암호와 복호가 동일한 SPN 블록 암호 SSB

조경연*

SPN Block cipher SSB having same structure in encryption and decryption

Gyeong-Yeon Cho*

요 약

블록 암호는 Feistel 구조와 SPN 구조로 나눌 수 있다. Feistel 구조는 암호 및 복호 알고리즘이 같은 구조이고, SPN 구조는 암호 및 복호 알고리즘이 다르다. 본 논문에서는 암호와 복호 과정이 동일한 SPN 구조 블록 암호 알고리즘인 가칭 SSB를 제안한다. SSB는 짝수 N 라운드로 구성하고, 각 라운드는 라운드 키 덧셈, 치환 계층, 바이트 교환 및 확산 계층으로 구성한다. 치환 계층은 홀수 라운드와 짝수 라운드가 서로 역의 관계를 이룬다. 확산 계층은 MDS 대합 행렬로 구성한다. SSB의 차분 및 선형 공격 확률은 2^{-306}로 AES와 동일하다. 본 논문에서 제안한 암호와 복호가 동일한 SPN 블럭 암호는 하드웨어 구성이 간단한 장점을 가지므로 제한적 하드웨어 및 소프트웨어 환경인 스마트카드와 전자 칩이 내장된 태그와 같은 RFID 환경에서 안전하고 효율적인 암호 시스템을 구성할 수 있다.

ABSTRACT

Feistel and SPN are the two main structures in a block cipher. Feistel is a symmetric structure which has the same structure in encryption and decryption, but SPN is not a symmetric structure. In this paper, we propose a SPN block cipher so called SSB which has a symmetric structure in encryption and decryption. The proposed SSB is composed of the even numbers of N rounds. Each round consists of a round key addition layer, a subsitution layer, a byte exchange layer and a diffusion layer. The subsitution layer of the odd round is inverse function of one of the even round. And the diffusion layer is a MDS involution matrix. The differential and linear attack probability of SSB is 2^{-306} which is same with AES. The proposed symmetric SPN block cipher SSB is believed to construct a safe and efficient cipher in Smart Card and RFID environments which is in limited hardware and software resources.

키워드

AES, ARIA, SPN, 암호, 복호

Key word

AES, ARIA, SPN, Cipher, Decipher

* 정회원 : 부경대학교(주저자, gycho@pknu.ac.kr)

접수일자 : 2011. 01. 05
심사완료일자 : 2011. 03. 23

Ⅰ. 서 론

통신 기술의 발전과 사회의 전반적인 활동이 무선 통신망과 인터넷과 같은 범용 통신망을 이용한 지식기반 정보화 사회로 빠르게 진행함에 따라 정보보호에 대한 인식이 점차 높아지고 있으며, 정보보호를 위한 기술적 대응 조치로 암호화 기술이 발전하고 있다. 또한 암호를 해독하는 공격 기술도 동반 발전하여, 그동안 표준 암호로 널리 사용되어 온 비밀키 블록 암호인 DES(Data Encryption Standard)[1]가 더 이상 안전하지 않게 되었다.

이에 미국을 비롯한 유럽 및 선진국들은 자국의 표준 비밀키 블록 암호 개발에 주력하고 나서게 되었다. 미국은 2000년에 AES(Advanced Encryption Standard)[2] 프로젝트를 통하여 Rijndael 알고리즘[3]을 표준 블록 암호 알고리즘으로 선정했으며, 유럽은 NESSIE(New European Schemes for Signatures. Integrity. and Encryption)[4], 일본은 CRYPTREC(Cryptography Research and Evaluation Committees)[5] 프로젝트를 통해 여러 가지 암호 기반 기술을 선정하고 있다. 우리나라도 SEED[6]와 ARIA[7]를 국가 표준으로 선정하였다.

비밀키 블록 암호는 암호 알고리즘 E에 비밀키 K를 적용하여 평문 P로부터 암호문 $C = E_K(P)$를 생성하며, 복호는 복호 알고리즘 D에 동일한 비밀키 K를 적용하여 암호문 C로부터 평문 $P = D_K(C)$를 얻는다. 이러한 비밀키 블록 암호의 암호 및 복호 알고리즘은 암호 및 복호 함수를 여러 라운드 반복 수행하는 구조이다. 비밀키 블록 암호는 라운드 함수의 구조에 따라서 Feistel 구조[8]와 SPN(Substitution Permutation Network) 구조로 나눌 수 있다.

