論文
4-4-11

확장된 Feistel 구조를 이용한 Block Cipher의 설계와 분석 (Design and Analysis of the Block Cipher Using Extended Feistel Structure)

임 웅 택(Ung-Taeg Lim)[1] 전 문 석(Moon-Seog Jun)[2]

요 약

본 논문에서는 확장된 Feistel 구조를 갖는 128-비트 블록 암호알고리즘 Lambda를 설계하고, 차분공격(differential cryptanalysis)과 선형공격(linear cryptanalysis)을 통해 안전성을 분석하였다. Lambda는 2-라운드 만에 완전 확산(diffusion)이 일어나도록 설계되었다. 이러한 우수한 확산효과로 인해 8-라운드 차분 특성이 구성될 확률은 2^{-192}로, 선형특성이 구성될 확률은 2^{-128}로 분석되었다. 결과적으로 Lambda는 128-비트 비밀키를 적용하였을 경우 8-라운드 이상이면 차분공격이나 선형공격이 전사(exhaustive)공격 방법보다 효율성이 떨어지는 것으로 분석되었다.

ABSTRACT

In this paper, we designed a 128-bit block cipher, Lambda, which has 16-round extended Feistel structure and analyzed its secureness by the differential cryptanalysis and linear cryptanalysis. We could have full diffusion effect from the two rounds of the Lambda. Because of the strong diffusion effect of the algorithm, we could get a 8-round differential characteristic with probability 2^{-192} and a linear characteristic with probability 2^{-128}. For the Lambda with 128-bit key, there is no shortcut attack, which is more efficient than the exhaustive key search, for more than 8 rounds of the algorithm.

1. 서론

NIST(National Institute of Standards & Technology)는 DES(data encryption standard) [3]를 대체할 차세대 암호 알고리즘 표준안인 AES(Advanced Encryption Standard)를 공모하여, 2000년 10월 2일 Rijndael[7]을 최종 선정하였다[10].

NIST는 AES 공모 시 차세대 암호 알고리즘은 다음과 같은 요구사항[6]을 만족하도록 요구하였다.

- symmetric block cipher
- Triple DES보다 안전하고, 효율적일 것
- 키 길이는 128, 192, 256 비트까지 가능할 것
- block size는 128 비트 (64, 256 등 선택적으로 가능)

1) 정회원 : 부천대학 전산정보처리과 조교수
2) 정회원 : 숭실대학교 컴퓨터학과 교수

논문심사 : 2003. 6. 20.
심사완료 : 2003. 7. 18.

• 하드웨어와 소프트웨어로 구현 가능할 것
• 무료로 공개할 것

본 논문에서는 AES 요구사항을 만족하는 독자적인 암호 알고리즘을 설계하고, 차분공격(differential cryptanalysis)[2]과 선형공격(linear cryptanalysis) [5] 분석을 통해 안전성을 입증하였다.

이 논문에서 설계하여 제시한 암호 알고리즘을 Lambda라 명명하였다.

Lambda의 전체적인 구조는 확장된 Feistel 구조로 설계하였다. 여기에서 확장된 Feistel 구조란 DES와 같은 전통적인 Feistel 구조에 확산효과를 증대시켜주는 확산연산을 SPN(substitution-permutation network) 구조로 부가한 설계방법을 말한다.

DES와 같이 전통적인 Feistel 구조로 설계된 암호 알고리즘이 보통 3-라운드를 거쳐야만 완전 확산(diffusion)이 이루어지는 것과 달리 Lambda는 단지 2-라운드 만에 완전 확산이 일어난다.

설계된 Lambda는 128-비트 비밀키를 적용하였을 경우 8-라운드 이상이면 차분이나 선형 공격 방법이 전사공격 방법보다 효율성이 떨어지는 것으로 분석되었다.

본 논문에서는 256-비트 비밀키를 적용하였을 경우를 고려하여 총 16-라운드 구조를 갖는 Lambda를 제안하였다.

2. Lambda의 설계

Lambda 설계 시 AES로 요구되는 기준을 따랐으며, 구체적으로 다음 두 가지에 중점을 두고 설계하였다.

- 차분공격과 선형공격에 강할 것
- 확산 특성이 좋을 것

2.1 Lambda 구조

설계된 Lambda의 전체적인 구조는 [그림 1]과 같다.

