Journal of Korea Multimedia Society Vol. 8, No. 1, January 2005(pp. 79-87)

공개키 연산기의 효율적인 통합 설계를 위한 임계 경로 분석

이완복[†], 노창현[††], 류대현[†††]

요 약

공개키 연산기는 전자상거래 환경에서 사용자 인증, 서명 및 키 분배 등에 있어서 매우 중요한 기반 요소이다. 유선망 환경에서는 공인 인증서를 이용한 전자상거래가 이미 활성화되었으나, 무선망 환경에서는 무선 단말기의 제약으로 인하여 아직까지 보편화되지 못하고 있는 실정이다. 본 논문에서는 소프트웨어 프로파일링 기법을 적용하여 공개키 연산기 내부 연산별 부담(Overhead)을 측정하고, 이 정보를 기반으로 통합 설계에 적용할 수 있는 방안을 제시한다. 또한, 통합 시스템의 성능 예측 방안 및 하드웨어와 소프트웨어간의 통신 부담을 줄일 수 있는 방안에 대해서 제시하며, 예제 모델로서 EC-Elgamal 알고리즘의 연산과정을 정량적으로 측정하고 분석하였다.

Critical Path Analysis for Codesign of Public Key Crypto-Systems

Wan-bok Lee[†], Chang-hyun Roh[††], Dae-hyun Ryu[†††]

ABSTRACT

In e-commerce applications, a public key cryptosystem is an important and indispensible element for the basic security operations such as authentication, digital signaturing, and key distribution. In wired network environments, the public key infrastructure certificate, which is based on X.509 specification, has been widely used. On the other hand, it still remains difficult to use the certificate information in wireless network environments due to the inherent limitations of the hand-held devices such as low computational power and short battery life. In this paper, we facilitate a codesign approach by implementing a software public-key cryptosystem and classifying its internal computation overheads quantitatively using a software profiling technique. Moreover, we propose a method to analyze the profiled data and apply it to the problem of software/hardware partitioning in a codesign approach. As an illustrative example, we analyze the computational overheads of an EC-Elfagamal application and examine a critical computational path.

Key words: Public Key(공개키), Codesign(통합설계), EC-Elgamal(타원곡선엘가말암호), ECC(타원곡선), Smart card(스마트카드)

※ 교신저자(Corresponding Author) : 류대현, 주소 : 경기도 군포시 당정동 604-5(435-742), 전화 : 031)450-5228, FAX : 031)450-5172, E-mail : dhryu@hansei.ac.kr
접수일 : 2004년 10월 25일, 완료일 : 2004년 1월 4일
[†] 정회원, 중부대학교 정보보호학과 전임강사
(E-mail : wblee@joongbu.ac.kr)
[††] 정회원, 중부대학교 게임학과 전임강사
(E-mail : chroh@joongbu.ac.kr)
[†††] 정회원, 한세대학교 IT학부 부교수
※ 본 논문은 2004년 한세대학교 교내 연구비 지원에 의하여 연구되었음.

1. 서 론

근래에 초고속 정보통신망이 널리 보급되면서, 증권거래와 인터넷 뱅킹 서비스를 비롯한 각종 전자상거래 활동이 활성화되고 있으며, 휴대폰을 비롯한 무선 단말 환경 하에서도 이러한 서비스가 부분적으로 준비되고 있다. 특히, 상거래 서비스에서는 키 관리와 분배가 용이할 수 있도록 개인키와 비밀키가

분리된 공개키 연산기가 매우 중요한 역할을 하고 있으나, 소요되는 계산량이 많기 때문에 소프트웨어적으로만 구현하는 것이 용이하지 않은 실정이다[1].

이러한 이유로 공개키 연산기를 하드웨어적으로 구현하거나, 하드웨어적인 도움을 통해 그 계산량을 줄이는 방법이 연구되었다[2,3]. 특히, 최근에 CAD Tool과 SoC 설계 기술이 발달함에 따라, 하드웨어와 소프트웨어의 각종 잇점을 최대한 살려서 설계할 수 있는 통합설계 기법이 제안된 바 있는데[4-6], 이러한 통합설계 기술은 설계자의 목적에 따라 시스템의 수행 속도, 소모 전력, 칩의 면적, 요구 메모리양 등의 여러 요소들을 최적화하면서 설계할 수 있기 때문에 매우 유용한 설계 기법이라 할 수 있다. 특히, 데이터 흐름(Data Flow)과정에서 많은 연산이 소요되는 코사인 변환이나 동영상 코덱과 같은 이미지 처리 부분은 하드웨어로 구현하고, 그 이외의 부분은 소프트웨어로 구현할 경우, 통합설계 기법이 매우 효과적으로 적용될 수 있다고 보고 된 바 있다[7].

