Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.2, Apr. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.2.123

블록암호 RECTANGLE에 대한 DLCT를 이용한 차분-선형 공격*

조 세 희,[1†] 백 승 준,[1] 김 종 성[2‡]
[1,2]국민대학교 (대학원생, 교수)

Differential-Linear Cryptanalysis Using DLCT on the Block Cipher RECTANGLE*

Sehee Cho,[1†] Seungjun Baek,[1] Jongsung Kim[2‡]
[1,2]Kookmin University (Graduate student, Professor)

요 약

블록암호 알고리즘의 안전성을 점검하거나 공격하는 대표적인 방법은 차분 공격과 선형 공격이다. 이 두 방법을 이용한 차분-선형 공격은 차분 구별자 및 선형 구별자가 서로 독립적이라 가정한 한계점이 있으며, 이를 보완하기 위한 도구로서 최근 DLCT (Differential Linear Connectivity Table)가 제안됐다[1]. 본 논문에서는 제안된 DLCT를 적용한 13-라운드 차분-선형 구별자를 이용해 15-라운드 RECTANGLE의 분석을 제시한다. 또한 2^{79}의 시간 복잡도와 2^{24}의 공간 복잡도를 통해 마스터 키 22 비트를 복구하는 자세한 키 복구 알고리즘을 제시한다.

ABSTRACT

Typical methods of checking or attacking the security of block cipher algorithms are differential cryptanalysis and linear cryptanalysis. Difference-linear cryptanalysis using these two methods have limitation assuming that the differential distinguisher and the linear distinguisher are independent of each other, and a Differential Linear Connectivity Table (DLCT) has recently been proposed as a tool to compensate for them[1]. This paper presents an analysis of block cipher 15-round RECTANGLE through a 13-round differential-linear distinguisher using the proposed DLCT. Also we present a detailed key recovery algorithm that recovers master key 22 bits through 2^{79} time complexity and 2^{24} memory complexity.
Keywords: DLCT, DLC, RECTANGLE, Block Cipher, Cryptanalysis

I. 서 론

차분 공격과 선형 공격은 블록 암호에 대한 대표적인 공격이다. 1994년에 Langford와 Hellman이 차분 공격과 선형 공격을 동시에 이용한 차분-선형 공격[2]을 제시했다. 이후 2002년에 Biham 등에 의해 향상된 차분-선형 공격[3]이 제안되었다. 2019년에는 Bar-On 등이 기존 차분-선형 공격의 한계점을 보완할 수 있는 DLCT (Differential-Linear Connectivity Table)를 이용한 향상된 차분-선형 공격을 제안했다[1]. 차분 구별자와 선형 구별자 간에 DLCT를 적용하면 기존 차분-선형 공격보다 정

Received(10. 15. 2020), Modified(01. 22. 2021), Accepted(01. 22. 2021)
* 본 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00520, SCR-Friendly 대칭키 암호 및 응용모드 개발)
* 본 논문은 2020년도 한국정보보호학회 하계학술대회에 발표한 우수논문을 개선 및 확장한 것임
† 주저자, ghfaos7708@kookmin.ac.kr
‡ 교신저자, jskim@kookmin.ac.kr(Corresponding author)

확한 차분-선형 구별자의 바이어스를 구할 수 있으며, 이를 통해 전체 공격 구별자의 라운드 수도 증가시킬 수 있다.

현재 RECTANGLE에 대한 단일 키 기준 안전성 분석은 제안 논문[4]과 [5]을 제외하고는 진행되지 않았다. RETANGLE 제안 논문[4]에서는 14-라운드 차분 구별자를 이용한 18-라운드 공격을 제안하였으나, 구체적인 공격 과정과 구체적인 공격 복잡도는 제시되지 않았다. 또한 [5]에서는 [4]보다 향상된 차분, 선형 구별자만 제시되어있고 제시된 구별자를 통한 키 복구는 제시되지 않았다. 이에 본 논문에서는 최신 암호 분석 기법인 DLCT를 차분-선형 공격에 적용하여 RECTANGLE의 안전성을 분석하고 이에 따른 키 복구 알고리즘과 복잡도를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 RECTANGLE 암호에 관해 간략하게 서술한다. 3장에서는 본 논문에 쓰이는 기호들을 정리하고 DLCT에 대한 정의와 차분-선형 공격을 서술한다. 또한 차분-선형 공격의 한계점과 DLCT의 필요성을 서술한다. 4장에서는 RECTANGLE-80에 DLCT를 이용한 차분-선형 공격을 적용한 구별자를 정의하고 정의한 구별자를 통해 키 복구 과정 및 이에 따른 공격 복잡도를 정리한다. 마지막으로 5장에서는 결론 및 향후 연구를 제시한다.

II. RECTANGLE 암호 알고리즘

본 장에서는 RECTANGLE 암호 알고리즘에 관해 서술한다. RECTANGLE은 SPN (Substitution Permutation Network) 구조를 갖는 암호 알고리즘이다.

