主題

차세대 대칭키 암호 알고리즘 AES의 하드웨어 구현 기술

최병윤, 박영수, 전성익

차 례

1. 서론

인터넷과 컴퓨터망 기술의 발달에 따라, 정보 공유라는 긍정적인 측면과 정보의 유출 가능성이 높아지는 부정적인 측면이 함께 존재한다. 암호화는 통신 채널을 통해 전달되는 정보 및 데이터를 제삼자가 가로채어 그 내용을 외부로 노출시키거나 의도적으로 내용을 조작·변경하는 등의 보안 공격으로부터 정보를 보호하기 위한 수단으로 사용된다[1]. 암호 알고리즘은 크게 암·복호화 키가 서로 동일한 비밀키(대칭키) 암호 알고리즘과 두 키가 서로 다른 공개키(비대칭키) 알고리즘으로 나뉜다. 두 가지 방식은 키 관리, 동작 속도 등에서 상반되는 장·단점이 존재하므로, 두 가지 방식이 접목되어 보안 시스템을 구현하게 된다. 대표적인 대칭키 알고리즘은 1977년 미국 연방 표준으로 채택되어 지금까지 널리 사용되고 있는 DES(Data Encryption Standard)가 있다. 그러나 DES는 56 비트의 짧은 키를 사용하므로 현재 컴퓨터의 뛰어난 계산 속도로 짧은 시간 안에 해독이 가능하다. 이와 같은 안전성에 대한 문제로 인해 미국 상무부 기술 표준국 NIST(National Institute of Standards and Technology)는 1997년에 DES를 대체할 새로운 표준 블록 알고리즘을 선정하기 위해 AES(Advanced Encryption Standard) 공모를 발표하였다. 3 단계의 평가 과정을 통해 2000년 10월 벨기에 Proton World International(PWI)사의 연구원인 John Daemen과 K.U.Leuven 대학의 Vincent Rijmen이 제안한 Rijndael 알고리즘을 최종 AES 알고리즘으로 채택하였다[2]. AES 알고리즘은 128-비트의 고정된 블록 크기에 대해 암호 키 길이로 128, 192, 256 비트를 지원하며, 보안성, 효율성, 융통성의 동작 특성을 갖추고 있다. 키 길이에 따라 AES-128, AES-192, AES-256으로 구분되며, 3 가지 키는 응용 분야에 따라 선택적으로 구현할 수 있도록 하고 있다. 그리고 알려진 모든 공격에 강하고 취약 키가 존재하지 않는 특징을 갖고 있다. AES 알고리즘으로 채택된 Rijndael은 소프트웨어 구현,

FPGA(Field Programmable Gate Array) 구현, 스마트 카드 구현, 대규모 집적회로 구현 평가를 통해, 가장 적합한 대칭키 암호 알고리즘으로 평가를 받았다[3].

암호 시스템은 크게 나누어 소프트웨어 구현과 하드웨어 구현으로 구현된다. 소프트웨어 구현은 암호 시스템간의 이식성과 유연성이 좋다는 장점이 있으나, 동작 속도가 느리고 해킹에 의한 키의 노출 가능성이 있어서 물리적인 안전성이 완벽히 보장되지 않는다는 단점이 있다. 반면에 하드웨어 구현은 외부의 침입자에 의한 암호 알고리즘과 암호 키의 노출 및 조작이 불가능하므로 물리적인 안전성이 보장된다는 장점을 갖는다.

현재 네트워 분야, 모바일 통신 등의 고속 정보 서비스 확대와 함께 물리적인 안전성이 강조되면서, 전용 하드웨어를 이용한 실시간 암호 시스템의 개발이 요구되고 있다.

본 고에서는 AES 알고리즘의 암호 및 복호 알고리즘을 설명한 후, 대칭키 암호 알고리즘의 하드웨어 구현 구조, AES 알고리즘의 각 모듈의 하드웨어 구현 기술을 서술하였다.

2. AES 알고리즘

DES를 비롯한 대부분의 대칭키 암호 시스템들은 Feistel 구조의 라운드 변환을 기반으로 하는데 비해, AES 암호 알고리즘은 Feistel 구조를 채택하지 않으며, 4개의 독립된 역변환 가능한 라운드 변환으로 구성된다. AES 알고리즘은 블록 길이를 128 비트로 고정하고, 3가지 키 길이 128, 192, 256 비트를 사용한다. 키 길이에 따라 AES-128, AES-192, AES-256으로 불리며, 암호에 필요한 라운드 수는 키 길이에 따라 각각 10, 12, 14 라운드로 다른 값을 갖는다. 표 1은 키 길이와 필요한 라운드 수 간의 관계를 나타낸다. 여기서 Nb, Nk는 블록 및 키 길이를 나타내며 32로 나눈 값이다.