SPN 구조는 C. E Shannon의 혼돈과 확산[9] 이론을 바탕으로 하였다. SPN 구조에서의 암호 라운드 함수는 키합산층과 혼돈을 수행하는 치환층 S 및 확산층 P의 세 단계로 구성된다. 복호 라운드 함수는 역확산층 P^{-1}과 역치환층 S^{-1} 및 키합산층의 세 단계로 구성한다. SPN 구조는 암호와 복호 알고리즘이 다르므로 구현이 복잡해지지만 각 라운드에서 입력이 전부 비선형 변환되므로 Feistel 구조에 비하여 라운드의 수가 적어진다. 따라서 하드웨어의 동작 속도가 빠른 장점을 가진다.

SPN 구조 블록 암호로는 AES, Hierocrypt-3[10], ARIA 등이 있다. ARIA는 암호 및 복호 알고리즘이 동일한 SPN 구조이다. ARIA는 치환층과 역치환층을 네 가지 S-박스$\{S1, S2, S1^{-1}, S2^{-1}\}$를 적절하게 조합하여 구성하였으며, 치환층을 대합 함수로 구성하여 치환층과 역치환층이 동일하다. ARIA의 치환층은 바이트 단위의 16 X 16 행렬식으로 구성되어 있으므로 소프트웨어로 구현하면 동작 속도가 AES 및 Hierocrypt-3에 비하여 크게 느려지는 단점이 있다.

본 논문에서는 암호와 복호 과정이 동일하며, AES와 동일한 하드웨어 및 소프트웨어 구현 복잡도 및 안전성을 가지는 SPN 구조 블록 암호 알고리즘인 가칭 SSB(Symmetric SPN Block cipher)를 제안한다. SSB는 짝수 N 라운드로 구성하고, 각 라운드는 라운드 키 덧셈 계층, 8 비트 S-박스에 의한 치환 계층, 바이트를 교환하는 교환 계층 그리고 MDS(Maximum Distance Separated) 대합 행렬로 구성한 확산 계층으로 구성한다. 치환 계층의 S-박스는 안전성이 검증된 AES의 S-박스를 사용한다. 홀수 라운드의 치환 계층은 S-박스와 S^{-1}-박스의 순서로 구성하고, 짝수 라운드의 치환 계층은 이와 반대 순서로 구성한다. 이렇게 구성하므로써 홀수 라운드 치환 계층은 짝수 라운드 치환 계층의 역함수가 된다.

본 논문에서 제안하는 블록 암호 SSB는 암호와 복호 알고리즘이 동일하며, 적용하는 라운드 키는 상호 역순으로 되어있다.

본 논문의 구성은 2장에서 확산 계층을 구성하는 MDS 대합 행렬의 구조를 기술하고, 3장에서 SSB의 구조를 기술한다. 4장에서는 SSB의 안전성을 검증하고, 5장에서 결론을 맺는다.

Ⅱ. MDS 대합 행렬

MDS 코드[11]를 암호 구조의 선형 변환 확산층에 사용하는 것은 Vaudenay[12]에 의하여 제안되었고, SHARK[13], SQUARE[14]에 채용되었으며 AES의 MixColumn() 및 InvMixColumn()도 MDS 코드를 구성되어 있다. 선형 코드 (n, k, d)에서 'd=n-k+1'일 때 이 코드

가 MDS 코드를 구성하는 조건은 다음의 lemma-1으로 정의된다.

Lemma-1[11] : 생성 행렬 $G=[I|A]$를 가지는 (n, k, d) 코드가 MDS이기 위한 필요충분조건은 모든 부분 정방 행렬이 비특이인 경우이다. 단, A는 'k * (n-k)' 행렬이다.

MDS 대합 행렬은 lemma-2로 구성할 수 있다.

Lemma-2[11] : $x_0,\cdots,x_{n-1}$과 $y_0,\cdots,y_{n-1}$이 주어졌을 때, 행렬 $A=[a_{ij}]$, $0\le i,j\le n-1$, $a_{ij}=\frac{1}{x_i\oplus y_j}$을 Cauchy 행렬이라고 한다. 단. 모든 i, j에 대하여 '$x_i\neq x_j$', '$y_i\neq y_j$', '$x_i\neq y_j$'이다. Cauchy 행렬 A는 $A^2=c^2I$, $c=\bigoplus_{i=1}^{n} a_{ij}^2$이 된다. 따라서 행렬 A의 모든 원소 a_{ij}를 $\sqrt{c}=\bigoplus_{i=1}^{n} a_{1i}$로 나누어 생성한 행렬은 MDS 대합 행렬이 된다.

