한국항행학회 논문지 제16권 제5호 2012년 10월

경량 블록 암호 LBlock에 대한 차분 오류 공격

Differential Fault Analysis on Lightweight Block Cipher LBlock

정기태*, 이창훈**

Ki-Tae Jeong*, Chang-Hoon Lee**

요 약

64-비트 블록 암호 LBlock은 무선 센서 네트워크 환경과 같이 제한된 환경에 적합하도록 설계된 경량 블록 암호이다. 본 논문에서는 LBlock에 대한 차분 오류 공격을 제안한다. 랜덤 니블 오류 주입 가정에 기반을 둔 이 공격은 평균 5개의 랜덤 니블 오류와 2^{25}의 전수조사를 이용하여, LBlock의 비밀키를 복구한다. 이는 일반적인 수 초 내에 가능함을 의미한다. 본 논문의 공격 결과는 기제안된 LBlock에 대한 차분 오류 공격 결과보다 더 효율적이다.

Abstract

LBlock is a 64-bit ultra-light block cipher suitable for the constrained environments such as wireless sensor network environments. In this paper, we propose a differential fault analysis on LBlock. Based on a random nibble fault model, our attack can recover the secret key of LBlock by using the exhaustive search of 2^{25} and five random nibble fault injection on average. It can be simulated on a general PC within a few seconds. This result is superior to known differential fault analytic result on LBlock.

Key words : Block Cipher, LBlock, Differential fault analysis

I. 서 론

ACNS 2011에 제안된 경량 블록 암호 LBlock은 80-비트 비밀키를 사용하는 64-비트 블록 암호로서, 무선 센서 네트워크 환경과 같이 제한된 환경에 적합하도록 설계되었다 [1]. 이 알고리즘은 Feistel 구조를 가지며, 전체 라운드 수는 32이다. 기제안된 LBlock에 대한 이론적인 분석 결과로는 축소된 버전에 대한 불능 차분 공격이 제안되었다 [2,3].

차분 오류 공격 (differential fault analysis, DFA)는 대표적인 부채널 공격 중의 하나로서, 1997년 Biham과 Shamir에 의해 처음 소개되었다 [4]. 이후 DFA는 SEED [5], ARIA-128 [6], PRESENT [7], LED-64 [8], Piccolo-80 [9] 등 대부분의 블록 암호에 적용되었다. 한편, COSADE 2012에서는 LBlock에 대한 DFA가 제안되었다 [10]. 랜덤 비트 오류 주입 가정에 기반을 둔 이 공격은 최소 7개의 오류 주입을 필요로 한다. 이때 필요한 계산 복잡도에 대한 언급되어 있지 않다.

본 논문에서는 LBlock에 대한 DFA를 제안한다. 본 논문에서 제안하는 공격은 라운드 29의 입력 레지스터에 랜덤 니블 오류를 주입한다는 가정을 이용

* 고려대학교 정보보호연구원(Center for Information Security Technologies(CIST), Korea University)
** 서울과학기술대학교 컴퓨터공학과(Department of Computer Science and Engineering, Seoul National University of Science and Technology)
· 교신저자 (Corresponding Author) : 이창훈
· 투고일자 : 2012년 10월 2일
· 심사(수정)일자 : 2012년 10월 4일 (수정일자 : 2012년 10월 23일)
· 게재일자 : 2012년 10월 30일

한다. 일반적인 PC에서 이 공격을 구현한 결과, 평균 5개의 오류 주입과 2^{25}의 전수조사를 이용하여, 수초 내에 LBlock의 비밀키를 복구할 수 있었다. 이 공격 결과는 [10]의 공격 보다 적은 수의 오류 주입을 필요로 한다.

본 논문은 다음과 같이 구성되어 있다. 먼저, 2절에서는 블록 암호 LBlock의 구조를 소개한다. 3절에서는 오류 주입 가정과 오류가 주입된 위치를 계산하는 방법을 소개한 후, 4절에서 LBlock에 대한 DFA를 소개한다. 마지막으로 5절에서 결론을 맺는다.

Ⅱ. LBlock

64-비트 블록 암호 LBlock은 그림 1과 같이 80-비트 비밀키를 사용하고 32-라운드 Feistel 구조를 갖는다.

본 논문에서는 다음과 같은 표기법을 사용한다. 여기서 32-비트 값은 4-비트 단위로 표기된다. 예를 들어, 32-비트 값 $X=(X_7,X_6,\cdots,X_0)$이다.

- $P=(P^L,P^R)$: 64-비트 평문.
- $C=(C^L,C^R)$: 64-비트 암호문.
- $I_r=(I_r^L,I_r^R)$: 라운드 r의 64-비트 입력값 $(r=1,2,\cdots,32)$.
- $K_r=(K_{r,7},K_{r,6},\cdots,K_{r,0})$: 라운드 r의 라운드 키.