표 1. 블록 길이와 키 길이에 따른 라운드 수

Nr(라운드 수)		블록 길이 Nb=4(128-bit)	AES 알고리즘
키 길이	Nk=4(128-bit)	10	AES-128
	Nk=6(192-bit)	12	AES-192
	Nk=8(256-bit)	14	AES-256

Rijndael에서는 처음 입력 데이터 값은 4행 × 4열의 2차원 배열로 구성된다. 2차원 매핑시 배열 순서는 그림 1과 같이 열 우선 순으로 채워진다.

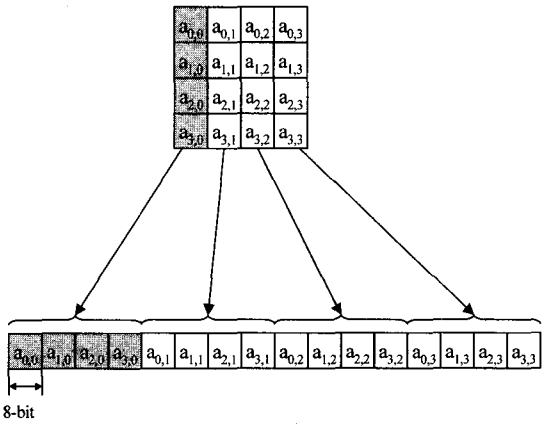

그림 1. 입력 데이터의 매핑

AES 대칭키 암호 알고리즘의 연산 처리 과정은 그림 2와 같이 초기 라운드 키 가산(AddRoundKey)후에 (Nr-1)번의 반복 라운드 및 최종 라운드의 순서로 처리된다. 최종 라운드를 제외한 각 라운드는 ByteSub, ShiftRow, MixColumn 및 AddRoundKey 등의 라운드 변환동작으로 구성된다. 이러한 라운드 변환 동작은 외부에서 주어진 1차원 형태의 128 비트 블록을 2차원의 4행 × 4열로 구성되는 State(128 비트 데이터를 4행 × 4열 바이트의 2차원 배열로 만든 것을 State라 함)에 대해 다음과 같은 연산을 수행한다.

첫째, ByteSub 변환은 그림 3에서 보는 바와 같

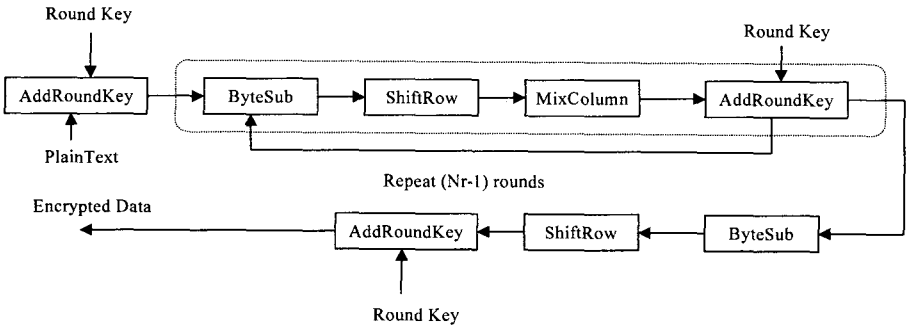

그림 2. AES 암호 알고리즘의 연산 처리 과정

이 State를 구성하는 각각의 바이트에 대해 서로 독립적인 비선형 치환을 한다. 여기에 사용되는 치환 테이블은 S-box라 하며, 각각의 8 비트 입력 값에 대해 2단계 동작으로 비선형 변환 동작을 구현한다. 첫 번째 단계에서 $GF(2^8)$상에서 각각의 8 비트 입력에 대해 곱셈에 대한 역원(multiplicative inverse)을 구한 후, 두 번째 단계에서는 식(1)과 같은 아핀(affine) 변환을 행한다.

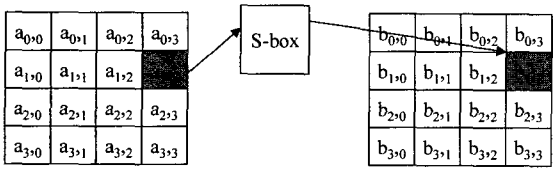

그림 3. ByteSub 변환

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (1)$$

둘째, ShiftRow 변환은 그림 4와 같이 첫째 행을 제외한 나머지 행들은 각각 1, 2, 3 바이트씩 왼쪽으로 순환이동 동작을 수행한다. 여기서 $a_{i,j}$ 와 $b_{i,j}$ 는 바이트 형태의 값을 나타낸다.

