http://dx.doi.org/10.3745/KTCCS.2013.2.7.311

Security Analysis on Block Cipher XSB

Changhoon Lee†

ABSTRACT

256-bit block cipher XSB(eXtended Spn Block cipher) was proposed in 2012 and has a symmetric strucrure in encryption and decryption process. In this paper, we propose a differential fault analysis on XSB. Based on a random byte fault model, our attack can recover the secret key of XSB by using only two random byte fault injection. This result is the first known cryptanalytic result on the target algorithm.

Keywords : Block Cipher, XSB Cipher, Differential Fault Anlalysis

블록 암호 XSB에 대한 안전성 분석

이 창 훈†

요 약

2012년에 제안된 256-비트 블록 암호 XSB(eXtended Spn Block cipher)는 암호화 과정과 복호화 과정이 동일하게 설계된 블록 암호 알고리즘이다. 본 논문에서는 XSB에 대한 차분 오류 공격을 제안한다. 랜덤 바이트 오류 주입 가정에 기반을 둔 이 공격은 2개의 랜덤 바이트 오류를 이용하여, XSB의 256-비트 비밀키를 복구한다. 본 논문의 공격 결과는 XSB에 대한 첫 번째 분석 결과이다.

키워드 : 블록 암호, XSB 암호, 차분 오류 분석

1. 서 론

차분 오류 공격 (differential fault analysis, DFA)는 대표적인 부채널 공격 중의 하나로서, 1997년 Biham과 Shamir에 의해 처음 소개되었다 [1]. 이후 DFA는 SEED [2], ARIA-128 [3], PRESENT [4], LED-64 [5], Piccolo-80 [6], LBlock [7] 등 대부분의 블록 암호에 적용되었다.

본 논문에서는 2012년에 제안된 256-비트 블록 암호 XSB [8]에 대한 DFA를 제안한다. XSB(eXtended Spn Block cipher)는 256-비트 비밀키를 사용하는 256-비트 블록 암호로서, 암호화 과정과 복호화 과정이 동일하다는 특징을 갖고 있다. 본 논문에서 제안하는 공격은 라운드 11의 입력 레지스터에 랜덤 바이트 오류를 주입한다는 가정을 이용한다. 본 논문에서 제안하는 공격은 2개의 랜덤 바이트 오류를 주입하여 $O(2^{45})$의 계산 복잡도로 XSB의 256-비트 비밀키를 복구할 수 있다. 이 공격 결과는 XSB에 대한 첫 번째 분석 결과이다.

본 논문은 다음과 같이 구성되어 있다. 먼저, 2절에서는 블록 암호 XSB의 구조를 소개한다. 3절에서는 XSB에 대한 DFA를 소개한다. 마지막으로 4절에서 결론을 맺는다.

2. XSB

256-비트 블록 암호 XSB는 그림 1과 같이 256-비트 비밀키를 사용하고 14-라운드 구조를 갖는다. 암호화 과정과 복호화 과정을 동일하게 하기 위해, 라운드 1 ~ 6에 Pre_stage 함수를 이용하여, 이 함수의 역함수인 Post_stage 함수를 라운드 8 ~ 14에 사용한다. 그리고 가운데 대칭 함수인 Symm_Funct 함수를 사용한다.

256-비트 내부 상태값은 AES에서의 표기법과 유사하게 다음과 같이 바이트 단위 4×8 행렬로 표기한다. 또한, 라운드 r의 입·출력값은 $I^r=(I^r_{0,0},\cdots,I^r_{3,7})$과 $O^r=(O^r_{0,0},\cdots,O^r_{3,7})$로 표기한다 $(r=1,\cdots,14)$.