라운드 1의 입력값 $I_1=P$는 라운드 함수 F와 좌측 8-비트 순환 이동 연산을 그림 1과 같이 32번 반복 적용하여 암호문 $C=(C^L,C^R)$이 된다. 32-비트 값을 입력받아 32-비트 값을 출력하는 라운드 함수 F는 그림 2와 같이 8개의 S-box S_7, S_6, …, S_0와 니블 단위 치환 함수로 구성된다. 이 함수들에 대한 자세한 소개는 [1]을 참조하라.

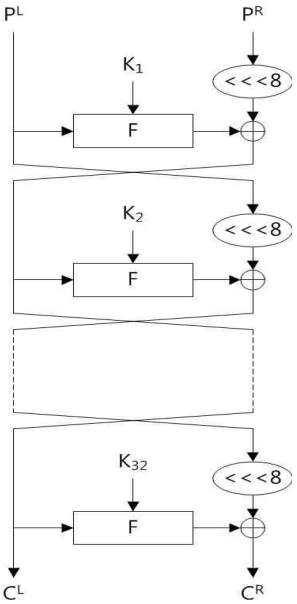

그림 1. LBlock의 전체 구조
Fig. 1. The structure of LBlock.

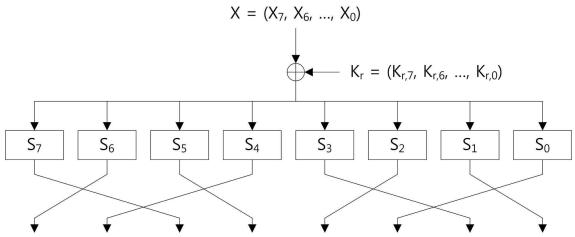

그림 2. 라운드 함수 F
Fig. 2. Round function F of LBlock.

LBlock의 키스케줄은 다음과 같은 단계로 구성된다. 먼저 80-비트 비밀키 $K=(k_{79},k_{78},\cdots,k_0)$은 80-비트 레지스터 K에 그대로 저장된다. 여기서 라운드 1의 라운드 키 K_1은 K의 왼쪽 최상위 32-비트 값이 된다. 그리고 나머지 라운드 키를 생성하기 위해 다음 단계를 수행한다. 여기서, $[r]_2$는 r의 비트 표현을 의미한다. 그리고 S-box S_9와 S_8에 대한 자세한 소개는 [1]을 참조하라.

- $r=1,2,\cdots,31$에 대해, 다음과 같이 레지스터 K를 갱신한다.

(1) $K \lll 29$.
(2) $(k_{79},k_{78},k_{77},k_{76})=S9[(k_{79},k_{78},k_{77},k_{76})]$.
(3) $(k_{75},k_{74},k_{73},k_{72})=S9[(k_{75},k_{74},k_{73},k_{72})]$.

(4) $(k_{50},k_{49},k_{48},k_{47},k_{46})\oplus[r]_2$.

(5) 레지스터 K의 왼쪽 최상위 32-비트 값을 라운드 $r+1$의 라운드 키 K_{r+1}로 출력한다.

표 1. 라운드 키에 사용된 비밀키
Table 1. Secret key used in round keys.

라운드 r	비밀키 정보
1	$(k_{79},k_{78},\cdots,k_{48})$
2	$(k_{50},k_{49},\cdots,k_{19})$
⋮	⋮
29	$(k_{67},k_{66},\cdots,k_{36})$
30	$(k_{38},k_{37},\cdots,k_{7})$
31	$(k_{9},k_{8},\cdots,k_{0},k_{79},k_{78},\cdots,k_{58})$
32	$(k_{60},k_{59},\cdots,k_{29})$

표 1은 각각의 라운드 키에 사용된 비밀키 정보를 나타낸 것이다. 예를 들어, 라운드 29의 라운드 키 K_{29}에 비밀키 정보 $(k_{67},k_{66},\cdots,k_{36})$이 적용되었다.

Ⅲ. 오류 주입 가정 및 오류 주입 위치 계산

본 절에서는 라운드 29의 입력 레지스터에 오류 주입을 가정한 후, 주입된 오류의 위치를 계산하는 방법을 소개한다.

3-1 오류 주입 가정

본 논문에서 제안하는 공격의 오류 주입 가정은 다음과 같다.

- 공격자는 한 개의 평문을 선택한 후, 오류가 발생하지 않은 알고리즘을 이용한 평문/암호문 쌍 (P,C)와, 오류가 발생한 알고리즘을 이용한 평문/암호문 쌍 (P,C^*)를 얻을 수 있다.
- 공격자는 라운드 29의 입력 레지스터에 랜덤 니블 오류를 주입할 수 있다.
- 공격자는 오류의 위치와 오류 주입을 통해 발생하는 차분값을 알 수 없다.

