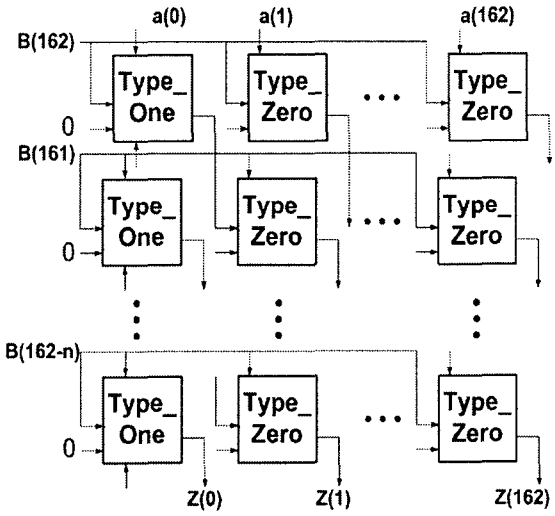
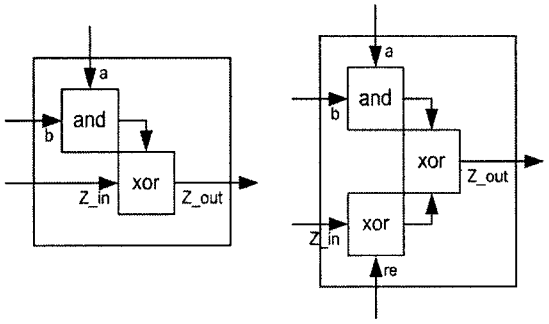


그림 6. Serial-Cell_array 곱셈기의 구조
Fig. 6. Architecture of Serial-Cell Array Multiplication



(a) (b)

그림 7. Type-Zero와 Type-One 셀의 구조: a) Type-Zero 셀, b) Type-One 셀
Fig. 7. Architecture of Cell both Type-Zero and Type-One

Type-One 과 Type-Zero 는 $f(x)$의 계수가 1인 경우와 0인 경우의 연산을 위한 cell을 나타낸다. Type_Zero의 경우는 이전 계산 값에서 reduction 되는 값이 없으므로 Type_One 보다 XOR gate 가 1 개 생략된 구조이다. 제안된 곱셈기를 이용하여 163bit의 유한체 곱셈을 수행할 경우 20 clock과 마지막 소수처리를 위해 1 clock이 필요하게 되므로 총 21 clock에 모든 연산이 완료하게 된다.

이 Serial-Cell_array 유한체 연산기의 장점은 Serial 유한체 곱셈기와 마찬가지로 규칙적인 구조를 가지면서도 추가적인 하드웨어의 부담은 그리 크지 않다는 것이다. Serial 곱셈기에 비해 레지스터의 수는 변동이 없고 하위체 연산 회로의 구현에만 주로 자원이 할당되므로 가격 대 성능 비가 매우 우수하다고 할 수 있다.

3.3.2. 나눗셈기

유한체 산술연산 중에 가장 많은 시간과 복잡한 구조를 필요로 하는 연산기가 바로 나눗셈기이다. 나눗셈 연산에서 역원을 구하는 연산에 추가의 곱셈연산을 하지 않고 나눗셈 연산을 수행하는 Modified 확장 유클리드 알고리즘을 사용하여 역원 연산과 나눗셈 연산에 모두 사용하였다. 유한체 GF(2^m) 상에서 나눗셈 연산은 유한체 곱셈에서와 마찬가지로 기약다항식에 의존하여 수행되는데, A(x)와 B(x)는 유한체 GF(2^m)상의 두 원소이고, F(x)는 차수 m의 기약 다항식이고, Z(x)는 B(x)/A(x) mod F(x)의 결과라고 하면, 각각의 다항식은 식 (4)와 같이 표현되며 계수들은 이진수 0 또는 1이다.

