Journal of The Korea Institute of Information Security & Cryptology(JKIISC) ISSN 1598-3986(Print)
VOL.23, NO.5, October 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.5.815

임베디드 마이크로 프로세서 상에서의 최신 암호 구현 동향*

서 화 정,† 김 호 원‡
부산대학교

Recent Trends in Implementing Cryptography with Embedded Microprocessors*

Hwa-jeong Seo,† Howon Kim‡
Pusan National University

요 약

임베디드 마이크로 프로세서는 기존의 컴퓨터에 비해 제한적인 컴퓨팅 파워로 인해 간단한 연산과 작업의 수행에 보다 적합한 기기로 간주되어 왔다. 하지만 최근 들어 임베디드 마이크로 프로세서의 발전으로 인해 다양한 서비스를 제공하는 것이 가능해 졌다. 이와 더불어 안전하고 신뢰성 높은 서비스의 제공을 위해 임베디드 장비 상에서의 보안의 중요성이 갈수록 높아지고 있다. 현재 임베디드 장비 상에서의 다양한 암호화 구현 기법들이 제시되고 있다. 본 논문에서는 대표적인 8-, 16-, 32-비트 임베디드 장비인 AVR, MSP, 그리고 ARM 상에서 진행된 다양한 보안 구현 결과들을 비교 분석한다. 이는 추후 연구자들의 임베디드 장비 상에서의 암호 구현 연구에 많은 도움이 될 것이다.

ABSTRACT

Traditionally embedded microprocessors is considered as a device for low- and simple-computations because of its limited computing power and constrained resources. However high-end embedded devices have been developed and many applications are getting feasible in the embedded devices. To provide secure and robust service environments, security on embedded devices are in order. Recently many research results on embedded devices have been proposed. In this paper, we explore various cryptography implementation results on representative 8-, 16- and 32-bit embedded processors including AVR, MSP and ARM. This report would be helpful for following researchers who are interested in cryptography implementation techniques on resource constrained devices.

Keywords: Embedded Microprocessors, Cryptography Implementation, RSA, ECC, MQPKs

I. 서 론

급속한 임베디드 마이크로 프로세서 기술의 발전은 우리가 사용하는 임베디드 장비의 성능을 대폭 향상시킴으로써 복잡한 연산이 포함된 다양한 서비스를 언제 어디서나 제공받는 것이 가능하게 해주고 있다. 하지만 서비스의 제공이 가능해진 임베디드 장비는 악의적인 공격의 대상이 되어 사용자의 안전을 위협하는 요소로 작용하고 있다. 따라서 임베디드 장비 상에서의

접수일(2013년 5월 23일), 수정일(2013년 9월 9일), 게재확정일(2013년 9월 9일)
* 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2010-0026621).
† 주저자, hwajeong@pusan.ac.kr
‡ 교신저자, howonkim@pusan.ac.kr(Corresponding author)

민감한 정보의 교환이 안전하고 빠르게 수행될 수 있는 구조의 정착은 그 무엇보다 우선시 되어야 한다. 만약 개인의 정보가 해당 장비를 통해 유출되게 된다면 물질적 피해가 발생하게 된다. 이를 방지하기 위해 현재 임베디드 장비 상에서의 안전한 암호화 구현 기술들이 활발히 연구되고 있다. 본 논문에서는 임베디드 장비 상에서의 최신 구현 기술에 대해 살펴봄으로써 현재의 암호화 구현 기술수준에 대해 알아보며 안전하고 효율적인 서비스 제공을 위해 암호화 구현 시 고려해야 하는 사항에 대해 살펴본다. 본 논문에서 소개된 모든 내용은 최근에 발표된 논문들의 결과를 토대로 작성되었다.

본 논문의 구성은 다음과 같다. 논문의 2장에서는 대표적인 임베디드 프로세서에 대해 알아본다. 3장에서는 대표적인 대칭키, 공개키 구현 연구 동향에 대해 알아보며 마지막으로 4장에서는 본 논문의 결론을 맺는다.

II. 임베디드 마이크로 프로세서

〔표 1〕 각 프로세서들의 특징

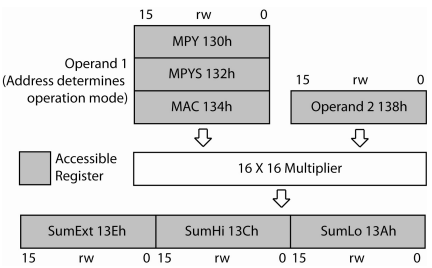
	ATmega128	MSP430(X)	ARM
Word size	8-bit	16-bit	32-bit
Frequency	8Mhz	8~20Mhz	13~416Mhz
Flash	128KB	32KB	32MB
SRAM	4KB	4KB	256~512KB
Multiplier	8-bit	16, 32-bit	32-bit
Crypto accelerator	DES, AES	AES	AES, SHA
Company	Atmel	TI	ARM