위의 오류 주입 가정에 따라, 라운드 29의 입력 레지스터 각각의 니블, 즉 $I^L_{29,i}\ (i=0,\cdots,7)$에 오류가 주입될 수 있다. 따라서 발생 가능한 오류 주입의 위치는 8가지이다 (그림 2 참조). 각각의 경우를 다음과 같이 표기하기로 한다. $E^L_{29,i}$. 예를 들어, $E^L_{29,7}$은 랜덤 니블 오류가 $I^L_{23,7}$에 주입된 경우를 의미한다.

3-2 오류 주입 위치 계산

본 소절에서는 암호문 차분으로부터 앞에서 정의한 4가지의 경우 중 정확한 오류 주입의 위치를 계산하는 방법을 소개한다.

먼저, 라운드 29의 입력 레지스터 중 $I^L_{29,7}$에 오류가 주입되었다고 가정한다. 그림 3은 이 경우에서의 차분 확산을 나타낸 것이다. 오류 주입 가정에 의해, I^L_{29}에서의 차분 ΔI^L_{29}은 다음과 같은 형태가 된다. 여기서 $a\neq 0$이다.

$$\Delta I^L_{29}=(a,0,0,0,0,0,0,0). \tag{1}$$

그러면, 라운드 함수 F의 출력 차분은 그림 3과 같이 $(0,0,b,0,0,0,0,0)$의 형태가 된다. 여기서 b는 S-box S_7의 입력 차분이 a일 때의 출력 차분을 의미한다. 따라서 라운드 30의 입력 차분은 다음과 같은 형태를 갖는다.

$$\Delta I_{30}=[(0,0,b,0,0,0,0,0),(a,0,0,0,0,0,0,0)]. \tag{2}$$

라운드 31의 입력 차분은 다음과 같은 형태를 갖는다. 여기서, c는 S-box S_5의 입력 차분이 b일 때의 출력 차분을 의미한다. 한편, 라운드 30에서 차분 a는 왼쪽 8-비트 순환 이동 연산에 의해 $\Delta I^R_{30,7}$에서 $\Delta I^L_{31,1}$로 이동하였음을 유의해라.

$$\Delta I_{31}=[(0,0,0,c,0,0,a,0),(0,0,b,0,0,0,0,0)]. \tag{3}$$

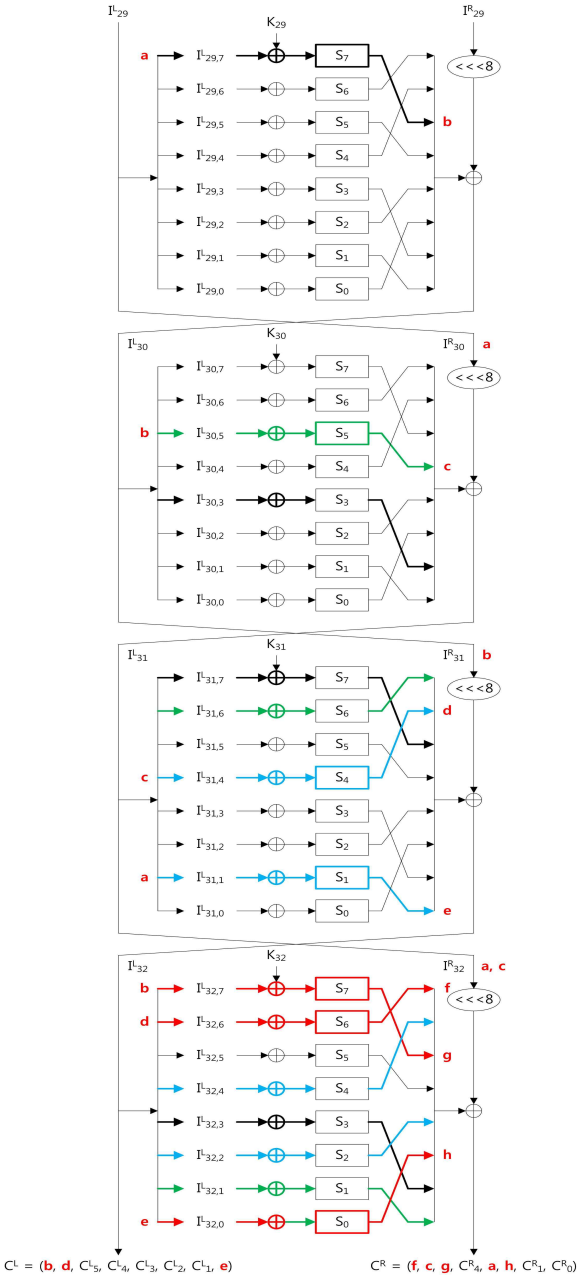

그림 3. $E_{29,7}^{L}$

Fig. 3. Event $E_{29,7}^{L}$.