본 논문에서는 lemma-2에 의하여 4X4 MDS 대합 행렬을 다음과 같이 정의한다.

$$M=\begin{bmatrix} 0x1b & 0x1c & 0x14 & 0x12 \\ 0x1c & 0x1b & 0x12 & 0x14 \\ 0x14 & 0x12 & 0x1b & 0x1c \\ 0x12 & 0x14 & 0x1c & 0x1b \end{bmatrix}$$

Ⅲ. SSB의 구조

본 논문에서는 AES의 표준인 FIPS-197[15]에서 정의한 기호를 사용한다. SSB의 주요 사항은 AES와 동일하게 구성한다. 즉, 기약다항식은 AES와 동일한 $m(x)=x^8+x^4+x^3+x+1$을 사용한다, 블록 크기는 128 비트이며, 이는 16 바이트 $\{a_0, a_1,\cdots, a_{15}\}$로 구성한다. 내부 연산은 '4 X 4' 바이트 정방 행렬인 스테이트(state) 단위로 수행한다. 키는 128, 192 및 256 비트 길이이며, 키 길이에 따라서 10, 12 및 14 라운드를 수행한다.

```
SSB_Encyption(byte in[16], byte out[16], byte
erk[NROUND+1][16])
 begin
  byte state[16]

  state = in ;

  AddRoundKey(state, erk[0])

  for round = 1 to NROUND step 2
   SubByteOdd(state)
   ExgRow(state)
   MixColumn(state)
   AddRoundKey(state, erk[round])

   SubByteEven(state)
   ExgRow(state)
   if (round != NROUND-1)
     MixColumn(state)
   AddRoundKey(state, erk[round+1])
  end for

  out = state ;
 end
```

그림 1. SSB 암호 의사 코드
Fig. 1 Pseudo code for SSB encryption

그림-1에 SSB 암호 의사 코드를 보인다. NROUND는 128비트, 192비트 및 256비트 키에서 각각 10, 12 및 14로 라운드 수이다. 그림-1의 각 함수의 정의는 다음과 같다.

- AddRoundKey() 함수는 라운드 키 덧셈 함수이다.
- SubByteOdd() 함수는 홀수 라운드에서의 8비트 S-박스를 이용한 비선형 바이트 치환 함수이다. 그림-2에 함수의 동작을 보인다. 그림-2에서 $S[x]$와 $S^{-1}[x]$는 입력 x를 S-박스 및 S^{-1}-박스에 의하여 각각 비선형 변환

한 것을 나타낸다. S-박스와 S^{-1}-박스는 AES에서 정의한 것을 사용한다.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

→

$S[s_{0,0}]$	$S[s_{0,1}]$	$S[s_{0,2}]$	$S[s_{0,3}]$
$S^{-1}[s_{1,0}]$	$S^{-1}[s_{1,1}]$	$S^{-1}[s_{1,2}]$	$S^{-1}[s_{1,3}]$
$S[s_{2,0}]$	$S[s_{2,1}]$	$S[s_{2,2}]$	$S[s_{2,3}]$
$S^{-1}[s_{3,0}]$	$S^{-1}[s_{3,1}]$	$S^{-1}[s_{3,2}]$	$S^{-1}[s_{3,3}]$

그림 2. SubByteOdd() 함수 동작
Fig. 2 SubByteOdd() function operation

• SubByteEven() 함수는 짝수 라운드에서의 8비트 S-박스를 이용한 비선형 바이트 치환 함수이다. 그림-3에 함수의 동작을 보인다.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

→

$S^{-1}[s_{0,0}]$	$S^{-1}[s_{0,1}]$	$S^{-1}[s_{0,2}]$	$S^{-1}[s_{0,3}]$
$S[s_{1,0}]$	$S[s_{1,1}]$	$S[s_{1,2}]$	$S[s_{1,3}]$
$S^{-1}[s_{2,0}]$	$S^{-1}[s_{2,1}]$	$S^{-1}[s_{2,2}]$	$S^{-1}[s_{2,3}]$
$S[s_{3,0}]$	$S[s_{3,1}]$	$S[s_{3,2}]$	$S[s_{3,3}]$

그림 3. SubByteEven() 함수의 동작
Fig. 3 SubByteEven() function operation

• ExgRow() 함수는 스테이트의 바이트를 그림-4와 같이 교환한다.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

→

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,0}$	$s_{1,3}$	$s_{1,2}$
$s_{2,3}$	$s_{2,2}$	$s_{2,1}$	$s_{2,0}$
$s_{3,2}$	$s_{3,3}$	$s_{3,0}$	$s_{3,1}$

그림 4. ExgRow() 함수 동작
Fig. 4 ExgRow() function operation

첫 번째 행은 교환하지 않으며, 두 번째 행은 (0,1)과 (2,3)을 교환하고, 세 번째 행은 (0,3)과 (1,2)를 교환하며, 네 번째 행은 (0,2)와 (1,3)을 교환한다.