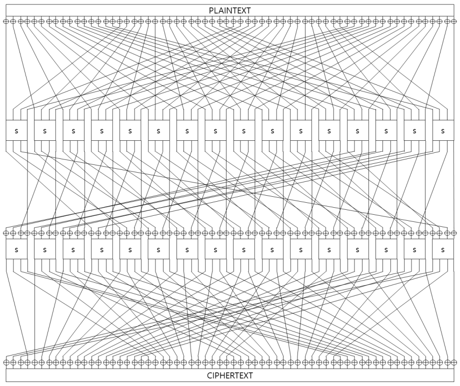

Fig. 1. First 2-round RECTANGLE Block Cipher

RECTANGLE 암호 알고리즘의 총 라운드 수는 25-라운드이고 블록 크키는 64 비트로 이루어져 있다. 키 길이는 80 비트 혹은 128 비트로 이루어진다. 본 논문에서는 80 비트의 키를 사용하는 암호에 대해 공격을 수행한다. 다음 Fig. 1.은 RECTANGLE 암호 알고리즘의 첫 2라운드에 대한 도식이다.

2.1 라운드 구성

각 라운드는 라운드 키를 XOR하는 AddRoundkey, 비선형 연산인 SubColumn, 비트를 확산시키는 ShiftRow로 구성된다. 마지막 라운드 이후 AddRoundKey를 한 번 더 수행한다. 다음 Fig. 2.는 RECTANGLE 블록 암호의 state를 나타낸다.

$$\begin{bmatrix} w_{15} \cdots & w_2 & w_1 & w_0 \\ w_{31} \cdots & w_{18} & w_{17} & w_{16} \\ w_{47} \cdots & w_{34} & w_{33} & w_{32} \\ w_{63} \cdots & w_{50} & w_{49} & w_{48} \end{bmatrix}$$

Fig. 2. State of RECTANGLE

Fig. 2.의 w_0는 Fig. 1.의 LSB (Least Significant Bit)이다.

2.2 SubColumn

RECTANGLE의 SubColumn은 sbox를 이용해 각 비트에 비선형 연산을 적용한다. Table 1.은 RECTANGLE의 sbox를 나타내며 sbox는 4 비트 입력을 4 비트 출력에 매핑한다. Fig. 2. 암호 state를 기준으로 각 열에 sbox를 적용한다. 즉, state의 64 비트는 16번의 sbox를 거친다.

Table 1. RECTANGLE sbox table

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
y	6	5	c	a	1	e	7	9	b	0	3	d	8	f	4	2

$(y = s(x))$

2.3 ShiftRow

RECTANGLE의 ShiftRow는 Fig. 3.과 같다. 두 번째 행은 왼쪽으로 1 비트 로테이션, 세 번째

$$\begin{bmatrix} w_{15} \cdots \ w_2 \ \ w_1 \ \ w_0 \\ w_{31} \cdots w_{18}\, w_{17}\, w_{16} \\ w_{47} \cdots w_{34}\, w_{33}\, w_{32} \\ w_{63} \cdots w_{50}\, w_{49}\, w_{48} \end{bmatrix} \begin{matrix} \lll 0 \\ \lll 1 \\ \lll 12 \\ \lll 13 \end{matrix}$$

Fig. 3. ShiftRow of RECTANGLE

행은 왼쪽으로 12 비트 로테이션, 마지막으로 네 번째 행은 왼쪽으로 13 비트 로테이션을 진행한다.

2.4 Key schedule

RECTANGLE의 Key schedule은 일련의 과정을 통해 총 26개의 라운드 키를 생성한다. Fig. 4.는 80 비트의 key state를 나타낸다.

$$\begin{bmatrix} v_{15} \cdots \ v_2 \ \ v_1 \ \ v_0 \\ v_{31} \cdots v_{18}\, v_{17}\, v_{16} \\ v_{47} \cdots v_{34}\, v_{33}\, v_{32} \\ v_{63} \cdots v_{50}\, v_{49}\, v_{48} \\ v_{79} \cdots v_{66}\, v_{65}\, v_{64} \end{bmatrix}$$

Fig. 4. An 80 bits Key state

key state에서 1행부터 4행을 이용해 64 비트 라운드 키를 얻는다. (라운드 키 = 4행||3행||2행||1행). key state에서 i 번째 라운드 키를 얻은 후 다음과 같은 과정을 통해 key state를 갱신한다. 1)의 S는 sbox, $Row_n(n=0,1,2,3,4)$는 key state의 n 번째 행 그리고 $RC[i]$는 i 번째 5 비트 라운드 상수를 뜻한다.

1) $v'_{(48+j)} \| v'_{(32+j)} \| v'_{(16+j)} \| v'_{(0+j)} = S(v_{(48+j)} \| v_{(32+j)} \| v_{(16+j)} \| v_{(0+j)})(j=0,1,2,3)$

2) $Row'_0 = (Row_0 \lll 8) \oplus Row_1$
$Row'_1 = Row_2$
$Row'_2 = Row_3$
$Row'_3 = (Row_3 \lll 12) \oplus Row_4$
$Row'_4 = Row_0$

3) $v'_4 \| v'_3 \| v'_2 \| v'_1 \| v'_0 = (v_4 \| v_3 \| v_2 \| v_1 \| v_0) \oplus RC[i]$

위와 같은 key state 갱신하고 다시 1행부터 4행을 이용해 64 비트 라운드 키를 얻는다. 이를 반복해 총 26개의 라운드 키를 생성한다.

본 논문에서 라운드 키를 통해 마스터 키를 추측할 때 비선형 연산인 sbox는 고려하지 않았다.

III. 차분-선형 공격과 DLCT

본 장에서는 차분-선형 공격을 설명하고, 차분 구별자와 선형 구별자 사이에 DLCT를 적용하는 방법론을 설명한다. 그리고 기존의 차분-선형 공격의 가정에 따른 한계점을 구체적으로 서술하고 DLCT의 필요성을 제시한다. 다음 Table 2.는 이후 사용될 기호들을 정리한 것이다.