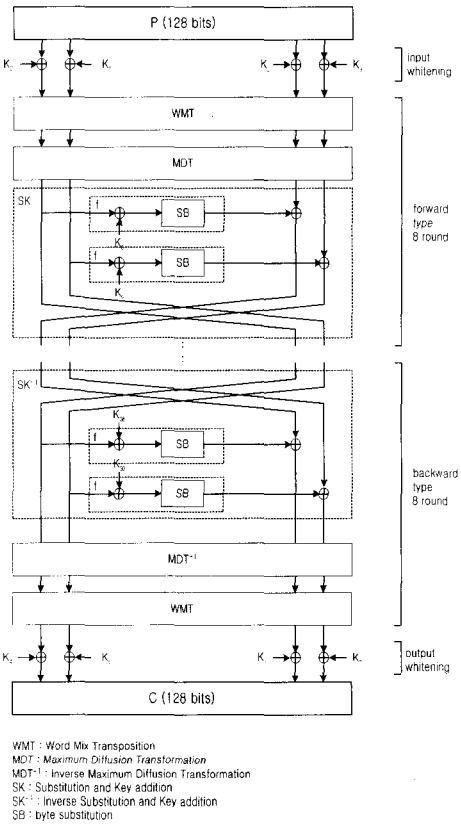
Lambda는 총 16-라운드 반복 구조로 설계되었으며 첫 라운드 시작 전과 마지막 라운드 후에 이미 효과가 입증된 XOR whitening 기능을 적용하였다. whitening은 공격자에게 첫 라운드와 마지막 라운드의 f-함수 입력 값을 예측하지 못하게 하는데 효과가 있는 것으로 [4]에서 이미 입증된 방식이다.

Lambda의 총 16-라운드 구조는 서로 다른 전위 8-라운드 구조와 후위 8-라운드 구조가 결합된 형태로 운영된다. 서로 다른 전위와 후위 구조를 결합시킨 이유는 암호화나 복호화 시 동일한 알고리즘을 사용하기 위해서이다. SPN 구조는 복호화 시의 암호 알고리즘은 암호화 시 구조의 역순으로 적용되어야 하고 적용되는 각 함수는 암호화 때의 역함수를 적용해야 한다. 이러한 SPN 구조 특성을 해결하기 위해서 Lambda에서는 전위 구조는 암호화 구조로 설계하고 후위 구조는 복호화 구조로 설계하였다. 이렇게 암호화 구조와 복호화 구조를 결합시킴으로서 SPN 구조를 갖는 암호 알고리즘을 암호화 구조와 복호화 구조를 동일하게 사용할 수 있게 된다.

Lambda의 전위와 후위의 라운드 구조는 후위 라운드 구조를 전위 라운드에서의 WMT와 MDT, SK 함수 적용과정을 역순으로 적용하고 각 함수의 역함수를 적용한다는 점이 틀리다.

전위의 각 라운드는 다음과 같은 함수를 차례로 적용한다.

- WMT : 워드 단위로 행과 열을 바꾼다.
- MDT : 확산 효과를 극대화 시키기 위해 4x4 MDT 행렬과 입력 워드를 vector 곱 연산한다.
- SK : 워드 단위로 라운드 키를 적용하고, S-box 치환 한 후에 워드 단위로 자리바꿈(swapping)을 한다.



[그림 1] Lambda의 구조
[Fig. 1] The structure of Lambda

후위의 각 라운드는 전위의 역순으로 전위 함수의 역함수를 차례로 적용한다.

- SK^{-1} : SK에서 워드 단위의 자리바꿈을 먼저 적용한다.
- MDT^{-1}: MDT 행렬의 역 행렬을 적용한다.
- WMT^{-1}: 전위 함수의 WMT와 동일하다.

2.2 WMT

WMT(word mix transposition)는 각 라운드를 거치면서 바이트 간 확산 효과를 극대화시키기 위해 워드 단위로 행과 열을 바꾸어주는 과정이다.

WMT를 두 번 적용하면 원래 위치로 돌아오므로 $WMT=WMT^{-1}$이 된다.

A[0]	a_{00}	a_{01}	a_{02}	a_{03}	WMT	B[0]	a_{00}	a_{10}	a_{20}	a_{30}
A[1]	a_{10}	a_{11}	a_{12}	a_{13}		B[1]	a_{01}	a_{11}	a_{21}	a_{31}
A[2]	a_{20}	a_{21}	a_{22}	a_{23}	⇒	B[2]	a_{02}	a_{12}	a_{22}	a_{32}
A[3]	a_{30}	a_{31}	a_{32}	a_{33}		B[3]	a_{03}	a_{13}	a_{23}	a_{33}

2.3 MDT

MDT(maximum diffusion transformation)는 입력 워드의 각 바이트 간에 확산 효과를 극대화하는 연산을 한다.

MDT 함수는 4개 워드를 입력으로 하여 각 워드 단위로 아래와 같이 고정된 MDT 행렬과 다항식 곱 연산을 수행하여 출력 워드를 결정한다.

$$\begin{Bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{Bmatrix} = \begin{Bmatrix} 01 & 01 & 46 & C8 \\ 46 & 01 & C8 & 01 \\ C8 & 46 & 01 & 01 \\ 01 & C8 & 01 & 46 \end{Bmatrix} \cdot \begin{Bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{Bmatrix}$$

MDT 연산을 식으로 표현하면 다음과 같다.

$$y_i = \oplus_{j=0}^{3}(m_{ij} \cdot x_j)$$

여기에서, m_{ij}는 MDT 행렬의 i행 j열의 값이고, $m_{ij} \cdot x_j$ 연산은 $GF(2^8)$ 상의 원시다항식 $p(x)=x^8+x^4+x^3+x^2+1$을 이용한 다항식 곱 연산이다.