유사한 방식으로, 라운드 32의 입력 차분은 다음과 같은 형태를 갖는다. 여기서, d와 e는 S-box S_4, S_1의 입력 차분이 각각 c, a일 때의 출력 차분을 의미한다 (그림 3 참조).

$$\Delta I_{32} = [(b,d,0,0,0,0,0,e),(0,0,0,c,0,0,a,0)]. \quad (4)$$

따라서 $I_{29,7}^{L}$에 랜덤 니블 오류가 주입되었을 경우, 즉 $E_{29,7}^{L}$, 암호문 차분은 다음과 같은 형태를 갖는다.

$$\Delta C = [(b,d,0,0,0,0,0,e),(f,c,g,0,a,h,0,0)]. \quad (5)$$

나머지 $E_{29,i}^{L}$ $(i=0,1,\cdots,6)$의 경우도 유사한 방식으로 계산할 수 있다. 표 2는 각각의 오류 주입 위치에 따른 암호문 차분의 형태를 나타낸 것이다. 여기서 '?'는 0이 아닌 임의의 4-비트 값을 의미한다. 표 2를 통해 알 수 있듯이, 각각의 경우마다 암호문 차분의 형태가 모두 다르다. 따라서 암호문 차분 형태를 통해 정확한 오류 주입의 위치를 계산할 수 있다.

표 2. 오류 주입 위치에 따른 암호문 차분의 형태

Table 2. Ciphertext differences for the positions of fault injections.

오류 주입 위치	암호문 차분
$E_{29,7}^{L}$	$[(?,?,0,0,0,0,0,?),(?,?,?,0,?,?,0,0)]$
$E_{29,6}^{L}$	$[(0,0,0,?,0,?,?,0),(?,?,0,0,?,?,0,?)]$
$E_{29,5}^{L}$	$[(?,?,?,0,0,0,0,0),(?,0,?,?,0,0,?,?)]$
$E_{29,4}^{L}$	$[(?,0,?,0,0,0,0,?),(0,0,?,?,0,?,?,?)]$
$E_{29,3}^{L}$	$[(0,0,0,?,?,?,0,0),(?,?,0,0,?,?,?,0)]$
$E_{29,2}^{L}$	$[(0,?,?,0,0,0,0,?),(?,?,0,?,?,?,0,0)]$
$E_{29,1}^{L}$	$[(0,0,0,0,?,?,?,0),(0,0,?,?,?,0,?,?)]$
$E_{29,0}^{L}$	$[(0,0,0,?,?,0,?,0),(0,?,?,?,0,0,?,?)]$

Ⅳ. LBlock에 대한 차분 오류 공격

본 절에서는 LBlock에 대한 차분 오류 공격을 제안한다. 공격 과정을 소개하기에 앞서, 오류가 주입된 위치를 계산한 후 각각의 경우에 대해 비밀키 정보를 얻을 수 있는 방법을 먼저 소개한다.

4-1 오류의 위치에 따른 라운드 키 복구

랜덤 니블 오류가 $I_{29,7}^{L}$에 주입되었다고 가정한다. 이 경우, 56-비트 라운드 키를 다음과 같은 과정

을 통해 얻을 수 있다.

(1) 4-비트 $K_{32,7}$을 추측하여, 라운드 32의 S-box S_7의 출력 차분이 ΔC_5^R와 동일한지 체크한다 (그림 3에서 라운드 32의 빨간색 부분 참조). 이 단계를 통과할 확률이 2^{-4}이므로, 옳은 $K_{32,7}$을 얻을 수 있다.

(2) 4-비트 $K_{32,6}$을 추측하여, 라운드 32의 S-box S_6의 출력 차분이 ΔC_7^R과 동일한지 체크한다 (그림 3에서 라운드 32의 빨간색 부분 참조). 이 단계를 통과할 확률이 2^{-4}이므로, 옳은 $K_{32,6}$을 얻을 수 있다.

(3) 4-비트 $K_{32,0}$을 추측하여, 라운드 32의 S-box S_0의 출력 차분이 ΔC_2^R와 동일한지 체크한다 (그림 3에서 라운드 32의 빨간색 부분 참조). 이 단계를 통과할 확률이 2^{-4}이므로, 옳은 $K_{32,0}$을 얻을 수 있다.

(4) 8-비트 $(K_{31,4}, K_{32,4})$를 추측하여, 라운드 31의 S-box S_4의 출력 차분이 ΔC_6^L과 동일한지 체크한다 (그림 3에서 파란색 부분 참조). 이 단계를 통과할 확률이 2^{-4}이므로, 2^4개의 후보 $(K_{31,4}, K_{32,4})$를 얻을 수 있다.