Table 2.의 ?는 4-라운드의 DLCT를 적용한 차분-선형 구별자를 표현한 Table 5.에서 0 또는 1의 비트를 뜻한다. 예를 들어 1의 입력 차분의 결과로 3, 6, 7, $0xb$, $0xe$ 그리고 $0xf$의 출력 차분이 가능하다. $\overline{DLCT}$를 계산하기 위해 입력 차분에 따른 가능한 모든 출력 차분을 고려하기 때문에 0 또는 1의 값을 가지는 각 비트들을 ?로 표현했다. 또한 출력 차분이 정해지면 ? 비트는 0 또는 1의 값이 정해진다.

Table 2. Notations

Notations	Explanation
P	Plaintext
C	Ciphertext
Ω_I	Input Difference
Ω_O	Output Difference
λ_I	Input Mask
λ_O	Output Mask
$\overline{DLCT}$	Probability of Differential-Linear distinguisher with DLCT
?	0 or 1 bit
K_i	ith round key

3.1 차분-선형 공격

[2]에서 처음 제안된 차분-선형 공격 도식은 Fig. 5.와 같다.

Fig. 5.에서 E_0는 차분 구별자, E_1은 선형 구별자이며, 이를 통해 전체 차분-선형 구별자를

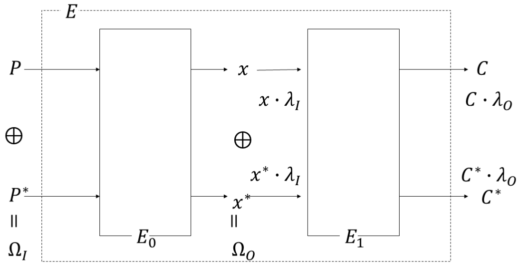

Fig. 5. Differential-Linear distinguisher

$E = E_1 \circ E_0$로 표현할 수 있다.

차분 구별자에서 $\Omega_I \rightarrow \Omega_O$의 확률이 p이고 선형 구별자에서 $\lambda_I \rightarrow \lambda_O$의 바이어스가 q라면 차분-선형 공격 구별자의 전체 바이어스는 식(1)과 같다.

$$
\begin{aligned}
& p((\frac{1}{2}+q)(\frac{1}{2}+q)+(1-(\frac{1}{2}+q))(1-(\frac{1}{2}+q))) \\
& (\because \Pr[x \bullet \lambda_I = C \bullet \lambda_O] = \frac{1}{2}+q, \\
& \Pr[x^* \bullet \lambda_I = C^* \bullet \lambda_O] = \frac{1}{2}+q) \\
& = 2pq^2
\end{aligned}
\tag{1}
$$

3.2 DLCT와 차분-선형 공격에 DLCT의 적용

다음 Table 3.은 식(2)의 관계를 통해 구성된 RECTANGLE sbox에 대한 DLCT이다. 행은 입력 차분을 의미하고 열은 출력 마스킹을 의미한다.

$$S(x) \bullet \lambda = S(x \oplus \Omega) \bullet \lambda \tag{2}$$

Table 3. DLCT of RECTANGLE sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
1	16	8	0	8	4	4	12	12	8	8	8	8	8	8	8	8
2	16	8	8	8	4	12	8	8	4	8	4	8	8	4	12	8
3	16	8	8	8	12	4	8	8	4	8	4	8	4	8	8	12
4	16	0	0	16	8	8	8	8	8	8	8	8	8	8	8	8
5	16	8	16	8	8	8	8	8	8	8	8	8	4	4	4	4
6	16	8	8	8	8	8	4	4	8	12	8	4	12	8	8	4
7	16	8	8	8	8	8	4	4	8	12	8	4	8	12	4	8
8	16	0	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	16	8	8	0	4	4	12	12	8	8	8	8	8	8	8	8
a	16	8	8	8	4	12	8	8	8	4	8	4	4	8	8	12
b	16	8	8	8	12	4	8	8	8	4	8	4	8	4	12	8
c	16	16	8	8	8	8	8	8	4	4	12	12	4	4	4	4
d	16	8	8	0	8	8	8	8	4	4	12	12	8	8	8	8
e	16	8	8	8	8	8	4	4	12	8	4	8	8	12	4	8
f	16	8	8	8	8	8	4	4	12	8	4	8	12	8	8	4

E_0와 E_1을 서로 연결하기 위해 식(2)와 같이 Ω_O과 λ_I의 관계식을 가지는 DLCT를 이용해 구성한 구별자 E_m을 넣는다. 이를 도식화하면 다음 Fig. 6.와 같다.

본 논문에서 선형 마스킹의 결괏값을 0을 기준으로 설정했다. 따라서 DLCT를 구성한 식(2)를 만족한다면 결과적으로 다음과 같은 식이 유도된다.

$$
\begin{aligned}
& \lambda_I \bullet x \oplus \lambda_O \bullet E_1(x) = 0 \\
& \lambda_I \bullet x^* \oplus \lambda_O \bullet E_1(x^*) = 0 \\
& (E_0(P) = x, E_0(P^*) = x^*) \\
& \lambda_I \bullet x \oplus \lambda_O \bullet E_1(x) = \lambda_I \bullet x^* \oplus \lambda_O \bullet E_1(x^*) \\
& \Rightarrow \lambda_I \bullet (x \oplus x^*) \oplus \lambda_O \bullet (E_1(x) \oplus E_1(x^*)) = 0 \\
& \Rightarrow \lambda_I \bullet \Omega_O \oplus \lambda_O (E_1(x) \oplus E_1(x^*)) = 0 \\
& If\, \lambda_I \bullet x = \lambda_I \bullet x^* \\
& \Rightarrow \lambda_O \bullet (E_1(x) \oplus E_1(x^*)) = 0\,(\because \lambda_I \bullet \Omega_O = 0)
\end{aligned}
$$