MDT 테이블의 요소 중에 46과 C8은 다항식 곱 연산을 하드웨어로 구성할 때에 테이블 참조 방식이 아닌 쉬프트만으로 연산이 가능하도록 아래와 같이 원시다항식 p(x)로부터 계산된 값이다.

$46 = p(x)/4 + 1$
$C8 = p(x)/4 + p(x)/2 + 1$

만약, MDT 연산의 입력 워드 중에 특정 바이트 a(x)를 다항식으로 다음과 같이 표현하였을 때

$$a(x)=a_7x^7+a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x^1+a_0$$

46·a(x)와 C8·a(x) 연산은 다음과 같이 쉬프트 연산 식으로 표현할 수 있다.

$46 \cdot a(x) = a(x) \gg 2 \oplus a(x) \oplus (a_0 == 1 ?\ p(x) \gg 2 : 0) \oplus (a_1 == 1 ?\ p(x) \gg 1 : 0)$

$C8 \cdot a(x) = a(x) \gg 2 \oplus a(x) \gg 1 \oplus a(x) \oplus (a_0 == 1 ?\ p(x) \gg 2 : 0) \oplus (a_0 == 1 ?\ p(x) \gg 1 : 0) \oplus (a_1 == 1 ?\ p(x) \gg 1 : 0)$

여기에서, $(a_0 == 1 ?\ p(x) \gg 2 : 0)$ 식은 $a_0 == 1$이면 $p(x)/4$를 $a_0 == 0$이면 0을 반환하는 연산이다.

따라서 MDT 연산은 입력 값의 비트 값 검사에 의한 쉬프트 연산과 XOR 연산만으로 운영이 가능하다.

MDT 테이블 구성요소 중에 01은 MDT 연산을 반으로 줄이기 위해 선택한 값이다. 즉, 어떤 입력 요소에 01을 곱하면 자신의 값이 되므로 연산이 일어나지 않는다.

MDT^{-1}은 MDT 행렬의 역 행렬을 적용하는 함수로서 MDT 역 행렬은 다음과 같다.

$$MDT^{-1} = \begin{Bmatrix} cd & 43 & ce & 42 \\ 42 & cd & 43 & ce \\ 43 & ce & 42 & cd \\ ce & 42 & cd & 43 \end{Bmatrix}$$

2.4 SK

SK(substitution and key-addition)는 라운드 키 적용과 S-box 치환을 하는 f-함수와 4개의 워드 간에 자리바꿈을 하는 부분으로 구성된다.

여기에서 f-함수는 SK 상위 두 워드를 입력으로 하여 라운드 키를 XOR 적용하고, 그 결과를 다시 S-box 치환하는 역할을 한다.

f-함수의 출력은 SK의 하위 두 입력 워드와 각각 XOR 시킨 후에 각 워드 간에 자리바꿈을 시킨다.

SK에 입력과 출력되는 4개 워드를 각각 ($L_{i,0}$, $L_{i,1}$, $R_{i,0}$, $R_{i,1}$)와 ($L_{o,0}$, $L_{o,1}$, $R_{o,0}$, $R_{o,1}$)라고 하고, f-함수에서 적용되는 두 라운드 키를 K_{2r+8}과 K_{2r+9}라고 했을 때 SK의 입력 워드와 출력 워드와의 관계식은 다음과 같다.

$$L_{o,0} = f(L_{i,0}, K_{2r+8}) \oplus R_{i,0} \quad r=0, 1, \ldots, 7$$
$$L_{o,1} = f(L_{i,1}, K_{2r+9}) \oplus R_{i,1} \quad r=0, 1, \ldots, 7$$
$$R_{o,0} = L_{i,0}$$
$$R_{o,1} = L_{i,1}$$

SK^{-1}은 SK의 역함수로서 SK의 입력 워드 간에 자리바꿈을 먼저 수행한 후에 f-함수를 적용시킨다. SK^{-1}은 다음과 같이 표현된다.

$$L_{o,0} = R_{i,0}$$
$$L_{o,1} = R_{i,1}$$
$$R_{o,0} = f(R_{i,0}, K_{2r+8}) \oplus L_{i,0} \quad r=8, 9, \ldots, 15$$
$$R_{o,1} = f(R_{i,1}, K_{2r+9}) \oplus L_{i,1} \quad r=8, 9, \ldots, 15$$

여기에서, f-함수는 다음과 같은 연산을 수행을 한다.

$$f(L_{i,0}, K_{2r+8}) = SB(L_{i,0} \oplus K_{2r+8}) \quad r=0, 1, \ldots, 7$$
$$f(L_{i,1}, K_{2r+9}) = SB(L_{i,1} \oplus K_{2r+9}) \quad r=0, 1, \ldots, 7$$
$$f(R_{i,0}, K_{2r+8}) = SB(R_{i,0} \oplus K_{2r+8}) \quad r=8, 9, \ldots, 15$$
$$f(R_{i,1}, K_{2r+9}) = SB(R_{i,1} \oplus K_{2r+9}) \quad r=8, 9, \ldots, 15$$

여기에서, SB는 바이트 단위로 S-box 치환을 하는 함수로서 S-box는 256개의 8-비트 요소들로 구성되며, 8-비트 입력을 index 값으로 하여 출력 8-비트 값을 결정한다.