$$
\begin{aligned}
A(x) &= a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0 \\
B(x) &= b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \ldots + b_1x + b_0 \\
F(x) &= x^m + f_{m-1}x^{m-1} + \ldots + f_1x + f_0 \\
Z(x) &= z_{m-1}x^{m-1} + z_{m-2}x^{m-2} + \ldots + z_1x + z_0
\end{aligned}
\tag{4}
$$

알고리즘 1은 [8]에서 제안한 유한체 GF(2^m)상의 나눗셈을 하기 위한 확장 유클리드 알고리즘이다. 본 논문에서는 일반적인 확장 유클리드 알고리즘을 참고하여 유한체 GF(2^m)상의 나눗셈을 위한 새로운 Modified 확장 유클리드 알고리즘을 식 (4)와 같이 제안한다. 식 (4)에서 B(x)는 GF(2^m)상의 0이 아닌 원소이고, W(x) • F(x) + U(x) • A(x) = 1을 만족하는 W(x)와 U(x)를 찾으면, U(x) • A(x) ≡ 1 mod F(x)가 되어 U(x)는 A(x)의 역원이 된다. 이때 U(x)에 초기값 1 대신 B(x)를 대입하면 알고리즘의 종료 후 나눗셈 결과 Z(x) = B(x) / A(X) mod F(x)를 얻을 수 있다.

알고리즘 1. 확장 유클리드 알고리즘

```
Input    : F(x), A(x), B(x)
 Output   : Z(x) : B(x)/A(x) mod F(x)
 Initialize  : R=A(x), S=F(x), G=F(x), U=B(x), Z=0
 ----------------------------------------
 while R ≠ 0
    while R0 == 0 do
       R= R/x
       if U0 == 0 then
          U = U/x mod G;
       else
          U = (U+G)/x;
       end if;
    end while;
    while S0 == 0 do
       S = S/x;
       if V0 == 0 then
          V = V/x mod G;
       else
          V = (V+G)/x;
       end if;
    end while;
    if R >= S then
       (S, R) = (S, R+S); (V, U) = (V, V+U);
    else
       (S, R) = (R+S, S); (V, U) = (V+U, V);
    end if;
 end while;
```

기존의 확장 유클리드 알고리즘에서 식 (4)의 나눗셈 연산 방법을 가지고 입력 레지스터 U의 값에 1 이 아닌 임의의 값이 주어질 경우 U값을 곱하는 과정까지 동시에 이루어지도록 수정하였다. 이 과정으로 인해서 타원곡선 Point 덧셈 연산과정 중 한 번의 곱셈 연산을 줄일 수 있게 된다.

그림 8은 [8]에서 제안한 확장 유클리드 알고리즘에서 각 열의 상태 전이를 나타낸다. 그림 8에서 dummy는 유한체 나눗셈을 하기 위한 자릿수를 맞추는 상태, switch는 제수와 피제수의 값을 바꾸는 상태, normal은 나눗셈의 정상적인 연산을 하는 상태, end는 한 단계가 끝남을 나타내는 상태이다.

Modified 확장 유클리드 알고리즘을 구현함에 있어서 가장 문제가 되는 부분은 알고리즘 내에 존재하는 반복 구문과 스위치 구문이다. 본 논문에서는 [8]의 일반적인 확장 유클리드 알고리즘의 상태 전이를 참고하여 반복 구문과 스위치 구문을 최적화한 상태 전이를 그림 9에 나타낸다.