정보보호 분야에서도 FPGA나 ASIC 으로 암호 연산을 구현한 사례가 있다. 그러나, 이들 접근 방안에서는 암호화 연산 전체를 하드웨어 하나로 실장하여 암호화 성능에만 초점을 두었으며, 게이트 수, 소모 전력, 유연성(Flexibility)등과 같은 여러 부가적인 요소들에 대해 복합적으로 고려하지는 못한 편이다. 본 논문에서는 최근 각광받고 있는 타원곡선 공개키 연산을 먼저 C언어로 구현한 후, 소프트웨어 프로파일링 기술을 이용하여 내부 연산들 간의 태스크 그래프를 생성하고, 그 부담(Overhead)을 연산별로 도출한다. 또한, 각각의 내부 연산들의 호출 횟수와 빈도 및 단위 계산 시간을 고려하여, 하드웨어 소프트웨어 분할 문제에 적용한다. 분석 결과, 타원 곡선 공개키 연산의 경우, 상수배 연산이 전체 소요 시간 중 대부분을 차지한다는 것을 알 수 있었으며, 이 부분을 하드웨어로 구현할 경우 많은 성능 향상이 이루어질 것으로 예측되었다.

휴대폰과 PDA를 비롯한 각종 무선 단말 환경에서 높은 비도의 각종 상거래 서비스를 제공하기 위해서는 공개키 연산을 가속화할 필요가 있다. 본 논문에서 분석한 공개키 연산의 기본 블록별 계산 시간과 호출 횟수의 분석은, 암호 연산을 더욱 잘 이해하고 효율적으로 시스템을 설계하기 위해서 반드시 필요하다. 더욱이 현재 보안 분야에서 각종 인증 수단으로 많이 사용되고 있는 스마트 카드의 경우에는 칩 크기와 소모 전력의 제약으로 말미암아 암호 가속 모듈 전체를 하드웨어로 구현하는 것이 쉽지 않은 실정이다. 이러한 경우 최소한의 내부 모듈만을 선별하여 하드웨어로 구현한다면, 비교적 적은 비용으로 암호 가속 성능, 소모 전력, 칩 면적 등의 다양한 디자인 목적을 동시에 해결할 수 있게 될 것이며, 본 연구의 결과는 그러한 방안에 응용될 수 있다는 측면에서 의의가 크다고 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 타원곡선 공개키 암호 연산의 특징과 연산의 계층적 구조에 대해 설명하고, 3장에서는 소프트웨어 프로파일링을 통하여 EC-Elgamal 암호 연산의 내부 태스크별 소요 계산 시간을 측정해본다. 4장에서는 프로파일링 결과를 통합설계 과정에 활용할 수 있는 방안을 소개하며 5장에서 결론을 맺는다.

2. 타원 곡선 공개키 암호 연산

2.1 타원 곡선 암호 연산의 특징

RSA, ElGamal, 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 소인수분해 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. 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.

현재까지 Java Card 상에서 공개키 연산을 소프트웨어적으로 수행하는 것은 현실적이지 못하다[1]. 아래 표 1에서 제시한 결과를 살펴보면 타원 곡선 암호의 기반이 되는 덧셈과 이배수 연산을 Java Card에서 구현하였을 경우, 모두 9분 이상의 시간이 소요되는 것을 알 수 있다[8]. 이러한 경우, 스칼라 상수 배 연산은 수 시간 단위로 이루어질 수 밖에 없으며, Java card 상에서 타원 곡선 연산을 하는 것은 전혀 실용적이지 못할 수밖에 없다.

2.2 타원 곡선 암호의 연산 계층

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

표 1. 세 종류의 Smart Card 비교 (ECC2-109 알고리즘 기준)

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.

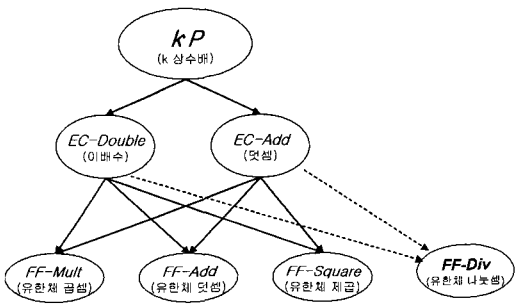


그림 1. 타원 곡선 암호의 연산 계층도

위 연산의 계층도에서 상위 계층 연산을 1회 수행하기 위해서는 하위 계층을 연산을 다수 회 수행하여야 한다. 그러므로, 최 하위 계층의 계산 속도는 전체 연산 속도와 밀접한 관계가 있다. 위 연산 중에서 유한체 나눗셈 모듈은 소프트웨어적으로 구현할 경우 페르마의 정리를 이용하여 유한체 곱셈과 유한체 제곱 모듈을 다수 회 호출하여 계산할 수 있다.

