情報保護學會論文誌
第16卷 第5號, 2006. 10

RFID 태그를 위한 초소형 AES 연산기의 구현

구 본 석,[1†] 유 권 호,[1] 양 상 운,[1] 장 태 주,[1] 이 상 진[2‡]
[1]국가보안기술연구소, [2]고려대학교 정보보호대학원

Low-cost AES Implementation for RFID tags

Bonseok Koo,[1†] Gwonho Ryu,[1] Sangwoon Yang,[1] Taejoo Chang,[1] Sangjin Lee[2‡]
[1]National Security Research Institute(NSRI),
[2]Graduate School of Information Security(GSIS), Korea University

요 약

Radio Frequency IDentification (RFID) 시스템은 최근 수많은 산업분야에서 각광받고 있는 근거리 자동 인식 기술이다. 이러한 RFID 시스템에서 전송 데이터에 대한 보안과 프라이버시 보호는 점차 심각한 문제로 인식되고 있으며, 이를 해결하기 위해서는 강도 높은 암호 알고리즘을 이용한 전송 데이터의 암호화가 필수적이다. 본 논문에서는 이러한 문제를 해결하기 위해 RFID 태그에 구현 가능한 초소형 Advanced Encryption Standard (AES) 연산기를 제안한다. 제안하는 연산기는 3,992 게이트 카운트의 작은 크기를 가지면서 암호화와 복호화가 모두 가능하다. 또한 128-비트 한 블록에 대해 암호화를 446 클락 사이클, 복호화를 607 클락 사이클에 처리하므로 기존에 발표된 초소형 AES 연산기들에 비해 각각 55%와 40% 이상 개선된 성능을 가진다.

ABSTRACT

Radio Frequency IDentification (RFID) will soon become an important technology in various industries. Therefore, security mechanisms for RFID systems are emerging crucial problems in RFID systems. In order to guarantee privacy and security, it is desirable to encrypt the transferred data with a strong crypto algorithm. In this paper, we present the ultra-light weight Advanced Encryption Standard (AES) processor which is suitable for RFID tags. The AES processor requires only 3,992 logic gates and is capable of both 128-bit encryption and decryption. The processor takes 446 clock cycles for encryption of a 128-bit data and 607 clock cycles for decryption. Therefore, it shows 55% improved result in encryption and 40% in decryption from previous cases.

Keywords : *AES processor, Low-cost, RFID tag, gate count*

Ⅰ. 서 론

RFID (Radio Frequency IDentification) 시스템은 RFID 태그가 붙어 있는 모든 대상 객체를 자동적으로 식별하고 분류할 수 있는 기술이며, 다가오는 유비쿼터스 시대의 핵심 기술 중 하나로서 최근 매우 큰 각광을 받고 있다. RFID 시스템은 태그, 리더기, 그리고 백-앤드 (back-end) 데이터베이스의 세 가지 요소로 구성된다. 먼저 태그는 마이크로칩 형태로 안테나와 고유 ID를 가지며 radio 주파수를 통해 리더기로 정보를 전송하는 역할을 담당한다.

접수일: 2006년 6월 13일; 채택일: 2006년 9월 21일
† 주저자, bskoo@etri.re.kr
‡ 교신저자, sangjin@korea.ac.kr

리더기는 radio 주파수를 통해 태그로 명령을 전송하여 그 응답으로 받은 정보를 백-앤드 데이터베이스로 전달하며, 백-앤드 데이터베이스는 각 태그들의 정보를 데이터베이스로 관리하는 서버 역할을 담당한다. 그림 1은 전형적인 RFID 시스템의 구조를 나타낸다.

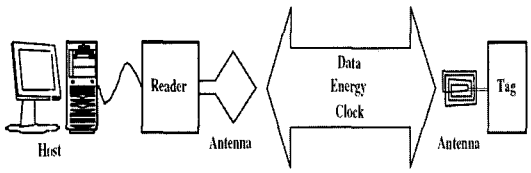

그림 1. RFID 시스템의 구조

RFID 시스템은 대중교통 요금의 자동 지불, 애완동물의 인식과 추적, 물류 관리의 자동화 등 다양한 분야에서 큰 파급효과가 예상된다. 반면에 태그와 리더기 사이에서 발생하는 radio 주파수 통신은 보안과 프라이버시 보호 문제를 야기하며 이러한 문제점은 RFID 기술의 도입과 확산에 적지 않은 장애요소로 지적되고 있다.

지난 몇 년간에는 이와 같은 RFID의 보안 문제점을 해결하기 위한 노력들이 진행되어 왔다. 먼저 'kill' 명령을 이용하여 한번 사용된 태그를 비활성화시키는 방법이 제안되었다.[1] 이 방법은 태그의 추적을 회피하는데 효과적이지만 한번 사용된 태그를 재활용할 수 없다는 단점을 가진다. 또한 태그를 차폐하는 'blocker tag' 방법도 제안되었다.[2] 이 방법 또한 효과적이지만 고의로 서비스를 거부하는 등의 악의적인 공격에 오용될 수 있는 단점을 가진다. 또 다른 접근 방법으로 암호학적 기법인 해쉬 함수를 이용하는 인증 프로토콜들도 제안되었으나, 이와 같은 방법은 하나의 태그에 대해 백-앤드 데이터베이스의 많은 계산량을 요구하므로 효과적이지 못하다.[3,4]

최근에는 보다 안전하고 효과적인 해결책을 위해 AES(Advanced Encryption Standard) 블록 암호를 이용하는 인증 프로토콜과 RFID 태그의 초소형, 저전력 환경에서도 구현 가능한 초소형 AES 연산기들이 제안되었다.[5,6] 근본적인 보안 해결책을 위해서는 RFID 시스템에서 리더기와 태그간의 통신은 모두 암호화하여 운용할 필요가 있으며, 이를 위해서는 대표적인 표준 블록 암호인 AES 알고리즘을 사용하는 것이 바람직하다.