(5) 8-비트 $(K_{31,1}, K_{32,2})$를 추측하여, 라운드 31의 S-box S_1의 출력 차분이 ΔC_0^L과 동일한지 체크한다 (그림 3에서 파란색 부분 참조). 이 단계를 통과할 확률이 2^{-4}이므로, 2^4개의 후보 $(K_{31,1}, K_{32,2})$를 얻을 수 있다.

(6) 12-비트 $(K_{30,5}, K_{31,6}, K_{32,1})$를 추측하여, 라운드 30의 S-box S_5의 출력 차분이 ΔC_6^R과 동일한지 체크한다 (그림 3에서 녹색 부분 참조). 이 단계를 통과할 확률이 2^{-4}이므로, 2^8개의 후보 $(K_{30,5}, K_{31,6}, K_{32,1})$를 얻을 수 있다.

(7) 16-비트 $(K_{29,7}, K_{30,3}, K_{31,7}, K_{32,3})$를 추측하여, 라운드 29의 S-box S_7의 출력 차분이 ΔC_7^L과 동일한지 체크한다 (그림 3에서 검정색 부분 참조). 이 단계를 통과할 확률이 2^{-4}이므로, 2^{12}개의 후보 $(K_{29,7}, K_{30,3}, K_{31,7}, K_{32,3})$를 얻을 수 있다.

위의 공격 과정을 통해, $I_{29,7}^L$에 주입된 1개의 랜덤 니블 오류를 이용하여 다음과 같은 2^{28}개의 후보 56-비트 라운드 키를 얻을 수 있다.

- $K_{29,7}$.
- $(K_{30,3}, K_{30,5})$.
- $(K_{31,1}, K_{31,4}, K_{31,6}, K_{31,7})$.
- $(K_{32,0}, K_{32,1}, K_{32,2}, K_{32,3}, K_{32,4}, K_{32,6}, K_{32,7})$.

나머지 $E_{29,i}^L$ $(i = 0,1,\cdots,6)$의 경우도 유사한 방식으로 계산할 수 있다. 각각의 경우에 대해, 1개의 랜덤 니블 오류로부터 다음과 같이 2^{28}개의 후보 56-비트 라운드 키를 얻을 수 있다.

- $E_{29,6}^L$
  (1) $K_{29,6}$.
  (2) $(K_{30,1}, K_{30,7})$.
  (3) $(K_{31,0}, K_{31,2}, K_{31,3}, K_{31,5})$.
  (4) $(K_{32,0}, K_{32,1}, K_{32,2}, K_{32,4}, K_{32,5}, K_{32,6}, K_{32,7})$.
- $E_{29,5}^L$
  (1) $K_{29,5}$.
  (2) $(K_{30,4}, K_{30,6})$.
  (3) $(K_{31,1}, K_{31,4}, K_{31,6}, K_{31,7})$.
  (4) $(K_{32,1}, K_{32,2}, K_{32,3}, K_{32,4}, K_{32,5}, K_{32,6}, K_{32,7})$.
- $E_{29,4}^L$
  (1) $K_{29,4}$.
  (2) $(K_{30,4}, K_{30,6})$.
  (3) $(K_{31,1}, K_{31,4}, K_{31,6}, K_{31,7})$.
  (4) $(K_{32,0}, K_{32,1}, K_{32,2}, K_{32,3}, K_{32,4}, K_{32,5}, K_{32,7})$.
- $E_{29,3}^L$
  (1) $K_{29,3}$.
  (2) $(K_{30,1}, K_{30,7})$.
  (3) $(K_{31,0}, K_{31,2}, K_{31,3}, K_{31,5})$.
  (4) $(K_{32,0}, K_{32,2}, K_{32,3}, K_{32,4}, K_{32,5}, K_{32,6}, K_{32,7})$.

- $E_{29,2}^{L}$

(1) $K_{29,2}$.

(2) $(K_{30,3}, K_{30,5})$.

(3) $(K_{31,1}, K_{31,4}, K_{31,6}, K_{31,7})$.

(4) $(K_{32,0}, K_{32,1}, K_{32,2}, K_{32,3}, K_{32,4}, K_{32,5}, K_{32,6})$.

- $E_{29,1}^{L}$

(1) $K_{29,1}$.

(2) $(K_{30,0}, K_{30,2})$.

(3) $(K_{31,0}, K_{31,2}, K_{31,3}, K_{31,5})$.

(4) $(K_{32,0}, K_{32,1}, K_{32,2}, K_{32,3}, K_{32,5}, K_{32,6}, K_{32,7})$.

- $E_{29,0}^{L}$

(1) $K_{29,0}$.

(2) $(K_{30,0}, K_{30,2})$.

(3) $(K_{31,0}, K_{31,2}, K_{31,3}, K_{31,5})$.

(4) $(K_{32,0}, K_{32,1}, K_{32,3}, K_{32,4}, K_{32,5}, K_{32,6}, K_{32,7})$.