2.3 EC-Elgamal 암호

타원곡선 연산을 이용한 암호 시스템에는 ECDH (Elliptic Curve Diffie-Hellman), EC-Elgamal, EC-Massey-Omura, ECDSA(Elliptic Curve DSA) 등이 있다. 여기서는 EC-Elgamal 알고리즘을 소개하고, 다음 장에서 이 알고리즘을 프로파일링하여 분석하고자 한다. EC-Elgamal 알고리즘은 기존의 Elgamal 알고리즘을 타원곡선 상의 연산을 이용하여 구현한 것으로 그 특성을 간략히 요약하면 아래 표 2에서 보이는 바와 같다.

표 2. Elgamal 알고리즘과 EC-Elgamal 알고리즘의 비교

	Elgamal 암호	EC-Elgamal 암호
공개키	(g, p, $y=g^x \bmod p$)	(G, p, Y=xG)
개인키	(x)	(x)
암호화	(c_1, c_2)=($g^k \bmod p$, $y^k m \bmod p$)	(C_1, C_2)=(kG, kY+M)
복호화	$c_2\ c_1^{-x} \bmod p$	$C_2 - x\ C_1$

(k는 임의의 난수임)

3. 프로파일링을 이용한 임계 경로 분석

공개키 연산기를 프로파일링 하기 위해 먼저 EC-Elgamal 암호 알고리즘을 윈도우즈 Cygwin[9] 환경에서 구현하였다. 여기에서는 이진 타원곡선 연산을 기반으로 암호 시스템을 구축하였다.

3.1 프로파일링을 위한 공개키 연산 테스트 프로그램 작성

공개키 연산기의 세부 연산들의 계산 부담(Overhead)를 측정하기 위한 테스트 프로그램은 만 개의 데이터 블록에 대해 EC-Elgamal 알고리즘을 적용하여 암호화/복호화 하였다. 먼저 구현한 프로그램의 동작성을 확인하기 위해 무작위로 생성된 스트링

에 대해 암호화와 복호화 과정을 연속적으로 적용하여 원문이 복구됨을 확인하였다. 이후에, 암호화 과정과 복호화 과정을 별도로 프로파일링 하기 위해 테스트 프로그램을 암호화용과 복호화용으로 분리하여 작성하였다. 이때 암호 알고리즘에 사용한 파라메터들은 다음과 같다.

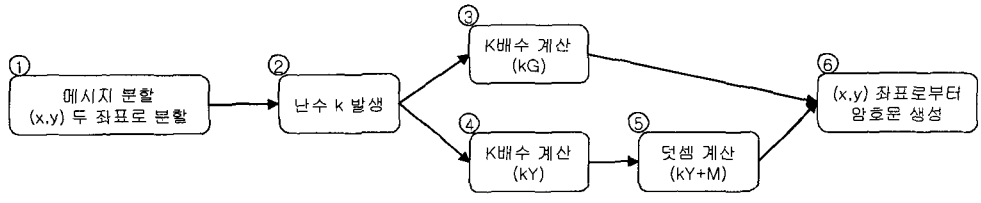
- T(m, f(x), a, b, G, n, h)
- m : 163
- f(x) : $x^{163}+x^7+x^6+x^3+1$
- a=07 B6882CAA EFA84F95 54FF8428 BD88E246 D2782AE2
- b=07 13612DCD DCB40AAB 946BDA29 CA91-F73A F958AFD9
- G(c)=0303 69979697 AB438977 89566789 567F-787A 7876A654
- G(uc)=040369 979697AB 43897789 56678956 7F787A78 76A65400 435EDB42 EFAFB298 9D51FEFC E3C80988 F41FF883
- n=03 FFFFFFFF FFFFFFFF FFFF48AA B6-89C29C A710279B

EC-Elgamal 암호 연산기의 내부 연산 과정은 아래 그림 2와 같다. 하나의 블록을 암호화하기 위해 스칼라 상수 배 연산이 두 번 계산되어야 하며, 이 두 번의 연산은 상호 연관성이 없기 때문에 병렬적으로 계산이 가능하다. 스칼라 상수 배 연산은 다시 하위 계층에 존재하는 타원곡선상의 덧셈과 이배수 연산으로 구성된다. 이러한 덧셈과 이배수 연산은 Affine 또는 Projective 좌표계에서 계산될 수 있다. 테스트 프로그램에서는 계산 속도 측면에서 전반적으로 유리한 Projective 좌표계를 사용하여 구현하였다. Projective 좌표계에서의 덧셈 연산은 그 하위 계층의 유한체 곱셈을 4번, 유한체 제곱을 1번, 유한체 덧셈을 2번 요구한다. 반면에 이배수 연산은 2번의 유한체 곱셈, 4번의 유한체 제곱, 1번의 유한체 덧셈을 계산하여야 한다.

