2003년 대한전자공학회 하계종합학술대회 제26권 제1호

대칭 및 비대칭 암호화 알고리즘 가속을 위한 명령어 집합 구조의 설계

김일관, 최 린
고려대학교 전자컴퓨터공학과

Design of Instruction Set for accelerating symmetric and asymmetric ciphers

Ilkwan Kim, Lynn Choi
The department of electronics and computer engineering
Korea University

Abstract

상거래와 통신을 위한 주된 매개체로써 등장한 인터넷뿐 아니라 새로이 대두되는 다양한 유무선 네트워크 환경, 그리고 정보 저장에 있어서 암호화 알고리즘은 보안의 중요한 요소로 자리 잡고 있다.

본 논문에서는 대칭 및 비대칭 암호화 알고리즘을 가속시키기 위한 암호화 프로세서의 명령어와 해당 Functional Unit 을 제안하였다. 현재 암호화 알고리즘을 가속시키기 위한 방법으로 사용되는 주문형 반도체(ASIC)는 알고리즘 가속 속도는 빠르지만, 새로운 암호화 알고리즘을 지원할 수가 없고, 지원하는 알고리즘을 사용하지 않는 경우 비효율성을 야기한다. 또한 범용 프로세서는 다양하고 새로운 암호화 알고리즘을 지원할 수 있지만 암/복호화 가속 속도가 느리다. 이는 암호화 알고리즘이 범용 프로세서에서는 지원하지 연산을 주로 사용하기 때문이다. 따라서 이 논문에서는 대칭 및 비대칭 암호화 알고리즘의 주된 연산을 분석하고, 각각의 연산을 가속시키기 위한 명령어 집합, 그리고 해당하는 Functional Unit 을 제안하여 Programmable 한 암호화 프로세서를 설계하기 위한 토대를 마련한다.

I. 서론

암호화 알고리즘은 Hashing 알고리즘과 같이 쓰여 기밀성과 실체 인증, 그리고 메시지의 무결성을 제공하는 보안의 핵심 요소로 자리 잡고 있다. 하지만 암호화 알고리즘을 자주 처리하는 시스템에서의 연산은 기존의 개인용 컴퓨터나 일반 컴퓨팅 시스템에서 이용되는 것과는 다른 성격을 지니고 있기 때문에 범용 마이크로프로세서에서의 암호화 알고리즘의 처리가 상당히 비효율적일 수 밖에 없다. 이는 암호화 알고리즘이 근본적으로 마이크로프로세서에서 구현하기 어렵기 때문이 아니라 암호화 알고리즘에서 자주 사용되는 연산에 대한 연산 장치(Functional Unit)와 이를 지원하는 명령어 집합이 프로세서에 정의되어 있지 않기 때문이다.

이러한 문제점을 극복하고자 Programmable 한 마이크로 프로세서에서 암호화 알고리즘을 효과적으로 가속시킬 수 있는 여러 명령어와 해당 Functional Unit 이 이전 연구에서 제안 되었다. Zhiie Shi 와 Ruby B. Lee 는 동/정적으로 명시되는 n 비트 Permutation 을 가속시키기 위해 각각 PPER3R/GPR 명령어를 제안하였으며, Jerome Burke et al 은 IDEA 에서 사용하는 16 비트 Modular Multiplication 을 위한 MULMOD 명령어, Rotation 과 XOR 를 연이어 수행하는 명령어들, 대입(Substitution)을

이 논문은 2002 년도 한국학술 지홍재단의 지원에 의해 연구되었음(KRF-2002-003-D00266)

수행하기 위한 SBOX 와 SBOXSYNC 명령어를 제안하였다.[3][1] Lisa Wu et al 은 기존의 범용 프로세서의 명령어들을 실행 시간에 따라 tiny, short, long 으로 분류하고 tiny & short, short & short 명령어들을 합성한 3 개의 Source 레지스터를 참조하는 명령어 집합들을 제안하고 Jerome Burke et al 에서 제안한 것과 함께 5-stage 파이프라인된 4-wide VLIW 아키텍쳐를 적용한 암호화 co-프로세서를 제안하였다.[2]

하지만 이전의 연구들이 대칭형 알고리즘을 가속시키기 위한 명령어 위주이므로 실제로 클락 사이클의 많은 부분을 차지하지만 긴 Operand 에 대한 Modular 연산이 많은 비대칭 암호화 알고리즘은 하드웨어적으로 전혀 가속시키기 힘들다. 또한 기존의 대칭형 알고리즘 가속을 위한 명령어 조차도 이들 명령어들에게는 포함되지 않는 부분이 존재한다. 따라서 대칭 및 비대칭 알고리즘의 주요 연산들을 분석하여 각각의 연산을 가속시키기 위한 명령어와 해당 Functional Unit 들을 제안한다.