• MixColumn() 함수는 열 단위로 혼합을 수행하는 48-비트 선형 변환 함수이다. 2장에서 정의한 MDS 대합 행렬을 사용하여, 그림-5와 같이 동작한다.

$$\begin{bmatrix} S'_{0,n} \\ S'_{1,n} \\ S'_{2,n} \\ S'_{3,n} \end{bmatrix} = \begin{vmatrix} 0x1b & 0x1c & 0x14 & 0x12 \\ 0x1c & 0x1b & 0x12 & 0x14 \\ 0x14 & 0x12 & 0x1b & 0x1c \\ 0x12 & 0x14 & 0x1c & 0x1b \end{vmatrix} \begin{vmatrix} S_{0,n} \\ S_{1,n} \\ S_{2,n} \\ S_{3,n} \end{vmatrix}$$

그림 5. MixColumn() 함수 동작
Fig. 5 MixColumn() function operation

그림-1을 소프트웨어로 효과적으로 구현하기 위해서 256 X 32 비트 테이블 T0-T7를 다음과 같이 정의한다.

$$T0[a] = \begin{bmatrix} S[a]*0x1b \\ S^{-1}[a]*0x1c \\ S[a]*0x14 \\ S^{-1}[a]*0x12 \end{bmatrix} \quad T1[a] = \begin{bmatrix} S[a]*0x1c \\ S^{-1}[a]*0x1b \\ S[a]*0x12 \\ S^{-1}[a]*0x14 \end{bmatrix}$$

$$T2[a] = \begin{bmatrix} S[a]*0x14 \\ S^{-1}[a]*0x12 \\ S[a]*0x1b \\ S^{-1}[a]*0x1c \end{bmatrix} \quad T3[a] = \begin{bmatrix} S[a]*0x12 \\ S^{-1}[a]*0x14 \\ S[a]*0x1c \\ S^{-1}[a]*0x1b \end{bmatrix}$$

$$T4[a] = \begin{bmatrix} S^{-1}[a]*0x1b \\ S[a]*0x1c \\ S^{-1}[a]*0x14 \\ S[a]*0x12 \end{bmatrix} \quad T5[a] = \begin{bmatrix} S^{-1}[a]*0x1c \\ S[a]*0x1b \\ S^{-1}[a]*0x12 \\ S[a]*0x14 \end{bmatrix}$$

$$T6[a] = \begin{bmatrix} S^{-1}[a]*0x14 \\ S[a]*0x12 \\ S^{-1}[a]*0x1b \\ S[a]*0x1c \end{bmatrix} \quad T7[a] = \begin{bmatrix} S^{-1}[a]*0x12 \\ S[a]*0x14 \\ S^{-1}[a]*0x1c \\ S[a]*0x1b \end{bmatrix}$$

T0-T3 테이블은 홀수 라운드에서 사용하여 SubByteOdd(), ExgRow(), MixColumn()을 4번의 테이블 참조와 3번의 XOR 연산으로 구현할 수 있다. T4-T7 테이블은 짝수 라운드에서 사용하여 SubByteEven(), ExgRow(), MixColumn()을 4번의 테이블 참조와 3번의 XOR 연산으로 구현할 수 있다.

```
SSB_Decyption(byte in[16], byte out[16], byte
drk[NROUND+1][16])
 begin
  byte state[16]

  state = in ;

  AddRoundKey(state, drk[0])

  for round = 1 to NROUND step 2
   SubByteOdd(state)
   ExgRow(state)
   MixColumn(state)
   AddRoundKey(state, drk[round])

   SubByteEven(state)
   ExgRow(state)
   if (round != NROUND-1)
     MixColumn(state)
   AddRoundKey(state, drk[round+1])
  end for

  out = state ;
 end
```

그림 6. SSB 복호 의사 코드
Fig. 6 Pseudo code for SSB decryption

SSB의 복호는 암호와 동일하며 적용하는 라운드 키만이 다르다. SSB의 복호 의사 코드를 그림-6에 보인다. 라운드 키는 AES와 유사하게 생성한다. AES에서는 4개의 S-박스를 사용했으나, SSB에서는 $\{S, S^{-1}, S, S^{-1}\}$를 사용한다. AES와 동일하게 생성한 라운드 키 {erk[0], erk[1],. erk[NROUND]}가 암호 라운드 키이다.