4-2 라운드 키로부터 비밀키 복구

앞에서 랜덤 니블 오류를 주입한 후, 주입된 위치에 따라 후보 라운드 키를 계산하는 방법을 살펴보았다. 본 소절에서는 복구한 후보 라운드 키로부터 후보 비밀키를 복구하는 방법을 소개한다.

표 1에서 라운드 키에 사용된 비밀키 정보를 나타내었다. 앞에서 복구한 $(K_{29}, K_{30}, K_{31}, K_{32})$에 사용된 비밀키 정보를 정리하면 다음과 같다.

- K_{29}: $(k_{67}, k_{66}, \cdots, k_{36})$.
- K_{30}: $(k_{38}, k_{37}, \cdots, k_{7})$.
- K_{31}: $(k_{9}, k_{8}, \cdots, k_{0}, k_{79}, k_{78}, \cdots, k_{58})$.
- K_{32}: $(k_{60}, k_{59}, \cdots, k_{29})$.

위에서 알 수 있듯이, $(K_{29}, K_{30}, K_{31}, K_{32})$에는 80-비트 비밀키 정보가 모두 포함되어 있다. 또한 LBlock의 특성에 의해, 80-비밀키 정보를 이용하여 비밀키 값을 계산할 수 있다. 따라서 이 라운드 키들로부터 후보 비밀키를 쉽게 계산할 수 있다. 하지만 후보 라운드 키의 수가 많을 경우, 계산된 후보 비밀키의 수도 많게 되므로, 매우 많은 전수조사가 필요하게 된다. 한편, 위의 식으로부터, 각각의 라운드 키마다 비밀키 정보가 중복되어 있음을 알 수 있다. 따라서 이 중복된 정보들에 대한 방정식을 구성하면, 후보 라운드 키의 수를 줄일 수 있다.

후보 라운드 키의 수를 줄이기 위해, 다음과 같은 방정식을 구성한다. 총 여과 확률은 2^{-41}이다. 예를 들어, (k_{58}, k_{59}, k_{60})에 대한 방정식을 구성하면 다음과 같다. 여기서 S_9^{-1}, S_8^{-1}는 S-box S_9, S_8의 역함수를 각각 의미한다.

$$\left\{\left(S_9^{-1}[K_{32,7}]\right) \gg 1\right\} = \left(K_{32,0} \,\&\, 0\text{x}7\right). \quad (6)$$

- (k_{58}, k_{59}, k_{60})
  - $\left\{\left(S_9^{-1}[K_{32,7}]\right) \gg 1\right\} = \left(K_{32,0} \,\&\, 0\text{x}7\right)$.
  - 여과 확률: 2^{-3}.
- (k_{37}, k_{38})
  - $\left(K_{32,2} \,\&\, 0\text{x}3\right) = \left(K_{30,7} \gg 2\right)$.
  - 여과 확률: 2^{-2}.
- (k_{35}, k_{36})
  - $\left(K_{32,1} \gg 2\right) = \left(K_{30,7} \,\&\, 0\text{x}3\right)$.
  - 여과 확률: 2^{-2}.
- (k_{33}, k_{34})
  - $\left(K_{32,1} \,\&\, 0\text{x}3\right) = \left(K_{30,6} \gg 2\right)$.
  - 여과 확률: 2^{-2}.
- (k_{31}, k_{32})
  - $\left\{\left(K_{32,0} \gg 2\right) \oplus 0\text{x}1\right\} = \left(K_{30,6} \,\&\, 0\text{x}3\right)$.
  - 여과 확률: 2^{-2}.
- (k_{29}, k_{30})
  - $\left\{\left(K_{32,0} \,\&\, 0\text{x}3\right) \oplus 0\text{x}3\right\} = \left(K_{30,5} \gg 2\right)$.
  - 여과 확률: 2^{-2}.
- (k_{7}, k_{8}, k_{9})
  - $\left\{\left(S_9^{-1}[K_{31,7}]\right) \gg 1\right\} = \left(K_{30,0} \,\&\, 0\text{x}7\right)$.
  - 여과 확률: 2^{-3}.
- (k_{60})
  - $\left[\left\{\left(S_9^{-1}[K_{32,7}]\right) \gg 3\right\} \oplus 0\text{x}1\right] = \left(K_{29,6} \,\&\, 0\text{x}1\right)$.
  - 여과 확률: 2^{-1}.
- (k_{57}, k_{58}, k_{59})

- $\left[\left\{\left(S_9^{-1}[K_{32,7}]\right)\&\,0\text{x}7\right\}\oplus 0\text{x}7\right]=\left(K_{29,5}\gg 1\right)$.
- 여과 확률: 2^{-3}.