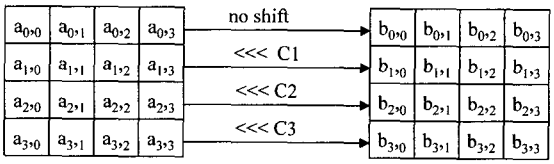

그림 4. ShiftRow 라운드 변환($C_1=1$, $C_2=2$, $C_3=3$)

셋째, MixColumn 라운드 변환은 State의 열(Column)을 유한체 $GF(2^8)$의 다항식으로 생각하여, 그림 5와 같은 b(x) = c(x) ⊗ a(x) 형태의 다항식 곱셈으로 연산한다. 여기서 c(x)는 $c(x)=\{03\}x^3+\{01\}x^2+\{01\}x^3+\{02\}$ 형태의 고정된 계수 값을 갖는다. 여기서 상수 값은 8 비트 16진 값을 나타낸다. 마지막으로 AddRoundKey 변환은 State 값과 라운드 키간의 XOR 연산으로 처리된다.

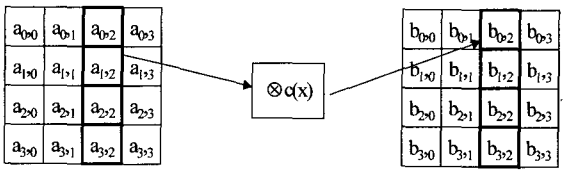

그림 5. MixColumn 라운드 변환

AES 알고리즘의 라운드 동작에 필요한 라운드 키는 라운드 키 생성 알고리즘을 사용하여 만들어진다.

그림 6은 키 길이가 128 또는 192인 경우에 대한 라운드 키 생성 알고리즘이다. 키 길이가 256인 경우에 대한 키 생성 알고리즘은 약간 변형된 구조를 갖고 있다. 이러한 라운드 키는 외부에서 오프라인(offline) 방식으로 생성되거나, 또는 암호 알고리즘을 수행하는 과정에 온라인(on-the-fly) 방식으로 생성될 수 있다. 라운드 키의 총 워드 수는 Nb × (Nr+1)이며 라운드 키 생성은 그림 7과 같이 키 확장(Key Expansion) 과 키 선택(Key Selection) 동작으로 구성된다.

복호화의 경우에는 그림 8과 같이 그림 2의 역 과정을 수행한다. 복호 과정의 경우 적용되는 라운드 키의 순서도 암호화의 반대가 되고, 라운드를 구성하는 4개의 변환들도 각 변환의 역(inverse)이 된다. 단, AddRoundKey는 XOR 연산을 하며, 역변환도 동일한 XOR값을 갖는다. 현재 AES 표준안에는 각각의 역 변환의 구현 방안이 공개되어 있다[4].

```
KeyExpansion(byte key[4*Nk]   word W[Nb * (Nr+1)] )
{for (i=0; i < Nk; i++)
        W[i] =(Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3]);
              // 처음 Nk word는 cipher key와 동일
   for (i=Nk; i < Nb * (Nr+1); i++)
      {  temp = W[i-1];  // 바로 이전 word 정보
         if (i % Nk == 0) // Nk word마다 temp값의 변화 처리 필요
              temp = SubByte(RotByte(temp)) ^ Rcon[i/Nk];
         W[i] = W[i-Nk] ^ temp;  // Nk 이전 워드 정보 유지 필요
      }
}
```