본 논문에서는 보다 개선된 RFID 태그용 초소형 AES 연산기의 구현 방법을 제안한다. 제안하는 AES 연산기는 128-비트 한 블록을 암호화하는데 446 클록 사이클을 소요하므로 기존 연산기들에 비해 55% 이상 향상된 성능을 가진다. 또한 상호 인증과 같은 복잡한 쌍방향 통신 프로토콜이 가능하도록 암호화와 복호화가 모두 가능하면서도 0.25um CMOS 공정에서 4,000 게이트 카운트 미만의 크기를 가지므로 RFID 태그에 적용이 가능하다는 장점을 가진다.

본 논문은 다음과 같이 구성된다. 2장에서는 AES 연산기의 구현에 관해 발표된 기존 연구들을 살펴보고, 3장과 4장에서는 AES 알고리즘의 개요와 본 논문에서 제안하는 AES 연산기의 아키텍쳐 및 구현방법에 대해 상세히 설명하며, 5장은 구현 결과 및 기존 결과와의 비교를 서술하며, 마지막 6장에서 결론을 맺는다.

II. 관련분야의 연구

AES 블록암호 알고리즘[7]은 지난 2001년 NIST(National Institute of Standards and Technology, 미국 국립 표준 기술원)에 의해 차세대 표준 알고리즘으로 채택되었으며, 현재까지 하드웨어 구현에 관한 수많은 연구들이 발표되었다. 초기에는 주로 ASIC 라이브러리[8,9]나 FPGA[10,11]를 이용하여 고성능 구현하거나 그 성능들을 비교 분석하는 연구가 주로 이루어졌다. 하지만 점차 임베디드(embedded) 기술과 시장이 발전하면서 소형화, 저전력화 구현에 관한 연구가 발표되었다.

V. Rijmen은 복합체(composite field) 연산을 이용하여 유한체(finite field) $GF(2^8)$상의 8-비트 원소에 대한 역원 연산을 유한체 $GF(2^4)$상의 4-비트 원소들의 연산으로 매핑(mapping)시킴으로써 S-box를 경량으로 구현하는 방법을 제안하였다.[12] A. Satoh 등은 이 개념을 더욱 확장하여 8-비트 원소를 복합체 $GF(((2^2)^2)^2)$상의 원소로 매핑하여 S-Box를 더욱 경량화하는 방법과 이를 이용하여 약 5,400게이트 카운트의 크기를 가지고 32-비트 단위로 동작하는 AES 경량 연산기를 제안하였다.[13] 또한 D. Canright는 정규기저(Normal Basis)를 사용하여 복합체 $GF(((2^2)^2)^2)$로 매핑하는 방법을 제안하고, 이를 이용하여 A. Satoh의 결과보다 20%

정도 더 작은 S-Box 구현 결과를 발표하였다. [14]

최근에는 소형, 저전력 구현에 관한 연구가 계속 발전하여 RFID 태그에서도 적용 가능한 초소형 AES 연산기에 대한 연구 결과가 발표되고 있다. M. Feldhofer 등은 RFID 태그에 적용이 가능한 크기를 약 5,000 게이트 카운트 미만으로 예측하고, 3,595 게이트 카운트의 크기를 가지는 8-비트 AES 연산기를 제안하였다. [5] M. Feldhofer 등이 제안한 연산기는 128-비트 한 블록의 데이터를 처리하는데 약 1,000 클락 사이클을 소모하며, 암호화만이 가능한 특징을 가진다. 또한 M. Jung 등은 8-비트 마이크로컨트롤러의 보조프로세서로 사용할 수 있는 AES 연산기를 제안하였다. [6] 이 연산기는 마이크로컨트롤러와 데이터 및 키 저장 메모리를 공유하고 마이크로컨트롤러의 컨트롤에 의해 전체적인 동작이 이루어진다. 따라서 2,168 게이트 카운트의 상대적으로 작은 크기를 가진다. 또한 키 확장에는 363 클락 사이클이, 암호화 및 복호화에는 645 클락 사이클이 각각 소요된다.

Ⅲ. AES 알고리즘

AES는 128-비트 평문에 대해 128-비트 암호문을 출력하는 블록암호이며, 128/192/256-비트의 키 크기를 사용할 수 있다. 현재까지 알려진 암호학적 공격에 대해 128비트 키 길이의 암호 강도는 RFID와 같은 일반적인 상용 환경에서 안전하다고 알려져 있으므로, 여기서는 128-비트 키 길이의 경우로 제한하여 생각한다. 그림 2에는 AES 알고리즘의 암호화 및 복호화 과정을 나타내었다. 그림에서 보는 것처럼 총 10 라운드의 연산과정으로 이루어지며, 각각의 라운드는 AddRoundKey, (Inv)SubBytes, (Inv)ShiftRows, (Inv)MixColumns의 4가지 연산으로 이루어진다. 각 연산의 내용은 다음과 같다.

- AddRoundKey: 128-비트 State(중간 결과값)와 128-비트 라운드 키의 비트별 XOR 연산이다.
- SubBytes: 128-비트 State의 각 바이트에 대해 비선형 특성을 가지는 S-Box를 적용한다. InvSubBytes는 SubBytes의 역연산(inverse operation)이다.
- ShiftRows: 128-비트 State 중 Row라고 정의된 32-비트에 대해 바이트 단위로 환형 쉬프트(cyclic shift)방식으로 자리바꿈. InvShiftRows는 ShiftRows의 역연산이다.
- MixColumns: 128-비트 State 중 Column이라고 정의된 32-비트에 대해 다항식 곱셈 연산을 수행. InvMixColumns는 MixColumns의 역연산이다.