Lambda에서는 한 개의 S-box로 각 워드의 입력 바이트를 치환한다.

설계된 S-box는 부록에 명시하였다.

2.5 키 스케줄

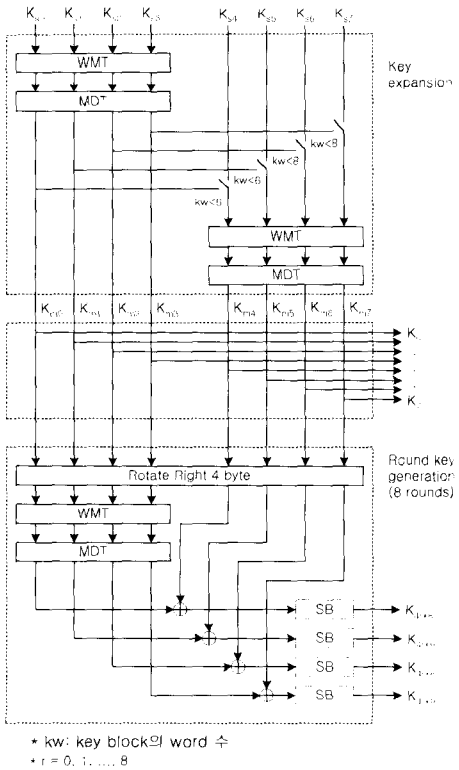
키 스케줄은 16-라운드에서 사용할 라운드 키 32개 워드와 최초 입력과 최종 출력에 대한 whitening용 8개를 포함하여 총 40개 워드를 생성한다. 키 스케줄 구조는 초기 키 확장 부분과 whitening 키 생성 부문, 라운드 키 생성 부분으로 구성된다.

키 확장 부분은 입력되는 가변길이 128, 192, 256 비트 비밀키를 입력으로 하여 라운드 키 생성에 사용할 256-비트로 확장한다.

키 확장으로 생성된 8개 워드는 whitening 키 생성에 사용되고, whitening key는 다시 라운드 키 생성 부분으로 입력되어 라운드 키 생성 부분을 총 8-라운드를 운영하여 각 라운드 당 4개의 라운드 키를 생성하여 총 32개의 라운드 키를 생성한다.

2.5.1 키 확장

키 확장은 128-비트 또는 192-비트, 256-비트로 입력되는 비밀키를 라운드 키 생성에서 요구되는 256-비트 길이로 키 길이를 확장하는 부분이다. 만약, 입력되는 비밀키의 길이가 요구되는 키 길이(128-비트 또는 192-비트, 256-비트)보다 짧을 경우 부족한 길이만큼 '0' 으로 채워서 키 확장의 입력으로 사용한다.



[그림 2] 키 스케줄 구조
[Fig. 2] The structure of key schedule

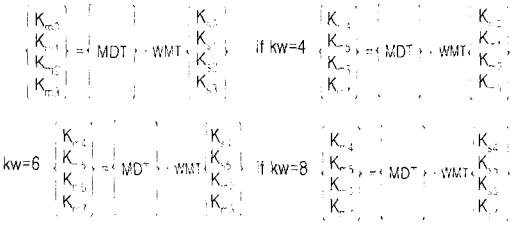
비밀키는 키 길이에 따라 다음과 같이 워드 길이로 표현된다.

$K_{s128}=(K_{s0}, K_{s1}, K_{s2}, K_{s3})$
$K_{s192}=(K_{s0}, K_{s1}, K_{s2}, K_{s3}, K_{s4}, K_{s5})$
$K_{s256}=(K_{s0}, K_{s1}, K_{s2}, K_{s3}, K_{s4}, K_{s5}, K_{s6}, K_{s7})$

키 확장은 입력 비트 간에 확산 효과를 최대화 될 수 있도록 MDT를 적용하였다. MDT는 입력되는 비밀키의 잉여부분이 '0' 으로 채워짐으로서 라운드 키 생성에 부정적인 영향을 미칠 수 있는 요소를 제거해 주는 역할을 하게 된다.

키 확장에 적용되는 WMT와 MDT 연산은 암호 알고리즘에서의 WMT와 MDT 연산과 동일하다.

확장된 키 K_m이 (K_{m0}, K_{m1}, K_{m2}, K_{m3}, K_{m4}, K_{m5}, K_{m6}, K_{m7})로 구성된다고 하였을 때, 적용되는 키의 워드 수 kw 값에 따라 K_m은 다음과 같은 WMT와 MDT 연산을 통해 구해진다.



여기에서, kw는 입력되는 비밀키의 워드 수이다.