MixColumn()은 선형 변환 함수이므로 '$M(A \oplus B) = MA \oplus MB$'가 성립한다. 따라서 복호 라운드 키 {drk[0], drk[1],. drk[NROUND]}는 다음과 같이 생성한다.

```
drk[0] = erk[NROUND],
drk[1] = MixCloumn(erk[NROUND-1]).
                    .....
drk[NROUND-1] = MixColumn(erk[1]),
drk[NROUND] = erk[0].
```

Ⅳ. SSB의 성능 및 안전성

그림-1과 그림-6의 SSB 암호와 복호 의사코드를 비교하면 라운드 키만 다르다는 것을 알 수 있다. 따라서 SSB는 암호와 복호 알고리즘이 동일한 SPN 블록 암호이다.

SSB를 소프트웨어로 구현하는 경우에 속도를 빠르게 하기 위하여 T0-T7의 256 X 32 비트 테이블을 사용하는데 이는 AES와 동일한 형태의 테이블이다. 따라서 소프트웨어 구현시에 AES와 SSB의 동작 속도는 동일하다. 단, SSB는 암복호가 동일한 알고리즘을 사용하므로 AES에 비하여 코드 크기가 작은 장점을 가진다.

ARIA는 암호와 복호 알고리즘이 동일한 SPN 블록 암호이다. ARIA는 확산 계층을 16X16 바이트 대합 행렬로 구성했다. [7]에서 제안한 32 비트 프로세서에서 소프트웨어 구현 방식에 의한 ARIA와 AES 및 SSB의 소프트웨어 속도 비교를 표-1에 보인다.

표 1. SSB의 소프트웨어 속도 비교
Table. 1 SSB software speed comparison

	SSB	AES	ARIA
Normalized time	100	100	360

GNU-C 컴파일러를 사용하였으며 Windows XP, 셀러론 2.8GHz, 700M RAM 환경에서 1억 개 블록에 대하여 10 라운드 AES 및 SSB와 12 라운드 ARIA를 비교하였다. SSB와 AES는 소프트웨어 속도가 동일하며, ARIA에 비하여 3.5배 정도 빠르다.

SSB를 하드웨어로 구현하면 T0-T7의 256 X 32 비트 테이블을 사용하지 않으며 8 비트 S-박스를 사용하는데 이는 AES의 암호 부분과 동일한 구조가 된다. 암호와 복호를 동시에 수행하지 않는 응용 분야에서는 SSB는 암호와 복호가 동일한 하드웨어로 구성되므로 이들이 상이한 하드웨어로 구성되는 AES와 비교하여 하드웨어가 반으로 줄어든다.

SSB는 AES와 동일한 구조를 가지고 있으므로 안전성 또한 AES와 동일하다. AES에 대한 안전성 분석은 차분 공격[16], 선형 공격[17], Square 공격[13], 부매랑 공격[18], 불능 차분 공격(Impossible Differentials Cryptanalysis)[19], 부정차분공격(Truncated Differentials Cryptanalysis) [20] 등이 수행되었다. 이들 공격은 공격 패스를 설정하고, 설정한 패스에서 활동성을 가지는 S-박스의 차분/선형 확률에 의하여 공격 복잡도를 계산한다. 그러므로 SSB의 안전성을 검증하기 위해서는 최적의 패스에서 활동성을 가지는 S-박스의 수를 산출하고 이를 AES와 비교하여 상대적인 안전성을 분석할 수 있다. 그림-7에 4 라운드까지 SSB에서 최소로 활동성을 가지는 크리티칼 패스(critical path)를 보인다.

1 라운드 초기 상태에서 한 개의 S-박스만이 활동성을 가지도록 평문을 설정한다. 그림-3에서는 활동성을 가지는 S-박스의 위치를 'x'로 표시했다. 1 라운드에서는 ExgRow()를 수행해도 여전히 한 개의 S-박스만이 활성화된다. 그러므로 1 라운드에서는 한 개의 S-박스만이 활성화된다. MixColumn() 함수를 수행하면 하나의 열에 속한 모든 S-박스가 활성화된다. 이는 SSB의 MixColumn()이 MDS 행렬을 이루기 때문이다.