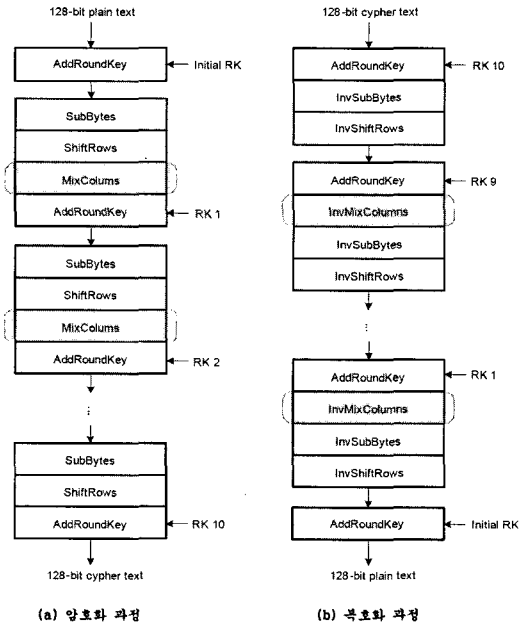

그림 2. AES 알고리즘의 암호화 및 복호화 과정

AES 알고리즘의 복호화 과정은 암호화 과정의 역순으로 처리된다. 이때 InvSubBytes와 InvShiftRows 연산은 128-비트 State를 바이트 단위로 처리하므로 연산 순서를 그림 2와 같이 서로 바꿔도 무관하다. 또한 초소형 구현을 위해서는 AES의 최소 연산 단위인 8-비트 아키텍처를 가지는 AES 연산기를 설계하는 것이 바람직하며, 이를 위해서는 AES의 암호화 및 복호화 과정을 그림 2와 같이 8-비트 단위 처리가 가능한 연산인 AddRoundKey, (Inv)SubBytes, (Inv)ShiftRows와 32-비트 단위로 처리되는 연산인 (Inv)MixColumns의 과정으로 구분할 필요가 있다. 따라서 AES 알고리즘의 암호화과정은 AddRoundKey, SubBytes, ShiftRows으로 구성되는 일련의 8-비트 단위 연산과 MixColumns 연산의 반복, 그리고 마지막에 수행되는 AddRoundKey 연산의 과정으로 생각할 수 있다. 또한 복호화 과정은 InvSubBytes, InvShiftRows, AddRoundKey으로 구성되는 일련의 8-비트 연산과 InvMixColumns 연산의 반복, 그리고 맨 처음에 수행되는 AddRoundKey 연산의 과정이

된다.

매 라운드에 사용되는 라운드 키는 초기 라운드 키로부터 확장된다. 암호화 시에는 각 라운드 함수를 수행하면서 동시에 해당 라운드의 키를 계산하는 'on-the-fly' 키 확장이 가능하다. 반면에 복호화 시에는 암호화 과정과 역순으로 라운드 키가 사용된다. 따라서 이때에는 모든 키를 미리 계산하여 레지스터에 저장한 다음 사용하는 방법을 생각할 수 있다. 하지만 이런 방법은 11개의 128-비트 키를 저장하기 위한 하드웨어 자원이 필요하므로 바람직하지 않다. 초소형 구현을 위해서는 복호화 계산을 시작하기 전에 초기 라운드 키로부터 마지막 라운드 키를 계산한 다음, 복호화를 수행하면서 각 라운드 키를 역순으로 'on-the-fly' 키 확장을 수행하는 방법을 사용할 수 있다.

Ⅳ. 초소형 AES 연산기의 아키텍쳐

그림 3에는 본 논문에서 제안하는 8-비트 AES 연산기의 아키텍쳐를 나타내었다. 제안하는 연산기는 데이터와 키를 저장하는 메모리, SubBytes나 InvSubBytes 연산을 수행하는 S-Box 블록, MixColumns나 InvMixColumns 연산을 수행하는 MC 블록, 데이터 경로를 위한 MUX-(Multiplexer), 그리고 제어를 담당하는 컨트롤러 등으로 구성된다.

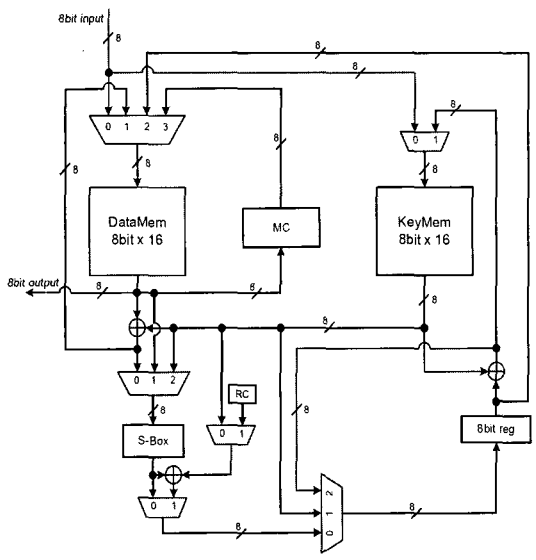

그림 3. 제안하는 초소형 8-비트 AES 연산기의 아키텍쳐

먼저 데이터 메모리(DataMem)는 처음에는 128-비트 입력 데이터를 저장하고 연산중에는 중간 결과값으로 갱신되며, 연산이 완료되면 결과값을 저장한다. 같은 방식으로 키 메모리(KeyMem)는 처음에는 128-비트 초기 라운드 키를 저장하고 연산중에는 매 라운드 키로 갱신된다. 두 개의 메모리는 동일한 구조를 가지며 모든 접근은 8-비트 단위로 이루어진다. 또한 레지스터를 기반으로 설계하여 쓰기 포트와 읽기 포트를 따로 가지므로 새로운 값으로 쓰기와 동시에 이전 값을 읽기가 가능하도록 구현하였다. 표 1에서는 본 논문에서 제안하는 AES 연산기의 내부 구성 요소별 기능에 대해 보다 자세히 나타내었다.