그림 6. 128(Nk=4), 192(Nk=6) 비트 키에 대한 키 확장 알고리즘

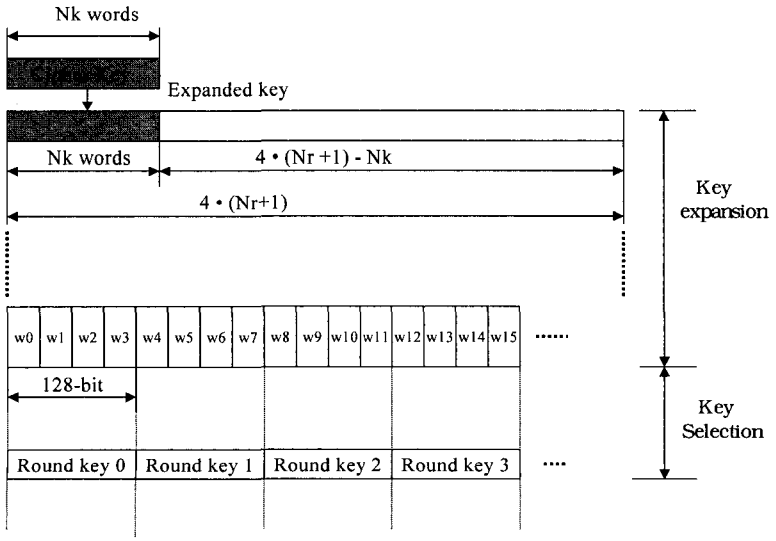

그림 7. 확장된 (Nr+1) 개의 키 와 키 선택 기법

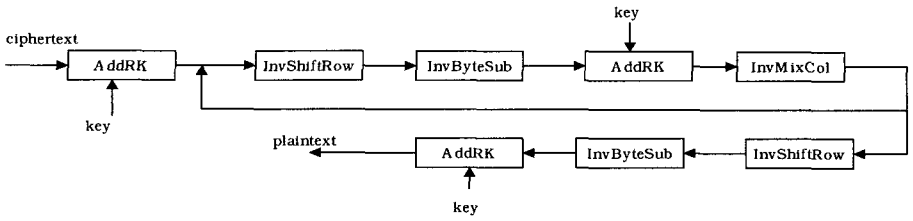

그림 8. 복호 동작에 대한 연산 흐름도

3. 대칭키 암호 알고리즘의 구현 기법

본 장에서는 AES 알고리즘을 구현하기 위한 연산 기법을 기술한다[5,6,7]. 본 장의 내용은 AES에 국한되지 않고 일반적인 대칭키 알고리즘의 하드웨어 구현에도 적용된다. 일반적인 대칭키 블록 암호 프로세서의 구조는 그림 9와 같다. 그림에서 키 스케줄링(key scheduling)과 내부 키 저장 메모리(Memory of internal key)는 라운드 키 생성 방식에 따라 1 가지 회로만 존재한다. 즉 암·복호 동작 중에 라운드 키를 생성하는 온라인(on-the-fly)방식의 경우 키 스케줄링 하드웨어만 존재하고, 내부 키(라운드 키) 저장용 메모리는 존재하지 않는다.

3.1 라운드 키 생성 방식

대칭 키 암호 프로세서에서 라운드 키를 생성하는 방식은 크게 외부에서 모든 라운드에 대한 라운드 키를 계산해서 내부 메모리에 저장해 두는 오프라인(offline) 방식과 외부에서는 암호 키만 제공해주면 암호 및 복호 동작 진행 중에 라운드 키를 생성하는 온라인(on-the-fly) 방식으로 나뉜다. 2가지 방식의 특징을 비교하면 표 2와 같다. 표에서 키 설정 시간은 암호 키가 바뀔 경우 프로세서 내부 키를 초기화해주는 것과 관련된 오버헤드(overhead)를 나타낸다. 표 2를 분석하면 키가 자주 바뀌는 응용의 경우 온라인 방식이 바람직하며, 키가 자주 바뀌지 않고, 저 전력 소모 특성이 필요한 분야에서는 오프라인 방식이 적절함을 알 수 있다.

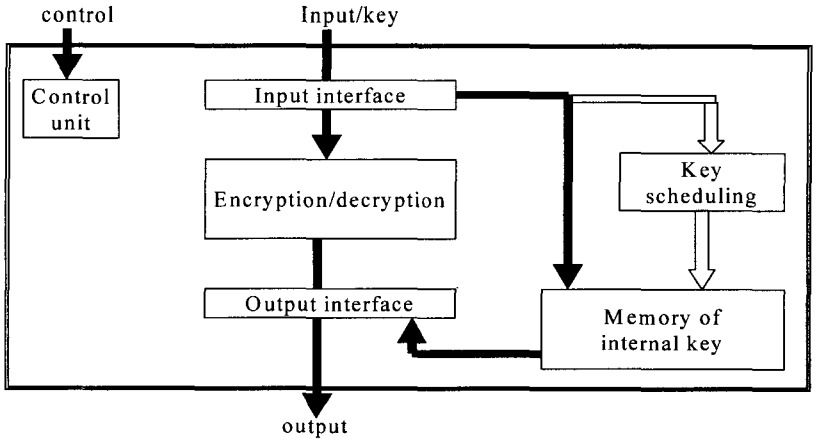

그림 9. 대칭키 블록 암호 프로세서 구조

표 2. 라운드 키 생성 방식의 특성 비교

	오프 라인 (offline)	온라인 (on-the-fly)
기억 메모리	large	small
라운드 키 생성회로	no	yes
전력 소모	small	large
키 설정 지연시간	large	small

3.2 라운드 연산 구조

대칭키 암호 알고리즘은 모두 라운드 연산을 반복해서 암호 및 복호 동작을 구현하기 때문에 암호 및 복호 동작을 구현할 때 라운드를 처리하는 구조는 암호 및 복호율을 결정하는데 큰 영향을 미친다. 따라서 대칭키 프로세서 설계자는 응용 분야에서 요구하는 면적과 동작 속도의 사양에 바탕을 두고 적절한 라운드 구현 구조를 선택해야 한다. 본 고에서는 기본적인 반복구조, 루프 펼침(loop unrolling), 자원 공유(Resource Sharing), 그리고 파이프라인 구조를 살펴본다. 일반적으로 암호 프로세서의 성능을 결정하는 요소는 크게 암·복호 율(throughput), 암호 및 복호 지연시간(latency) 그리고 면적(하드웨어 크기) 등이 있다. 여기서 latency는 하나의 입력 블록이 암호 프로세서에 들어가서 출력으로 나올 때까지의 소요 시간을 나타내며, 암·복호 율과의 관계는 식 (2)과 같다. 동시에 처리되는 블록수는 파이프라인 구조에서의 병렬 연산 특성을 반영하는 항이다.