라운드 8 ~ 14에 대해 동작하는 Post_stage 함수에서는 다음과 같이 서브 함수 SubBytePost (SBP), ExgRow (ER),

※ 본 연구는 서울과학기술대학교 교내 학술연구 지원비로 수행되었음.
† 종신회원 : 서울과학기술대학교 컴퓨터공학과 조교수
논문접수 : 2013년 2월 26일
수 정 일 : 1차 2013년 4월 26일
심사완료 : 2013년 4월 26일
* Corresponding Author : Changhoon Lee(chlee@seoultech.ac.kr)

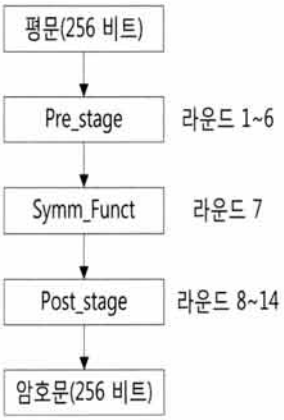

Fig. 1. The structure of XSB.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$	$S_{0,4}$	$S_{0,5}$	$S_{0,6}$	$S_{0,7}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	$S_{1,5}$	$S_{1,6}$	$S_{1,7}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$	$S_{2,5}$	$S_{2,6}$	$S_{2,7}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,4}$	$S_{3,5}$	$S_{3,6}$	$S_{3,7}$

MixColumn (MC), AddRoundKey (ARK) 함수가 반복 적용된다. Pre_stage와 Symm_Funct 함수는 본 논문에서 제안하는 공격에서는 사용되지 않으므로, 생략한다. 이 함수에 대한 자세한 설명은 [8]을 참조하기 바란다.

- Post_stage
  - 라운드 8~13에 대해 다음을 반복 수행한다.
    * SubBytePost
    * ExgRow
    * MixColumn
    * AddRoundKey
  - SubBytePost
  - ExgRow
  - AddRoundKey

Post_stage 함수에서 사용되는 각각의 서브 함수는 다음과 같다.

- ARK: 라운드 키 덧셈 함수
- SBP: 바이트 단위 S-box를 이용한 비선형 치환 함수 (자세한 설명은 [8] 참조)
- ER: 내부 상태값을 다음과 같이 치환
  - 첫 번째 행: 교환하지 않음
  - 두 번째 행: (0, 1), (2, 3), (4, 5), (6, 7) 열을 서로 교환
  - 세 번째 행: (0, 7), (1, 2), (3, 4), (5, 6) 열을 서로 교환
  - 네 번째 행: (0, 4), (1, 5), (2, 6), (3, 7) 열을 서로 교환
- MC: 다음의 선형 변환을 사용하는 바이트 단위 선형 변환 함수

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

XSB의 키스케줄은 AES-256의 키스케줄과 유사하다. 하지만 XSB는 AES-256의 키스케줄의 256-비트 내부 상태값 전부를 256-비트 라운드 키로 사용한다.

3. XSB에 대한 차분 오류 공격

본 절에서는 라운드 11의 입력 레지스터에 오류 주입을 가정한 후, 256-비트 비밀키를 복구할 수 있는 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 11의 입력 레지스터에 랜덤 바이트 오류를 주입할 수 있다.
- 공격자는 오류의 위치와 오류 주입을 통해 발생하는 차분값을 알 수 없다.

위의 오류 주입 가정에 따라, 라운드 11의 입력 레지스터 각각의 바이트, 즉 $I^{11}_{i,j}$에 오류가 주입될 수 있다 $(i=0,\cdots,3,\ j=0,\cdots,7)$. 따라서 발생 가능한 오류 주입의 위치는 32가지이다. 각각의 경우를 다음과 같이 표기하기로 한다: $E^{11}_{i,j}$. 예를 들어, $E^{11}_{0,0}$은 랜덤 바이트 오류가 $I^{11}_{0,0}$에 주입된 경우를 의미한다.

3.2 차분 확산 특징

먼저, $E^{11}_{0,0}$의 경우를 고려한다. 즉, 랜덤 바이트 오류가 $I^{11}_{0,0}$에 주입되었다고 가정한다. 이 경우의 차분 확산 형태는 (Fig. 2)와 같다. 오류 주입 가정에 의해, I^{11}에서의 차분 ΔI^{11}은 다음과 같은 형태가 된다. 여기서 $x \neq 0$이다.

$$\Delta I^{11} = (x, 0, \cdots, 0).$$

라운드 11의 입력 차분 ΔI^{11}은 비선형 함수 SBP를 거쳐

$(x^*,0,\cdots,0)$의 형태가 된다. 여기서 x^*는 S-box의 입력 차분 x에 대한 출력 차분값을 의미한다. 그리고 MC에 의해 MC의 출력 차분은 다음과 같은 형태를 갖는다.