2.1 AT(X)mega(8-비트)

ATmega는 대표적인 8비트 마이크로 프로세서로써 8Mhz 이상의 클락으로 동작하며 128Kbyte 이상의 EEPROM과 4Kbyte 이상의 RAM을 제공한다〔6〕. 총 32개의 레지스터를 제공하며 사용자는 효율적인 연산 수행을 위해 메모리에 비해 소모되는 연산 복잡도가 낮은 레지스터를 잘 활용해야 한다. 초기 ATmega 모델에서는 가장 기본적인 대칭키 암호화인 DES를 제공했다. 이는 레지스터 주소에 암호화하고자 하는 값 정보를 넣으면 일정한 클락이 지난 이후에 값이 지정된 주소 레지스터에 저장되는 방식으로 수행된다. 최근 모델인 ATXmega에서는 AES기능 또한 제공하며 해당 기술의 경우 지정된 메모리 주소에 암호화를 위한 환경 셋팅값과 평문과 비밀키 값을 인가해 준 후 시작 신호를 주어 일정 시간이후에 암호문이 특정한 주소에 저장된다. 대표적인 대칭키 연산이 프로세서 상에서 제공되므로 장비 상에서의 대칭키 암호화기술의 소프트웨어 구현은 더 이상 큰 의미를 가지지 못한다. 반면에 비대칭키 암호화의 경우 연산 복잡도가 높아서 대칭키 구현에서와 같은 작은 모듈 형식의 라이브러리는 제공되지 않는다. 따라서 RSA, ECC 그리고 Pairing과 같은 연산은 사용자에 의해 직접 작성되어야 한다. 이를 보다 효과적으로 구현하기 위해서는 임베디드 디바이스에서 제공하는 하드웨어 곱셈기를 사용하는 것이 보다 효율적이다. 해당 곱셈기를 사용하게 될 경우 정수에 대한 곱셈이 짧은 시간 안에 가능함으로 실용적인 형태의 연산이 가능하다.

2.2 MSP430(X)(16-비트)

MSP430 보드는 대표적인 16비트 마이크로프로세서로써 8Mhz이상으로 동작하며 32Kbyte 이상의 플래시 메모리와 8Kbyte의 이상의 RAM 그리고 12개의 레지스터를 제공한다〔7〕. 이중에서 3개의 레지스터는 메모리 주소를 가리키는 포인터형식으로 사용되게 된다. MSP430 보드에서는 기본적으로 AES 모듈을 제공하며 동작형식은 ATmega128에서와 같이 모듈의 메모리 주소에 값을 저장하는 방식하고 시작신호를 주는 방식으로 연산이 수행된다. MSP430은 ATmega128보다 강력한 형태의 곱셈 모듈을 제공한다. 〔그림 1〕에서는 16비트 곱셈기의 구조를 보여주고 있다. 총 3가지의 곱셈 기법을 제공하며 이는 첫 번째 인자값을 MPY, MPYS 그리고 MAC에 인가하면 각각 정수곱셈과 곱셈 및 축적형 곱셈 기법을 선택할 수 있다. 두 번째 인자 주소에 값을 인가하게 되면 결과값은 SumHi, SumLo에 출력되어 나오게 되며 MAC 기법의 경우에는 SumExt에 올림이 발생하는 값만을 지속적으로 저장하게 된다. 최신 보드인

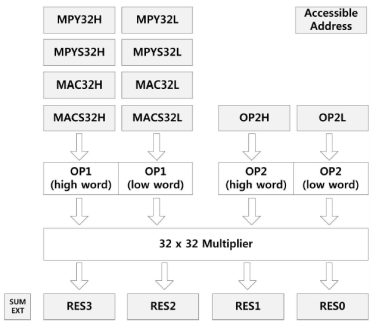


〔그림 1〕 16비트 곱셈기

〔표 2〕 암호화 기법의 소프트웨어 구현 비교[9](Enc: Encryption, Dec: Decryption, Tp: Throughput)

Cipher	Key (bit)	Block (bit)	Enc (cycle/block)	Tp at 4MHz (Kbps)	Dec (cycle/block)	Relative Tp (% of AES)	Code (byte)	SRAM (byte)	Relative code (% of AES)
Hardware-oriented block ciphers									
DES	56	64	8,633	29.6	8,154	38.4	4,314	0	152.4
DESXL	184	64	8,531	30.4	7.961	39.4	3,192	0	112.8
Hight	128	64	2,964	80.3	2,964	104.2	5,672	0	200.4
Present	80	64	10,723	23.7	11,239	30.7	936	0	33.1
Software-oriented block ciphers									
AES	128	128	6,637	77.1	7,429	100.0	2,606	224	100.0
IDEA	128	64	2,700	94.8	15,393	123.0	596	0	21.1
TEA	128	64	6,271	40.8	6,229	53.0	1,140	0	40.3
SEA	96	96	9,654	39.7	9,654	51.5	2,132	0	75.3
Software-oriented stream ciphers									
Salsa20	128	512	18,400	111.3	NA	144.4	1,452	280	61.2
LEX	128	320	5,963	214.6	NA	287.3	1,598	304	67.2

MSP430X 보드는 32비트 곱셈기법을 제공하며 〔그림 2〕에 나타난 바와 같이 32비트의 인자값을 16비트 곱셈기에서 사용한 방식으로 인가하게 될 경우 값이 도출되는 형식을 따르게 된다.