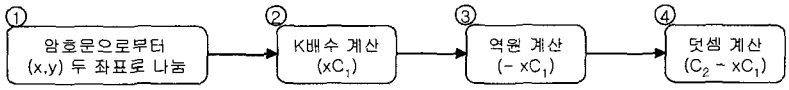
3.2 타원 곡선 공개키 연산기의 프로파일링

다음 표 3은 위 테스트 프로그램을 실행한 후 소요되는 시간을 위 그림 1에서 분류한 기본 연산별로 측정한 것이다. 암호화 과정 중 k배수 연산은 각각 kG와 kY를 계산하기 위해서 한 개의 데이터 블록을 암호화하기 위해 두 번 계산이 된다. 테스트 프로그램에서는 두 번의 k배수 연산이 동일한 함수를 통하여 연산되기 때문에 아래 표에서는 k배수 연산에 소요된 전체 시간을 측정한 후 절반으로 나누어서 표기하였다. 또한 각 함수별 호출 횟수도 측정하였는데, 이것은 하드웨어-소프트웨어 분리과정에 고려되어야 할 중요한 파라메터들 중 하나이다. 암호화 과정에 소요된 총 시간은 10.8 초였으며, 복호화 과정에 소요된 시간은 6.9 초였다. 암호화 과정에 더 많은 시간이 소요된 것은 그림 1에서, 암호화 연산이 복호화 과정에 비하여 복잡하며, 시간을 많이 소요하는 k 상수배 연산을 두 배 정도 많이 계산하기 때문이다. 다음 표 3, 4, 5에서 단위 소요 시간은 전체 소요시간을 각 호출 횟수로 나누어서 계산한 값이다.

연산과정을 프로파일링한 결과, 스칼라 상수배 연산이 연산 시간의 대부분을 차지하는 것을 알 수 있으며, 각 연산별 호출 횟수를 측정하면, 그림 1의 하



a) 암호화 과정



b) 복호화 과정

그림 2. EC-Elgamal 암호 연산의 태스크 그래프

표 3. 내부 연산 태스크별 계산 소요 시간

	태스크 이름	단위 소요시간(usec)	호출 횟수	소요 시간(sec)	비율(%)
암호화	1) 메시지 분할	1	10000	0.01	0.1
	2) 난수 k 발생	1	10000	0.01	0.1
	3) k배수 계산	527	10000	5.27	48.9
	4) k배수 계산	527	10000	5.27	48.9
	5) 덧셈 계산	1	10000	0.01	0.1
	6) 암호문 생성	21	10000	0.21	1.9
복호화	1) 두 좌표 나눔	154	10000	1.54	22.3
	2) k배수 계산	496	10000	4.96	71.9
	3) 역원 계산	19.5	20000	0.39	5.7
	4) 덧셈 계산	1	10000	0.01	0.1

표 4. 암호화 과정 내부 하위 레벨 연산의 프로파일링

	세부 연산	단위 소요시간(usec)	호출 횟수	소요 시간(sec)	비율(%)
k상수배	이배수 연산	1.6	3220161	5.13	51
	덧셈 연산	1.5	3220161	4.92	49
유한체 연산	유한체 덧셈	0.1	89358984	4.74	49.5
	유한체 곱셈	0.2	19609071	3.12	32.6
	유한체 제곱	0.1	16160970	1.71	17.9

표 5. 복호화 과정 내부 하위 레벨 연산의 프로파일링

	세부 연산	단위 소요시간(usec)	호출 횟수	소요 시간(sec)	비율(%)
k상수배	이배수 연산	1.5	161048	2.41	51
	덧셈 연산	1.4	1610483	2.32	49
유한체 연산	유한체 덧셈	0.0	6525361	3.26	54
	유한체 곱셈	0.1	987293	1.48	24.5
	유한체 제곱	0.1	1296258	1.30	21.5

위 계층 연산일수록 호출횟수가 증가된다는 것을 확인할 수 있었다. 또한 유한체 곱셈이나 유한체 제곱 연산은 내부적으로 유한체 덧셈 연산을 빈번하게 호출하기 때문에, 유한체 덧셈 연산이 가장 많이 호출되는 것으로 예측할 수 있는데, 측정 결과, 유한체 덧셈 연산은 유한체 곱셈이나 제곱 연산보다 5~6배 가량 빈번하게 호출됨을 알 수 있었으며, 전체 속도에 가장 중요한 영향을 미친다는 것을 확인할 수 있었다. 다음 장에서는 소프트웨어 프로파일링 결과를 분석하여 하드웨어-소프트웨어 분할 문제에 그 정보를 활용할 수 있는 방안을 소개한다.