2.5.2 whitening키 생성

whitening 키는 8개 워드가 필요하며 확장된 키 $K_m=(K_{m0}, K_{m1}, K_{m2}, K_{m3}, K_{m4}, K_{m5}, K_{m6}, K_{m7})$을 그대로 whitening 키로 사용한다.

$K_i=K_{mi}$ $\quad i=0,1, \ldots, 7$

whitening 키는 다시 라운드 키 생성의 입력 워드로 사용된다.

2.5.3 라운드키 생성

Lambda에서 필요로 하는 총 40개의 라운드 키는 한 라운드에 4개의 라운드 키를 생성하는

라운드 키 생성 라운드 구조를 8회 반복 수행시킴으로서 얻어진다.

라운드 키 생성에 필요한 초기 입력 값은 8 워드 길이의 whitening 키를 사용하고, 매 라운드마다 rotate right 4 바이트씩 하여 매 라운드의 입력으로 계속 사용한다.

4 바이트씩 rotate right된 입력 워드 중에 최하위 4개 워드를 선택하여 암호화 알고리즘에서와 같이 WMT와 MDT 과정을 거쳐 4개의 각 f-함수의 입력으로 하여 각 라운드 당 4개씩의 라운드 키를 생성한다. 이 때 각 f-함수의 입력 워드와 XOR되는 값은 rotate right된 워드 중에 최상위 4개 워드를 각각 사용한다.

키 스케줄의 f-함수에서 적용하는 S-box는 암호화 알고리즘에서와 같은 S-box를 사용한다.

3. 설계 특성

Lambda의 각 구성 부분에 대한 설계 특성은 다음과 같다.

3.1 S-box 특성

Lambda의 S-box는 Rijndael의 S-box 설계사상[7]을 활용하여 다음과 같은 절차에 의해 설계하였다.

① g: a → b= a^{-1}

② c = f(b)

여기에서 g는 S-box 입력 a에 대하여 GF(2^8)에서 원시근 $p(x)=x^8+x^4+x^3+x^2+1$를 이용하여 multiplication inverse를 구하는 연산이고, f는 다음과 같은 연산을 한다.

$$\begin{Bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{Bmatrix} = \begin{Bmatrix} 1\,1\,1\,1\,0\,1\,0\,0 \\ 1\,1\,1\,0\,1\,0\,0\,1 \\ 1\,1\,0\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,1\,1 \\ 1\,0\,0\,1\,1\,1\,1\,0 \\ 0\,0\,1\,1\,1\,1\,0\,1 \\ 0\,1\,1\,1\,1\,0\,1\,0 \end{Bmatrix} \cdot \begin{Bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{Bmatrix} \oplus \begin{Bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{Bmatrix}$$

이와 같은 방식으로 설계된 S-box는 부록에 명시하였으며, 설계된 S-box의 특성은 다음과 같다.

- 최대 차분특성 구성 확률: 2^{-6}
- 최대 선형특성 구성 확률: 2^{-4}
- 평균 SAC 특성 확률: 0.563

여기에서 SAC(strict avalanche criterion)[9]은 S-box의 입력 각 비트가 출력의 각 비트에 변화에 영향을 미치는 확률 값으로 0.5가 가장 이상적인 확률 값이 된다.

3.2 WMT 특성

WMT는 각 라운드를 거치면서 특정 위치의 바이트 간 다른 위치에 있는 바이트 값에 골고루 영향을 미치도록 MDT 확산 수행 이전에 이루어지는 바이트 간의 위치 변환 과정이다.

WMT는 4x4 바이트 입력에 대해 바이트 간에 행과 열의 위치를 바꾸는 과정으로서 WMT를 두 번 적용하게 되면 원래 위치로 돌아오게 된다. 즉, WMT의 역함수 WMT^{-1}=WMT 관계가 성립된다.

3.3 MDT 특성

고정된 4x4 바이트 테이블 값과 입력 4x1 바이트간의 다항식 곱 연산 결과는 대단히 높은 확산효과를 갖는 점이 [7]과 [8]에서 입증되었다.

또한, 다항식 연산은 table look-up 방식으로 빠르게 연산이 가능한 특성을 갖는다[1].

Lambda에서 사용하는 MDT 테이블의 확산 효과에 대한 두 특성은 다음과 같다.

- ***특성-1:*** *MDT 입력 워드의 4-바이트 중에 3-바이트를 모두 '0' 으로 하고, 나머지 1-바이트를 '0' 이 아닌 값으로 했을 때 입력 값과 출력 값에 대한 최소 hamming distance 값은 6이다.*

  특성-1은 MDT 연산을 하게 되면 입력 32-비트 중에 최소한 6개 비트는 바뀌어서 출력된다는 의미이다. 본 논문에서 사용된 MDT 연산은 최대 29개 비트까지 바뀌어서 출력되는 것으로 측정되었다.

- ***특성-2:*** *branch number는 5이다.*

  branch number는 선형 변환에 대하여 확산효과를 측정하는 한 방법으로서 선형변환의 입력과 출력에 나타나는 non-zero 바이트의 최소수로 측정된다. W(a)를 MDT에 입력되는 워드 a의 non-zero 바이트의 수라고 정의했을 때 branch number는 다음과 같은 식으로 표현할 수 있다.