따라서 E_m에서는 Ω_O와 λ_I의 종속성을 고려하기 위해 $S(x) \bullet \lambda_I = S(x \oplus \Omega_O) \bullet \lambda_I$의 관계를 차분 구별자와 선형 구별자 사이에 적용한다. 이때, 전체 구별자를 $E = E_1 \circ E_m \circ E_0$으로 표현한다.

본 논문에서는 E_m을 총 5-라운드로 구성했다. E_m은 총 19개의 active sbox로 구성되었으며 E_m의 마지막 active sbox의 입력 차분에 대해 식 $S(x) \bullet \lambda_I = S(x \oplus \Omega_O) \bullet \lambda_I$의 값을 만족하는지 확인하였다. 위의 과정을 통해 5-라운드 E_m의 확률을 구했으며, 자세한 과정은 4.1.2에서 서술한다.

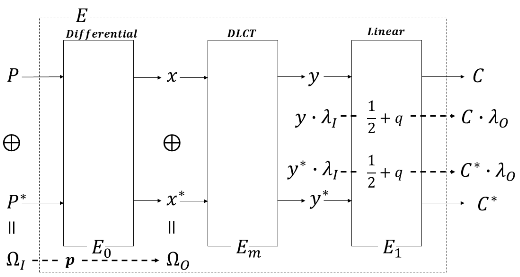

Fig. 6. Differential Linear distinguisher with DLCT

E_m을 포함한 전체 구별자 E의 바이어스는 식(3)과 같이 나타낸다.

$$2p\overline{DLCT}q^2 \quad (3)$$
$$(\overline{DLCT} = \Pr[S(x) \cdot \lambda_I = S(x \oplus \Omega_O) \cdot \lambda_I] - \frac{1}{2})$$

3.3 차분-선형 공격의 한계점과 DLCT의 필요성

기존 차분-선형 공격에서는 E_0와 E_1이 독립적이고, 또한 차분 구별자, 선형 구별자의 active sbox 확산 양상이 서로 독립적이라고 가정했다. 그러나 차분 구별자 E_0의 입력값은 $P \oplus P^* = \Omega_I$의 관계에 있는 종속적인 두 개의 평문 P, P^*이므로 $E_0(P)$와 $E_0(P^*)$는 확률적으로 $E_0(P) \oplus E_0(P^*) = \Omega_O$의 관계에 있다. 즉, $E_0(P)$와 $E_0(P^*)$는 서로 독립적이지 않다. 따라서 E_1의 입력값인 $E_0(P)$와 $E_0(P^*)$가 종속적이므로 E_0와 E_1은 서로 종속적이라 할 수 있다. 또한 E_0의 확률, 즉, $\Pr[\Omega_I \rightarrow \Omega_O]$이 1보다 작은 경우 차분 구별자 E_0를 만족하지 않는 평문 쌍 (P, P^*)에 대해 $E_0(P) \cdot \lambda_I = E_0(P^*) \cdot \lambda_I$의 관계를 만족할 확률이 1보다 작기에 $E_0(P) \cdot \lambda_I = E_0(P^*) \cdot \lambda_I$을 만족할 확률에 대한 계산이 필요하다. 따라서 E_0와 E_1사이에 E_m을 추가하여 E_0의 출력 차분에 대해 $E_0(P) \cdot \lambda_I = E_0(P^*) \cdot \lambda_I$의 확률을 계산함과 동시에 E_0와 E_1을 연결하여 종속성을 따진다. 기존 차분-선형 구별자의 E_0에서 E_1의 active sbox를 E_m을 통해 연결한다. E_m의 입력 차분에 따른 모든 출력 차분을 전부 고려하여 E_0와 E_1의 active sbox의 확산 양상을 종속적으로 따진다.

Table 3.의 (색칠된 부분)을 보면 가능한 16개의 입력값에 대한 식(2)의 값이 모두 한쪽으로 치우쳐있는 것을 볼 수 있다. 이는 식(2)를 만족할 확률이 0 또는 1이라는 것을 뜻한다. 이 확률은 〔3〕에서 식(2)를 만족할 확률인 $\frac{1}{2}$과 차이가 크다.(〔2〕에서는 $\Pr[S(x) \cdot \lambda = S(x \oplus \Omega) \cdot \lambda] = 1$을 가정했다. 〔3〕에서는 $S(x) \cdot \lambda$와 $S(x \oplus \Omega) \cdot \lambda$의 값이 서로 독립적이고 균일하다고 가정했다. 따라서 차분 구별자를 성립하지 않는 경우의 입력 쌍 $(x, x \oplus \Omega)$에 대해 $\Pr[S(x) \cdot \lambda = S(x \oplus \Omega) \cdot \lambda] = \frac{1}{2}$로 설정했다). 이는 E_0와 E_1이 종속적이라는 것으로 해석할 수 있다. 따라서 E_0와 E_1 사이에 E_1의 첫 라운드를 포함하는 DLCT를 이용해 구성된 구별자 E_m을 적용하면 〔2〕, 〔3〕에서 E_0와 E_1이 독립적이라고 가정한 한계를 극복할 수 있으며, 이를 통해 더욱 정확한 차분-선형 구별자의 확률을 구할 수 있다.