〔그림 2〕 32비트 곱셈기

2.3 ARM(32-비트)

ARM 보드는 대표적인 32비트 마이크로 프로세서로써 13Mhz이상의 클락으로 동작하며 256Kbyte의 SRAM과 32MB의 이상의 플래시 메모리를 제공한다[8]. 인텔의 PXA271 보드는 센서 네트워크 구현에 가장 많이 인용되는 ARM기반 프로세서로써 강력한 병렬 연산 구조인 wmmx 모드를 제공한다. wmmx모드는 32비트 마이크로 프로세서가 계산할 수 있는 연산 크기를 64비트까지 확장시켜 한번에 동일한 연산을 보다 큰 비트 단위로 수행할 수 있게 해준다. 곱셈 연산은 또한 wmmx의 기능을 이용하여 4개의 16비트 인자에 대한 곱셈 결과를 한 번에 계산할 수 있다. 최근에 개발된 프로세서인 Cortex시리즈는 이전의 ARM 보드들에 비해 전체적으로 높은 성능을 제시하여 임베디드 장비의 한계를 극복하고 다양하고 복잡한 서비스에 대한 구현이 가능하게 하였다. Cortex-M 시리즈의 경우 AES와 SHA 함수에 대한 하드웨어 가속모듈을 제공한다. 따라서 기존의 소프트웨어를 통한 구현 기법들 보다 적은 전력과 빠른 속도로 연산을 하는 것이 가능해 졌다[1]. Cortex-A 시리즈는 성능이 대폭적으로 개선한 제품으로써 스마트폰과 같은 높은 성능을 요구하는 장비의 프로세서로 사용된다. 기존의 장비와는 달리 NEON 기능을 이용하여 SIMD(Single Instruction Multiple Data)연산이 가능하여 연산의 벡터화를 통해 성능의 효율적인 개선이 가능하며 최근에는 이를 이용한 연구 논문이 많이 제시되고 있다. 〔표 1〕에서는 본장에서 살펴본 대표적인 프로세서들의 특징을 정리하여 나타내고 있다.

III. 암호화 구현 결과

3.1 대칭키 암호화

대칭키 암호화는 크게 DES와 AES로 구성되며 DES의 경우에는 낮은 보안강도로 인해 현재는 채택되지 않고 있는 보안기법이다. 반면에 AES는 강력한 암호화 강도와 효율적인 구현이 가능한 구조를 제공하

여 보다 많은 분야에서 기본적인 암호화 기법으로 사용되고 있다. 지금까지 센서네트워크와 RFID와 같은 저전력, 저성능의 보드에 적합한 대칭키 암호화의 제안 및 구현은 오랜 시간동안 연구되어 왔다. 〔표 2〕에는 대표적인 암호화 기법에 대한 소프트 웨어 구현 결과를 나타내고 있다. 해당 표에서는 대칭키 암호화를 세 가지 분류로 나누어 나타내며 이는 해당 암호화 기법의 초기 최적화 모델을 나타낸다. 하드웨어를 목표로 제시된 암호화 기법의 경우 소프트웨어 구현 시 Hight의 경우 throughput이 높게 나오는 대신 코드 크기가 커지는 문제가 있고 present의 경우에는 그와 반대로 코드크기가 줄지만 연산 속도 성능이 좋지 못하다〔23,24〕. 소프트웨어 기반 구현의 경우 표준으로 사용되는 AES의 경우 가장 안정적인 성능을 나타내었고 IDEA의 경우에는 암호화 과정과 코드 사이즈측면에서 AES보다 높은 성능을 나타내지만 복호화에 소비되는 시간이 증가하는 문제를 가지며. TEA와 SEA의 경우 초경량에 초점을 맞춘 암호화로써 throughput과 코드 사이즈가 같이 줄어드는 특징을 가진다. 소프트웨어 기반 스트림 암호기법의 경우 throughput과 코드사이즈 측면에서 AES보다 좋은 성능을 나타내었다. 하지만 스트림 암호화 기법은 암호화의 특징 상 암호화하고자 하는 평문의 크기가 커야 성능이 좋게 나오는 특징을 가진다.

이처럼 임베디드 장비 상에서의 대칭키 암호화는 속도와 코드 크기 면에서 프로세서에 부담을 주지 않는다. 또한 현재는 대칭키 암호화를 보다 가속화하기 위해 MSP430과 ATXmega와 같은 다양한 보드에서 기본적으로 AES 가속기를 제공함으로써 보다 빠른 속도로 연산하는 것이 가능하다. 최근에 제시된 MSP430 상에서의 AES 가속기를 이용하여 소프트웨어를 통한 기존 구현에 비해 성능을 월등히 향상시켰으며 〔표 3〕에는 이에 대한 구현 결과를 나타내고 있다. 표에서와 같이 암호화 모드에 따라 많게는 8배 이상의 성능 차이가 발생함을 확인할 수 있다. 성능에 있어 큰 차이가 발생하지 않는 GCM의 경우 AES연산 이외에도 유한체 연산을 수행해야 하는 추가적인 부하의 발생으로 인해 큰 성능차이가 발생하지 않는다. AES 가속기를 사용하게 될 때에는 추가적으로 소프트웨어로 암호화를 구현하지 않아도 된다. 따라서 〔표 4〕에서와 같이 프로그램을 작성하는 데 추가적으로 ROM이 사용되지 않으므로 속도와 크기 면에서 AES 가속기를 사용하는 것이 보다 효율적인 암호화 구현이 된다.