$1bx^*$	0	0	0	0	0	0	0
$1cx^*$	0	0	0	0	0	0	0
$14x^*$	0	0	0	0	0	0	0
$12x^*$	0	0	0	0	0	0	0

이 차분은 라운드 12에서의 SBP에 의해 다음과 같은 형태를 갖는다. 여기서 X_0, X_1, X_2, X_3은 모두 0이 아닌 값을 의미한다.

X_0	0	0	0	0	0	0	0
X_1	0	0	0	0	0	0	0
X_2	0	0	0	0	0	0	0
X_3	0	0	0	0	0	0	0

유사한 과정을 거쳐, 라운드 13에서의 SBP의 출력 차분은 다음과 같이 열 (0, 1, 4, 7)에만 0이 아닌 차분을 갖는다.

Y_0	Y_4	0	0	Y_8	0	0	Y_{12}
Y_1	Y_5	0	0	Y_9	0	0	Y_{13}
Y_2	Y_6	0	0	Y_{10}	0	0	Y_{14}
Y_3	Y_7	0	0	Y_{11}	0	0	Y_{15}

이 차분을 이용하여 라운드 13의 출력 차분은 다음과 같이 열 (0)에는 4개의 0이 아닌 차분값으로 계산된 형태, 열 (2,6)에는 1개의 0이 아닌 차분값으로 계산된 형태, 나머지 열 (1, 3, 4, 5, 7)에는 2개의 0이 아닌 차분값으로 계산된 형태를 갖는다.

4	2	1	2	2	2	1	2
4	2	1	2	2	2	1	2
4	2	1	2	2	2	1	2
4	2	1	2	2	2	1	2

본 논문에서 제안하는 공격은 위의 차분 형태를 이용한다. 예를 들어, 위의 차분 형태에서 열 (2, 6)에 해당하는 라운드 14의 라운드 키를 각각 추측하면, 암호문으로부터 열 (2, 6)의 정확한 값을 계산할 수 있다. 그러면 가능한 차분 형태를 체크함으로써 후보 라운드 키의 수를 줄일 수 있다. 유사한 방식으로 각각의 모든 열에 대해 후보 라운드 키의 수를 줄일 수 있다. 각각의 열에 대해, 32-비트 라운드 키를 추측하면 열의 형태에 따라 통과하는 라운드 키의 수는 다음과 같다.

- 열 (0): $2^{32} \rightarrow 2^{32}$ (확률: 1)
- 열 (2, 6): $2^{32} \rightarrow 2^{8}$ (확률: 2^{-24})
- 열 (1, 3, 4, 5, 7): $2^{32} \rightarrow 2^{16}$ (확률: 2^{-16})

(1)

Fig. 3은 $E^{11}_{0,1}$에서의 차분 확산을 나타낸 것이다. $E^{11}_{0,0}$ Fig. 2와 $E^{11}_{0,1}$(Fig. 3)을 비교해 보면, 다음과 같은 유사한 특징을 가지고 있음을 알 수 있다.

- [특징 1] 라운드 13에서의 SBP의 출력 차분은 4개의 열에만 0이 아닌 차분을 갖는다.
- [특징 2] 라운드 13의 출력 차분 형태는 4개의 0이 아닌 차분값으로 계산되는 열 1개, 1개의 0이 아닌 차분값으로 계산되는 열 2개, 2개의 0이 아닌 차분값으로 계산되는 열 5개를 갖는다.

위의 두 가지 특징에 따라, 32개의 가능한 오류 주입 위치를 분류하면 다음과 같이 8개의 집합으로 분류할 수 있다.

