하드웨어 공유 구조를 이용한 RBC 알고리즘의 설계

박형근*, 김선엽**, 나유찬**
*남서울대학교 전자공학과
**남서울대학교 정보통신공학과
e-mail:phk315@nsu.ac.kr

Design of the RBC Algorithm using Shared Hardware Architecture

Hyoung-Keun Park*, Sun-Youb Kim**, Yu-Chan Ra**
*Dept of Electronic Eng., Namseoul University
**Dept of Information Communication Eng., Namseoul University

요 약

본 논문에서는 차세대 블록 암호 시스템으로 선정되었으며 미연방정부의 표준으로 제정된 RBC(Rijndael Block Cipher) 알고리즘을 하드웨어로 구현하였다. 구현된 블록 암호 시스템은 임베디드 시스템에 적용이 가능하도록 암호화 블록과 복호화 블록을 따로 설계하지 않고 하드웨어를 공유하여 하나의 블록에서 선택에 따라 암호화와 복호화가 모두 이루어질 수 있도록 설계함으로써 하드웨어의 면적을 최소화하였다.

1. 서론

컴퓨터의 사용이 보편화되면서 현대 사회는 점차 정보 통신 및 정보화 기술의 활용이 증가하고 있다. 특히, 국방 및 외교분야 등에서 특수목적에 국한되어 국가 독점으로 사용되어 오던 암호기술에 대한 민간 사용에 대한 요구가 증가하게 되었고, 현재 암호기술은 정보사회 구축의 초석으로 간주되고 있다. 더욱이 암호기술이 기밀성 서비스 등 소극적 기능 제공에서 인증 서비스 등 적극적 기능 제공으로 영역을 확대하면서 새로운 정보보호서비스가 창출되고 그에 따르는 시장이 형성되고 있다. 이에 각국은 암호기술 및 암호제품 개발의 주도권을 확보하고자 노력을 경주하고 있다.[1] 따라서 본 논문에서는 차세대 블록 암호 시스템으로 선정되었으며 미연방정부의 표준으로 제정된 RBC(Rijndael Block Cipher) 알고리즘을 하드웨어로 구현하였다.

2. RBC 알고리즘 분석

Rijndael은 가변 블록 길이와 가변 키 길이를 갖는 반복 구조의 블록 암호 알고리즘이다. 따라서 블록 길이와 키 길이는 128, 192, 256비트로 독립적으로 지정될 수 있으며, 블록 길이와 키 길이에 따라 라운드 수가 결정된다. 표 1은 블록 길이와 키 길이에 따른 라운드 수와의 관계를 나타낸 것이다.[2][3]

[표 1] 블록 길이와 키 길이에 따른 라운드 수

Nr	Nb=4 (128bit)	Nb=6 (192bit)	Nb=8 (256bit)
Nk=4(128bit)	10	12	14
Nk=6(192bit)	12	12	14
Nk=8(256bit)	14	14	14

암호화와 복호화는 다음의 네가지 연산으로 구성된 라운드를 이용한다

-. ByteSub : byte 단위의 비선형 치환 테이블(S-box)을 이용하여 각 byte를 state의 다른 byte 값에 독립적으로 치환한다.

-. ShiftRow : state의 값을 변경시키지 않고 state들의 위치를 변환시킨다.

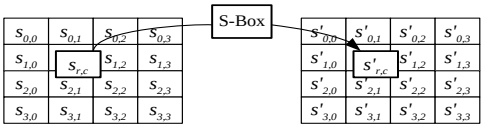
-. MixColumn : state 값을 column에 대해서 $GF(2^8)$상에서 정의된 행렬곱셈 연산을 한다.

-. AddRoundKey : 각 state에 대해 키 확장에 의해 생성된 라운드 키와 XOR연산을 한다.

2.1 암호화(Encryption)

암호화될 데이터 입력은 state 배열에 복사된다. 초기 라운드 키와의 덧셈이 수행된 후, state 배열은 키 길이와 블록의 크기에 따른 라운드 수(표)만큼 실행됨으로써 변환된다. 위에서 설명한 단계의 연산 과정이 한 라운드의 실행과정이다. 그리고 마지막 라운드는 위의 연산중 MixColumn 변환이 제외된다. 모든 라운드가 끝난 state는 출력 배열에 복사된다.