2.1 절에서는 대칭 암호화 알고리즘의 주요 연산, 2.2 절에서는 비대칭 암호화 알고리즘들을 분석한다. 3.1 절에서는 대칭 알고리즘을 가속시키기 위한 추가 명령어와 해당 Functional Unit, 3.2 절에서는 비대칭 암호화 알고리즘을 가속하기 위한 명령어와 Functional Unit 를 제안하고 4 장에서는 이들과 관련된 향후 연구 방향을 제시하고 5 장에서는 결론을 짓는다.

II. 암호화 알고리즘 연산 분석

기존 [1] 의 대칭형 암호화 알고리즘의 커널은 Arithmetic, Logic, Rotates, Multiplies, Substitions, Permutes, Load/Store, Control 의 8 가지 카테고리로 나누어 진다.[1] 이중 대부분의 연산을 위한 명령어는 기존 연구에서 제안 되었다. 하지만 최근에 AES(Advanced Encyryption Standard)의 후보[2]로 제안된 알고리즘들은 기존의 대칭형 알고리즘에 비교해서 다음과 같은 새로운 특징을 가지고 있다.[5] 정수형 Modular Multiplication (IDEA 에서 사용하는 modulus 와는 다르지만 새로운 알고리즘에서 채택하고 있는 정수형 Modular Multiplication 연산은 Modulus 가 2^{32} 이기 때문에 기존의 Multiplication 의 하위 32 비트에 해당하므로 특별한 명령어가 필요가 없다.), GF(2^8) field 에서의 Modular Multiplication, Endian 변환, 다른 sub-block 사이 byte-level shift 가 새롭게 사용되고 있으며, 비트단위의 Permutation 은 거의 사용되지 않고 있는 것이다. 이는 기존의 범용 프로세서에서도 효율적으로 암호/복호화 할 수 있으면서도 대칭 암호화 알고리즘의 매우 중요한 특성인 Confusion 과 Diffusion 요소를 더 강화시키기 위함이다.

반면 비대칭형 암호화 알고리즘은, 각 알고리즘이 기초로 하는 수학적인 문제를 해결하기 위한 어려움 때문에 그들의 보안을 유지할 수 있다. [6] 이들은 3 가지 카테고리로 나눌 수 있는데 이는 [표 1]에 요약하였다.

분류	예
IFP[3]	RSA, Robin-Williams
DLP[4]	DSA(U.S.), Diffie-Hellman KE, ELGamal, Nyberg-Rueppel
ECDLP[5]	ECDSA, ECC anlalogs of Diffie-Hellman KE, ECC versions of ELGamal & Nyberg-Rueppel

표 1. 비대칭 알고리즘의 수학적 분류

[표 1]에서 IFP 와 DLP 와 관련된 비대칭 알고리즘은 암호/복호화시 긴 Operand 의 Integer Modular Exponentiation 을 집중적으로 사용하기 때문에 이 계산이 성능에 중대한 영향을 미친다. 한편 ECDLP 와 관련된 알고리즘은 암호/복호화시 Elliptic Curve 위의 기하학적으로 정의된 점들간 Addition(또는 Doubling)을 집중적으로 사용되는 데, 이 기하학적인 점들간의 Addition 은 GF(p) 또는 GF(2^m) 필드에서 정의되는 긴 operand 의 Modular Multiplication 과 Modular Addition 그리고 Multiplicative Inverse Modulo 으로 구성되어 있다.

더불어 키 생성시 난수 발생, 소수 생성, *Divisibility* 여부, 상대 소수 여부 같은 연산이 사용되는데 이중 난수 발생 연산은 모든 비대칭 알고리즘에서 사용된다.

III. 대칭 및 비대칭 알고리즘을 위한 명령어 집합과 Functional Unit

이 장에서는 앞장에서 분석한 대칭 및 비대칭 알고

[1] DES(3DES), RC4,RC5, IDEA, Blowfish,CAST-128
[2] Rijndael - AES, RC6, Twofish, Serpent, MARS 등
[3] IFP (Integer Factorization Problem)
[4] DLP (Discrete Logarithm Problem)
[5] ECDLP (Elliptic Curve Discrete Logarithm Problem)

리즘의 주요 연산을 토대로 새로이 필요한 명령어들을 제시하고 해당 명령어의 연산 방법 또는 Functional Unit 을 제안한다.

3.1 대칭 알고리즘 가속을 위한 명령어 집합

3.1.1 EndCov 명령어

EndCov 명령어는 Big-endian 으로 표현된 워드를 Little-endian 으로 표현된 워드로 변환(역도 마찬가지)하는 명령어다. 네트워크에서는 Big-endian 으로 데이터를 전송하는 것이 표준이지만, little-endian 으로 데이터를 사용하는 많은 범용 프로세서에서 효율성을 높이기 위해 새로이 정의되는 많은 대칭 알고리즘들이 little-endian convention 을 요구하고 있기 때문에 endian 변환이 자주 발생할 수 있다. 명령어 셋과 연산은 [그림 1]과 같다.