4. 통합 설계 틀에서의 프로파일링 결과 활용 방안

Smart card, PC Card, 또는 휴대폰에 내장될 암호 연산기는 그 특성상 고속 및 저전력으로 동작될 필요가 있다. 일반적으로 시스템의 모든 요소를 하드웨어로 구현할 경우, 우수한 성능을 보장할 수 있으나 생산 비용이 비싸지며, 전력소모도 많아지는 문제점이 발생한다. 따라서, 시스템의 구성 요소들을 분석하여 적절한 수준의 하드웨어와 소프트웨어로 분할하여 개발하는 통합설계 방안이 매우 바람직할 수 있다. 그러나, 통합 설계 방안은 그 설계 과정이 복잡하고 어려워지는 경향이 있다. 통합설계 과정에서는 먼저 시스템의 Control Data Flow Graph(CDFG)를 분석한 후, 개별적인 작업의 단위를 하드웨어 또는 소프트웨어로 분할 및 매핑하게 된다. 이러한 매핑 과정 이후에, 전체 시스템이 수행 속도, 소비 전력, 칩 면적 등의 특성들을 만족하는지 살펴보기 위해 하드웨어 소프트웨어 통합시뮬레이션을 통하여 디자인의 적절성을 평가하는 과정을 거치게 된다. 그러나, 통합 시뮬레이션 과정에는 시간과 노력이 매우 많이 소요되며, 하드웨어와 소프트웨어로 분할할 수 있는 경우의 수가 매우 크기 때문에 효율적인 설계의 대안을 찾는 것은 어려운 작업이 된다.

특히, 통합설계에서 하드웨어-소프트웨어 분할 문제는 시스템의 여러 메트릭들-소비 전력, 계산 속도, 칩 면적, 유연성 등-에 미치는 영향이 크기 때문에 분할 방법에 대한 연구가 많이 보고된 바 있다[10]. 대표적으로 VULCAN[11] 시스템에서는 HardwareC 언어로 표현된 시스템 모델로부터 CDFG를 추출하고 greedy 알고리즘을 적용하여 성능에 별 영향을 미치지 않는 연산들을 표준형 프로세서로 옮겨서 실행하는 방안을 채택하고 있다. 또한, COSYMA 시스템에서는 시스템을 소프트웨어로 표현하고 특정 블록을 소프트웨어에서 하드웨어로 옮김으로써 적절한 디자인 대안을 찾는다[12]. 그러나, 이들 기존 연구들에서 분할 과정을 주로 상위 레벨에서 수행하며, 각 블록이 실행되는데 소요되는 시간에 대한 정보나 호출 빈도에 대해서는 구체적인 방안을 제시하지 못하고 있다. 본 논문에서는 소프트웨어 프로파일링 기법을 적용하여 이러한 정보를 얻어낼 수 있는 방법에 대해 논의한다.

4.1 하드웨어-소프트웨어 분할 및 매핑 과정에서의 적용

소프트웨어로만 구성된 시스템을 하드웨어와 소프트웨어로 통합 설계할 경우, 전용의 하드웨어를 통해 일부 태스크가 가속화될 수 있지만, 하드웨어 소프트웨어간의 통신 부담이 새로이 생기게 된다. 또한, 이러한 통신부담은 하드웨어와 소프트웨어의 통신 방식, 버스의 대역폭 등에 따라 달라지기 마련이다. 하드웨어 추가로 인해 많은 속도 증가를 이루려면, 통신 부담을 줄여야 한다. 통신 부담은, 통신 빈도와 1회 통신시에 전송하게 되는 데이터의 양에 비례하는 특성이 있다. 그러므로, 프로파일링 결과는 하드웨어-소프트웨어 분할로 인해 발생하는 통신 부담을 유추하는데 매우 요긴하게 사용될 수 있다.