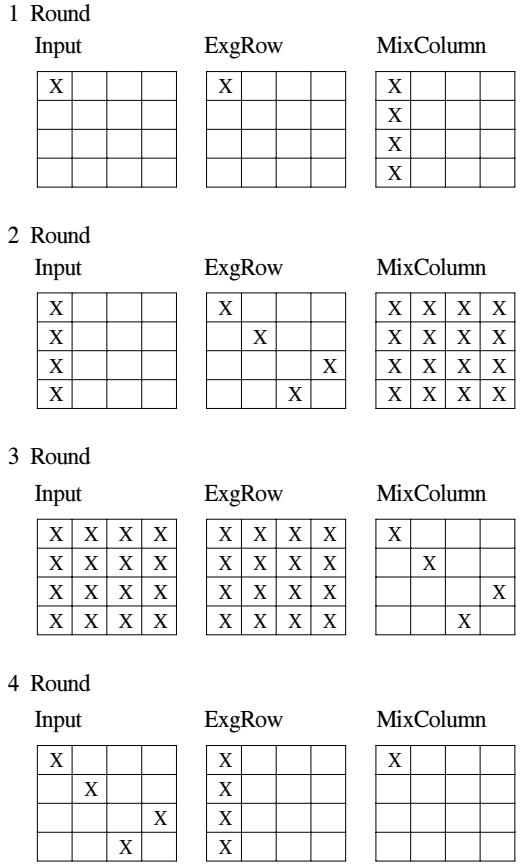

그림 7. SSB의 최소 활성 S-박스
Fig. 7 Minimum active S-box of SSB

2 라운드의 입력은 1 라운드의 MixColumn() 결과이다. 라운드 키 덧셈은 선형변환이므로 S-박스의 활성 상태에 영향을 주지 않는다. ExgRow()를 수행하면 활성 S-박스의 위치가 변화한다. 2 라운드에서는 네 개의 S-박스가 활성화된다. MixColumn() 함수를 수행하면 모든 S-박스가 활성화된다.

3 라운드에서는 16개의 S-박스가 활성화된다. MixColumn() 함수를 수행하면 각 열에서 오직 한 개의 S-박스만이 활성화되는 경우가 발생한다. 이는 MixColumn()가 MDS 행렬을 이루기 때문이다. 따라서 공격이 가장 용이한 경우로 각 열에서 오직 한 개의 S-박스만이 활성화되는 경우를 가정하였다.

4 라운드의 입력에는 4개의 S-박스가 활성화되는데, ExgRow()를 수행하면 이들이 하나의 열에 정렬된다.

MixColumn() 함수를 수행하면 한 개의 S-박스만이 활성화된다. 이는 공격이 가장 용이한 경우를 가정한 것이다. 3 라운드와 4 라운드와 같은 경우가 실제로 발생하는 입력을 찾지 못했으나 공격이 가장 용이한 패스로 설정하였다.

5 라운드는 1 라운드와 입력이 동일하므로 S-박스의 활성화 상태 또한 동일하다. 6-9 라운드는 2-5 라운드와 동일하다

표-2에 그림-3에서 분석한 SSB의 최소 패스에서의 활성화 S-박스 수와 동일한 방법으로 분석한 AES의 최소 패스에서의 활성화 S-박스 수를 비교한다.

표 2. 활성 S-박스의 비교
Table. 2 Number of active S-Box

	1	2	3	4	5	6	7	8	9	10	계
SSB	1	4	16	4	1	4	16	4	1	4	55
AES	1	4	16	4	1	4	16	4	1	4	55

SPN 구조 암호 알고리즘의 안전성 평가는 Hong[21] 등에 의해서 제안된 바 있다. 차분/선형 분석법은 차분/선형 활동성을 가지는 S-박스의 수를 구해서 안전성을 분석할 수 있다. AES S-박스의 최대 차분/선형 공격 확률은 각각 2^{-6}이다. S-박스와 S^{-1}-박스는 일대일 대응되는 전단사 함수로 같은 차분/선형 확률 값을 가진다.

표-2에서 10 라운드 SSB에서 활성화되는 S-박스의 수는 총 55개이다. 실제의 차분/선형 공격은 마지막 라운드의 라운드 키를 탐색하므로 마지막 라운드의 S-box는 안전도에 영향을 주지 않는다. 따라서 10 라운드 SSB의 공격 확률은 $(2^{-6})^{51} = 2^{-306}$으로 AES와 동일하다.