표 1. 제안하는 AES 연산기의 구성 요소별 기능 설명

구성 요소	기능
DataMem (8bit x 16)	128-비트 데이터 메모리
KeyMem (8bit x 16)	128-bit 라운드 키 메모리
S-Box	SubBytes 또는 InvSubBytes 연산을 바이트 단위로 처리하는 8-비트 S-box 블록
MC	MixColumns 또는 InvMixColumns 연산을 바이트 단위로 처리하는 블록
RC	라운드 키 확장에 필요한 8-비트 상수를 선택적으로 출력하는 블록
8bit_Reg	데이터나 키 정보의 중간 결과값을 임시로 저장하는 8-비트 레지스터

1. AddRoundKey/SubBytes/ShiftRows 연산

S-Box 블록에서 처리하는 SubBytes나 InvSubBytes 연산은 흔히 룩업-테이블(LUT, LookUp-Table) 방식으로 구현한다. 하지만 이 방법은 2 x 256 x 8-비트 크기의 롬(ROM)이 소요되므로 초소형 구현에는 적합하지 않다. 따라서 보다 소형 구현을 위해 (Inv)S-Box를 조합논리(combinational logic)로 구현하는 기법들이 발표되어 왔다.[12,13,14] 여기서는 그 중에서 최적의 결과를 나타내는 D. Canright의 방법[14]을 사용하였다.

그림 3을 살펴보면 데이터 메모리로부터 읽혀진 데이터는 암호화 시에는 키 메모리로부터 전달되는 라운드 키와 XOR되어 S-Box 블록을 통과하고, 복호화 시에는 S-Box 블록을 거친 다음 라운드 키와 XOR될 수 있다. 따라서 제안하는 연산기는 3장에

서 언급한 일련의 8-비트 단위 연산 과정: AddRoundKey, SubBytes, ShiftRows나 InvSubBytes, InvShiftRows, AddRoundKey를 하나의 클락 사이클 안에 처리가 가능하다. 이때 별도의 임시 저장 공간을 사용하지 않고 ShiftRows나 InvShiftRows가 처리될 수 있도록 데이터 메모리와 키 메모리에 대한 읽기 주소와 쓰기 주소를 적절히 제어해야 한다. 그림 4에는 128-비트 State와 라운드 키에 대한 바이트 단위의 연산 처리 순서를 암호화와 복호화 각각에 대해 나타내었다. 그림에서 각 바이트마다 표기된 번호는 처리되는 순서를 나타낸다.

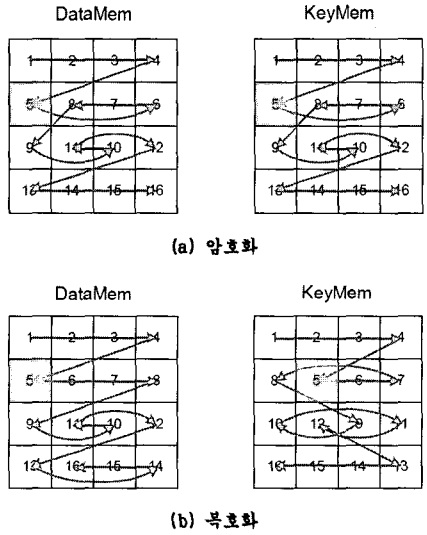

그림 4. 8-비트 단위 연산의 처리 순서

먼저 암호화 시에는 ShiftRows 연산의 자리바꿈에 맞춰서 바이트 단위의 연산 순서가 정해진다. 예를 들어 위쪽에서 두 번째 Row의 경우, 첫 번째 클락 사이클에는 State와 라운드 키의 맨 왼쪽 바이트를 각각 읽어서 AddRoundKey, SubBytes의 연산을 거친 다음, 그 결과를 그림 3의 8-비트 레지스터(8-bit reg)에 저장한다. 두 번째 클락 사이클에는 맨 왼쪽 바이트가 ShiftRows 처리된 후의 위치인 맨 오른쪽 바이트의 State와 라운드 키를 읽어서 이전 클락 사이클과 동일한 연산을 수행한다. 이때 8-비트 레지스터에 저장된 이전 바이트의 결과값은 현재 처리 중인 맨 오른쪽 바이트에 갱신된다.

반면에 복호화 시에는 InvShiftRows 연산의 자리바꿈에 맞춰서 연산 순서가 정해진다. 예를 들어 두 번째 Row의 경우, 먼저 State의 맨 왼쪽 바이트를 읽어서 InvSubBytes 연산을 거친 다음, 해당 바이트를 InvShiftRows 처리한 후의 위치인 라운드 키의 왼쪽 두 번째 바이트와 AddRoundKey 연산을 수행하고 그 결과값을 8-비트 레지스터에 저장한다. 두 번째 클락 사이클에는 InvShiftRows 수행 후의 위치인 왼쪽 두 번째 바이트 State를 읽어서 오른쪽 두 번째 바이트의 라운드 키와 함께 이전과 동일한 연산을 수행한다. 이때 암호화 경우와 마찬가지로 8-비트 레지스터에 저장된 이전 바이트의 결과값은 현재 처리 중인 왼쪽 두 번째 바이트에 갱신된다. 결과적으로 제안하는 AES 연산기는 128-비트 State에 대해 AddRoundKey, SubBytes, ShiftRows나 InvSubBytes, InvShiftRows, AddRoundKey의 연산을 8-비트 단위로 총 16 클락 사이클 만에 처리하며 1 클락 사이클의 지연 시간을 가진다.