- Set 0: $E^{11}_{0,0}$, $E^{11}_{1,1}$, $E^{11}_{2,7}$, $E^{11}_{3,4}$
- Set 1: $E^{11}_{0,1}$, $E^{11}_{1,0}$, $E^{11}_{2,2}$, $E^{11}_{3,5}$
- Set 2: $E^{11}_{0,2}$, $E^{11}_{1,3}$, $E^{11}_{2,1}$, $E^{11}_{3,6}$
- Set 3: $E^{11}_{0,3}$, $E^{11}_{1,2}$, $E^{11}_{2,4}$, $E^{11}_{3,7}$
- Set 4: $E^{11}_{0,4}$, $E^{11}_{1,5}$, $E^{11}_{2,3}$, $E^{11}_{3,0}$
- Set 5: $E^{11}_{0,5}$, $E^{11}_{1,4}$, $E^{11}_{2,6}$, $E^{11}_{3,1}$
- Set 6: $E^{11}_{0,6}$, $E^{11}_{1,7}$, $E^{11}_{2,5}$, $E^{11}_{3,2}$
- Set 7: $E^{11}_{0,7}$, $E^{11}_{1,6}$, $E^{11}_{2,0}$, $E^{11}_{3,3}$

각각의 집합에 속한 오류 주입 위치는 동일한 위치에 [특징 1]과 [특징 2]가 발생함을 쉽게 확인할 수 있다.

3.3 라운드 14의 라운드 키 복구

대부분의 차분 오류 공격에서는 오류가 주입된 암호문의 형태를 파악하여, 오류가 주입된 위치를 계산한다. 하지만, XSB에 대한 공격에서는 암호문의 형태를 통해 오류가 주입된 위치를 파악할 수 없다. 따라서 본 논문에서 제안하는 공격에서는 가능한 오류 주입 위치를 모두 고려해야 한다.

본 논문에서 제안하는 XSB에 대한 차분 오류 공격은 크게 두 단계로 구성된다. 먼저, 라운드 14의 256-비트 라운드 키를 추측하여, 모든 가능한 오류 주입 위치에 대해 라운드 14의 입력 차분 형태를 체크한다. 이 단계를 통과하는 라운드 키는 라운드 14의 후보 라운드 키가 된다. 그리고 마지막 라운드 키로부터 XSB의 256-비트 후보 비밀키를 얻는다.

예를 들어, $E^{11}_{0,0}$의 경우를 고려한다. 즉, 랜덤 바이트 오류가 $I^{11}_{0,0}$에 주입되었다고 가정한다. 그러면 차분 확산 형태는 그림 1과 같다. 그러면 256-비트 마지막 라운드 키를 추