〔표 3〕 20MHz 동작 주파수에서의 4KB 메시지 암호화 처리량〔2〕

스킴	AES 가속기 이용	AES 소프트웨어 이용
CTR	6,915	830
CCM	4,391	427
GCM	891	509
SGCM	1,832	640
OCB3	4,209	779

〔표 4〕 암호화 구현에 소모되는 ROM과 RAM 크기〔2〕

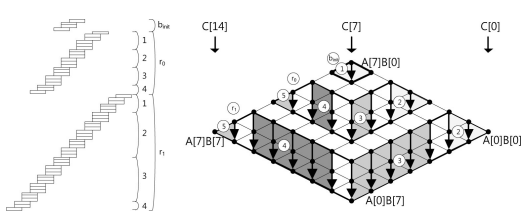
스킴	AES 가속기 이용		AES 소프트웨어 이용	
	ROM	RAM	ROM	RAM
CTR	130	100	3,034	100
CCM	1,094	258	3,998	258
GCM	4,680	886	7,584	886
SGCM	2,172	322	5,076	322
OCB	1,724	538	7,584	538

3.2 공개키 암호화

3.2.1 곱셈기법

현재 가장 많이 사용되는 공개키 암호화인 RSA, ECC 그리고 Pairing연산 모두 유한체 상에서의 덧셈, 곱셈 그리고 역위 연산에 기반을 두고 있기 때문이다. 특히 곱셈 연산은 모든 공개키 암호화에서 가장 높은 부하를 차지하며 그 범위는 80%까지 차지한다. 따라서 지금까지 수많은 논문 결과들이 곱셈에 치중하여 진행되어 왔고 최근 연구들도 유한체 상에서의 곱셈에 대한 연구가 가장 활발히 진행되고 있다. 본 장에서는 최근에 제시된 가장 빠른 프라임, 바이너리 필드 곱셈을 설명한다. 아래 그림은 프라임 필드 곱셈을 마름모형으로 나타낸 것이며 각각의 점은 부분 곱셈을 의미하며 화살표는 곱셈 방향을 의미한다.

〔그림 3〕에 나타난 Operand Caching 기법은 기존의 기법들이 최대한 중간 결과값에 대한 접근을 피했다면 해당 기법은 중간 결과값에 대한 접근을 적절



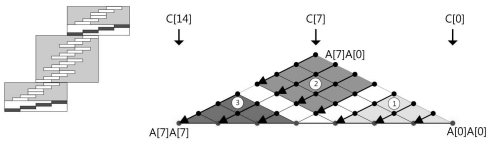
〔그림 3〕 Operand-Caching 곱셈기법

히 수용하면서 대신 인자들에 대한 접근을 효율적으로 줄여 곱셈 성능을 향상시켰다[9]. 예를들어 곱셈에 수행되어야 하는 인자가 각각 10개씩 존재한다면 부분곱셈을 10개의 쌍에 대해 취한 결과값을 구하는 것이 가능하다. 그 다음에 결과값을 구하기 위해 다음 인자를 선택해야 하는 시점이 오면 이전에 사용했던 인자의 쌍 중 하나의 인자들의 집합을 다음 연산 결과에도 사용하여 인자에 대한 접근을 효율적으로 줄일 수 있는 구조로 되어 있다.

[표 5] Operand-caching 성능비교

	[19]	[20]	[21]	[22]	[10]
Clock cycle	3,106	2,881	2,651	2,865	2,395

[그림 3]에 나타난 바와 같이 곱셈 연산 순서는 먼저 가장 상위 단에 위치하는 b_{init}을 계산한 후 다음 열에 해당하는 r_0을 계산한다. 이때 그림에서와 같이 2, 3, 4 그리고 5의 순서로 부분곱셈이 수행되게 된다. 여기서 2와 3은 A인자들의 집합을 공유하게 되며 3, 4 그리고 5의 연산에서는 B인자들의 집합을 공유하게 된다. 따라서 지속적으로 동일한 인자를 유지하며 사용하게 되므로 인자를 불러오기 위한 메모리 접근을 효율적으로 줄일 수 있다. [표 5]에서는 operand-caching기법의 효용성을 나타내며 기존 기법에 비해 성능이 월등히 향상됨을 확인할 수 있다.