표 3, 4, 5를 분석해보면, 상수배 연산보다는 이배수 연산과 덧셈 연산이 300배 정도 많이 호출되는 것을 알 수 있으며, 또한 유한체상의 연산들은 이배수 연산이나 덧셈 연산보다 5~20 여배 더 많이 호출됨을 알 수 있다. 그러므로, 통신 부담은 상위 계층의 연산을 하드웨어로 구현할 경우보다 하위 계층의 연산을 하드웨어로 구현할 경우 급속히 커질 것을 예측할 수 있다. 현재 보편적으로 사용되어 지고 있는 PCI 버스의 대역폭은 32bit×33MHz 이므로 1Gbit/sec가 된다. 128비트 블록을 하나 전송하기에는 최소 128usec이 소요된다. 그러므로, 앞의 타겟 플랫폼에서는 k배수 연산 아래 계층의 연산을 하드웨어로 분할할 경우에는 소프트웨어로 구현할 경우보다 느려지게 계산될 수밖에 없음을 알 수 있다. 이러한 결과는 위 프로파일링 실험이 수행 속도가 비교적 빠른 펜티엄4 프로세서상에서 이루어졌기 때문이다. 만일, Arm7이나 8051과 같이 비교적 느린 프로세서에서는 타원곡선상의 이배수 연산이나 덧셈 연산들의 단위 계산 소요시간이 훨씬 클 것이기 때문에 하위 레벨의 연산을 하드웨어로 분할한다고 하더라도 버스의 대역폭이 크다면 가속화 될 수 있을 것이다. Linux 운영체제에서 프로세서의 계산 성능을 측정하는 단위인 BogoMips를 기준으로 삼을 경우, Arm7 프로세서는 펜티엄4 프로세서보다 계산 성능이 400배 이상 떨어진다. 그러므로 앞의 프로파일링 결과를 고려해볼 때, Arm7 프로세서가 상기의 이배수 연산이나 덧셈 연산을 수행하기 위해서는 600 usec 이상의 시간을 소요하게 될 것이며, 만일 사용하는 버스의 전송 속도가 PCI 수준이라면 하드웨어로 구현하는 것이 바람직하다. 요컨대, 소프트웨어 프로파일링을 통하여 태스크 그래프 상의 한 모듈에서 소요하는 단위 계산 시간을 측정한 후에, 그 모듈이 호출되는 빈도를 고려하여 하드웨어-소프트웨어 분할 문제에 활용할 수 있게 된다.

4.2 하드웨어-소프트웨어 Interface 통신 부담의 축소화 방안

하드웨어 소프트웨어간의 통신 부담은 소프트웨어가 하드웨어를 제어하기 위해서 디바이스 드라이버, 운영체제의 시스템 콜 등과 같은 여러 계층의 서비스를 거치기 때문인데, 하드웨어 제어 방안을 경량화시켜서 극복할 수 있다. 그러한 방안 중 두 가지를 소개한다.

첫 번째 방안은, 마이크로프로세서의 명령어 레벨에서 하드웨어를 구동하는 것이다. 현재 개발된 프로세서 중에서 Xtensa, ARC-tangent, Jazz 등은 기본적인 마이크로 프로세서 코어 이외에 기타 명령어 집합을 제공한다. 이들은 응용에 따라 다르게 구성되거나 확장될 수 있기 때문에 재구성 가능한 프로세서(Reconfigurable processor) 라고 부른다. 이러한 프로세서들은 임베디드 시스템이나, 특정 목적을 위해 명령어 집합을 재구성할 수 있기 때문에, 특정 어플

리케이션에서 좋은 성능을 보여줄 수 있다. 이 중에서 Tensilica에서 제작한 Xtensa의 경우, 범용적인 DES 알고리즘의 가속을 위해 특정 명령들을 제작한 결과, 40배 이상의 성능 향상이 가능함을 보고하고 있다. Tensilica에서는 Tensilica Instruction Extension (TIE)를 이용하여 DES 블록 암호 알고리즘을 가속화한 경우를 발표하고 있다. DES 알고리즘은 매우 대중적인 암호 알고리즘이지만, 일반 프로세서에서 소프트웨어적으로 가속화할 경우, 다음과 같은 이유로 인해 높은 성능을 얻기가 어려운 편이다. 3DES 알고리즘은 내부적으로 매우 빈번한 비트 치환을 사용하고 있다. 이러한 연산은 하드웨어적으로는 매우 간단히 구현 가능하지만, 소프트웨어로 구현할 경우에는 순차적으로 처리할 수 없기 때문에 느려진다. 또한 이 알고리즘에서는 28비트 블록을 기준으로 순환 연산을 사용하는데, 이것 또한 일반 마이크로프로세서들은 16비트 또는 32비트의 레지스터 크기를 갖기 때문에 비효율적으로 계산되어진다. 이외에도 비트 패킹, 언패킹과 테이블 룩업 등의 암호 연산들은 소프트웨어로 구현할 경우, 하드웨어로 구현할 경우보다 현저히 속도가 느려질 수 밖에 없는 요소들이 된다.