IV. 15라운드 RECTANGLE-80에 대한 DLCT 를 적용한 차분-선형 공격

본 장에서는 RECTANGLE에 DLCT를 적용한 13-라운드 차분-선형 공격 구별자를 설명한다. 또한 이를 기반으로 앞, 뒤로 1라운드씩 키 복구 라운드를 붙여 총 15-라운드 RECTANGLE-80에 대한 키 복구 공격을 제안한다. 표로 표현된 공격 구별자들은 sbox의 입, 출력을 기준으로 구성했고 라운드는 암호 알고리즘의 라운드를 기준으로 작성했다.

4.1 공격 구별자

공격 구별자는 4-라운드의 E_0, 5-라운드의 E_m 그리고 4-라운드의 E_1로 구성되며, 총 13-라운드이다.

4.1.1 차분 구별자

Fig. 7.은 E_0의 첫 1라운드를 도식화한 것이다. 우측부터 7, 12번째 sbox의 입력 차분은 $0x6$, $0x5$이고 출력 차분은 $0x2$, $0x4$이다. 이 비트들은 다음 라운드의 8번째 sbox의 $0x6$의 입력 차분이다.

전체 E_0는 Table 4.와 같다. E_0는 〔4〕에서 제안된 최적의 확률을 갖는 5-라운드 차분 구별자를 이용하여 구성했으며, 마지막 라운드의 active sbox가 E_m과 연결될 수 있도록 했다.

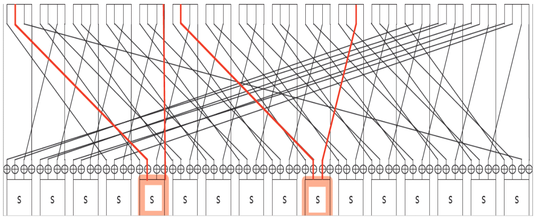

Fig. 7. 1st-round of Differential distinguisher

Table 4. 4-round Differential distinguisher

Round	Bias $(2^{-\alpha})$	Input difference	Output difference
2	4	0000100000000000 0000000001000000 0000100001000000 0000000000000000	0000000000000000 0000000001000000 0000100000000000 0000000000000000
3	2	0000000000000000 0000000010000000 0000000010000000 0000000000000000	0000000000000000 0000000010000000 0000000000000000 0000000000000000
4	3	0000000000000000 0000000100000000 0000000000000000 0000000000000000	0000000000000000 0000000000000000 0000000000000000 0000000100000000
5	3	0000000000000000 0000000000000000 0000000000000000 0000000000100000	0000000000100000 0000000000000000 0000000000000000 0000000000000000

4.1.2 DLCT를 이용해 구성된 구별자 E_m

E_m의 입력 차분은 구별자의 계산 복잡도를 최소화하기 위해 1개의 active sbox로 제한했다. 5-라운드 E_m은 Table 5.와 같다.
DLCT를 적용한 5-라운드 E_m의 도식은 Fig. 8.과 같다.

Table 5. 5-round DLCT distinguisher E_m

Round	Input difference	Output difference
6	0000000000100000 0000000000000000 0000000000000000 0000000000000000	0000000000?00000 0000000000?00000 0000000000?00000 0000000000?00000
7	0000000000?00000 000000000?000000 00000000000000?0 0000000000000?00	000000000??00??0 000000000??00??0 000000000??00??0 000000000??00??0
8	000000000??00??0 00000000??00??00 0??00000000000??0 ??000000000??00	??000000000??00 0??00000000000??0 00000000??00??00 000000000??00??0
9	??000000000??00 ??000000000??00 ??000000000??00 ??000000000??00	?000000000000000 0?00000000000000 000000000000?000 0000000000000?00
10	?000000000000000 ?000000000000000 ?000000000000000 ?000000000000000	1000000000000000 0000000000000000 0000000000000000 0000000000000000

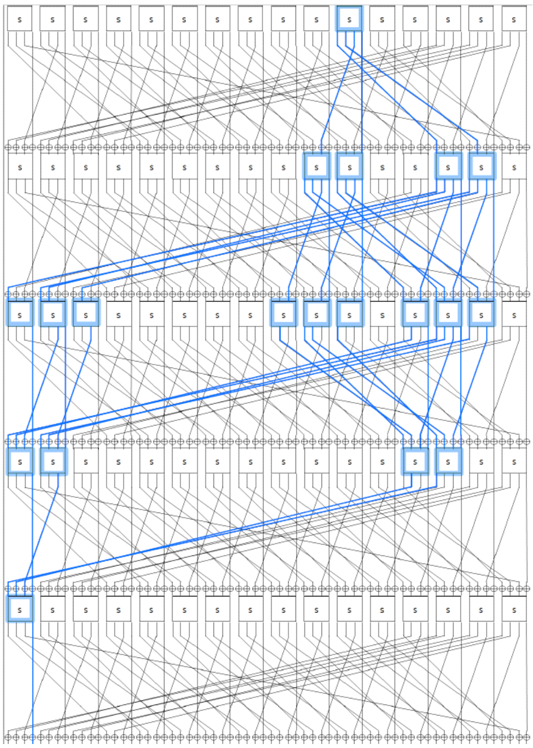

Fig. 8. 5-round DLCT distinguisher E_m

Table 6.은 E_m의 마지막 라운드에 대한 표이다. Table 6.을 식(4)와 같이 표현할 수 있다.