- $\left(k_{56}\right)$
- $\left\{\left(S_8^{-1}[K_{32,6}]\right)\gg 3\right\}=\left(K_{29,5}\&\,0\text{x}1\right)$.
- 여과 확률: 2^{-1}.

- $\left(k_{53},k_{54},k_{55}\right)$
- $\left\{\left(S_8^{-1}[K_{32,6}]\right)\&\,0\text{x}7\right\}=\left(K_{29,4}\gg 3\right)$.
- 여과 확률: 2^{-3}.

- $\left(k_{52}\right)$
- $\left(K_{32,5}\gg 3\right)=\left(K_{29,4}\&\,0\text{x}1\right)$.
- 여과 확률: 2^{-1}.

- $\left(k_{49},k_{50},k_{51}\right)$
- $\left(K_{32,5}\&\,0\text{x}7\right)=\left(K_{29,3}\gg 1\right)$.
- 여과 확률: 2^{-3}.

- $\left(k_{48}\right)$
- $\left(K_{32,4}\gg 3\right)=\left(K_{29,3}\&\,0\text{x}1\right)$.
- 여과 확률: 2^{-1}.

- $\left(k_{45},k_{46},k_{47}\right)$
- $\left(K_{32,4}\&\,0\text{x}7\right)=\left(K_{29,2}\gg 1\right)$.
- 여과 확률: 2^{-3}.

- $\left(k_{44}\right)$
- $\left(K_{32,3}\gg 3\right)=\left(K_{29,2}\&\,0\text{x}1\right)$.
- 여과 확률: 2^{-1}.

- $\left(k_{41},k_{42},k_{43}\right)$
- $\left(K_{32,3}\&\,0\text{x}7\right)=\left(K_{29,1}\gg 1\right)$.
- 여과 확률: 2^{-3}.

- $\left(k_{40}\right)$
- $\left(K_{32,2}\gg 3\right)=\left(K_{29,1}\&\,0\text{x}1\right)$.
- 여과 확률: 2^{-1}.

- $\left(k_{39}\right)$
- $\left\{\left(K_{32,2}\gg 2\right)\&\,0\text{x}1\right\}=\left(K_{29,0}\gg 3\right)$.
- 여과 확률: 2^{-1}.

- $\left(k_{36},k_{37},k_{38}\right)$
- $\left\{\left(S_9^{-1}[K_{30,7}]\right)\gg 1\right\}=\left(K_{29,0}\&\,0\text{x}7\right)$.
- 여과 확률: 2^{-3}.

4-3 LBlock에 대한 DFA

본 논문에서 제안하는 LBlock에 대한 차분 오류 공격은 크게 세 단계로 구성된다. 먼저, 암호문의 차분 형태를 통해 오류가 주입된 위치를 계산한다. 그리고 계산된 위치에 따라 후보 라운드 키를 얻는다. 마지막으로 계산된 후보 라운드 키로부터 후보 비밀키를 얻는다.

LBlock에 대한 DFA 공격 과정은 다음과 같다.

(1) [오류가 발생하지 않은 데이터 수집] 오류가 발생하지 않은 알고리즘을 이용하여 평문 P에 대한 암호문 $C=(C^L,C^R)$을 얻는다.
(2) [오류가 발생한 데이터 수집] 라운드 29의 입력 레지스터 $I_{29}^L=(I_{29,7}^L,I_{29,6}^L,\cdots,I_{29,0}^L)$에 랜덤 니블 오류 Δ^i를 n번 주입한 후, 해당 오류에 대한 암호문 C^i를 얻는다 $(i=1,\cdots,n)$.
(3) [오류 위치 계산] (C,C^i)로부터 ΔC^i를 계산한 후, 표 2를 이용하여 정확한 오류 주입의 위치를 각각 계산한다.
(4) [후보 $(K_{29},K_{30},K_{31},K_{32})$ 계산] 단계 (3)에서 계산한 오류의 위치에 따라, 4-1절에서 소개한 방법을 이용하여 후보 $(K_{29},K_{30},K_{31},K_{32})$를 모든 Δ^i에 대해 계산한다.
(5) [후보 $(K_{29},K_{30},K_{31},K_{32})$ 계산] 단계 (4)를 통과한 후보 라운드 키 $(K_{29},K_{30},K_{31},K_{32})$에 대해, 4-2절에서 소개한 방정식에 대입하여 통과하지 않는 후보 라운드 키를 걸러낸다.
(6) [LBlock의 80-비트 비밀키 복구] 단계 (5)를 통과한 각각의 후보 $(K_{29},K_{30},K_{31},K_{32})$에 대해, 후보 비밀키를 계산한다. 계산한 후보 비밀키에 대해 한 개의 평문/암호문 쌍에 대해 전수조사를 수행한다. 이 단계를 통과한 후보 비밀키를 LBlock의 80-비트 비밀키로 출력한다.