ARIA는 12 라운드이며, 확산 계층의 branch number가 8이다. 즉, 최악의 경우에 두 라운드에서 8개의 S-박스만이 활성화된다. 그러므로 ARIA에서 최소로 활성화되는 S-박스는 48개인데, 마지막 라운드는 차분/선형 공격에 영향을 주지 않으므로 ARIA의 차분/선형 공격 확률은 $(2^{-6})^{41} = 2^{-246}$으로 AES나 SSB에 미치지 못한다. 표-3에 SSB의 차분 및 선형 공격 확률을 보인다.

표 3. 차분 및 선형 공격 비교
Table. 3 Comparison of differential and linear attack

	Differential attack	Linear attack
SSB	2^{-306}	2^{-306}
AES	2^{-306}	2^{-306}
ARIA	2^{-246}	2^{-246}

부정차분공격은 바이트 단위로 변환되는 암호에서 바이트 패턴이 서로 다른 경우 '1', 같은 경우 '0'으로 정의된 차이 값을 가지고 분석하는 것으로 입력차분과 출력차분의 전부를 고려해야하는 차분분석보다 용이하게 공격할 수 있다. 불능차분공격은 차분확률이 '0'에 수렴하거나, 차분이 존재하지 않는 경우 평문쌍에 이와 같은 차분을 가지는 라운드 키는 제외한 후 남은 라운드 키를 가지고 틀린 키를 제외하는 분석방법으로 가능한 후보 서브키에 한정해서 적용하는 방법이다. Square 공격 및 부매랑 공격은 바이트 패턴이 각각의 라운드 사이에서 전파되는 특성을 이용한 공격이다.

이들 공격은 S-박스의 크기가 모두 동일한 경우에 유효한 공격 방법이다. 본 논문에서 제안하는 SSB는 AES와 동일한 바이트 패턴을 가진다. 따라서 부정차분공격, 불능차분공격, Square 공격 및 부매랑 공격 등은 AES와 동일한 안전성을 가지므로 이들 공격으로부터 또한 안전하다.

Ⅴ. 결 론

SPN 비밀키 블록 암호 및 복호 알고리즘은 여러 라운드의 암호 또는 복호 함수를 반복 수행한다. 암호 라운드 함수는 키합산층과 치환층 S 및 확산층 P의 세 단계로 구성된다. 복호 라운드 함수는 역확산층 P^{-1}과 역치환층 S^{-1} 및 키합산층의 세 단계로 구성된다. 이러한 SPN 구조는 암호와 복호 알고리즘이 다르므로 구현이 복잡해지는 단점을 가진다.

본 논문에서는 암호와 복호 과정이 동일한 SPN 구조 블록 암호 알고리즘인 가칭 SSB를 제안했다. SSB는 짝수 라운드로 구성하고, 각 라운드는 라운드 키 덧셈 계층, 8 비트 S-박스에 의한 치환 계층, 바이트를 교환하는

교환 계층 그리고 MDS 대합 행렬로 구성한 확산 계층으로 구성했다. 치환 계층의 S-박스는 안전성이 검증된 AES의 S-박스를 사용했다. 홀수 라운드의 치환 계층은 $\{S, S^{-1}, \ldots S, S^{-1}\}$으로 구성하고, 짝수 라운드의 치환 계층은 이와 반대 순서로 구성했다. SSB는 암호와 복호 알고리즘이 동일하며, 적용하는 라운드 키는 상호 역순으로 되어있다.

제안한 SSB는 암호와 복호 알고리즘이 동일하므로 하드웨어 구현이 단순한 장점을 가진다. 소프트웨어 수행 속도는 AES와 동일하며, 암호와 복호가 동일하므로 코드 크기가 AES 보다 줄어드는 장점을 가진다. ARIA는 SSB와 유사하게 암복호가 동일한 SPN 블록 암호이다. SSB는 ARIA과 비교하여 소프트웨어 수행 속도가 약 3.5배 빠르다,

SSB는 AES와 동일한 구조를 가지고 있으므로 안전성은 AES와 동일한 방식으로 검증할 수 있다. 블록 암호의 안전성 분석은 차분 공격, 선형 공격, Square 공격, 부매랑 공격, 불능 차분 공격, 부정차분공격 등이 있다. 본 논문에서는 공격 패스를 설정하고, 설정한 패스에서 활동성을 가지는 S-박스의 차분/선형 확률에 의하여 공격 복잡도를 계산하였다. SSB는 AES와 동일한 전파 경로를 가지며 동일한 S-박스를 사용하므로 안전성 또한 동일하여 선형 및 차분 공격확률은 2^{-306}이다. ARIA는 MDS를 구성하지 않으므로 활성화되는 S-박스의 수가 적으므로 선형 및 차분 공격확률은 2^{-246}로 계산되었다.