암호율(throughput) =
(블록 크기 × 동시에 처리되는 블록 수) / latency (2)

가. 기본적인 반복(Iterative) 구조

대칭키 암호 알고리즘의 가장 일반적인 하드웨어 구현 방식으로 그림 10과 같이 하나의 라운드 하드웨어를 갖추고, 이를 반복 활용해서 전체 암호 및 복호 동작을 구현하는 방식이다. 여기서 라운드 하드웨어에 1 클록의 연산 시간이 할당된다. latency와 암·복호율은 식 (3)과 같다.

지연 시간 = NT
암·복호율 = 블록 크기 / NT (3)
여기서 N은 라운드 수, T는 클록 주기

나. 루프 펼침(Loop Unrolling) 구조

그림 10과 유사한 반복 구조이지만, 그림 11과 같이 단일 클록에 여러 개의 라운드를 동시에 수행하는 구조이다. 그림에서 동시에 처리되는 라운드 수 k는 N의 약수가 되어야 한다. 이러한 방식은 최대 1 클록에 모든 라운드를 처리하는 구조로 확대될 수 있다. 그러나 이 방식은 암·복호 동작에 소요되는 클록 수는 줄이지만 클록 주기를 k 배 증가시키므로 암·복호 율 특성이 거의 변화가 없다. 루프 펼침 구조의 성능은 다음과 같다. 식(4)의 T는 기본 반복 구조의 클록 주기를 나타낸다.

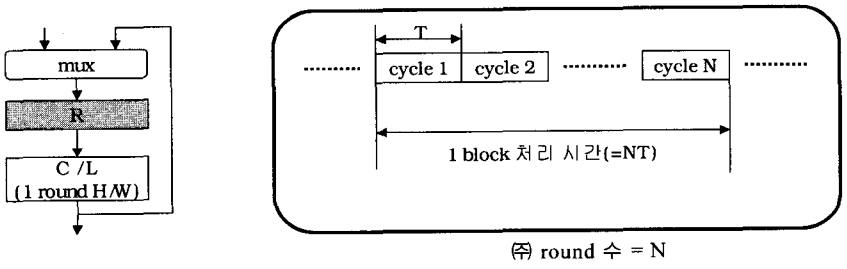

그림 10. 기본적인 반복 구조

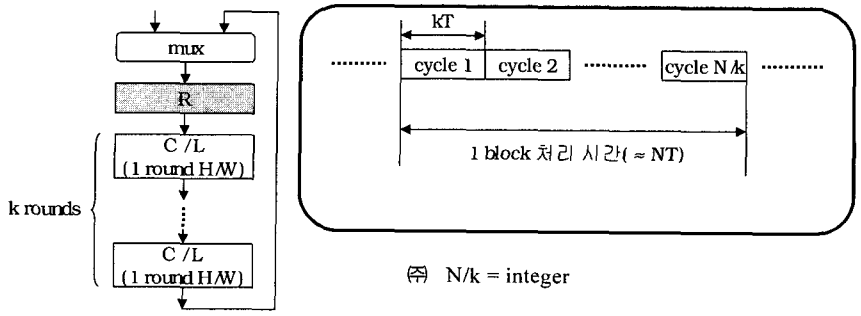

그림 11. 루프 펼침(Loop Unrolling) 구조

지연 시간(latency) ≒ (N/k) × (kT) = NT
암·복호율 ≒ 블록 크기 / (N/k × kT) =
블록 크기 / NT (4)

다. 자원 공유(Resource Sharing) 구조

이 방식은 하나의 라운드 연산에 동일한 연산 모듈이 병렬로 사용되는 경우에 적합한 방식으로, 성능은 감소하지만 하드웨어 면적을 최소화하는 설계에 널리 사용된다. 그림 12와 같이 하나의 라운드에 2개의 F 함수가 포함되어 있는 경우 하나의 F 함수만 하드웨어로 갖추고 이를 반복해서 활용하는 방식이다. 이렇게 하면 라운드에 소요되는 클록 수는 2배로 증가하지만 면적은 약 1/2로 감소하는 특징이 있다. 따라서 이러한 구조는 각 라운드가 바이트 단위의 병렬 연산으로 구성된 AES 알고리즘에 적용되어 면적 최소화