[그림 4] Lazy doubling 제곱

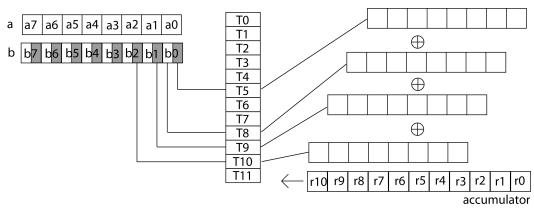
[그림 4]는 제곱 연산을 효율적으로 수행하기 위해 제시된 lazy doubling 기법을 triangular 형식으로 나타내고 있다. 곱셈 연산과 달리 모든 곱셈결과를 얻을 필요가 없이 약 절반에 해당하는 곱셈 연산을 수행하고 해당 중간 결과값에 대해 2배 연산을 하여 최종 결과값을 얻게 된다. 기존 기법과 달리 제시된 기법에서는 hybrid 기법을 기본적인 곱셈 구조로 선택하여 내부적으로 곱셈구조를 나눈다. 나누어진 곱셈

[표 6] Lazy doubling 성능비교[3,21]

	Scott et al	Lazy doubling
Clock cycle on ATmega128	2,065	1,509

구조에서 먼저 중복되어 계산되어야 하는 곱셈 부분을 먼저 모두 연산한다. 모든 곱셈 연산이 끝나면 이에 대한 2배 연산을 수행하고 남아 있는 곱셈을 처리하는 방식으로 수행하게 된다. 2배 연산을 미룸으로써 연산의 복잡도를 효율적으로 줄일 수 있는 기법이다. [표 6]에서는 이에 대한 성능을 나타내며 약 25% 이상의 성능 향상이 있음을 확인할 수 있다.

바이너리 곱셈은 지속적으로 인자들에 대한 비트연산을 수행해야 하므로 성능이 떨어지게 된다. 반면에 comb-window는 해당 비트연산을 효율적으로 대치하기 위해 곱셈을 연산 수행단계에 하지 않고 미리 계산해 놓은 연산결과에 접근하여 해당 값을 할당하는 방식으로 복잡한 곱셈연산을 메모리에 대한 접근으로 대신 계산하여 성능을 향상시켰다[11, 12]. 즉 사전에 하나의 인자 집합에 대한 모든 곱셈 결과값을 테이블 형식으로 계산하여 저장한 후 해당 결과값을 다른 인자의 값의 주소값으로 사용하여 순차적으로 메모리에 접근하여 결과값을 결과 레지스터에 xor하는 형식으로 저장하게 된다.



[그림 5] Comb-window 곱셈기법

[그림 5]에 이에 대한 예시가 나와 있다. 사전에 인자 a에 대한 모든 곱셈 결과값을 계산한다. 계산된 결과값은 b인자의 값을 주소로 하여 접근하게 된다. 여기서 a의 크기에 해당하는 결과값을 사전에 모두 계산하여 저장한 후 b인자를 통해 계산된 주소로 사전에 계산된 테이블에 접근하고 해당 결과값을 다시 결과 레지스터에 xor 연산을 하여 저장하게 된다. [표 7]에서는 바이너리 comb-window 곱셈기법을 사용하여 연산 시 기존의 기법보다 효율적임을 나타내고 있다.

〔표 7〕 Comb Window 성능비교(cycles)

Algorithm	ATmega128		MSP430		PXA271	
	〔11〕	〔15〕	〔11〕	〔15〕	〔11〕	〔15〕
Squaring	1,439	1,581	889	1,363	187	499
Square-root	1,182	1,730	769	1,644	185	546
Multiplication	11,727	13,557	8,706	10,147	1,411	4,926

3.2.2 RSA

2004년도에 제시된 논문에서는 RSA와 ECC 연산을 각각 임베디드 보드에 구현한 결과를 보임으로써 2004년도까지 가장 많은 분야에서 연구되고 적용되었던 RSA 연산에 대한 임베디드 보드의 구현이 ECC에 비해 효율적이지 않음을 시사하였다〔19〕. 〔표 8〕에서와 같이 ECC 연산의 경우 sign과 verify가 1초 안으로 수행이 되지만 RSA의 경우 sign의 경우 80비트의 강도를 제공하는 RSA-1024의 경우 2.7초 160비트의 강도를 제공하는 RSA-2048의 경우 20.8초가 걸림을 확인할 수 있다. 또한 인터넷 진흥원에서는 현재 RSA-1024가 보안적 취약성을 가짐으로 RSA-2048을 사용하도록 권장하고 있다. 이러한 상황도 더불어 비효율적인 RSA보다는 동일한 강도를 제공하고 키 크기가 작은 ECC에 대한 관심이 보다 높다고 할 수 있다.

〔표 8〕 ATmega 상에서의 RSA와 ECC 성능비교〔23〕

Method	Time〔ms〕@32MHz	
	sign	verify
ECC-P160(SECG)	203	203
ECC-P192(SECG)	310	310
ECC-P224(SECG)	548	548
RSA-1024	2,748	108
RSA-2048	20,815	485

3.2.3 TinyECC

2008년도에 제안된 TinyECC는 마이크로 프로세서 상에서의 타원곡선 알고리즘의 기본이 되는 프로그램을 작성하여 지금까지도 많은 후속 개발의 참조 논문으로 사용되고 있는 대표적인 논문이다〔12〕. 본 구현에서는 ATmega128, MSP430, Imote2에서 동작하는 소스코드를 tinyos환경에서 제공하고 있다. 사용자는 쉽게 홈페이지로부터 해당 코드를 다운받아 사용하는 것이 가능하다. 사용된 기법으로는 Barret reduction을 이용하여 reduction연산이 보다 효율적으로 수행되도록 하였으며 projective 상에서의 그룹 연산 구현을 통해 복잡한 역위 연산의 수를 줄여 연산이 효율적으로 수행되도록 하였다. 또한 hybrid 곱셈과 슬라이딩 윈도우 기법을 통해 유한체와 그룹 곱셈이 보다 효율적으로 수행되도록 하였다. 따라서 TinyECC는 뛰어난 범용성과 확장성을 제공하며 이를 통해 TinyOS에서 제공하는 다양한 서비스들을 TinyECC를 통해 보안 통신이 가능하도록 할 수 있다. 〔표 9〕에서는 TinyECC에서 구현된 알고리즘의 성능을 나타내고 있다. Imote2의 경우를 보면 1초안에 연산이 가능할 정도로 성능이 향상되었음을 확인할 수 있으며 이는 센서의 프로세서가 가지는 비트크기가 커질수록 성능이 향상되는 특성을 보인다.