  $Min_{a\neq 0}(W(a) + W(MDT(a)))$

  MDT 변환의 branch number가 5라고 하는 의미는 입력 워드 중에 1-바이트만을 non-zero 바이트로 입력한다면 반드시 출력 4-바이트가 모두 non-zero 바이트가 된다는 의미가 된다. 또한, 입력 2-바이트만을 non-zero로 한다면 최소한 출력의 3-바이트는 non-zero가 된다.

  위의 MDT 연산에 대한 특성-1과 특성-2는 입력 값 간에 확산을 극대화 시켜주는 효과를 주는 요소로서 입력 쌍에 대한 차분 값에 대해서도 동일하게 적용된다.

  이러한 MDT 연산의 확산 효과는 Lambda에 대한 차분이나 선형 특성이 구성될 확률을 낮추는 효과를 주게 된다.

- ***특성-3:*** *branch number가 5가 되도록 조건하에 MDT 출력의 특정 바이트 위치에 원하는 값을 의도적으로 만드는 것이 항상 가능하지 않다.*

branch number=5일 때, input non-zero 바이트 위치를 고정 시켜놓고 특정 위치 output 바이트만 non-zero 바이트가 되는 경우는 항상 255개이다. MDT 연산은 선형 변환이므로 이 때 출력되는 255개는 중복된 값으로 나타나지 않는다.

예를 들어, 입력 4개 바이트가 모두 non-zero이고 출력 첫 번째 바이트만 non-zero 바이트가 되는 경우는 1에서 255까지 모두 255개이다. 이것은 입력 non-zero 바이트가 4이면 출력의 특정 바이트 위치에 원하는 값을 항상 얻을 수 있다는 말과 같다.

그러나 입력 3개가 non-zero 바이트일 때 특정 위치의 출력 2개 바이트가 non-zero가 되는 경우는 모두 255개이므로 출력 2개 바이트 값을 의도하는 값으로 만들 수 있는 확률은 $255/(2^8-1)^2 \approx 2^{-8}$이 된다.

마찬가지로 특정 위치의 출력 3개 바이트를 원하는 값으로 만들 수 있는 확률은 $255/(2^8-1)^3 \approx 2^{-16}$이 되고, 특정 위치의 출력 4개 바이트를 원하는 값으로 만들 수 있는 확률은 $255/(2^8-1)^4 \approx 2^{-24}$가 된다.

이와 같은 MDT 연산의 제한된 출력 구성 특성은 SK 연산 입력의 우측 두 워드를 의도하는 XOR 값으로 입력하여 비교적 좋은 차분 특성 구성하여 공격하는 전통적인 Feistel 구조에 대한 차분공격을 어렵게 만든다.

3.4 2-라운드 완전 확산효과

MDT 연산은 입력 한 워드에 대해서 완전 확산 효과를 갖는다. 이러한 MDT가 갖는 워드에 대한 완전 확산 효과는 WMT를 함께 적용함으로서 각 워드 간 완전 확산 효과로 나타난다.

실제, WMT 입력 중에 non-zero 한 바이트는 한 라운드 만에 한 워드 내의 다른 나머지 바이트들에 영향을 미치고, 2-라운드 만에 다른 워드의 모든 바이트들에게 완전하게 영향을 미친다.

DES와 같은 전통적인 Feistel 구조를 갖는 암호 알고리즘들이 3-라운드 만에 완전 확산 효과를 갖는 반면 Lambda는 2-라운드 만에 완전 확산 효과를 갖는다.

3.5 키 스케줄

라운드 키 생성 시 f-함수를 적용함으로서 라운드 키의 일부를 알아낸 결과로 다른 라운드에 적용된 라운드 키를 유추하지 못하도록 하였다.

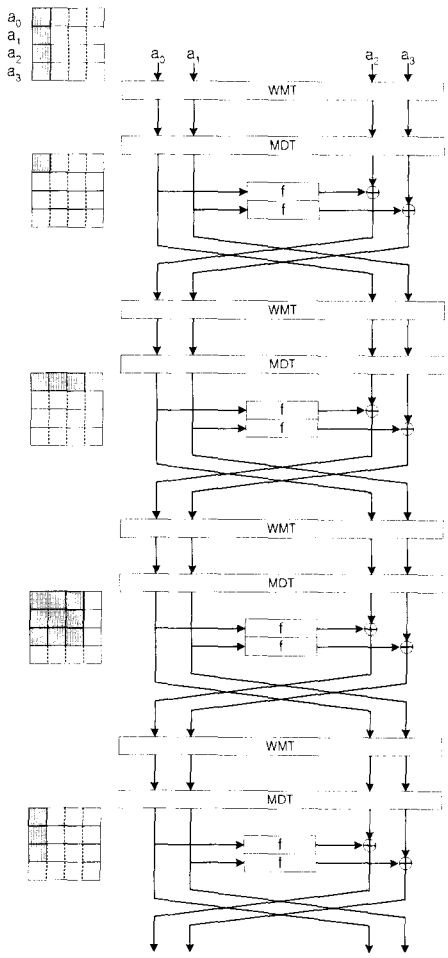
3.6 라운드 수

일반적으로 암호 알고리즘의 총 라운드 수는 전사공격보다 효율적인 공격에 안전한 라운드 수에 보다 나은 안전성을 위해 잉여 라운드를 추가하여 결정한다.