2. MixColumns 연산

MixClomns와 InvMixColumns 연산은 그림 3에서 설명한 MC 블록에서 선택적으로 수행된다. MixColumns와 InvMixColumns는 앞서의 다른 연산들과는 달리 32-비트 데이터가 한꺼번에 처리되어야 하는 특징이 있다. 암호화 시에 사용되는 MixColumns는 32-비트 Column 데이터 $a(x)$에 대해 수식 (1)과 같이 행렬 연산을 수행하여 32-비트 $b(x)$를 출력한다.

$$\begin{pmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 02\,03\,01\,01 \\ 01\,02\,03\,01 \\ 01\,01\,02\,03 \\ 03\,01\,01\,02 \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \quad (1)$$

MixColumns의 행렬 연산은 일반화하여 $b_i = b_i = \{02\}a_i + \{03\}a_{i+1} + \{01\}a_{i+2} + \{01\}a_{i+3}$로 표현할 수 있다. 따라서 MixColumns 연산의 네 개의 출력 b_i는 동일한 방식으로 계산되며, 단지 입력 바이트 a_i의 순서만 달라진다. M. Feldhofer 등은 이러한 MixColumns 연산의 규칙성을 이용하여 전체 연산의 1/4만 구현하는 소형화 기법을 사용하였다.[5] 여기서는 이 개념을 복호화까지 적용하여 MixColumns뿐만 아니라 InvMixColumns도 구현 가능하도록 MC 블록을 설계하였다. InvMix-Columns

연산은 32-비트 데이터 $a(x)$에 대해 수식 (2)와 같이 행렬 연산을 수행하여 32-비트 $b'(x)$를 출력한다.

$$\begin{pmatrix} b'_3 \\ b'_2 \\ b'_1 \\ b'_0 \end{pmatrix} = \begin{pmatrix} 0e\ 0b\ 0d\ 09 \\ 09\ 0e\ 0b\ 0d \\ 0d\ 09\ 0e\ 0b \\ 0b\ 0d\ 09\ 0e \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \quad (2)$$

$$= \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} + \begin{pmatrix} 0c\ 08\ 0c\ 08 \\ 08\ 0c\ 08\ 0c \\ 0c\ 08\ 0c\ 08 \\ 08\ 0c\ 08\ 0c \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix}$$

InvMixColumns의 행렬 연산을 일반화하면 $b'_i = \{0e\}a_i + \{0b\}a_{i+1} + \{0d\}a_{i+2} + \{09\}a_{i+3}$로 표현되며, 이는 다시 $b'_i = b_i + \{0c\}a_i + \{08\}a_{i+1} + \{0c\}a_{i+2} + \{08\}a_{i+3}$로 표현이 가능하다. 따라서 InvMixColumns는 MixColumns와 부가적인 로직으로 구현이 가능하며, 네 개의 출력 b'_i는 MixColumns와 마찬가지로 a_i의 순서만 달라질 뿐 동일한 방식으로 계산된다. 따라서 MC 블록은 MixColumns와 InvMixColumns 연산의 1/4 로직으로 구성되며 그 구조를 그림 5에 나타내었다.

그림에서 보듯이 MC 블록은 3개의 8비트 레지스터를 포함하며, 8-비트 데이터 a_i를 차례로 입력받아 쉬프트하면서 연산을 수행한다. 최초에 3개의 8-비트 데이터가 순서대로 매 클락 사이클마다 쉬프트하면서 입력된 후, 네 번째 입력부터는 3개의 8-비트 레지스터에 저장된 값과 데이터 메모리로부터 직접 입력받은 8-비트 값을 이용하여 8-비트 출력값 b_i를 순서대로 출력하게 된다. 따라서 MC 블록은 하나의 32-비트 Column 데이터를 7 클락 사이클동안 처리하며, 128-비트 State에 대해서는 28 클락 사이클이 소요된다. 또한 암호화 및 복호화를 선택하는 /enc 신호에 따라 MixColumns나 InvMixColumns 연산을 선택적으로 수행할 수 있다.

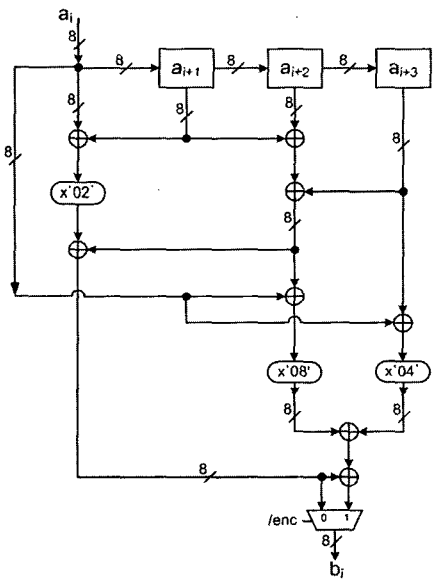

그림 5. MC 블록의 내부 구조

3. 라운드 키 확장

라운드 키 확장은 키 메모리, RC 블록, 두 개의 XOR 게이트, 8-비트 레지스터를 이용하여 계산된다. 그림 3을 살펴보면 키 메모리에서 읽혀진 8-비트 라운드 키는 S-Box 블록을 거쳐 RC 블록의 출력과 XOR되거나, 직접 8-비트 레지스터에 저장될 수 있다. 이렇게 8-비트 레지스터에 저장된 값은 다음 클락 사이클에 출력되는 8-비트 라운드 키와 XOR되어 원하는 키 메모리 공간에 갱신된다. 이와 같은 메카니즘을 이용하여 'on-the-fly' 방식으로 암호화 및 복호화의 라운드 키 확장이 가능하다. 여기서 RC 블록은 선택 신호에 따라 바이트 단위의 상수값을 출력하며, 룩업-테이블 방식으로 구현한다. 그림 6에는 라운드 키 확장 시 키 메모리를 8-비트 단위 처리하는 순서를 암호화와 복호화 각각에 대해 나타내었다. 그림에서 각 바이트마다 표기된 번호는 갱신 순서를 나타낸다.

여기서 점선은 첫 번째 클락 사이클에는 시작점의 한 바이트를 읽어서 8-비트 레지스터에 저장하고, 그 다음 클락 사이클에 끝점에 해당하는 한 바이트를 갱

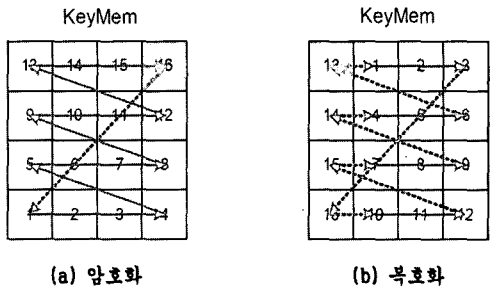

그림 6. 라운드 키 확장 시의 처리 순서

신하는 과정을 나타내며, 실선은 매 클락 바이트 단위로 갱신하는 과정을 나타낸다.