두 번째 방안은, 프로세서가 지원하는 고속의 로컬 버스에 메모리 매핑이나 DMA 방식으로 하드웨어를 연동시키는 것이다. 이러한 플랫폼을 지원하는 프로세서로는 Altera사의 Excalibur 칩이 대표적이다. Excalibur칩은 ARM9 프로세서와 100만게이트 로직을 하나의 칩에 집적하였기 때문에 내부의 AMBA 버스와 연동되는 로직을 설계하여 특정 메모리에 매핑되도록 할 수 있기 때문에 통신 인터페이스 속도를 매우 빠르게 설계할 수 있다.

4.3 통합 시스템의 성능 예측을 위한 활용

위 프로파일링 데이터는 하드웨어-소프트웨어 분할이 완료된 시스템의 성능 예측에도 직접적으로 사용될 수 있다. 전체 시스템의 태스크 집합을 $K=K_1 \cup K_2$ 라고 하자. 이중 K_1 내의 원소들은 하드웨어-소프트웨어 분할 과정에서 소프트웨어로 매핑되는 태스크들이며, K_2 내의 원소들은 하드웨어로 매핑되는 태스크들이라고 하자. 이때 소프트웨어로 시스템을 구현할 경우, 전체 소요되는 시간은 다음과 같다. $k_{i,sw}$는 k_i 태스크를 소프트웨어로 구현하였을 경우 소요되는 시간이며, $k_{i,hw}$는 하드웨어로 구현하였을 경우 소요되는 시간을 의미한다.

$$t_{sw} = \sum_{k_i \in K_1} k_{i,sw} + \sum_{k_i \in K_2} k_{i,sw} \quad (1)$$

기존 COSYMA 시스템[12]의 연구에서, 시스템의 전체 수행 시간은 소프트웨어 블록과 하드웨어 블록의 수행 시간과 상호 통신에 소요된 시간의 합으로 계산하였다. 여기에서는 통신에 소요되는 시간을 더욱 정확하게 계산할 수 있도록 수식을 구체화하였다. 즉, 전체 시스템에서 소요되는 시간을 계산하면 다음과 같다.

$$t_{co} = \sum_{k_i \in K_1} k_{i,sw} + \sum_{k_i \in K_2} k_{i,hw} + \sum_{i,k_i \in K_2} W_i * R_i * U_i \quad (2)$$

여기서, R_i는 i번째 하드웨어 요소의 호출되는 회수를 뜻하며, U_i는 단위 계산 소요시간, 그리고 W_i는 가중치를 의미한다. 수식 (2)의 첫 번째 항목인 소프트웨어적 소요 시간들은 앞에서 보인 바와 같이 소프트웨어 프로파일링 기술을 적용하여 구할 수 있으며, 하드웨어적 소요 시간들은 해당 IP내에 제공되는 성능 관련 정보나 Cycle Count 등을 고려하여 계산할 수 있다. 마지막 항목은 하드웨어-소프트웨어 간의 통신에 소요되는 비용인데, 이 비용은 기본적으로 통신 빈도와 통신량의 데이터 크기에 의해 결정된다. 프로파일링 결과는 이러한 통신부담을 예측하는데 정량적으로 사용될 수 있다.

예를 들어, 본 논문에서 검토한 EC-Elgamal 타원곡선 암호 연산기에서는, 하위 계층의 연산일수록 그 호출 횟수가 증가함을 프로파일링을 통하여 확인할 수 있었으며 그 횟수를 정량적으로 측정할 수 있었다. 특히, 계층도에서 하위의 연산일수록 그 호출횟수가 기하급수적으로 증가하므로 하위 레벨 모듈의 단위 계산 시간이 전체 계산 시간에 큰 영향을 미칠 것으로 파악할 수 있다. 그러나 버스의 통신 속도가 느린 경우에는 빈번한 하드웨어 모듈 호출로 인해 전체 계산속도가 느려질 수 있다.

