한국항행학회 논문지 제16권 제3호 2012년 6월

블록 암호 Piccolo-80에 대한 차분 오류 공격

Differential Fault Analysis on Block Cipher Piccolo-80

정기태*

Ki-Tae Jeong*

요 약

64-비트 블록 암호 Piccolo-80은 무선 센서 네트워크 환경과 같이 제한된 환경에 적합하도록 설계된 경량 블록 암호이다. 본 논문에서는 Piccolo-80에 대한 차분 오류 공격을 제안한다. 랜덤 바이트 오류 주입 가정에 기반을 둔 이 공격은 평균 6개의 랜덤 바이트 오류와 2^{24}의 전수조사를 이용하여, Piccolo-80의 비밀키를 복구한다. 이는 일반적인 PC에서 수 초 내에 가능함을 의미한다. 본 논문의 공격 결과는 Piccolo-80에 대한 첫 번째 부채널 분석 결과이다.

Abstract

Piccolo-80 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 Piccolo-80. Based on a random byte fault model, our attack can the secret key of Piccolo-80 by using the exhaustive search of 2^{24} and six random byte fault injections on average. It can be simulated on a general PC within a few seconds. This result is the first known side-channel attack result on Piccolo-80.

Key words : Block cipher, Piccolo, Differential fault analysis

I. 서 론

CHES 2011에 제안된 경량 블록 암호 Piccolo는 80/128-비트 비밀키를 사용하는 64-비트 블록 암호로서, 무선 센서 네트워크 환경과 같이 제한된 환경에 적합하도록 설계되었다 [1]. 이 알고리즘은 일반화된 Feistel 구조를 가지며, 비밀키의 길이에 따라, 각각 Piccolo-80, Piccolo-128로 표기된다. Piccolo-80의 전체 라운드 수는 25이고, Piccolo-128의 전체 라운드 수는 31이다. 기제안된 Piccolo에 대한 분석 결과는 ISPEC 2012에서 제안된 Biclique 공격이 유일하다 [2]. 하지만 지금까지 부채널 공격에 대한 분석 결과는 제안되지 않았다.

차분 오류 공격 (differential fault analysis, DFA)는 대표적인 부채널 공격 중의 하나로서, 1997년 Biham과 Shamir에 의해 처음 소개되었다 [3]. 이후 DFA는 SEED [4, 5], ARIA-128 [6], PRESENT [7], LED-64 [8] 등 대부분의 블록 암호에 적용되었다.

본 논문에서는 Piccolo-80에 대한 차분 오류 공격을 제안한다. 본 논문에서 제안하는 공격은 라운드 23의 입력 레지스터에 랜덤 바이트 오류를 주입한다는 가정을 이용한다. 따라서 주입 가능한 오류의 위

* 고려대학교 정보보호연구원(Center for Information Security Technologies(CIST), Korea University)
· 제1저자 (First Author) : 정기태
· 투고일자 : 2012년 5월 15일
· 심사(수정)일자 : 2012년 5월 17일 (수정일자 : 2012년 6월 13일)
· 게재일자 : 2012년 6월 30일

치는 8가지가 존재한다. 각각의 위치는 암호문의 차분을 확인함으로서 계산할 수 있다. 그래서 발생한 오류의 위치에 따라, 비밀키의 부분 정보를 얻을 수 있다. 일반적인 PC에서 이 공격을 구현한 결과, 평균 6개의 오류 주입과 2^{24}의 전수조사를 이용하여, 수초 내에 Piccolo-80의 비밀키를 복구할 수 있었다. 이 공격 결과는 Piccolo-80에 대한 첫 번째 부채널 분석 결과이다.

본 논문은 다음과 같이 구성되어 있다. 먼저, 2절에서는 블록 암호 Piccolo-80의 구조를 소개한다. 3절에서는 오류 주입 가정과 오류가 주입된 위치를 계산하는 방법을 소개한 후, 4절에서 Piccolo-80에 대한 DFA를 소개한다. 마지막으로 5절에서 결론을 맺는다.

Ⅱ. Piccolo-80

64-비트 블록 암호 Piccolo-80은 그림 1과 같이 80-비트 비밀키를 사용하고 25-라운드 일반화된 Feistel 구조를 갖는다. 본 논문에서는 다음과 같은 표기법을 사용한다.