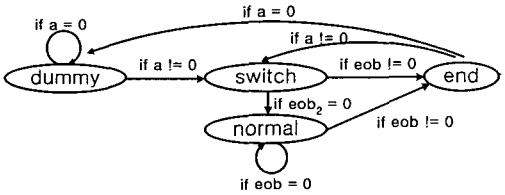


그림 8. 확장 유클리드 알고리즘에서 상태 전이도
Fig. 8. State Shift with Extended Euclid Algorithm

알고리즘 2. Modified 확장 유클리드 알고리즘

```
Input    : A(x) : 역원을 구하고자 하는 수
           B(x) : 역원에 곱하고자 하는 수
           F(x) : 유한체의 모듈러 상수
Output   : Z(x) : B(x)/A(x) mod F(x)
Initialize : U=B(x), S=F(x), F=F(x), Z=0, V=A(x)
           state=0
-----------------------------------------
while(state != 1 )
    if V0 == 0 then
        V = V >> 1;
        if U0 == 1 then
            U = U ^ S;
            U = U >> 1;
        else
            U = U ^ S;
        end if;
    elseif F0 == 0 then
        F = F >> 1;
        if Z0 == 1 then
            Z = Z ^ S;
            Z = Z >> 1;
        else
            Z = Z ^ S;
        end if;
    else
        if V >= F then
            V = V ^ F;
            U = U ^ Z;
        elseif V < F then
            F = V ^ F;
            Z = U ^ Z;
        end if;
        if (V ^ F) == 0 then
            state = 1;
            return Z;
        end if;
    end if;
end while;
```

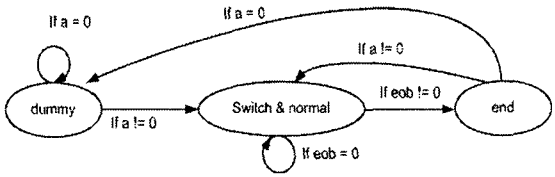


그림 9. Modified 확장 유클리드 알고리즘에서 상태 전이도
Fig. 9 State Shift with Modified Extended Euclid Algorithm

Ⅳ. 구현 및 시뮬레이션 결과

제안된 ECC 프로세서는 Verilog 언어를 사용하여 구현 하였으며, Xilinx ISE 5.2i 툴과 Mentor Modelsim을 이용하여 합성 및, 시뮬레이션을 수행하였다. 타겟 디바이스로는 Xilinx사 FPGA xc8000을 사용하였다. 그림 10은 ECC 프로세서의 합성 결과이다.

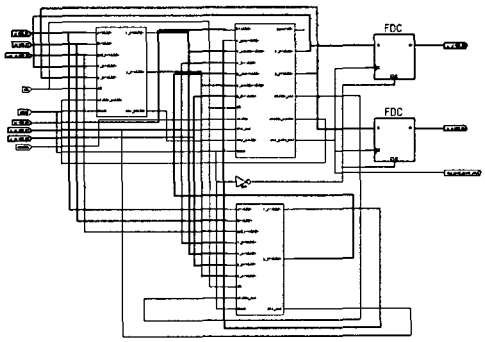


그림 10. ECC 프로세서 합성 결과
Fig. 10. Syntheses Result of ECC Processor

그림 11은 ECC의 암호화 과정에 대한 시뮬레이션 결과이다. 임의의 공개점 Xp, Yp, 타원곡선 계수a, 개인키 k를 입력으로 Xr, Yr의 좌표가 출력 되었고 [5]에 제공하는 결과와 일치함을 확인하였다. 복호화 과정의 검증을 위해서 마찬가지로 [5]에서 제공되는 테스트 벡터를 사용하였으며, 동일한 결과가 얻어짐을 확인하였다.

a : 0000000000000000000000000000000000000000001
pol_x : 8000000000000000000000000000000000000000c9
k : 0000000000000000000000000000000000000000003
Xp : 3f0eba16286a2d57ea0991168d4994637e8343e36
Yp : 0d51fbc6c71a0094fa2cdd545b11c5c0c797324f1
Xr : 634000577f86aa315009d6f9b906691f6edd691fe
Yr : 401a3d20d6c2ec014e6fba5653587bd45dc2230be

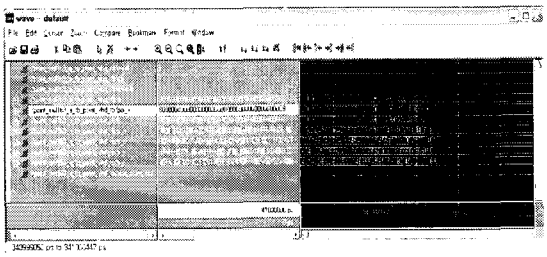


그림 11. 시뮬레이션 결과 화면
Fig. 11. Simulation Result

시뮬레이션 결과 구현된 시스템의 동작 주파수는 약 66 MHz이며, Gates 수는 229,284이고, 약 60Kbps 일 때 초당 380회, 665Kbps 일 때 4,200회 정도의 서명이 가능함을 알 수 있다.