5. 결 론

본 논문에서는 소프트웨어 프로파일링 방법을 통하여 EC-Elgamal 공개키 연산기의 내부 연산별 부담을 측정하고 분석하였다. 또한 측정된 데이터를 기

반으로 통합 설계시 하드웨어-소프트웨어간의 분할 및 매핑 문제와 성능 예측 과정에 적용할 수 있는 방안을 소개하였다. 통합설계는 비교적 저렴한 하드웨어 가격으로 고성능을 발휘할 수 있는 시스템을 구성하는데 효과적일 수 있다. 그러나, 하드웨어와 소프트웨어간의 통신 부담이 클 경우에는 오히려 소프트웨어만으로 구성된 시스템보다 느릴 수 있기 때문에 주의가 필요하다. 이러한 문제점을 극복하기 위해 본 논문에서 소개하는 방안은 참조될 수 있다. 특히, EC-Elgamal 연산기의 프로파일링 실험 결과, 스칼라 곱셈 연산이 계산력을 가장 많이 소모하는 것으로 나타났다. 그러므로 스칼라 곱셈 연산이 핵심적인 역할을 하는 기타 타원 곡선 암호 연산기의 경우에서도, 본 논문에서 소개한 방안이 효율적으로 적용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] T. Elo, "Lessons Learned on Implementing ECDSA on a Java Smart Card", Proceedings of NordSec2000, Oct. 2000.

[2] J. Goodman and A.P. Chandrakasan, "An Energy-efficient Reconfigurable Public-key Cryptography Processor", IEEE Journal of Solid-State Circuits, Vol.36, No.11, 2001.

[3] L. Batina, G. Bruin-Muurling and S.B. Ors, "Flexible Hardware Design for RSA and Elliptic Curve Cryptosystems," Okamoto (Ed.): CT-RSA 2004, LNCS 2964, Springer-Verlag, pp. 250-263, 2004.

[4] S. Janssens, J. Thomas, W. Borremans, P. Gijsels, I. Verbauwhede, F. Vercauteren, B. Preneel and J. Vandewalle, "Hardware/software Co-design of an Elliptic Curve Public-key Cryptosystem", 2001 IEEE Workshop on Signal Processing Systems (SiPS 2001), pp.209-216, Sep. 2001.

[5] 조성제, 권용진, "타원곡선 암호시스템을 위한 기저체 연산기의 FPGA 구현", '00 대한전자공학회 추계종합학술대회 논문집, pp.148-151, 2000.

[6] 신형철, "하드웨어-소프트웨어 통합 설계 기술", 대한전자공학회 논문지, 제24권, 제12호, pp. 69-78, 1997.

[7] W. Sung, M. Oh, C. Im and S. Ha, "Demonstration of Hardware Software Codesign Workflow in PeaCE", Proceedings of International Conference on VLSI and CAD, October 1997.

[8] I.Z. Berta et. al., "Implementing Elliptic Curve Cryptography on PC and Smart Card", Per. Pol. Electrical Engineering, Vol.46, No.1-2, pp.47-73, 2002.

[9] Cygwin Project Homepage, Available at: http://www.cygwin.com/

[10] 김남훈, 신현철, "하드웨어-소프트웨어 통합설계에서의 새로운 분할 방법", 대한전자공학회 논문지, 제35권 C편 5호, pp.1-10, 1998.

[11] R. Gupta and G. De Micheli, "System-level Synthesis Using-Reprogrammable Component", Proc. of EDAC, pp.2-7, 1992.

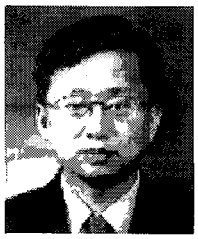
[12] R. Ernst, J. Henkel, and T. Benner, "Hardware-Software Cosynthesis for Microcontrollers", IEEE Design & Test of Computers, Vol.10, pp. 64-75, Dec. 1993.

이 완 복

1993년 2월 한국과학기술원 전기 및 전자공학과 학사 졸업
1995년 2월 한국과학기술원 전기 및 전자공학과 석사
2004년 2월 한국과학기술원 전자전산학과 박사
2000년 3월~2003년 2월 (주)시큐어넥서스 개발팀장
2003년 3월~현재 중부대학교 정보보호학과 전임강사
관심분야 : 정보보호, 컴퓨터 네트워크, 이산사건 시스템, 시뮬레이션

노 창 현

1993년 2월 한국과학기술원 원자력공학과 학사 졸업
1995년 2월 한국과학기술원 원자력공학과 석사
2001년 2월 한국과학기술원 원자력공학과 박사
2002년 8월~2003년 6월 (주)에스포라 연구소장
2002년 3월~현재 중부대학교 게임학과 전임강사
관심분야 : 컴퓨터게임, VR, Interactive Media



류 대 현

1983년 2월 부산대학교 전기기계 공학과 졸업
1985년 2월 부산대학교 전자공학과 석사
1997년 2월 부산대학교 전자공학과 박사
1987년 2월~1998년 2월 한국전자통신연구원 선임연구원
1998년 3월~현재 한세대학교 IT학부 부교수
관심분야 : 정보보호, 영상처리, 통신공학