Lambda의 축소된 4-라운드 차분특성이 구성될 확률은 2^{-96}이다. 따라서 Lambda를 8-라운드로 운영할 경우 차분공격을 위해 약 2^{192}개의 암호문 쌍이 필요하게 되므로, 만약 128-비트 키를 적용하였을 경우 8-라운드 구조면 차분공격이 전사공격 방법보다 효과가 없게 된다.

16-라운드 선형공격을 하기 위해서는 적어도 2^{258}개 평문에 대한 암호화 과정이 필요하다.

따라서 256-비트까지의 가변 키를 수용해야 하는 암호 알고리즘의 조건을 충족시키기 위해서는 16-라운드면 충분하다.



[그림 3] 4-라운드 차분 특성
[Fig. 3] DC Characteristic of reduced 4-round

4. 공격 분석

Lambda에 대한 차분공격과 선형공격 분석은 비교적 구성될 확률이 높은 4-라운드 특성을 반복 적용하여 분석하였다.
Lambda의 축소된 4-라운드 특성이 구성될 확률은 다음과 같다.

- 4-라운드 차분특성이 구성될 확률은 2^{-96}
- 4-라운드 선형특성이 구성될 확률은 2^{-64}

4.1 차분공격 분석

본 논문에서는 16-라운드 Lambda에 대한 차분공격 분석을 위해 비교적 구성 확률이 높은 축소된 4-라운드 차분 특성을 구성하여 이 특성을 반복 적용하는 방법으로 분석하였다.

비교적 구성 확률이 높은 축소된 4-라운드 차분 특성 구조는 [그림 3]과 같다.

[그림 3]에서 ▨는 차분 값이 0이 아닌 바이트를 의미하는 것으로서 4-라운드 차분 특성이 구성될 확률은 각 라운드의 f-함수에 특정 차분 바이트 값이 입력되었을 때 특정 차분 바이트가 출

력될 확률의 곱으로 계산된다[2].

결국, n-라운드 차분 특성이 구성될 최대 확률은 f-함수의 non-zero 입력 차분 바이트 수 즉, active s-box의 수에 의해 결정된다.

따라서 [그림 3]에서 f-함수의 non-zero 입력 차분 바이트는 모두 16개이고, S-box의 최대 차분특성 확률은 2^{-6}이므로 4-라운드 차분 특성이 구성될 최대 확률은 $(2^{-6})^{16}=2^{-96}$이 된다.

이러한 4-라운드 특성을 2번 반복 적용한 축소된 8-라운드 특성이 구성될 확률은 2^{-192}가 되고, 4-라운드 특성을 4번 반복 적용한 16-라운드 차분 특성이 구성될 확률은 2^{-384}가 된다.

결국, 128-비트 키를 적용하였을 경우 차분공격 방법이 전사공격 방법보다 효율성이 떨어짐을 알 수 있다.

16-라운드 Lambda의 마지막 라운드에 적용된 라운드 키 중에 32-비트를 예측하기 위해서는 적어도 2^{386}개의 선택된 평문이 필요하며, 각 라운드에 적용된 라운드 키를 모두 찾기 위해서는 2^{388}번의 암호화 과정이 필요하다.

따라서 256-비트의 비밀키를 적용한 16-라운드 Lambda에 대한 차분공격은 전사공격 방법보다 효과가 없음을 알 수 있다.

4.2 선형공격 분석

선형특성이 구성될 확률은 n-라운드 차분특성에서 non-zero 차분 바이트에 대하여 최대 선형특성 확률을 적용시킴으로서 간단하게 구할 수 있다[5].

S-box의 최대 선형특성 확률은 2^{-4}이므로 4-라운드 선형특성이 구성될 확률은 2^{-64}가 된다. 8-라운드 선형 특성은 2^{-128} 확률로, 16-라운드 선형 특성은 2^{256} 확률로 구성 가능하다.

따라서 256-비트의 비밀키를 적용한 16-라운드 Lambda에 대한 선형공격은 전사공격 방법보다 효과가 없음을 알 수 있다.

5. 결론

본 논문에서는 Lambda 설계 시 전통적인 Feistel 구조가 갖는 확산 효과의 취약점을 보완하기 위해 Feistel 구조에 SPN 구조를 부가한 확장된 Feistel 구조로 설계하였다. 이렇게 설계된 Lambda는 2-라운드 만에 완전 확산이 이루어졌다.