$$S(x) \cdot \lambda_I = S(x \oplus \Omega_O) \cdot \lambda_I$$
$$(\Omega_O = \{0,1,2 \cdots 15\}, \lambda_I = 1) \qquad (4)$$

Table 5.에서 확인할 수 있듯이 E_m의 입력 차분은 $0x0000000000100000$이다. 따라서 6번째 sbox의 입력 차분 $0x1$에 대해 DDT (Differential Distribution Table)를 통해 알 수 있는 가능한 출력 차분 $\{3, 6, 7, 0xB, 0xE, 0xF\}$을 전부 고려한다. 6라운드의 active sbox에 의해 생긴 출력 차분에 대해 7라운드의 active sbox와 각각의 입력 차

Table 6. 5th round of E_m

Round	Prob $(2^{-\alpha})$	Input difference	Output mask
10	1.09	?000000000000000 ?000000000000000 ?000000000000000 ?000000000000000	1000000000000000 0000000000000000 0000000000000000 0000000000000000

분이 정해진다. 7라운드의 정해진 각각의 active sbox에 대해 다시 가능한 출력 차분을 계산한다. 이와 같은 방식으로 선형 구별자의 λ_I에 해당하는 값들에 대해 $S(x) \cdot \lambda_I = S(x \oplus \Omega_O) \cdot \lambda_I$을 계산한다. 6라운드의 가능한 출력 차분 $\{3, 6, 7, 0xB, 0xE, 0xF\}$에 대해 모두 수행한다. 즉, 10라운드의 active sbox에 DLCT를 적용해 가능한 모든 입력 차분에 대해 $S(x) \cdot \lambda_I = S(x \oplus \Omega_O) \cdot \lambda_I$이 되는 확률을 구한다. 본 논문에서는 E_1의 최대 라운드를 구성하기 위해 $\lambda_I = 1$로 고정했다.

$\overline{DLCT}$ 계산에 대한 알고리즘은 다음과 같다.

1) temp와 카운터를 초기화한다.

2) 고정된 입력 차분 $0x1$에 대해 DDT를 통해 알 수 있는 가능한 출력 차분 $\{0x3, 0x6, 0xB, 0xE, 0xF\}$중 하나를 선택한다.

3) 2)에서 선택된 출력 차분에 대해 7라운드 1, 2, 5, 6번째 각각의 sbox 입력 차분이 정해진다. 각 sbox에서 정해진 입력 차분에 따라 가능한 출력 차분 중 하나를 각각 선택한다.

4) 3)에서 정해진 7라운드 sbox의 출력 차분에 대해 8라운드 1, 2, 3, 5, 6, 7, 13, 14, 15번째 sbox에 영향을 주는 차분을 각각의 sbox 입력 차분으로 한다.

5) 8라운드 sbox에 대해 입력 차분이 정해지면 가능한 출력 차분 중 하나를 선택한다. 각 출력 차분에 대해 9라운드 2, 3, 14, 15번째 sbox에 영향을 주는 차분을 9라운드 각각의 sbox 입력 차분으로 한다.

6) 9라운드 각각의 sbox 입력 차분에 대해 가능한 출력 차분 중 하나를 선택한다. 선택된 각 sbox의 출력 차분 중 10라운드 15번째 sbox에 영향을 주는 비트만 선택하여 각각의 sbox 입력 차분으로 설정한다.

7) 설정된 입력 차분을 Ω^*라 하자. 만약 $S(x) \cdot 1 = S(x \oplus \Omega^*) \cdot 1$을 만족하면 Table 3.을 이용해 확률을 구하고 temp에 더한다. 이후 해당 카운터를 1 증가시킨다.

8) 각 단계에서 가능한 모든 입력 차분과 출력 차분을 고려할 때까지 2)-7)의 과정을 반복한다.

9) temp의 값을 카운터로 나누어 $\overline{DLCT}$의 확률을 구한다.

위의 알고리즘을 통해 $E_m \simeq 2^{-1.09}$임을 알 수 있다.

4.1.3 선형 구별자

계산 복잡도를 최소화하면서 최대한 많은 키 복구를 위해 E_1을 두 가지 경우로 나누었다. 두 가지 경우의 E_1은 Table 7.과 Table 8.과 같다. 두 개의 E_1은 11-13라운드의 구성은 같고 14라운드의 λ_O는 각각 $0x0000006000060000$, $0x000000B0000E0000$이다.

Table 7. First 4-round Linear distinguisher

Round	Bias ($2^{-\alpha}$)	Input mask	Output mask
11	3	1000000000000000 0000000000000000 0000000000000000 0000000000000000	0000000000000000 0000000000000000 0000000000000000 1000000000000000
12	3	0000000000000000 0000000000000000 0000000000000000 0001000000000000	0000000000000000 0000000000000000 0001000000000000 0000000000000000
13	2	0000000000000000 0000000000000000 0000000100000000 0000000000000000	0000000000000000 0000000100000000 0000000100000000 0000000000000000
14	4	0000000000000000 0000001000000000 0000000000010000 0000000000000000	0000000000000000 0000001000010000 0000001000010000 0000000000000000