위의 공격 과정을 일반적인 PC에서 $10{,}000$번 구현한 결과, 평균 5개의 랜덤 니블 오류를 주입하면 약 2^{25}개의 후보 비밀키를 얻을 수 있다. 따라서 이

2^{25}개의 후보 비밀키에 대해 전수조사를 수행한다. 이 단계를 통과할 확률은 2^{-64}이므로, 틀린 후보 비밀키가 이 단계를 통과할 확률은 2^{-39} $(=2^{25}\cdot 2^{-64})$이다. 이는 본 논문에서 제안하는 공격이 항상 옳은 80-비트 비밀키를 복구할 수 있음을 의미한다. 일반적인 PC에서 실제 구현한 결과, 수 초 내에 항상 옳은 80-비트 비밀키를 복구할 수 있었다.

Ⅴ. 결 론

본 논문에서는 LBlock에 대한 차분 오류 공격 결과를 제안하였다. 본 논문에서 제안한 공격을 이용하여, 랜덤 니블 오류 가정 하에서, 라운드 29의 입력 레지스터에 랜덤 니블 오류를 평균 5번 주입하여 비밀키를 복구할 수 있음을 보였다. 일반적인 PC에서 이 공격을 구현한 결과, 2^{25}의 전수조사를 이용하여 수 초 내에 LBlock의 비밀키를 복구할 수 있었다.

기제안된 LBlock에 대한 DFA 결과는 최소 7개의 랜덤 비트 오류 주입을 필요로 하기 때문에, 본 공격이 기제안된 공격 보다 더 효율적이다.

감사의 글

이 연구는 서울과학기술대학교 교내 학술연구비 지원으로 수행되었습니다.

참 고 문 헌

[1] W. Wu and L. Zhang, “LBlock: A Lightweight Block Cipher”, *ACNS 2011, LNCS 6715*, pp. 327-344, Springer-Verlag, 2011.

[2] Y. Liu, D. Gu, Z. Liu and W. Li, “Impossible Differential Attacks on Reduced-Round LBlock”, *ISPEC 2012, LNCS 7232*, pp. 97-108, Springer-Verlag, 2012.

[3] F. Karakoc, J. Demirci and A. Harmanci, “Impossible Differential Cryptanalysis on Reduced-Round LBlock”, *WISTP 2012, LNCS 7322*, pp. 179-188, Springer-Verlag, 2012.

[4] E. Biham and A. Shamir, “Differential Fault Analysis of Secret Key Cryptosystems”, *Crypto 1997, LNCS 1294*, pp. 513-525, Springer-Verlag, 1997.

[5] K. Jeong, Y. Lee, J. Sung and S. Hong, “Differential fault analysis on block cipher SEED”, *Mathematical and Computer Modelling*, Vol. 55, pp. 26-34, Elsevier, 2012.

[6] 박세현, 정기태, 이유섭, 성재철, 홍석희, “블록 암호 ARIA-128에 대한 차분 오류 공격”, *정보보호학회논문지*, 제 21권, 제 5호, pp. 15-25, 2011.

[7] 박세현, 정기태, 이유섭, 성재철, 홍석희, “PRESENT-80/128에 대한 향상된 차분 오류 공격”, *정보보호학회논문지*, 제 22권, 제 1호, pp. 33-41, 2012.

[8] 정기태, “무선 센터 네트워크 환경에 적합한 블록 암호 LED-64에 대한 안전성 분석”, *한국항행학회 논문지*, 제 16권, 제 1호, pp. 70-75, 2012.

[9] 정기태, “블록 암호 Piccolo-80에 대한 차분 오류 공격”, *한국항행학회논문지*, 제 16권, 제 3호, pp. 510-517, 2012.

[10] L. Zhao, T. Nishide and K. Sakurai, “Differential Fault Analysis of Full LBlock”, *COSADE 2012, LNCS 7275*, pp. 135-150, Springer-Verlag, 2012.

정 기 태 (鄭基台)

2004년 2월 : 고려대학교 수학과 이학사
2006년 2월 : 고려대학교 정보보호대학원 공학석사
2011년 8월 : 고려대학교 정보보호대학원 공학박사
2011년 9월~현재 : 고려대학교 정보보호 연구원 연구교수
관심분야 : 대칭키 암호에 대한 분석 및 설계

이 창 훈 (李昌勳)

2001년 2월 : 한양대학교 수학과 (이학사)
2003년 2월 : 고려대학교 정보보호 대학원(공학석사)
2008년 2월 : 고려대학교 정보보호 대학원(공학박사)
2009년 3월 ~ 2012년 2월 : 한신대학교 컴퓨터공학부 조교수
2012년 3월 ~ 현재 : 서울과학기술대학교 컴퓨터공학과 조교수
관심분야 : 정보보호, 암호학, 디지털포렌식, 컴퓨터이론