V. 결 론

본 논문에서는 PDA나 휴대폰을 비롯한 휴대용 단말기기나 스마트 카드 등에서 암호 강도를 높이기 위해 설계한 타원곡선 암호 프로세서 설계에 대해 소개하였다. 특히, 전자상거래와 온라인 정보통신이 점차 활성화되고 있는 최근의 기조에서는 고비도의 암호시스템 구현이 필연적으로 요구되고 있기 때문에 앞으로 하드웨어적 암호 시스템 구현은 더욱 활성화되리라 기대된다.

효율적인 타원곡선 암호 하드웨어를 설계하기 위해서 먼저 암호 연산 계층을 분석해보았으며, 암호연산에 핵심적으로 필요한 k상수배 연산기를 하드웨어적으로 설계하였다. k상수배 연산기 내부의 유한체 곱셈기로는 Serial Cell Array 방식을 적용하고, 나눗셈기로는 수정된 확장 유클리드 알고리즘을 적용하여 설계하였다. Mentor 툴을 사용하여 시뮬레이션 해 본 결과, 올바르게 동작하는 것을 확인할 수 있었으며, 초당 수백~수천회의 서명이 가능한 암호처리가 이루어짐을 알 수 있었다. 현재, 덧셈기와 나눗셈기의 제어 논리회로는 더욱 최적화되어 칩의 면적을 줄일 수 있는 여지가 있다. 또한 현재 설계단계에서는 163비트 타원곡선 암호 알고리즘만 지원하는데, 다양한 모드를 지원할 수 있도록 설계를 확장할 필요가 있다.

참고문헌

[1] 이만영외, 현대암호학및응용, 생능출판사, 2002.

[2] T. Elgamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory, 1985.

[3] Tommi Elo, "Lessons Learned on Implementing ECDSA on a Java Smart Card", Proceedings of NordSec2000, Oct. 2000, Reykjavik, Iceland.

[4] 이완복, 노창현, 류대현, "공개키 연산기의 효율적인 통합 설계를 위한 임계 경로 분석", 한국멀티미디어학회 논문지, 제8권, 제1호, pp:79-87, 2005년.

[5] Certicom Research, "SEC2: Recommended Elliptic Curve Cryptography Domain Parameters", 1999.

[6] 문상국, "타원곡선 암호용 프로세서를 위한 고속 VLSI 알고리즘의 연구와 구현", 연세대학교 박사논문, 2001.

[7] D. Hankerson, J. L. Hernandez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," LNCS, vol. 1965, 2001.

[8] Y. Jeong and W. Burleson, "VLSI Array Synthesis for Polynomial GCD Computation and Application to Finite Field Division", IEEE Transact- ions on Circuits and Systems, pp. 891-897, Dec. 1994.

저자소개

이 완 복(Wan-Bok Lee)

2004년 2월 KAIST 전자전산학과 박사
2003년 3월~2007.2 중부대학교 이공대학 교수
2007년 3월~현재 공주대학교 공과대학 게임 디자인학과 교수
※관심분야: Simulation, Computer Game, Information Security, Discrete Event System

김 정 태(Jung-Tae Kim)

2001년 8월 연세대학교 대학원 전자공학과 박사
1991년 8월~1996년 2월 한국전자통신연구원 (ETRI) 선임연구원
2002년 10월~현재 목원대학교 공과대학 정보전자영상공학부 교수
※관심분야 : Optically fed wireless communi cation system design, Information security system design, Network Security, ASIC Design.