ByteSub 변환은 S-box를 이용하여 state의 각 바이트에 독립적으로 작용하는 비 선형 바이트 치환이다. 그림 2의 S-box 연산은 역(inverse)이 가능하며 다음의 두 변환을 거쳐 이루어진다.[4]



[그림 2] ByteSub 변환

변환 1. 유한체 $GF(2^8)$에서 곱의 역을 취한다.
변환 2. 다음과 같이 정의된 아핀(affine) 변환을 $GF(2^8)$상에서 적용한다.

$$b_i = b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \quad (1)$$

i 의 변화 범위가 $0 \le i < 8$ 일 때, b_i는 바이트의 i 번째 비트이고, c_i는 {63}인 {01100011} 값을 갖는 c 바이트의 i 번째 비트이다.

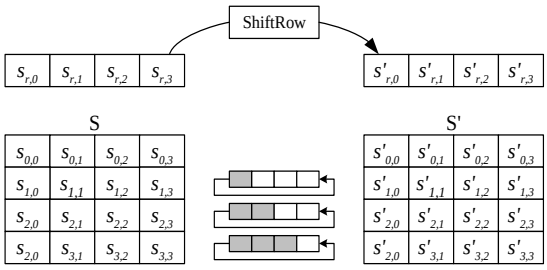
2.2 ShiftRow

ShiftRow 변환은 State의 첫째행을 제외한 나머지 행들을 각각 서로 다른 offset으로 byte 단위 cyclic shift를 통해 위치를 교환하는 변환이다. 즉, State의 첫째 행 Row_0는 shift 시키지 않으며, 두 번째 행 Row_1은 C1 byte 만큼 쉬프트 시키고, Row_2는 C2만큼 그리고, Row_3는 C3만큼 shift 시킨다. shift offset C1, C2, C3는 블록길이 Nb에 따라서 결정되며, 표 2에 블록길이에 따른 ShiftRow 변환의 offset 값을 나타냈다.[5]

[표 2] 블록길이에 따른 cyclic shift offsets

Nb	C1	C2	C3
Nb=4(128bit)	1	2	3
Nb=6(192bit)	1	2	3
Nb=8(256bit)	1	3	4

ShiftRow 변환동작은 그림 3과 같다.



[그림 3] ShiftRow 변환

2.3 MixColumn

MixColumn 변환은 State의 Column들을 $GF(2^8)$ 상에서의 다항식들 a(x)로 보고, 고정된 다항식 c(x)에 대해서 다음의 연산을 수행한다.

$$a(x) \otimes c(x) = \bmod x^4 + 1 \quad (2)$$

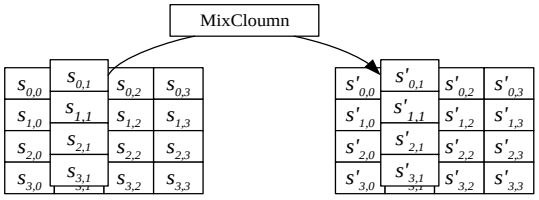
여기서 c(x)는 다음과 같다.

$$c(x) = \{03\}x^3 + \{02\}x^2 + \{01\}x + \{01\} \quad (3)$$

따라서, $s'(x) = c(x) \otimes s(x)$를 다음과 같이 행렬식으로 나타낼 수 있다.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad (4)$$
$$\text{for } 0 \le c < Nb$$

MixColumn 변환은 그림 4와 같다.



[그림 4] MixColumn 변환

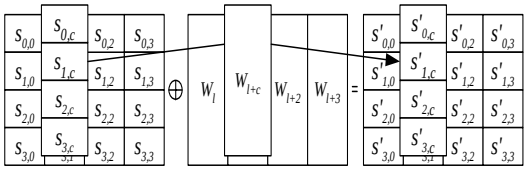
2.4 AddRoundKey

AddRoundKey 변환은 라운드 키와 State를 bitwise XOR 연산한다. 각 라운드 키는 키 스케쥴러로부터 확장된 Nb개의 Word로 구성된다. 다음 식과 같이 Nb개의 Word는 State의 column과 더해진다. 여기서 w_i는 키 스케쥴 워드, round의 범위는 $0 \le round \le Nr$ 이다.

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{\partial * Nb+c}] \quad (5)$$

여기서, c의 범위는 $0 \le c < Nb$ 이다.