제안한 SSB는 AES와 동일한 안전성을 가지며, 암호와 복호가 동일하기 때문에 하드웨어 구성이 간단해서 스마트카드 및 RFID 태그와 같은 제한된 하드웨어 및 소프트웨어 환경에서도 쉽게 구현 가능하다.

참고문헌

[1] National Bureau of Standards, Data Encryption Standard, FIPS-Pub. 46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977

[2] “Report on the Development of the Advanced Encryption Standard(AES)”, http://www.csrc.nist.gov/encryption/aes/.

[3] J. Daemen, and V. Rijmen, “AES Proposal: Rijndael,” http://www.csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf, 1999.

[4] “New European Schemes for Signatures. Integrity. and Encryption(NESSIE).” http:// cryptonessie.org/.

[5] “Cryptography Research and Evaluation Committees (CRYPTREC).” http: //www.cryptrec.go.jp/

[6] SEED, http://www.kisa.or.kr/seed/.

[7] Daesung Kwon, et. al., ″New block cipher : ARIA,″ Information security and cryptology - ICISC 2003, 6th international, pp. 432-445, 2003

[8] H. Feistel, “Cryptography and Computer Privacy.” Scientific American, Vol.228, No.5, pp. 15-23, 1973.

[9] C.E. Shannon, “Communication Theory of Secrecy System″ Bell System Technical Journal, Vol. 28, No. 4, page 656-715, 1949.

[10] P. Barreto, V. Rijmen, J. Nakahara Jr., B. Preneel, J. Vandewalle and H.Y. Kim, ″Improved SQUARE attacks against reduced-round HIEROCRYPT,″ 8th International Workshop on Fast Software Encryption, Springer-Verlag. pp. pp. 165 – 173, Apr. 2001.

[11] A. M. Youssef, S. Mister, and S. E. Tavares, “On the Design of linear Transformation for Substitution and Permutation Encryption Networks,” in the Workshop Record of the Workshop on Selected Areas in Cryptography (SAC ‘97), pp. 40-48, Aug. 1997.

[12] S. Vaudenay, ″On the need for multipermutations: Cryptanalysis of MD4 and SAFER,″ Proc. of Fast Software Encryption(2), LNCS 1008, Springer-Verlag, pp. 286-297, 1995

[13] V. Rijmen, J. Daemen, B. Preneel, A. Bosselares, and E. De Win, ″The cipher SHARK,″ Fast Software Encryption, LNCS 1-39, D. Gollmann Ed., Springer-Verlag, pp. 99-112, 1996

[14] J. Daemen, L. Knudsan, and V. Rijmen, “The Block Cipher Square,″ Proceeding of FSE‘97, LNCS Vol.1267, pp. 149-165, 1997.

[15] Federal Information Processing Standards Publication 197, ″Announcing the ADVANCED ENCRYPTION STANDARD(AES),″ Nov. 2001, csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[16] E. Biham and A. Shamir, "Differential Cryptanalysis of the Full 16-Round DES", LNCS 537, page 2-21, 1990.
[17] M. Matsui, "Linear Cryptanalysis Method for DES", LNCS 765, page 386-397, 1994.
[18] A. Birynkov, "The Boomerang attack on 5 and 6-round reduced AES", LNCS 3373, page 42-57, 2005.
[19] J. Cheon, M. Kim, K. Kim, J. Lee and S. Kang, "Improved impossible differential cryptanalysis of Rijndael and Crypton", LNCS 2288, page 39-49, 2001.
[20] L. R. Knudsen, "Truncated and higher order differential," Fast Software Encryption-Second International Workshop, LNCS Vol.1008, pp. 196-211, 1995.
[21] S. Hong, S. Lee, J. Lim, J. Sung, and D. Cheon, "Provable security against differential and linear cryptanalysis for the SPN structure," In Fast Software Encryption 2000, LNCS Vol.1978, pp. 273-283, 2001.

저자소개

조경연(Gyeong-Yeon Cho)

1990년 2월 인하대학교 전자공학과 박사
1983년～1991년 삼보컴퓨터 기술연구소 책임연구원
1991년～2003년 삼보컴퓨터 기술연구소 기술고문
1998년～2003년 에이디칩스 기술고문
1991년～현재 부경대학교 공과대학 IT융합응용공학과 교수
※관심분야 : 전산기구조, 반도체회로 설계, 암호 알고리즘