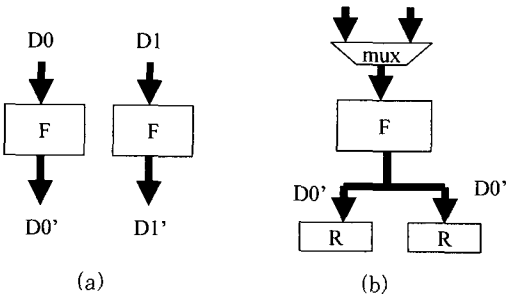

그림 12. 자원 공유(Resource Sharing) 구조
(a). 자원이 공유되지 않은 구조
(b). 자원이 공유된 구조

설계를 할 수 있다. 각 라운드는 반복 구조이고 라운드 내부에 자원 공유(공유도 m=2)가 적용된 경우 성능은 식(5)과 같다. 라운드 당 클록 수가 m배 증가하여 되어, 성능이 (1/m)로 감소한다.

지연 시간(latency) = (N) × (T) × m = mNT
암·복호율 = (블록 크기) / (N × T × m)
= (블록 크기) / mNT (5)

라. 파이프라인 구조

이 방식은 대칭키 암호 알고리즘에서 CBC (Cipher Block Chaining)과 같은 귀환(feedback)이 존재하는 모드에서는 사용할 수 없는 제약이 있지만, ECB(Electronic Codebook) 방식만을 사용하는 대칭키 프로세서에서 최대의 성능을 제공한다. 그림 13과 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. 파이프라인 구조의 성능은 식 (6)과 같다. 동시에 처리되는 블록 수가 N이므로 이론적인 최대 성능은 (블록 크기)/T이다.

지연 시간(latency) = (N) × (T) = NT
암·복호율 ≒ (블록 크기 × N) / (N × T)
≒ (블록 크기) / T (6)

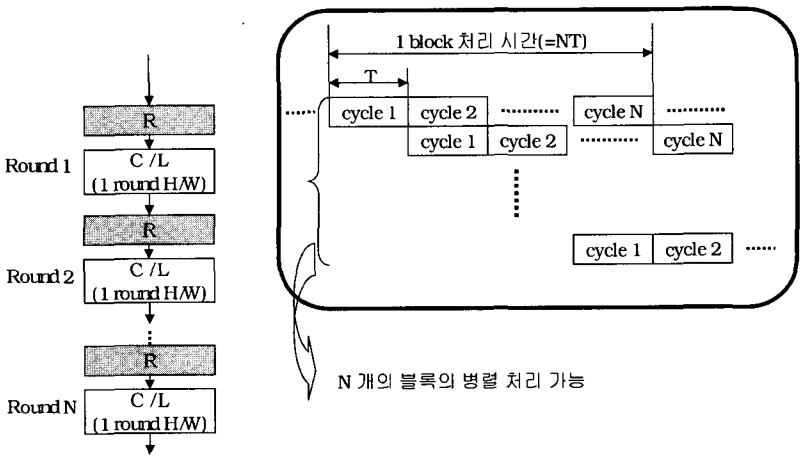

그림 13. 파이프라인 구조

4. AES 알고리즘의 라운드 변환 모듈의 하드웨어 구현 기법

본 장에서는 2장에서 설명한 AES 알고리즘의 개별 라운드 변환을 위한 하드웨어 구현 방안을 소개한다. 단, 연산 구조로 일반적으로 널리 사용하는 그림 10의 기본 반복 구조를 토대로 한다. 즉 매 클록에 하나의 라운드를 처리하므로, 자원 공유와 파이프라인 기법은 사용하지 않는다.

4.1 ByteSub와 InvByteSub 라운드 변환 하드웨어

ByteSub 변환은 그림 14와 같이 16개의 S-box로 구현될 수 있다. 각각의 S-box는 $2^8 \times 8$-bit ROM(Read-only memory)으로 구현된다. 유사하게 InvByteSub 블록은 16개의 SI-box로 구현될 수 있다. 그리고 식(1)의 ByteSub 동작의 두 단계를 독립적인 하드웨어로 구현하는 방안도 있다. 이 방식을 이용하면 ByteSub 와 InvByteSub 연산시 곱셈의 역원을 계산하는 룩업 테이블(look-up table)을 공유해서 사용할 수 있다는 장점이 있다.

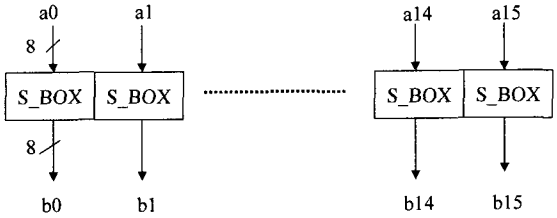

그림 14. 16개의 S-box로 구현된 ByteSub 변환

4.2 ShiftRow와 InvShiftRow 라운드 변환 하드웨어

이러한 두 변환은 별도의 하드웨어가 필요치 않고, 단순한 배선만으로 하드웨어가 구현 가능하다. 그림 15는 1 차원의 바이트 배열로 매핑된 입력 state에 대해 단순한 배선으로 구현된 ByteSub 변환을 나타낸다. 유사한 구조로 InvShiftRow 변환도 구현할 수 있다.