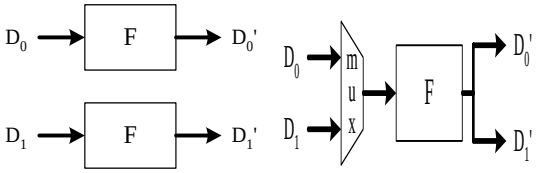
AddRoundKey 변환의 동작은 그림 5와 같다. 여기서, $l = round \times Nb$ 이다.



[그림 5] AddRoundKey 변환

3. 블록 암호 알고리즘의 하드웨어 구현

하드웨어 공유(Hardware Sharing)는 하나의 라운드 연산에 동일한 연산 모듈이 병렬로 사용되는 경우에 적합한 방식으로 하드웨어 면적을 최소화하는 설계에 널리 사용된다. 그림 6과 같이 하나의 라운드에 두 개의 연산 모듈이 포함되어 있는 경우 하나의 모듈만 하드웨어로 갖추고 이를 반복해서 사용하는 구조이다.



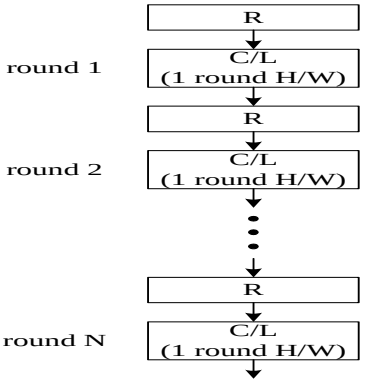
(a) 비공유구조 (b) 공유구조

[그림 6] 하드웨어 공유 구조

또한, 파이프라인(Pipeline) 방식은 대칭키 암호 알고리즘에서 귀환(feedback)이 존재하는 모드에서는 사용할 수 없는 제약이 있지만 귀환이 존재하지 않는 모드를 사용하는 대칭키 프로세서에서 최대의 성능을 제공한다. 그림 7과 같이 각각의 라운드 뒤에 레지스터를 두어 매 클록마다 새로운 블록이 제공될 수 있어서 높은 암/복호율을 갖는 반면 하드웨어 양은 다른 방식에 비해 크게 증가하는 결점이 있다.

3.1. ByteSub / InvByteSub 블록

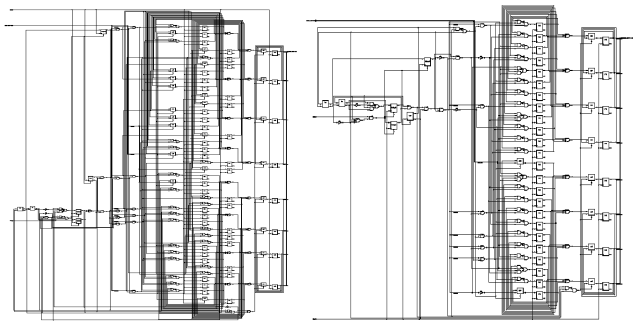
그림 8은 ByteSub 변환과 InvByteSub 변환 중 $in \times M + c$ 연산 회로와 $(in + c) \times M^{-1}$연산 회로의 합성도이다. 곱셈의 역원은 256×8 RAM 룩업 테이블을 이용하여 설계하였다.



[그림 7] 파이프라인 구조

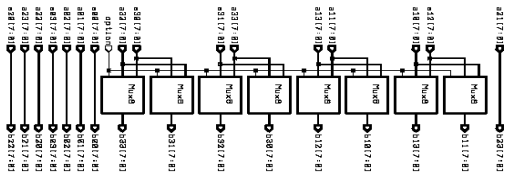
3.2 ShiftRow / InvShiftRow 블록

ShiftRow와 InvShiftRow는 하드웨어 배선과 Mux를 이용하여 설계하였다. 두 변환은 byte 단위로 이루어지며 총 16개의 바이트가 변환된다. 하드웨어 배선 중 8byte는 같고 나머지 8byte 만이 다르기 때문에 배선이 다른 8byte 만을 Mux를 이용하여 선택적으로 연결하였고 나머지 8byte는 직접 연결하였다. 그림 9에 합성도를 나타내었다.