$P=(P_0,P_1,P_2,P_3)$: 64-비트 평문.
$C=(C_0,C_1,C_2,C_3)$: 64-비트 암호문.
$I_i=(I_{i,0},I_{i,1},I_{i,2},I_{i,3})$: 라운드 i의 64-비트 입력값 $(i=0,1,\cdots,24)$.
(rk_{2i},rk_{2i+1}): 라운드 i의 라운드 키.
(wk_0,wk_1,wk_2,wk_3): 화이트닝 키.

64-비트 평문 $P=(P_0,P_1,P_2,P_3)$는 먼저 다음과 같은 화이트닝 키 단계를 거쳐 라운드 0의 입력값 $I_0=(I_{0,0},I_{0,1},I_{0,2},I_{0,3})$이 된다.

$$I_{0,0}=P_0\oplus wk_0,\ I_{0,1}=P_1,$$
$$I_{0,2}=P_2\oplus wk_1,\ I_{0,3}=P_3.$$

$I_0=(I_{0,0},I_{0,1},I_{0,2},I_{0,3})$는 라운드 함수 F와 라운드 치환 RP를 24번 반복 적용하여 라운드 24의 입력값 $I_{24}=(I_{24,0},I_{24,1},I_{24,2},I_{24,3})$가 된다. 이 값으로부터 암호문 $C=(C_0,C_1,C_2,C_3)$는 다음과 같이 계산된다.

$$C_0=I_{24,0}\oplus wk_2,\ C_1=F(I_{24,0})\oplus I_{24,1}\oplus rk_{48},$$
$$C_2=I_{24,2}\oplus wk_3,\ C_3=F(I_{24,2})\oplus I_{24,3}\oplus rk_{49}.$$

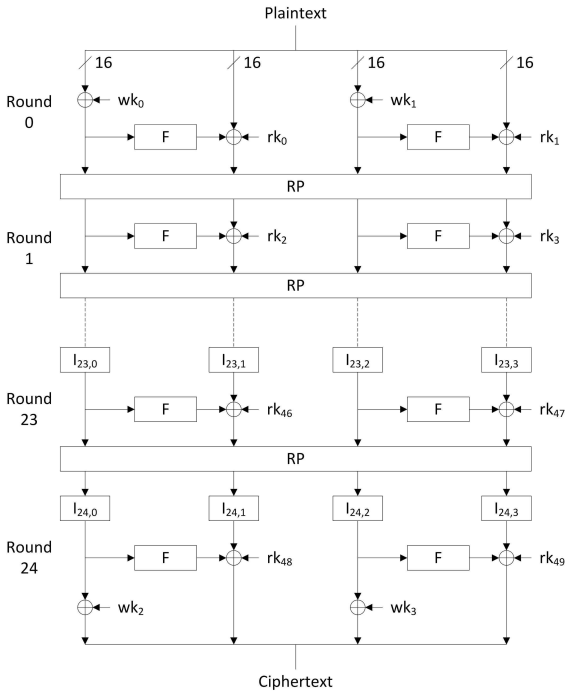

그림 1. Piccolo-80의 전체 구조
Fig. 1. The structure of Piccolo-80.

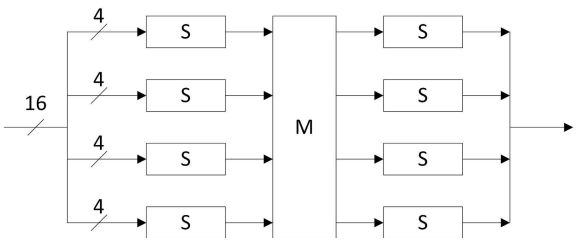

그림 2. 라운드 함수 F
Fig. 2. Round function F of Piccolo-80.

라운드 함수 F는 그림 2와 같이 16-비트 입출력을 갖는 함수로 SMS 구조이다. 본 논문에서 제안하는 공격은 4×4 S-box S와 4×4 행렬 M의 구체적인 성질을 이용하지 않으므로 생략하기로 한다. 이

함수에 대한 자세한 소개는 [1]을 참조하라. 라운드 치환 RP는 그림 3과 같이 64-비트 입력값 $X=(x_0,x_1,x_2,x_3)$을 입력받아 64-비트 출력값 $Y=(y_0,y_1,y_2,y_3)$을 생성하는 바이트 단위 치환이다. 여기서 $x_i=(x_i^L,x_i^R)$이다.