먼저 암호화 시에는 첫 번째 Row의 맨 오른쪽 바이트를 읽어서 8-비트 레지스터에 저장하고, 다음 클락 사이클부터는 네 번째 Row의 맨 왼쪽 바이트부터 화살표 방향으로 차례대로 갱신된다. 각 바이트의 정보는 자기 자신의 값과 8-비트 레지스터에 저장된 값을 XOR시킨 결과로 매 클락 사이클마다 갱신시킨다. 이때 8-비트 레지스터는 각 바이트에서 읽혀진 값으로 갱신된다. 한편 네 번째 Row의 맨 오른쪽 바이트에서 세 번째 Row의 맨 왼쪽 바이트로 진행하는 경우와 같이 4번의 대각선 과정에서는 S-Box 블록과 RC 블록이 사용된다. 결과적으로 한 라운드의 암호화 키 확장을 위해서는 총 17 클락 사이클이 소요된다.

복호화 시에는 암호화의 경우와 거의 유사하지만 조금 더 복잡한 순서로 키 확장이 이루어진다. 먼저 첫 번째 Row의 맨 왼쪽 바이트를 읽어서 8-비트 레지스터에 저장하고, 다음 3 클락 사이클 동안 오른쪽 세 개의 바이트를 화살표 방향으로 차례대로 갱신한다. 다음에는 두 번째 Row의 맨 왼쪽 바이트를 읽어서 8-비트 레지스터에 저장하고, 다음 3 클락 사이클 동안 같은 방법으로 오른쪽 세 개의 바이트를 차례대로 갱신한다. 이러한 순서로 맨 왼쪽 Column 4 바이트 이외의 12 바이트 정보를 갱신시킨다. 마지막으로 맨 왼쪽 Column 4 바이트의 갱신을 위해서는 그림 6처럼 대각선 위치의 바이트 정보를 읽어서 미리 8-비트 레지스터에 저장한다. 예를 들어, 맨 오른쪽 Column의 두 번째 바이트를 읽어서 8-비트 레지스터에 저장하고, 다음 클락 사이클에 맨 왼쪽 Column의 첫 번째 바이트를 갱신한다. 결과적으로 한 라운드의 복호화 키 확장은 오른쪽 3 개의 Column을 갱신하는데 16 클락 사이클, 맨 왼쪽 Column을 갱신하는데 8 클락 사이클, 총 24 클락 사이클이 소요된다.

라운드 키 확장 시에는 MixColumns나 InvMixColumns 연산에서 사용하는 데이터 메모리와 MC 블록이 사용되지 않는다. 따라서 라운드 키는 매 라운드의 MixColumns나 InvMixColumns 연산과 동시에 병렬로 계산이 가능하다. MixColumns나 InvMixColumns 연산은 28 클락 사이클이 소요되는 반면, 라운드 키 확장은 이보다 작은 17 클락 사이클 혹은 24 클락 사이클이 소요된다. 따라서 본 논문에서 제안하는 AES 연산기는 라운드 키 확장을 위해 별도의 클락 사이클이 소모되지 않는 장점을 가진다. 한편 복호화 시에는 암호화의 마지막 라운드 키로부터 키 확산이 이루어지기 때문에, 초기에 키 확장을 위해 약 170 클락 사이클 정도가 부가적으로 소요된다.

V. 구현결과

제안하는 AES 연산기는 Synopsys사의 Design Compiler 툴을 이용하여 동부-아남 0.25um standard cell CMOS 공정으로 합성하였으며, 그 합성 결과 로그를 그림 7에 나타내었다. 그림에서 보는 바와 같이 제안하는 연산기는 3,992 게이트의 작은 크기를 가진다. 또한 표 2에는 연산기의 내부 구성 요소별 하드웨어 복잡도를 정리하여 나타내었으며, 여기서 Memory는 그림 3의 DataMem와 KeyMem에 공통으로 사용되는 8비트 x 16 크기의 메모리를 나타낸다. 한편 하드웨어 복잡도를 나타내는 게이트 카운트는 ASIC 공정마다 다소 달라질 수 있으므로 기존 연산기들과의 정확한 비교는 어렵지만, 약 4,000 게이트 미만의 크기는 RFID 태그 구현에 매우 타당한 결과임을 판단할 수 있다.

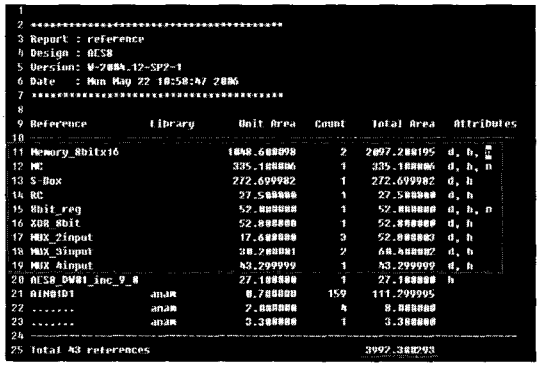

그림 7. 제안하는 AES 연산기의 합성 결과 로그 (동부-아남 0.25um 공정)

제안하는 연산기는 MixColumns나 InvMixColumns 연산과 라운드 키 확장 연산을 동시에 수행하는 특징을 가진다. 따라서 라운드 키 확장에 별도의 클록 사이클을 소모하는 기존의 연산기들이 1,000 클록 사이클 정도를 소모하는 반면에, 본 논문에서 제안하는 연산기는 암호화 시에는 446 클락 사이클, 복호화 시에는 607 사이클을 소모한다.