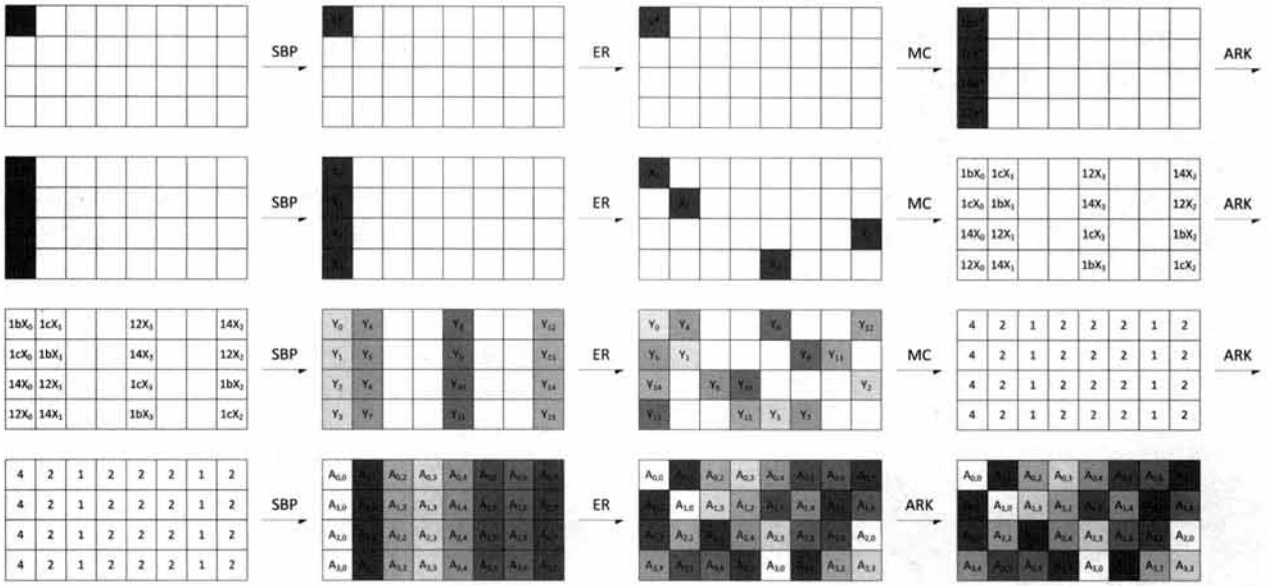

Fig. 2. Differential propagation in the case of $E_{0,0}^{11}$.

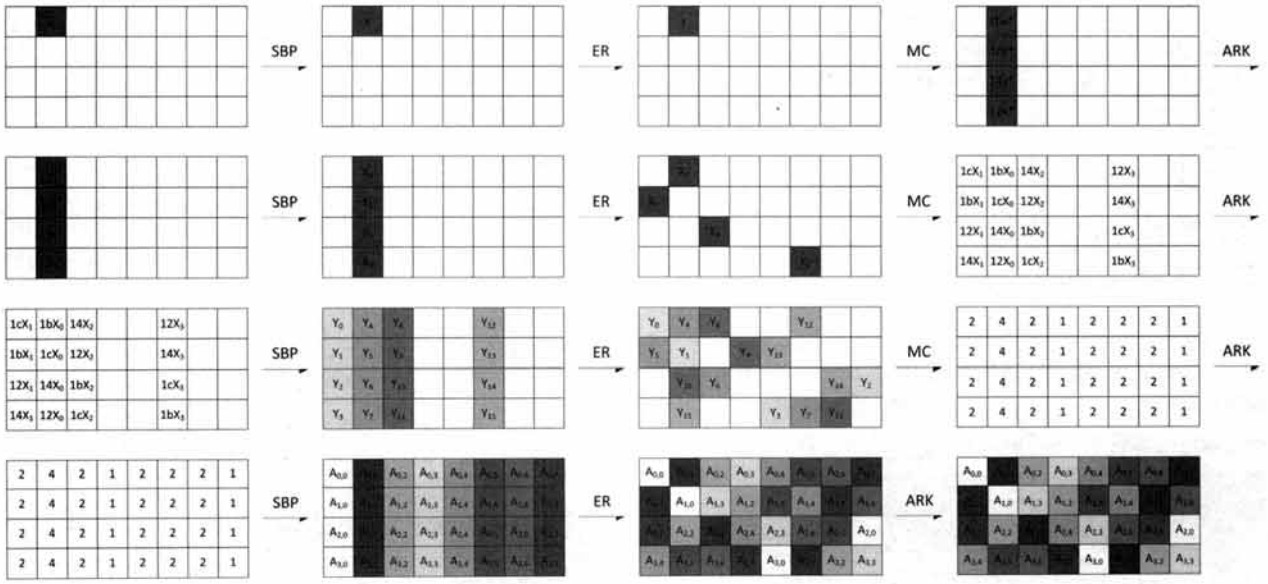

Fig. 3. Differential propagation in the case of $E_{0,1}^{11}$.

측하여 암호문으로부터 라운드 14의 입력값을 계산한 후, 가능한 입력 차분 형태를 갖는지 확인하다. 하지만 이 경우의 계산 복잡도는 2^{256}이므로, 너무 비효율적이다. 따라서 3-2절에서 소개한 바와 같이, 32-비트 부분 라운드 키를 추측하여 다음과 같이 식 (1)에 적용한다. 예를 들어, (Fig. 2)에서 암호문 $(A_{0,1},A_{1,1},A_{2,1},A_{3,1})$ (흰색)에 대응하는 32-비트 부분 라운드 키를 추측하면, 라운드 14의 입력값 중 열 (1)에 해당하는 32-비트 값을 계산할 수 있다. 식 (1)에 따르면 열 (1)은 2개의 0이 아닌 차분값으로 계산된 열이므로, 이 테스트를 통과할 확률은 2^{-16}이다. 따라서 이 테스트를 통과하는 32-비트 후보 라운드 키의 수의 기댓값은 2^{16}이다.