Lambda의 축소된 8-라운드 차분특성이 구성될 확률은 2^{192}로, 16-라운드 차분특성이 구성될 확률은 2^{384}로 분석되었다. 또한, 8-라운드 선형특성이 구성될 확률은 2^{128}로, 16-라운드 선형특성이 구성될 확률은 2^{256}로 분석되었다.

결국, Lambda는 128-비트 비밀키를 적용하였을 경우 8-라운드 이상이면 차분공격이나 선형공격이 전사공격 방법보다 효율성이 떨어짐을 알 수 있었고, 16-라운드로 구성하였을 경우 256-비트 비밀키에 대해서도 전사공격 방법보다 효과가 없음을 증명하였다.

향후 연구로서 Lambda에 대한 부정차분이나 불능차분과 같은 향상된 공격 방법에 대한 연구가 필요하다.

※ 참 고 문 헌

[1] 이문호, "정보이론 -실용 오류정정의 기초", 북두출판사, pp207-212.

[2] E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.

[3] D. E. Denning, Cryptography and Data Security, Addison-Wesley, 1983, pp90-101.

[4] J.Kilian and P.Robshaw, "How to Protect DES Against Exhaustive Key Search," Advances in Cryptology - EUCROCRYPT'96, Springer-Verlag, 1996.

[5] M. Matsui, "Linear Cryptanalysis of DES Cipher (I)", Symposium on Cryptography and Information Security '93, 1993.

[6] National Institute of Standards & Technology, "Announcing Development

of a Federal Information Standard for Advanced Encryption Standard," Federal Register, v.62, 1997.

[7] J.Daemen and V.Rijmen, "The Rijndael Block Cipher," AES Proposal, 1999.

[8] B.Schncier, J.Kelsey, D.Whiting, D. Wagner, C.Hall, N.Ferguson, "Twofish: A 128-Bit Block Cipher," AES Proposal, 1998.

[9] A.F.Webster and S.E.Tavares, "On the Design of S-boxes," Proc. of CRYPTO' 85, Springer-Verlag, 1985.

[10] http://www.nist.gov/aes

※ 부록

```
unsigned int S-box[256] = {
  99,  61, 153,  48, 203, 173, 202,  96, 226, 157,
4, 219,  98,  78,  55,  64,
163,  27, 201, 228,   5, 184,  63, 120, 227,  36,
  32,  80, 156, 244,  39, 222,
3,   8,  95, 171,  54,  33, 117, 119, 133,  45,  91,
161,  77, 141,  59, 103,
  35, 211,  21, 180,  23, 225, 250,  72,  73, 106,
168,  86,  65, 114, 189,  42,
  83, 194, 214, 164, 125,  19, 210, 147,  28,  70,
151, 100, 104, 143, 188, 207,
 197, 154,  68, 149, 127, 209, 215,  11, 116, 122,
20, 245,  79, 155,  97,  29,
  67,  38, 238, 220,  88,  66, 136,  22, 140, 111,
247,  87, 175,  75, 246,  93,
 118, 249,  50, 107, 134,  69,  44, 198, 167, 218,
62, 166, 217, 187,  18, 240,
 174, 237, 179,  76, 108,  56,  85, 212, 185, 158,
142, 146, 110, 223, 206, 254,
   9, 113, 241, 234, 204,   2,  53, 102, 230, 255,
192, 190,  89, 181, 224, 231,
 229,  81, 159,  57,  37, 144,  24,   6, 109,  41,
58, 186, 236, 139, 130, 131,
 232,  16, 239, 121, 216, 253,  40, 132, 160, 123,
31, 152, 183, 233, 137, 138,
 115,  90, 193, 101, 112,  71, 105,  10,  43,  51,
243,  26, 150, 126,  12, 213,
 148, 248, 176, 221, 252,  46, 172,  94, 208,  92,
162,  25, 124,   7, 169,  47,
  60, 182, 251,   0,  30,  84, 178,  82, 145, 195,
165, 242, 196,  17, 177, 128,
   1,  74, 191,  52, 205,  49, 129, 199, 235,  34,
15, 135,  14, 200, 170,  13
};
```

임 웅 택

1985년 2월 : 금오공과대학 전자공학과(전산전공) 졸업
1992년 1월 : 국방대학원 전자계산과 석사
1997년 3월~현재 : 부천대학 전산정보처리과 조교수
1998년 3월~현재 : 숭실대학교 컴퓨터학과 박사과정 수료
관심분야 :정보보호, 블록 암호 설계 및 분석

전 문 석

1981년 2월 : 숭실대학교 전자계산학과 졸업
1986년 2월 : University of Maryland, Computer Science(석사)
1989년 2월 : University of Maryland, Computer Science(박사)
1989년 : Morgan State Univ. 부설 Physical Science Lab. 책임연구원
1989년~1991년 : New Mexico State University 부설 Physical Science Lab. 책임연구원
1991년 3월~현재 : 숭실대학교 컴퓨터학과 정교수
관심분야 : PKI, 침입탐지시스템, 인증시스템, 전자상거래보안, 병렬처리시스템