〔표 9〕 TinyECC 성능비교〔12〕

secp160r1 Time(ms)	ATmega128 8MHz	MSP430 8MHz	PXA271 13MHz
ECDSA (init)	3,493	2,548	341
ECDSA (sign)	2,001	1,580	363
ECDSA (verify)	2,436	2,016	446
ECIES (init)	1,838	1,319	178
ECIES (encrypt)	3,907	3,271	736
ECIES (decrypt)	2,632	2,127	466
ECDH (init)	1,838	1,319	179
ECDH (key est)	2,117	1,755	392

3.2.4 TinyPBC

TinyECC가 프라임 필드 상에서의 구현에 중점을 맞추었다면 TinyPBC는 바이너리 필드 상에서의 구현을 효과적으로 제안하였다〔11〕. 〔그림 5〕에 나타난 comb-window 기법을 사용하여 바이너리 필드 상에

〔표 10〕 TinyPBC 성능비교〔11,15,16,17,18〕

80-bit security pairing (Time(sec))	TinyTate	NanoECC	Ishiguro	Oliveira	Szczechowiak	TinyPBC
ATmega128	30.20	10.96	5.80	5.45	2.66	1.90
MSP430	-	5.25	-	-	1.71	1.27
PXA271	-	-	-	-	0.46	0.14

서의 곱셈 연산을 사전 연산 테이블에 대한 메모리 접근 연산으로 효율적으로 대치하였다. 더불어 TinyPBC가 TinyECC와 다른점은 pairing연산에 적합한 다양한 pairing 기법들이 구현되었으며 곱셈 구현 시 MSP430에서의 MAC기술과 IMote2에서의 WMMX기술을 이용하여 보드에 특성화된 성능 향상을 도출보였다. MAC기술은 부분 곱셈 결과값을 곱셈기 메모리에 축적시킴으로써 메모리 접근 횟수를 줄인 기술이며 WMMX는 SIMD기술로써 한 번에 여러 연산의 수행이 가능하도록 하여 큰 워드 사이즈 계산이 한 번에 가능한 장점을 가진다. TinyPBC에서도 relic이라는 라이브러리를 통해 구현된 기술 코드를 제공하고 있다. 〔표 10〕에서는 TinyPBC를 통해 Pairing연산을 센서노드에 구현하게 될 때 수행되는 연산을 나타낸다. 제안된 구현 기법의 경우 1초에 가까운 성능을 나타낸다. 이는 초기 30초가 걸렸던 모델에 비하면 성능이 대폭 향상되었음을 확인할 수 있다.

3.2.5 MQPKs

Multivariate quadratic public key scheme은 앞으로 발명될 퀀텀 컴퓨터로 인해 기존의 number theory를 기반으로 둔 RSA와 ECC와 같은 암호화 기법의 factoring과 discrete logarithm 문제가 깨지는 것을 대비하기 위해 연구되고 있는 분야이다〔5〕. 기본적인 암호화 연산은 많은 수의 변수를 가진 행렬 연산을 통해 공격자가 변수의 값을 알 수 없도록 하여 안전하게 암호화하게 된다. 하지만 해당 기법은 자원 한정적인 임베디드 장비 상에서의 구현은 제한적이었다. 그 이유는 비밀키와 공개키의 크기가 수백Kbyte에 이르기 때문에 이를 저장할 공간이 임베디드 환경에서는 여의치 않기 때문이다. 하지만 ches2012에서는 보드 상에서의 구현결과를 실제로 보임으로써 여전히 비밀키와 공개키의 크기는 크지만 최적화된 구현 기법을 통해 구현가능함을 보였다〔13〕. 〔표 11〕에서는 구현된 결과를 나타낸다. 먼저 uov 스킴은 Unbalanced Oil and Vinegar 스킴의 약자로써 두 개의 Oil과 Vinegar을 매트릭스 연산의 인자로 하여 결과값을 도출한다. Rainbow스킴은 uov 스킴을 무지개와 같이 여러번 수행하여 킷값을 줄인 형식이다. enTTS는 기존의 UOV와 Rainbow와는 전혀 다른 형식으로 서명 및 검증이 수행되며 이는 키의 길이를 효율적으로 줄였다. 〔표 11〕에서와 같이 128kb의 용량 안에 비밀키와 공개키가 저장되며 MQPKs의 연산은 간단한 사칙연산을 통해 수행됨으로 성능이 기존의 Number Theory 기반 암호화에 비해 빠름을 확인할 수 있다. 따라서 앞으로 임베디드 상에서 MQPKs에 대한 연구가 보다 활발