- 라운드 키 $(A_{0,0},A_{1,0},A_{2,0},A_{3,0})$ 추측
  - 체크 열: (0)
  - 통과 확률: 1
  - 통과 라운드 키의 수: 2^{32}
- 라운드 키 $(A_{0,1},A_{1,1},A_{2,1},A_{3,1})$ 추측
  - 체크 열: (1)
  - 통과 확률: 2^{-16}
  - 통과 라운드 키의 수: 2^{16}
- 라운드 키 $(A_{0,2},A_{1,2},A_{2,2},A_{3,2})$ 추측
  - 체크 열: (2)
  - 통과 확률: 2^{-24}

- 통과 라운드 키의 수: 2^8

• 라운드 키 $(A_{0,3}, A_{1,3}, A_{2,3}, A_{3,3})$ 추측
- 체크 열: (3)
- 통과 확률: 2^{-16}
- 통과 라운드 키의 수: 2^{16}

• 라운드 키 $(A_{0,4}, A_{1,4}, A_{2,4}, A_{3,4})$ 추측
- 체크 열: (4)
- 통과 확률: 2^{-16}
- 통과 라운드 키의 수: 2^{16}

• 라운드 키 $(A_{0,5}, A_{1,5}, A_{2,5}, A_{3,5})$ 추측
- 체크 열: (5)
- 통과 확률: 2^{-16}
- 통과 라운드 키의 수: 2^{16}

• 라운드 키 $(A_{0,6}, A_{1,6}, A_{2,6}, A_{3,6})$ 추측
- 체크 열: (6)
- 통과 확률: 2^{-24}
- 통과 라운드 키의 수: 2^8

• 라운드 키 $(A_{0,7}, A_{1,7}, A_{2,7}, A_{3,7})$ 추측
- 체크 열: (7)
- 통과 확률: 2^{-16}
- 통과 라운드 키의 수: 2^{16}

유사한 방식으로 모든 $E_{i,j}^{11}$에 적용할 수 있다. 각각의 오류 주입 위치에 따라, 모든 테스트를 통과할 확률은 2^{-128}이다. 그러나 정확한 오류 주입 위치를 알 수 없기 때문에 모든 가능한 오류 주입 위치에 대해 모두 적용해야 하므로, 이 테스트를 통과하는 256-비트 후보 라운드 키의 수의 기댓값은 $2^{133}(=2^{256} \cdot 2^{-128} \cdot 32)$이다.

3.4 XSB에 대한 차분 오류 공격

XSB에 대한 DFA 공격 과정은 다음과 같다.

(1) [오류가 발생하지 않은 데이터 수집] 오류가 발생하지 않은 알고리즘을 이용하여 평문 P에 대한 암호문 $C=(C_{0,0}, C_{3,7})$를 얻는다.

(2) [오류가 발생한 데이터 수집] 라운드 11의 입력 레지스터 $I^{11}=(I_{0,0}^{11}, \cdots, I_{3,7}^{11})$에 랜덤 바이트 오류 Δ^i를 2번 주입한 후, 해당 오류에 대한 암호문 C^i를 얻는다 $(i=1,2)$.

(3) [라운드 14의 256-비트 라운드 키 계산] (C, C^i)로부터 3.3절에서 소개한 방법을 이용하여, 라운드 14의 후보 256-비트 라운드 키를 계산한다.

(4) [XSB의 256-비트 비밀키 복구] 단계 (3)를 통과한 라운드 14의 후보 256-비트 라운드 키를 이용하여 XSB의 키스케줄로부터 후보 비밀키를 계산한다. 계산한 후보 비밀키에 대해 한 개의 평문/암호문 쌍에 대해 전수조사를 수행한다. 이 단계를 통과한 후보 비밀키를 XSB의 256-비트 비밀키로 출력한다.