Table 8. Second 4-round Linear distinguisher

Round	Bias ($2^{-\alpha}$)	Input mask	Output mask
11	3	1000000000000000 0000000000000000 0000000000000000 0000000000000000	0000000000000000 0000000000000000 0000000000000000 1000000000000000
12	3	0000000000000000 0000000000000000 0000000000000000 0001000000000000	0000000000000000 0000000000000000 0001000000000000 0000000000000000
13	2	0000000000000000 0000000000000000 0000000100000000 0000000000000000	0000000000000000 0000000100000000 0000000100000000 0000000000000000
14	4	0000000000000000 0000001000000000 0000000000010000 0000000000000000	0000001000000000 0000001000010000 0000000000010000 0000001000010000

4.2 키 복구 공격

본 절에서는 앞에서 정의한 13-라운드의 E를 통해 RECTANGLE 암호 알고리즘의 15-라운드를 공격하고 공격 복잡도를 정리한다. E는 2-5라운드의 E_0, 6-10라운드는 E_m 그리고 11-14라운드는 E_1로 구성했다.

4.2.1 키 복구 과정

키 복구 과정은 총 7단계의 과정을 통해 $K1$ 16 비트와 whitening key $K16$ 24 비트를 복구한다. $K1$과 $K16$의 복구되는 키 비트는 Table 9.와 같다.

Table 7.의 첫 번째 E_1을 먼저 이용하여 공격한다. 공격 알고리즘은 다음과 같다.

1) 1라운드의 부분 키 16 비트의 키 후보를 하나 결정한다.

2) 1라운드의 복구하려는 부분 키 16 비트에서는 $0\sim(2^{16}-1)$의 값을 가지고 나머지 48 비트들이 고정된 상수를 갖는 $2^{40.18}$개의 스트럭쳐를 준비한다. 1)에서 결정한 키 후보로부터 주어진 스트럭쳐를 부분 암호화한 값이 $0\sim(2^{16}-1)$의 값을 갖는 평문 $P^0,\cdots,P^{2^{16}-1}$을 획득하고, 각각을 암호화하여 $C^0,\cdots,C^{2^{16}-1}$을 얻는다.

3) 2)에서 얻은 2^{16}개의 평문, 암호문 쌍으로 구성된 하나의 스트럭쳐로부터 두 평문을 1라운드의 부분 키 16 비트들의 자리에 대해 부분 암호화했을 때, $\Omega_I=0x0000500006000000$가 되도록 쌍을 지어 2^{15}개의 평문 쌍을 얻는다.

4) 평문 P^i와 암호문 C^i에 대해 $Y^i=Subcolumn^{-1}(ShiftRow^{-1}(C^i\oplus K16))$라 하자. $K1$ 16 비트 후보가 하나 결정됐을 때 부분키 $K16$ 12 비트를 복구하기 위한 2^{24}의 카운터(T)를 초기화한다. 그리고 모든 평문 쌍 (P^i,P^j)에 대하여, Y^i와 Y^j의 값이 같은 쌍의 개수를 저장한다.

5) $2^{40.18}$개의 스트럭쳐들로부터 얻을 수 있는 전체 쌍의 개수를 N이라 하자. 모든 값이 있는 T가 $\left|T-\frac{N}{2}\right|\approx 2^{-40.18}\times N$을 만족할 경우, 각각의 키 후보들을 실제 $K16$의 부분 키로 추정한다. 그리고 1)에서 결정한 $K1$ 16 비트를 실제 $K1$의 부분 키로 추정한다. 만족하지 않으면 1)로 돌아가 키 후보를 변경하고 2)~5)의 과정을 반복한다.

6) T를 초기화하고 위에서 얻은 $2^{55.18}$개의 평문 쌍에 대해 두 번째 선형 구별자를 적용한다. $K16$ 24 비트 부분 키 후보를 하나 결정한다. 결정한 키 후보를 통해 $Y_1^i\oplus Y_2^i\oplus Y_6^i\oplus Y_7^i\oplus Y_{10}^i\oplus Y_{11}^i$와 $Y_1^j\oplus Y_2^j\oplus Y_6^j\oplus Y_7^j\oplus Y_{10}^j\oplus Y_{11}^j$ (Y_j^i에서 j는 j번째 sbox)의 값이 같은 경우 카운터를 증가시킨다.

7) 카운터 중 가장 큰 값을 가지는 값을 실제 15라운드의 whitening key $K16$의 부분 키로 추정한다.

Table 9. Recovered key bits of $K1$ and $K16$

Key	Recovered key bits
$K1$	5, 10, 11, 15, 21, 26, 27, 31, 37, 42, 43, 47, 53, 58, 59, 63
$K16$	0, 1, 5, 6, 9, 10, 17, 18, 22, 23, 26, 27, 33, 34, 37, 38, 44, 45, 50, 51, 54, 55, 61, 62

위의 공격 알고리즘을 통해 $K1$ 16 비트와 $K16$ 24 비트를 복구할 수 있다. RECTANGLE-128에 대한 공격은 전수조사의 크기를 늘리는 것으로 공격할 수 있다.

4.2.2 공격 복잡도

각 구별자에 대한 확률과 바이어스는 다음과 같다. 선형 바이어스는 [6]의 Piling-up Lemma에 의해 계산된다. Piling-up Lemma는 식(5)과 같다.

$$2^{n-1}\prod_{i=1}^{n}(p_i-\frac{1}{2}) \tag{5}$$

차분 확률 : 2^{-12}, 선형 바이어스 : 2^{-8}, E_m 확률 : $2^{-1.09}$으로 총 13-라운드의 구별자의 확률은 $\frac{1}{2}+2^{-28.09}$이다. 따라서 해당 공격을 위한 필요 평문 쌍은 $2^{55.18}$쌍이고 84.13%의 성공 확률을 갖는다 [6].