4.3 MixColumn과 AddRoundKey 라운드 변환 하드웨어

그림 5의 MixColumn은 식 (7)과 같이 행렬 곱

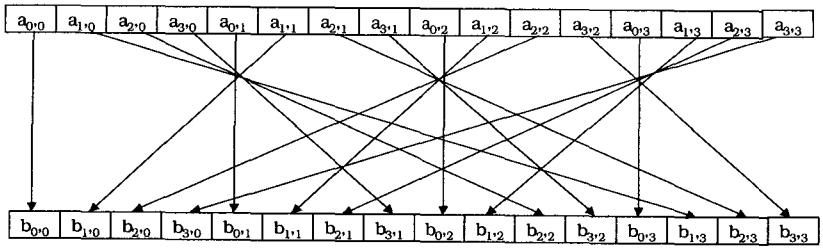

그림 15. ShiftRow 변환을 위한 배선 연결

셈 연산으로 표현될 수 있다. 여기서 ⊗는 $GF(2^8)$ 상의 다항식 곱셈을 나타낸다. 16진수로 표현된 상수 행렬은 분석해보면 각 행(row)간에 1 바이트씩 좌측 순환 이동 형태를 갖고 있음을 알 수 있다. 따라서 그림 16과 같이 하나의 상수 값만 유지하고 이를 적절히 순환 이동하여 4개의 행렬 곱셈에 활용할 수 있다. 단, MixColumn 연산에 사용하는 곱셈은 유한체 상의 곱셈이므로 길이가 확대되지 않는 특성을 갖고 있다. 그리고 상수값만 다를 뿐 동일한 하드웨어를 공유하여 InvMixColumn 연산을 구현할 수 있다. AddRoundKey 변환 모듈은 라운드 키와 비트 단위의 XOR 연산으로 구현된다.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (7)$$

4.4 라운드 키 생성회로

라운드 구조로 기본적인 반복 구조를 사용할 경우 그림 6의 라운드 키 생성 알고리즘도 매 라운드에 128 비트의 라운드 키만 생성하는 반복구조로 설계 가능하다. 그림 17은 키 길이가 128 비트인 경우 매 클록마다 128 비트 라운드 키를 생성하는 기법을 나타낸다. 그림에서 직렬로 연결된 XOR 모듈은 병렬 연산 기법을 사용하여 지연 시간을 축소하는 것이 가능하다. 복호 동작시 라운드 키의 온라인 계산 기법은 그림 17의 역 과정을 사용해서 구현 가능하다. 이러한 역동작은 A ⊕ B = C는 B ⊕ C=A라는 XOR 연산 특성을 활용하여 쉽게 회로로 구현할 수 있다. 단, 복호 동작의 온라인 라운드 키 생성 회로 설계시 시작 라운드 키 값(암호 동작의 마지막 라운드 키)은 외부에서 제공되는 암호 키에서 직접적으로

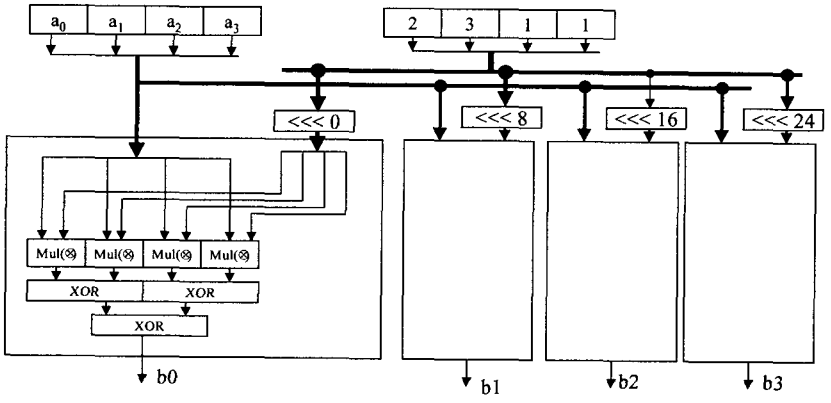

그림 16. MixColumn 연산용 하드웨어 구조(1 column 만 표현)