표 2. 제안하는 AES 연산기의 하드웨어 복잡도

구성 요소	개수	게이트 카운트
Memory(8bit x16)	2	2,097
MC	1	335
S-Box	1	273
RC	1	28
8-bit Reg	1	52
XOR(8bit)	3	53
MUX(2 input)	3	53
MUX(3 input)	2	60
MUX(4 input)	1	43
Controller	1	998
Total		3,992

결과적으로 제안하는 연산기는 기존의 연산기들에 비해 암호화는 55%, 복호화는 40% 정도 향상된 성능을 보인다. 연산기의 동작을 검증하기 위해 Cadence사의 NCVerilog 시뮬레이션 툴을 이용하였으며, 그림 8과 9에는 100KHz 클락 속도를 기준으로 암호화와 복호화의 경우에 대해 시뮬레이션을 실시한 출력결과 파형을 나타내었다.

그림에서 nEnc 신호는 연산기가 암호화('0') 또는 복호화('1')를 수행할 지를 결정하는 신호이며, aes_busy 신호는 aes 연산기가 암호화 및 복호화를 수행 중임을 알리는 신호이다. 또한 state는 유한 상태 기계(FSM, Finite State Machine) 기반으로 동작하는 연산기의 내부 동작 상태를 나타내며, di8은 연산기에 대한 8-비트 입력 신호, do8은 8-비트 출력 신호를 나타낸다.

제안하는 연산기는 RFID 태그에 적용이 가능한 초경량 크기이면서, 암호화와 복호화가 둘 다 가능한 최초의 단일 AES 연산기이다. M. Feldhofer 등이 제안한 연산기 [5] 는 암호화만 가능하며 Mark Jung

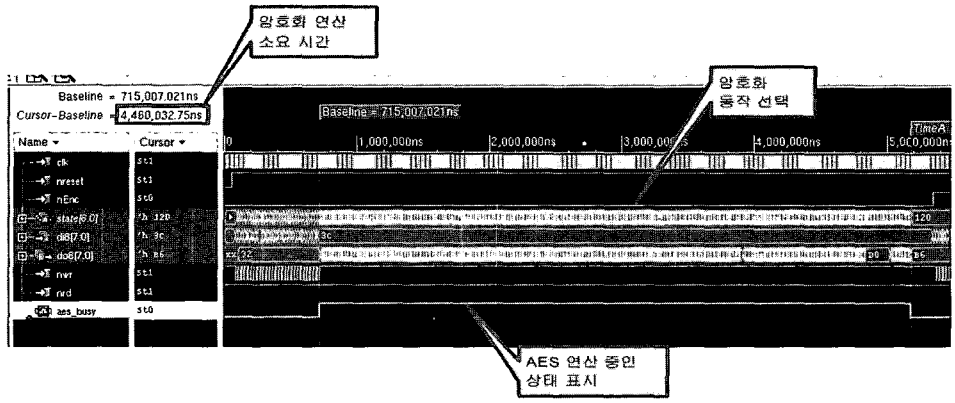

그림 8. 암호화 검증 시뮬레이션의 출력 파형

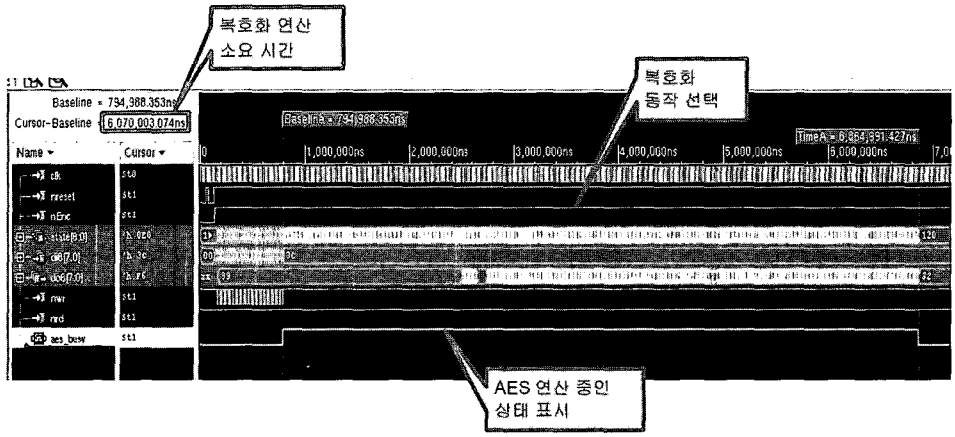

그림 9. 복호화 검증 시뮬레이션의 출력 파형

등이 제안한 연산기[6]는 단일 연산기가 아니고, 8-비트 마이크로컨트롤러의 보조 프로세서 형태이다. 또한 기존 연산기들은 라운드 함수와 키 확장 연산을 순서대로 수행하므로, 암호화나 복호화 시에 약 1,000 클락 사이클을 소모하는 반면에, 제안하는 연산기는 MixColumns나 InvMixColumns 연산과 동시에 라운드 키 확장 연산을 수행함으로써, 암호화 시에는 446 클락 사이클, 복호화 시에는 607 사이클을 소모한다. 따라서 제안하는 연산기는 기존의 연산기들에 비해 암호화는 55%, 복호화는 40% 각각 향상된 성능을 보인다.

표 3. 지원 기능, 게이트 카운트, 클락 사이클에 대한 기존 연산기와의 비교

	지원 기능	게이트 카운트	클락 사이클
제안하는 연산기	암복호화	3,992	446/607
M. Feldhofer 등[5]	암호화	3,595	1,016
Mark Jung 등[6]	암복호화 (보조프로세서)	3,089	1,008

Ⅵ. 구현결과

본 논문에서는 8-비트 기반의 초소형 AES 연산기를 제안하였다. 제안하는 연산기는 128-비트 암호화 및 복호화가 모두 가능하고 동부-아남 0.25um standard CMOS 공정으로 합성 시, 3,992 게이트 카운트의 크기를 가지므로 RFID 태그와 같은 경량 환경의 암호 구현에 적합하다. 또한 제안하는 연산기는 라운드 키 확장을 위해 별도의 클록 사이클을 소모하지 않고, MixColumns나 InvMixColumns 연산과 동시에 라운드 키 확장을 'on-the-fly' 방식으로 수행하는 특징을 가진다. 따라서 128-비트 한 블록에 대한 암호화를 수행하는데 446 클락 사이클, 복호화에는 607 클락 사이클을 소요한다. 결과적으로 제안하는 연산기는 기존에 발표된 초소형 AES 연산기들과 비슷한 4,000 게이트 카운트 미만의 크기를 차지하고 암호화와 복호화가 모두 가능하면서, 암호화는 55%, 복호화는 40% 이상 개선된 성능을 나타낸다.