다음 Table 10.은 복구하는 키 비트와 그에 따른 시간 복잡도와 공간 복잡도이다.

Table 10. Complexity

Time	$2^{56.18}\times\left[\left(2^{16}\times2^{12}\times\frac{7}{240}\right)+\left(2^{24}\times\frac{6}{240}\right)\right]\approx2^{79}$
Data	2^{24}

V. 결론 및 향후 계획

본 논문에서는 경량 블록 암호인 RECTANGLE-80에 DLCT를 적용한 차분-선형 13-라운드 E를 제안했다. 이 E를 이용해 15-라운드의 공격을 수행했으며, $K1$의 16 비트와 $K16$의 24 비트를 통해 마스터 키 22 비트를 복구했다. 이는 RECTANGLE 제안 논문[4]에서 제시한 최대 라운드 공격인 18-라운드 차분 공격에 비해 적은 라운드이다. 그러나 해당 차분 공격에서는 구체적인 키 복구 과정과 공격 복잡도는 제시되어있지 않다. 또한 [4], [5]의 결과를 제외하고는 단일 키 공격을 통한 분석 논문은 찾을 수 없었다. 따라서 본 논문의 공격은 신규 암호 분석 기법을 통한 암호 분석 관점에서 의미 있는 결과라고 할 수 있다.

본 논문의 5-라운드 E_m의 확률은 $2^{-1.09}$이다. 이는 같은 라운드의 차분 구별자의 확률, 선형 구별자의 바이어스와 비교해 확실히 좋은 확률을 갖는다. 하지만 E_m의 라운드 수는 블록 암호의 차분 관점의 Full-diffusion 라운드 수에 종속되며, 특히 RECTANGLE의 Full-diffusion 라운드는 4-라운드이다. 따라서 E_m의 4번째 라운드에서 모든 sbox를 active로 설정하면, 선형 구별자의 첫 라운드에서 active sbox를 2개 이상으로 설정한 후 라운드를 늘릴 수 있을 것으로 예상한다. 하지만 이를 위해서는 최소 16개의 active sbox를 추가로 고려하므로 확률 계산을 위해서는 더 높은 계산 복잡도가 필요하다. 관련 연구를 통해 16-라운드 이상의 RECTANGLE 분석을 수행하는 것이 향후 계획이다.

References

[1] Achiya Bar-On, Orr Dunkelman, Nathan Keller and Ariel Weizman, "DLCT : A New Tool for Differential-Linear Cryptanalysis," Advances in Cryptology - EUROCRYPT'19, LNCS 11476, pp. 313-342. May. 2019

[2] Susan K Langford and Martin E Hellman, "Differential-Linear Cryptanalysis," Advances in Cryptology - CRYPTO'94, vol. 839, pp. 17-25, Aug. 1994

[3] Eli Biham, Orr Dunkelman and Nathan Keller, "Enhancing Differential-Linear Cryptanalysis", Advances in Cryptology - ASIACRYPT'02, LNCS 2501, pp. 254-266, Dec. 2002

[4] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Nincent Rijmen, Bohan Yang and Ingrid Verbauwhede, "RECTANGLE: A Bit-slice Lightweight Block Cipher Suitable for Multiple Platforms," Science China Information Sciences, 58, pp. 1-15, Nov. 2015

[5] Hall-Andersen, Mathias, and Philip S. Vejre, "Generating graphs packed with paths estimation of linear approximations and differentials," Iacr Transactions on Symmetric Cryptology, pp. 265-289, Sep. 2018

[6] Mitsuru Matsui, "Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology - EUROCRYPT'93, vol. 765, pp. 386-397, May. 1993

[7] Eli Biham, Orr Dunkelman and Nathan Keller, "Differential-Linear Cryptanalysis of Serpent," International Workshop on Fast Software Encryption, LNCS 2887, pp. 9-21, Feb. 2003

[8] Jinyong Shan, Lei Hu, Ling Song, Siwei Sun, and Xiaoshuang Ma, "Related-Key Differential Attack on Round Reduced RECTANGLE-80," IACR Cryptology ePrint Archive, 986, 2014

〈저자소개〉

조 세 희 (Sehee Cho) 학생회원
2021년 2월: 국민대학교 정보보안암호수학과 졸업
2021년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 정보보호, 암호 알고리즘

백 승 준 (Seungjun Baek) 학생회원
2019년 2월: 국민대학교 수학과 졸업
2020년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 정보보호, 암호 알고리즘

김 종 성 (Jongsung Kim) 종신회원
2000년 8월/2002년 8월: 고려대학교 수학 전공 학사/이학석사
2006년 11월: K.U.Leuven. ESAT/SCD-COSIC 정보보호 전공 공학박사
2007년 2월: 고려대학교 정보보호대학원 공학박사
2007년 3월~2009년 8월: 고려대학교 정보보호기술연구센터 연구교수
2009년 9월~2013년 2월: 경남대학교 e-비즈니스학과 조교수
2013년 3월~2017년 2월: 국민대학교 수학과 부교수
2014년 3월~2020년 8월: 국민대학교 일반대학원 금융정보보안학과 부교수
2017년 3월~2020년 8월: 국민대학교 정보보안암호수학과 부교수
2020년 9월~현재: 국민대학교 정보보안암호수학과/일반대학원 금융정보보안학과 교수
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식