〔표 11〕 자원한정적 장비 상에서의 MQPKs 성능비교

Scheme	n	m	Key Size (Byte)		System parameter		Time〔ms〕@32MHz		Code Size (Byte)	
			private	public	private	public	sign	verify	sign	verify
uov(21,28)	49	21	21462	25725	-	-	50.49	52.83	2188	466
0/1 uov(21,28)	49	21	12936	4851	8526	20874	49.29	43.60	2258	578
rainbow(15,10,10)	35	20	9250	12600	-	-	26.51	31.58	4162	466
enTTS(7,28,40)	40	28	2731	22960	-	-	10.37	79.95	24898	827
uov(28,37)	65	28	49728	60060	-	-	113.66	122.23	2188	466
0/1 uov(28,37)	65	28	30044	11368	19684	48692	110.20	100.37	2258	578
rainbow(18,13,14)	45	27	19682	27945	-	-	54.38	69.19	4162	466
enTTS(9,36,52)	52	36	4591	49608	-	-	19.03	208.07	41232	827
uov(44,59)	103	44	194700	235664	-	-	416.07	441.70	2188	466
0/1 uov(44,59)	103	44	116820	43560	77880	192104	399.43	424.04	2258	578
rainbow(36,21,22)	79	43	97675	135880	-	-	257.11	288.01	4162	466
enTTS(15,60,88)	88	60	13051	234960	-	-	66.94	962.17	116698	827

히 진행되리라 생각된다.

3.2.6 NEON Crypto

본 논문에서는 ARM 보드 상의 vector 명령어 셋을 가지는 NEON을 통해 암호화를 보다 효율적으로 수행할 수 있는 기법을 제안하였다[4]. NEON 명령어는 기존의 ARM이 가지는 32-bit 레지스터가 아닌 128-bit로 확장된 vector 레지스터를 통해 연산이 보다 효율적으로 수행 가능한 구조를 제시한다. 주요 아이디어는 128-bit 곱셈으로써 곱셈을 함과 동시에 이전 결과와 덧셈을 수행하는 연산을 이용하여 유한체 상에서의 곱셈을 수행함과 동시에 reduction이 되어야 하는 비트에 대한 연산이 자동적으로 수행되도록 한 부분이다. 이를 통해 두 번으로 나누어 계산되었던 곱셈과 reduction 부분을 합쳐서 연산이 가능하도록 하였다. [표 12]에서는 NEON을 이용한 암호화 구현 성능을 평가하고 있다. 기존의 기법과 달리 연산 인자의 vectorization을 통해 향상시킬 수 있었다. 이처럼 기존의 컴퓨터에서 제공하던 다양한 기술들이 임베디드 장비에도 적용이 되고 있으며 이를 통해 임베디드 장비 상에서도 높은 성능을 나타낼 수 있는 수준에 도달하였다.

[표 12] NEON을 이용한 암호화 구현

Scheme	연산	결과
Salsa20	Encryption	5.60 cycles/byte
Poly1305	Authentication	2.30 cycles/byte
Curve25519	Shared Key 생성	527,102 cycles
Ed25519	Sign	244,655 cycles
Ed25519	Verify	624,846 cycles

IV. 결론

본 논문에서는 대표적인 임베디드 프로세서 상에서의 안전한 서비스 제공을 위한 보안 기술에 대해 확인해 보았다. 자원 한정적인 임베디드 마이크로 프로세서에 보다 적합한 성능을 위해 다양한 최적화 알고리즘 및 기술들이 개발되어 왔으며 현재는 고도화된 암호화 구현 기술을 통해 안전한 서비스 제공의 실용화 단계에 이르렀다. 하지만 아직도 공격자에 의해 쉽게 물리적으로 노출되는 문제를 가지며 이로 인해 물리적으로 임베디드 장비 상에서 동작하는 연산에 대한 기밀성 또한 보장이 되어야 하는 어려움에 봉착하고 있다. 따라서 효율적인 암호화 구현을 제공함과 동시에 물리적인 공격에도 강인한 암호화 구현기술 연구가 동시에 수행되어야 보다 안전한 구현이 가능하다.

참고문헌

[1] Cortex-M Datasheet, APR. 2013, available from http://www.freescale.com/

[2] Gouvêa, Conrado PL, and Julio López. "High speed implementation of authenticated encryption for the MSP430X microcontroller," In Progress in Cryptology LATINCRYPT 2012, pp. 288-304. Springer Berlin Heidelberg, 2012.

[3] Lee, Younho, Ill-Hee Kim, and Yongsu Park. "Improved multi-precision squaring for low-end RISC microcontrollers," Journal of Systems and Software, 2012.

[4] Bernstein, Daniel J., and Peter Schwabe. "NEON crypto," In Cryptographic Hardware and Embedded Systems-CHES 2012, pp. 320-339. Springer Berlin Heidelberg, 2012.

[5] Peter W. Shor. "Polynomial-time algorithms for prime factoriztion and discrete logarithms on a quantum computer," SIAM Journal on Computing, vol. 26, no. 5, pp.1484-1509, October, 1997.