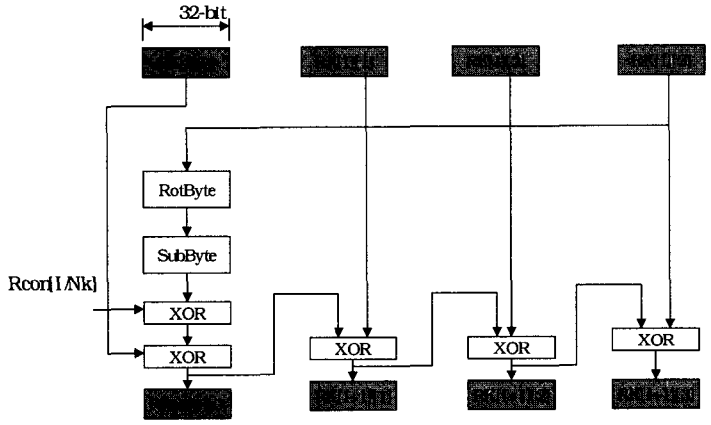

그림 17. 라운드 키를 생성 기법
(RK[I] : I 번째 라운드 키, RK[I+1] : (I+1)번째 라운드 키)

얻을 수 없기 때문에, 암호 동작을 통해 마지막 라운드 키를 계산해서 레지스터에 저장해두는 동작이 한 번 필요하다.

5. 결론

지금까지 차세대 대칭키 암호 알고리즘으로 채택된 AES(Advanced Encryption Standard) 알고리즘의 동작을 소개하고, 알고리즘을 하드웨어로 구현하기 위한 라운드 구조 구현 방식을 분석해보았다. 그리고 AES 알고리즘을 구성하는 내부 라운드 변환 모듈을 하드웨어로 구현하는 방안을 기술하였다. 기술한 하드웨어 구현 방안은 가장 보편적인 구현을 중심으로 기술하였으므로, 이를 바탕으로 다양한 개선 구조가 가능하다. 특히 AES 알고리즘은 유한체(finite field) 연산을 바탕으로 하기 때문에, 최근에 면적과 동작 속도를 향상시키기 위해 Composite field를 활용하는 연산 방식과 암호 및 복호 하드웨어를 최적으로 공유하는 방안에 대한 연구가 활발하다. 그리고 DES 알고리즘의 표준 4가지 모드(ECB, CBC, CFB, OFB)외에 CTR(counter) 모드와 같은 새로운 동작 모드를 정의하려는 연구가 활발히 진행되고 있다. 그리고 IPsec과 같은 네트워 보안 분야와 연계된 AES 알고리즘의 하드웨어 구현 연구가 필요하다. AES Rijndael 알고리즘은 속도와 하드웨어 구현 측면에서 뛰어난 장점을 갖고 있어, 네트워 보안, 모바일 통신, 스마트 카드 보안 등 다양한 보안 응용 시스템에 활발히 적용될 예정이다. 따라서 AES 알고리즘에 대한 고속 하드웨어 구현 연구와 함께 저 전력 특성을 갖는 구조 연구가 필요하다.

참고문헌

[1] William Stalling, Cryptography and Network Security, Prentice Hall, 1999.
[2] John Daemen and Vincent Rijnmen, The Design of Rijndael, Springerm 2002.
[3] Andreas Dandalis, "A Comparative Study of Performance of AES Final Candidates Using FPGAs", pp.126-141, CHESS '2000.
[4] NIST, "Announcing the Advanced Encryption Standard(AES)", FIPS PUB-197, Nov, 2002, http://www.nist.

gov/aes.

[5] 최 병 윤, "Rijndael 암호 기반의 정보 보호 시스템", IDEC 원격 강좌 네트워크 정보보호 시스템, 경북 대학교 반도체 설계 교육 센터 2002. 3.27-3.29.

[6] Christof Parr, Efficient VLSI Architectures for Bit-Parallel Computations in Galois Fields, Ph.D thesis, Institute for Experimental Mathematics, University of Essen, Essen, Germany, June, 1994.

[7] Henry Kuo and Ingrid Verbauwhede, "An Embedded Cryptographic Processor for the Rijndael AES Algorithm", Annual Research Review, UCLA Internal Report, Sept. 2000. http://www.ee.ucla.edu/~ingrid/Presentations/ARRHenryKuo.pdf.

최 병 윤

1985년 2월 : 연세대학교 전자공학과(공학사), 1992년 8월 : 연세대학교 전자공학과(공학박사), 1997년 8월~1998년 8월 : University of Illinois at UC 방문 연구 교수 1993년~현재 : 동의대학교 부교수 〈주관심 분야〉 RISC 마이크로프로세서 설계, 통신, 네트웍 및 암호 알고리즘의 VLSI 설계

박 영 수

1990년~현재 : 한국 전자통신 연구원 선임 연구원 〈주관심 분야〉 CAD 및 VLSI 설계, 암호 프로세서 설계, IC 카드 설계 등

전 성 익

1987년~현재 : 한국 전자통신 연구원 책임 연구원, IC 카드 연구팀 팀장 〈주관심 분야〉 운영체제, 시스템 소프트웨어, 스마트 카드 기술 등