참 고 문 헌

[1] S. E. Sarma, S. A. Weis, and D. W. Engels, "RFID Systems and Security and Privacy Implications," *CHES'02*, LNCS 2523, pp. 454-469, Springer-Verlag, 2002.

[2] A. Juels, R. L. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID tags for Consumer Privacy," *ACM Conference on Computer and Communication Security, 2003, Proceedings*, pp. 103-111, ACM Press, 2003.

[3] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," *Security in Pervasive Computing, 2003*, LNCS 2802, pp. 201-212, Springer-Verlag, 2003.

[4] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic approach to a provacy friendly tags," *RFID Privacy Workshop*, MIT, 2003.

[5] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm," *CHES'04*, LNCS 3156, pp. 357-370, Springer- Verlag, 2004.

[6] M. Jung, Horst Fiedler, and Reneé Lerch, "8-Bit Microcontroller System with Area Efficient AES Coprocessor for Transponder Applications," *Workshop on RFID and Lightweight Crypto*, pp. 32-43, July, 2005.

[7] FIPS Pub. 197: Specification for the AES, Nov. 2001, available at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[8] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and Performance Testing of a 2.29Gb/s Rijndael Processor," *IEEE Journal of Solid-State Circuits*, pp. 569-572, March 2003.

[9] S. Morioka and A. Satoh, "A 10Gbps full-AES crypto design with a twisted-BDD S-box architecture," *IEEE Inter-*

national Conference on Computer Design, IEEE, 2002.

[10] K. U. Jarvinen, M. T. Tommiska, and J. O. Skytta, "A fully pipelined memory-less 17.8Gbps AES128 encryptor," *FPGA'03*, ACM, 2003.

[11] M. McLoone and J. V. McCanny, "High performance single-chip FPGA Rijndael algorithm implementations," *CHES'01*, LNCS 2162, pp. 65-76, Springer- Verlag, 2001.

[12] Vincent Rijmen, "Efficient implemen -tation of the Rijndael S-box," available at : http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf, 2001.

[13] A. Satoh, S. Mroioka, K. Takano, and Seiji Munetoh, "A compact Rijndael hardware architecture with S-box optimization," *Advances in Cryptology - ASIACRYPT 2001*, LNCS 2248, pp. 239-254, Springer-Verlag, 2001.

[14] D. Canright, "A Very Compact S-Box for AES," *CHES'05*, LNCS 3659, pp. 441-455, Springer-Verlag, 2005.

[15] 조용국, 송정환, 강성우, "AES(Advanced Encryption Standard) 안전성 평가에 대한 고찰," *한국정보보호학회 논문지*, 제11권, 제6호, pp. 67-76, 2001.

[16] 안하기, 신경욱, "AES Rijndael 블록 암호 알고리듬의 효율적인 하드웨어 구현," *정보보호학회 논문지*, 제 12권, 제2호, pp. 57-67, 2002.

[17] 최병윤, 박영수, 전성익, "모듈화된 라운드 키 생성회로를 갖는 AES 암호 프로세서의 설계," *정보보호학회 논문지*, 제12권, 제5호, pp. 15-25, 2002.

〈著者紹介〉

구 본 석 (Bonseok Koo) 정회원
1998년 2월: 경북대학교 전자공학과 학사
2000년 2월: 포항공과대학교 전자공학과 석사
2000년 3월~2000년 9월: LG 정보통신 중앙 연구소 연구원
2000년 10월~현재: 국가보안기술연구소 선임연구원
〈관심분야〉 암호칩 설계, 공개키 암호, 부채널 공격

유 권 호 (Gwonho Ryu) 정회원
1999년 2월: 포항공과대학교 전자공학과 학사
2001년 2월: 포항공과대학교 전자공학과 석사
2002년 9월~현재: 국가보안기술연구소 연구원
〈관심분야〉 암호칩 설계, 블록 암호, 부채널 공격

양 상 운 (Sangwoon Yang) 정회원
1992년 2월: 충북대학교 정보통신공학과 학사
1998년 2월: 충북대학교 정보통신공학과 석사
1992년~2000년: 국방과학연구소 연구원
2000년~현재: 국가보안기술연구소 선임연구원
〈관심분야〉 암호칩 설계, Computer Arithmetic, 정보보호, 반도체

장 태 주 (Taejoo Chang) 정회원
1982년 2월: 울산대학교 전기공학과 학사
1990년: 한국과학기술원 전기 및 전자공학과 석사
1998년: 한국과학기술원 전기 및 전자공학과 공학박사
1982년~2000년: 국방과학연구소 선임연구원
2000년~현재: 국가보안기술연구소 책임연구원
〈관심분야〉 암호칩 설계, 정보보호, 통계학적 신호처리

이 상 진 (Sangjin Lee) 종신회원
1987년 2월: 고려대학교 수학과 학사
1989년 2월: 고려대학교 수학과 석사
1994년 2월: 고려대학교 수학과 박사
1989년 2월~1999년 2월: 한국전자통신연구원 선임연구원
1999년 2월~2001년 8월: 고려대학교 자연과학대학 조교수
2000년 2월~현재: 고려대학교 정보보호대학원 부교수
〈관심분야〉 대칭키 암호의 분석 및 설계, 정보은닉이론, 컴퓨터 포렌식