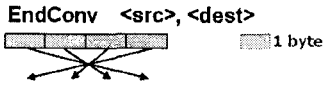


그림 1 EndConv 명령어와 동작

3.1.2 ByteExch 명령어

ByteExch 명령어는 다른 워드간 바이트 단위 위치 이동 또는 한 바이트 교환을 위해 정의 되었다. AES 의 경우 데이터 매트릭스(State)에서 한 행에 대해 바이트 열이 Shift 되는 연산이 정의되었는데, 이는 다른 데이터 워드간 바이트 이동을 의미한다. 명령어 셋과 연산은 [그림 2]와 같다. 여기서 x (1bit)로 0 이면 <src>와 <dest> 레지스터 사이에 y(2bits)번째 바이트를 서로 교환하고, 1 이면 <dest>레지스터의 z(2bits)번째 바이트를 tmp 에 저장하고 <src>레지스터의 y 번째 바이트를 <dest>레지스터의 z 번째 바이트에 저장함을 의미한다. <src>와 <dest>이 같을 경우 tmp 에 저장된 바이트를 <dest>레지스터의 z 번째 바이트에 저장한다. EndConv 명령어는 4 개의 ByteExch 명령어들로 표현될 수 있다.

그림 2 ByteExch 명령어 포맷과 동작

3.1.3 BPMOD4 명령어

BPMOD 명령어는 GF(2^8) 필드에서 4 개의 Modular Multiplication 연산을 하는 명령어이다. AES, Twofish 알고리즘의 경우 GF(2^8) 필드에서 Mat(4*4)*Mat(4*1) 연산을 한다. BPMOD4 명령어의 구체적인 포맷과 동작은 [그림 3]과 같다. 여기서 4(8bits)는 GF(2^8) 필드를 정의하는 Irriducible Polynomial 을 나타낸다. GF(2^8) 필드에서 임의의 Irriducible Polynomal 에 대한 Modular Multiplication 연산을 위한 Functional Unit 은 Groβ schädl 이 제안한 Bit-serial Unified Multiplier 에서 GF(p) 영역의 연산을 위해 필요한 부분을 제외한 부분을 사용한다.[7]

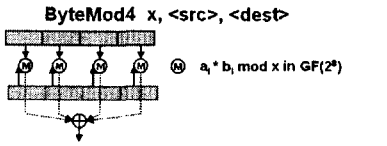


그림 3 BPMod4 명령어와 동작

3.2 비대칭 알고리즘 가속을 위한 명령어 집합

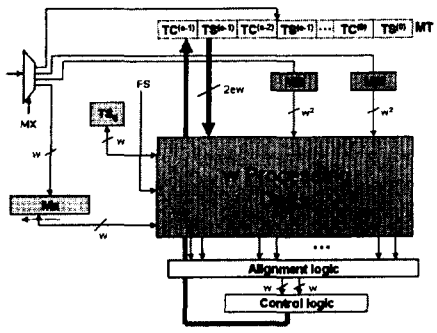
비대칭 알고리즘은 암호/복호화시에 긴 operand 에 대해 Modular 연산을 실행한다. 따라서 본질적으로 워드단위의 연산을 실행하는 마이크로 프로세서에서의 비대칭 알고리즘 가속이 쉽지 않다. 또한 마이크로 프로세서가 어느 길이의 Operand 에 대해서도 Programmability 를 제공해야 하는 제한적 요소는 기존의 비대칭 알고리즘을 가속하기 위한 ASIC 블락을 그대로 Functional Unit 으로 구성하는데 한계를 제공한다.

이 절에서 제시하는 긴 Operand 의 Modular Multiplication 명령은 GF(p)와 GF(2^m) 필드에서 모두 연산이 가능하도록 했는데 이 Modular Multiplication 명령어는 소프트웨어적인 방법을 통해 Modular Exponentiation 연산 또한 가속 시킬 수 있다.

3.2.1 MModInit/MModMult/MModBU 명령어

MModInit/MModMult/MModBU 명령어는 긴 Operand 의 Montgomery Multiplication 을 실행하는 명령어이다. Montgomery Multiplication 알고리즘은 Montgomery Reduction 알고리즘의 변형된 형태로 Precomputation 과 Conversion 을 요구하지만, 일반적으로 Classical Reduction 알고리즘, Barret Reduction 알고리즘보다 긴 Operand 에 대해 빨리 Reduction 을 수행하는 것으로 알려져 있다.[9] 또한 많은 변형 형태가 존재하며, GF(p) 필드뿐 아니라 GF(2^m) 필드에서도 약간의 하드웨어를 추가하여 동작하도록 만들 수 있을 뿐 아니라 Radix-level 병렬성이 존재하여 실제로 그 계산 속도를 높일 수 있다.[10][11] 본 논문에서는 E. Savarş et al 이 제안한 Unified Multiplier 에 기반하여 명령어를 설계하였다. 이들이 제안한 Unified Multiplier 는 Scalability 를 제공해 주기 때문에 몇 개의 PU(Processing Unit)를 사용하느냐