[6] ATmega Datasheet, OCT. 2012, available from www.atmel.com/

[7] MSP430 Datasheet, OCT. 2012, available from www.ti.com

[8] PXA270 Datasheet, OCT. 2012, available from http://www.phytec.com/pdf/datasheets/PXA270_DS.pdf

[9] Thomas Eisenbarth, Christof Paar, Axel Poschmann, Sandeep Kumar, Lelf Uhsadel, "A Survey of Lightweight Cryptography Implementations," IEEE Design & Test of Computers, vol. 24, no. 6, pp. 522-533, Dec. 2007.

[10] Michael Hutter and Erich Wenger. "Fast multi-precision multiplication for public key cryptography on embedded microprocessors," Cryptographic Hardware and Embedded Systems - CHES 2011, vol 6917 of Lecture Notes in Computer Science, pp. 459-474. 2011.

[11] Lopez, R, Dahab, "High-speed software multiplication in GF(2m)," in: B.K. Roy, E. Okamoto (Eds.), First International Conference in Cryptology in India (INDOCRYPT'00), LNCS, Vol. 1977, pp. 203-212, 2000.

[12] Leonardo B. Oliveira, Diego F. Aranha, Conrado P.L. Gouvea, Michael Scott, Danilo F. Camara, Julio Lopez, Ricardo Dahab, "TinyPBC: Pairings for authenticated identity based non-interactive key distribution in sensor networks," Computer Communications, vol. 34, pp. 485-493, 2011.

[13] An Liu, Peng Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track, pp. 245-256, April 2008.

[14] Peter Czypek, Stefan Heyse, Enrico Thomae, "Efficient Implementations of MQPKS on Constrained Devices," Cryptographic Hardware and Embedded Systems - CHES 2012, pp. 374-389, 2012.

[15] P. Szczechowiak, L.B. Oliveira, M. Scott, M. Collier, R. Dahab, "NanoECC: testing the limits of elliptic curve cryptography in sensor networks," in: Fifth European Conference on Wireless Sensor Networks (EWSN'08), pp. 305 - 320, 2008.

[16] D. Galindo, R. Roman, J. Lopez, "A killer application for pairings: authenticated key establishment in underwater wireless sensor networks," in: M.K. Franklin, L.C.K. Hui, D.S. Wong (Eds.), Seventh International Conference on Cryptology and Network Security (CANS'08), LNCS, vol. 5339, Springer, pp. 120 - 132, 2008.

[17] L.B. Oliveira, A. Kansal, B. Priyantha, M. Goraczko, F. Zhao, "Secure-TWS: authenticating node to multi-user communication in shared sensor networks," in: Eighth ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'08), pp. 289 - 300, 2009.

[18] W. Du, R. Wang, P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in: P.R. Kumar, A.T. Campbell, R. Wattenhofer (Eds.), Sixth ACM International Symposium on Mobile ad hoc Networking and Computing (MOBIHOC'05), ACM Press, pp. 58 - 67, 2005.

[19] Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 119 - 132, 2004.

[20] Uhsadel, L., Poschmann, A., Paar, C, "Enabling Full-Size Public-Key Algorithms on 8-bit Sensor Nodes," In: 4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks, ESAS 2007, Cambridge, UK, July, 2007.

[21] Scott, M., Szczechowiak, P, "Optimizing Multiprecision Multiplication for Public Key Cryptography," Cryptology ePrint Archive, Report 299 http://eprint.iacr.org/, 2007.

[22] Liu, Z., Großsch¨adl, J., Kizhvatov, I, "Efficient and Side-Channel Resistant RSA Implementation for 8-bit AVR Microcontrollers," In: Workshop on the Security of the Internet of Things - SOCIOT 2010, 1st International Workshop, Tokyo, Japan, November 29.

IEEE Computer Society, Los Alamitos, 2010.

〔23〕 Hong, Deukjo, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee et al. "Hight: A new block cipher suitable for low-resource device," In Cryptographic Hardware and Embedded Systems-CHES 2006, pp. 46-59. Springer Berlin Heidelberg, 2006.

〔24〕 Bogdanov, Andrey, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. "PRESENT: An ultra-lightweight block cipher," In Cryptographic Hardware and Embedded Systems-CHES 2007, pp. 450-466. Springer Berlin Heidelberg, 2007.

〈저자소개〉

서 화 정 (Hwa-jeong Seo) 정회원
2010년 2월: 부산대학교 컴퓨터공학과 학사 졸업
2012년 2월: 부산대학교 컴퓨터공학과 석사 졸업
2012년 3월~현재: 부산대학교 컴퓨터공학과 박사과정
〈관심분야〉 정보보호, 암호화 구현

김 호 원 (Ho-won Kim) 종신회원
1993년 2월: 경북대학교 전자공학과 학사 졸업
1995년 2월: 포항공과대학교 전자전기공학과 석사 졸업
1999년 2월: 포항공과대학교 전자전기공학과 박사 졸업
2008년 2월: 한국전자통신연구원 정보보호연구단 선임연구원/팀장
2008년 3월~현재: 부산대학교 정보컴퓨터공학부 부교수
〈관심분야〉 스마트그리드 보안, RFID/USN 정보보호 기술, PKC 암호, VLSI 설계, embedded system 보안, IoT