본 논문에서 제안하는 공격은 2개의 랜덤 바이트 오류를 사용하였다. 1개의 랜덤 바이트 오류에 대해 후보 라운드 키가 테스트를 통과할 확률이 $2^{-123}(=2^{-128} \cdot 32)$이므로, 단계 (3)을 통과하는 후보 라운드 키의 수의 기댓값은 이론적으로 $2^{10}(=2^{256} \cdot 2^{-123 \cdot 2})$이다. 하지만 실제 구현 결과, 단계 (3)을 통과하는 후보 라운드 키의 수는 최대 2^{32}인 것으로 나타났다. 그 이유는 2개의 랜덤 바이트 오류가 같은 위치 또는 3.2절에서 소개한 동일한 Set에 속할 경우, 확률 1로 모든 후보 키가 통과하는 열(4개의 0이 아닌 차분값으로 계산된 열)에 대해서는 모든 32-비트 후보 라운드 키가 테스트를 통과하기 때문이다. 다른 경우에서는 대부분이 옳은 라운드 키만이 테스트를 통과하였다.

이 공격의 계산 복잡도는 단계 (3)에 가장 많은 영향을 받는다. 단계 (3)은 8개의 32-비트 부분 라운드 키를 2개의 랜덤 바이트 오류에 대한 모든 가능한 오류 주입 위치에 대해 체크를 하므로, 계산 복잡도는 $2^{45}(=2^{32} \cdot 8 \cdot 32 \cdot 8)$이다. 따라서 본 공격의 전체 계산 복잡도는 $O(2^{45})$이다.

4. 결 론

본 논문에서는 XSB에 대한 차분 오류 공격 결과를 제안하였다. 본 논문에서 제안한 공격을 이용하여, 랜덤 바이트 오류 가정 하에서, 라운드 11의 입력 레지스터에 랜덤 바이트 오류를 2번 주입하여 비밀키를 복구할 수 있음을 보였다. 본 공격 결과는 XSB에 대한 첫 번째 분석 결과이다.

참 고 문 헌

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

[2] 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.

[3] S. Park, K. Jeong, Y. Lee, J. Sung and S. Hong, "Differential Fault Analysis on Block Cipher ARIA-128", *Journal of The Korean Institute of Information Security & Cryptology* (in Korean), Vol.21, No. pp.15-25, 2011.

[4] S. Park, K. Jeong, Y. Lee, J. Sung and S. Hong, "Improved Differential Fault Analysis on Block Cipher PRESENT-80/128", Journal of The Korean Institute of Information Security & Cryptology (in Korean), Vol.22, No.1, pp.33-41, 2012.

[5] K. Jeong, "Security Analysis of Block Cipher LED-64 Suitable for Wireless Sensor Network Environments", *The Journal of*

Korea Navigation Institute (in Korean), Vol. 16, No. 1, pp. 70-75, 2012.

[6] K. Jeong, "Differential Fault Analysis on Block Cipher Piccolo-80", *The Journal of Korea Navigation Institute* (in Korean), Vol.16, No.3, pp.510-517, 2012.

[7] K. Jeong and C. Lee, "Differential Fault Analysis on Lightweight Block Cipher LBlock", *The Journal of Korea Navigation Institute* (in Korean), Vol.16, No.5, pp.871-878, 2012.

[8] G. Cho, "256 bit Symmetric SPN Block cipher XSB", *Journal of the Korea Industrial Information System Society* (in Korean), Vol.17, No.3, pp.9-17, 2012.

이 창 훈

e-mail : chlee@seoultech.ac.kr

2001년 2월 한양대학교 수학과(이학사)

2003년 2월 고려대학교 정보보호대학원 (공학석사)

2008년 2월 고려대학교 정보보호대학원 (공학박사)

2009년 3월~2011년 2월 한신대학교 컴퓨터공학부 전임강사

2011년 3월~2012년 2월 한신대학교 컴퓨터공학부 조교수

2012년 3월~현 재 서울과학기술대학교 컴퓨터공학과 조교수

관심분야 : 정보보호, 암호학, 디지털포렌식, 컴퓨터이론