에 따라 명령어 포맷이 달라 질 수 있다.




```
MModInit   x, y , MX
MModMult   FS, x, <MT>, <MT>
MModBU     <MT>, x, y
```

그림 4 MModMult 명령어 포맷과 Functional Unit

[그림 4]는 w 개의 PU 을 하나의 Functional Unit 으로 구성한 것이다. MModInit 명령어는 각각 x 필드는 메모리 위치(offset), y 필드는 해당 메모리 위치로부터 읽어드려야 하는 바이트 수, MT 는 Funtional Unit 의 레지스터 위치(MT, Ma, MB)를 나타낸다. MModMult 명령어는 각각 FS 는 GF(p)와 GF(2^m)의 필드를 선택하는 것이고 피가수의 몇번째 워드인가를 나타낸다. MModBU 는 각각 x 필드는 메모리 위치(offset), y 필드는 메모리로 저장되어야 하는 바이트 수를 의미한다.

IV. 향후 관련 작업

3 장에서 제시한 명령어들은 아직 그 효율성에 대해서 검증되지 않았다. 따라서 시뮬레이션을 통해 그 효율성에 대해 알아보는 작업이 요구된다. 특히, 대칭 알고리즘과 달리 비대칭 알고리즘은 긴 Operand 에 대해 연산을 수행하기 때문에, 마이크로 프로세서에서 대칭형 알고리즘을 실행할 경우 비대칭 알고리즘에 대해 설계된 Functional Unit 이 사용되지 않을 가능성이 매우 크다. 따라서 비대칭 알고리즘 가속을 위해 설계된 명령어 집합은 마이크로 프로세서의 High-level 설계시 이 명령어의 사용 여부를 결정해야 한다. 마지막으로 2 장에서 제시했지만 해당 명령어를 제시하지 않은 명령어 집합은 소프트웨어적으로 쉽게 기존의 명령어를 가지고 가속할 수 있을 것인지, 아니면 하드웨어적인 가속이 필요한 것인지 실험을 통해 알아보는 것이 요구된다.

V. 결론

본 논문에서는 기존의 대칭 알고리즘을 위한 명령어 집합외에 추가로 제시될 수 있는 Endian 변환을 위한 EndConv 명령어, 다른 워드간 바이트 교환을 위한 ByteExch 명령어, 그리고 4 개의 byte 를 동시에 GF(2^8) 필드에서 계산할 수 있는 BPMOD4 명령어를 제안하였으며, 비대칭 알고리즘 가속의 핵심인 긴 Operand 에 대해 Modular Multiplication 을 하기 위한 MModMult 명령어 집합을 제안하였다. 이들은 Programmable 한 마이크로 프로세서의 설계에 핵심이 되는 부분으로써, 향후 검증 작업과 함께 대칭 및 비대칭 알고리즘 모두를 효율적으로 가속시킬 수 있는 토대를 마련한 점에서 의미가 있다고 하겠다.

참고문헌

[1] Jerome Burke, John McDonald, and Todd Austin, "Architectural Support for Fast Symmetric-key Cryptography ", ASPLOS, 2000

[2] Lisa Wu, Chris Weaver, and Todd Austin, "CryptoManiac: A Fast Flexible Architecture for Secure Communication", ISCA, 2001

[3] Zhiie Shi and Ruby B. Lee, "Bit permutation instruction for accelerating software cryptography", ASAP, 2000

[4] Yunjong choi, Sukhee cho, "Limitation on 3D realvideo coding using MAC", ISO/ IEC JTC1/ SC29/WG11 M8627, July 2002.

[5] Advanced Encryption Standard Development Effort, NIST, http://csrc.nist.gov/CryptoToolkit/aes/index2.html

[6] "Current Public-Key Cryptography systems", certicom http://www.certicom.com/resources/w_papers/w_papers.html

[7] Groβschädl, "A Bit-serial Unified Multipler Architecture for Finite Fields GF(p) and GF(2^m)", CHES 2001, 2001

[8] Alfred J. Menezes et al, Handbook of applied cryptography, CRC press

[9] Antoon Bosselaers et al, "Comparison of three modular reduction functions", CRYPTO'93, 1993

[10] E. Savaş et al, "A Scalable and Unified Multiplier Architecture for Finite Field GF(p) and GF(2^m)", CHES, 2000

[11] Gunnar Gaubatz, "Versatile Montgomery Multiplier Architectures", Worcester Plytechnic Institute, 2002