(a) in×M+c 연산 (b) (in+c)×M-1 연산

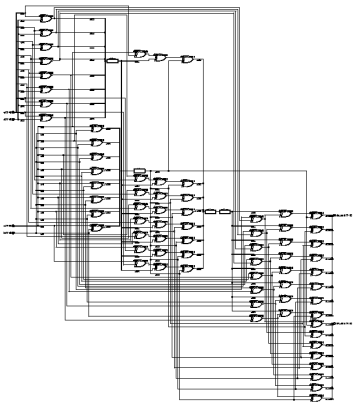
[그림 8] 아핀 변환과 역아핀 변환 블록



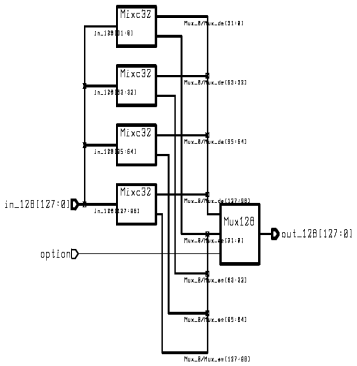
[그림 9] ShitRow와 InvShiftRow 블록

3.3 MixColumn / InvMixColumn 블록

그림 10(a)의 Mixc_byte 블록은 8bit의 입력을 받아 8bit의 두 출력이 나오게 되는데 각각 암호화와 복호화에 쓰이게 된다. 32bit의 입출력을 갖는 4개의 Mixc_word와 option 신호에 의해 암/복호 출력을 선택하는 Mux128을 사용하여 최종 1state를 수행하는 Mixc_state를 그림 10(b)와 같이 설계하였다.



(a) Mixc_byte

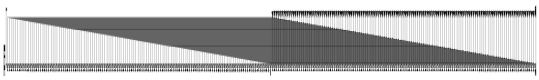


(b) Mixc_state

[그림 10] MixColumn과 InvMixColumn 블록

3.4 AddRoundKey

AddRoundKey는 키와 데이터 두 개의 128 bit 입력을 받아 한 개의 128bit AddRoundKey 연산 결과를 출력한다. 그림 3-11은 설계된AddRoundKey의 합성도이다.



[그림 11] AddRoundKey 블록

3.5 KeyExpansion

KeyExpansion블록은 표 3과 같이 설계하였다.

[표 3] KeyExpansion 블록

8bit xor	1개	Rcon과 sbox 출력과의 연산
32bit xor	4개	word 단위 연산
32bit mux	4개	word 단위 암/복호 data path 선택
rotword	1개	하드웨어 배선으로 설계
sbox	4개	256×8 RAM으로 look-up table

4. 결 론

본 논문에서는 새로운 블록암호 알고리즘의 표준으로 채택된 Rijndael 알고리즘을 작은 면적의 하드웨어로 설계하였다. 기존의 암호화 블록과 복호화 블록을 각각 설계하는 방식과 달리 하드웨어를 공유함으로써 암호화와 복호화를 하나의 블록으로 설계하였다. 또한 키 확장도 하나의 블록에 의해 선택적으로 역확장이 가능한 on-the-fly 방식으로 설계하였다.

본 논문에서 설계된 Rijndael 암호화 칩은 암호블록과 복호블록의 리소스를 공유하여 처리율의 저하 없이 하드웨어 리소스를 감소시킴으로써 전체적인 암호 시스템의 성능을 향상시켰으며 특히 하드웨어가 제한되는 스마트카드, PDA, 휴대폰 등과 같은 고속의 휴대용 임베디드 시스템의 응용에 적합하리라 사료된다.

참 고 문 헌

[1] 류희수 외 1인, “차세대 암호 알고리듬 동향”, ETRI 주간기술동향, 한국 전자통신 연구원, pp. 14-26, 2002

[2] NIST, ″ Advanced Encryption Standard(AES) Development Effort″, http://csrc.nist.gov/encryption/aes/index2.html

[3] NIST, ″AES Hompage″, http://csrc.nist.gov/encryption/aes

[4] 최병윤 외 2인, “차세대 대칭키 암호 알고리즘 AES의 하드웨어 구현 기술”,한국통신학회지, Vol. 19, No. 8, pp. 1262-1272, Aug. 2002

[5] 이선근 외 4인, “자기키 생성 기능을 가진 혼합형 암호 시스템 설계”,CAD 및 VLSI 설계연구회 논문집, pp. 39-42, May. 2002