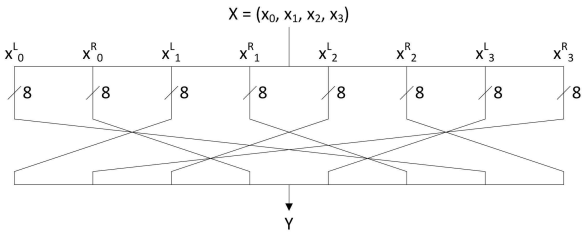

그림 3. 라운드 치환 RP
Fig. 3. Round permutation RP of Piccolo-80.

Piccolo-80의 키스케줄은 매우 단순하다. 먼저 80-비트 비밀키 K를 다음과 같이 5개의 k_j로 나눈다 $(j=0,1,2,3,4)$. 여기서 $k_j=(k_j^L,k_j^R)$이다.

$$K=(k_0,k_1,k_2,k_3,k_4).$$

4개의 화이트닝 키 (wk_0,wk_1,wk_2,wk_3)와 25개의 라운드 키 (rk_{2i},rk_{2i+1})는 다음과 같이 생성된다 $(i=0,1,\cdots,24)$. 여기서 $(con_{2i}^{80},con_{2i+1}^{80})$은 각각 16-비트 라운드 상수이다. 이 상수들을 계산하는 방법은 [1]을 참조하라.

화이트닝 키

$$wk_0=k_0^L\parallel k_1^R,\ wk_1=k_1^L\parallel k_0^R,$$
$$wk_2=k_4^L\parallel k_3^R,\ wk_3=k_3^L\parallel k_4^R.$$

라운드 키

$$(rk_{2i},rk_{2i+1})=(con_{2i}^{80},con_{2i+1}^{80})\oplus\begin{cases}(k_2,k_3), & i(\mathrm{mod}\,5)\equiv 0\ \text{or}\ 2,\\ (k_0,k_1), & i(\mathrm{mod}\,5)\equiv 1\ \text{or}\ 4,\\ (k_4,k_4), & i(\mathrm{mod}\,5)\equiv 3.\end{cases}$$

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

라운드 i	비밀키 정보
0	(k_2,k_3)
1	(k_0,k_1)
⋮	⋮
23	(k_4,k_4)
24	(k_0,k_1)

표 1은 각각의 라운드 키에 사용된 비밀키 정보를 나타낸 것이다. 예를 들어, 라운드 24의 라운드 키 (rk_{48},rk_{49})에 비밀키 정보 (k_0,k_1)이 각각 적용되었다.

Ⅲ. 오류 주입 가정 및 오류 주입 위치 계산

본 절에서는 우선 라운드 23의 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 23의 입력 레지스터에 랜덤 바이트 오류를 주입할 수 있다.
- 오류의 위치와 오류 주입을 통해 발생하는 차분값을 알 수 없다.

위의 오류 주입 가정에 따라, 라운드 23의 입력 레지스터 각각의 바이트, 즉 $I_{23,i}^j$ $(i=0,1,2,3,\ j=L,R)$에 오류가 주입될 수 있다. 따라서 발생 가능한 오류 주입의 위치는 8가지이다. 각각의 경우를 다음과 같이 표기하기로 한다: $E_{23,i}^j$. 예를 들어,

$E^L_{23,0}$ 은 랜덤 바이트 오류가 $I^L_{23,0}$ 에 주입된 경우를 의미한다.

3-2 오류 주입 위치 계산

본 소절에서는 암호문 차분으로부터 앞에서 정의한 8가지의 경우 중 정확한 오류 주입의 위치를 계산하는 방법을 소개한다.

3-2-1 $E_{23,0}$

라운드 23의 입력 레지스터 중 $I_{23,0}$에 오류가 주입되었다고 가정한다 (즉, $E^L_{23,0}$ 과 $E^R_{23,0}$). 그림 4는 $E^L_{23,0}$ 과 $E^R_{23,0}$ 에서의 차분 확산을 나타낸 것이다. 그림에서 파란색 선은 $E^L_{23,0}$ 을 의미하고, 빨간색 선은 $E^R_{23,0}$ 을 의미한다.

가정에 의해, $I_{23,0}$에서의 차분 $\Delta I_{23,0}$은 $(\alpha,0)$ 또는 $(0,\alpha)$ 형태가 된다 $(\alpha \neq 0)$. 그러면, 라운드 함수 F의 특성 상, 입력 차분이 $(\alpha,0)$ 또는 $(0,\alpha)$의 형태일 때 출력 차분은 (β,γ)의 형태가 된다 $(\beta \neq 0, \gamma \neq 0)$. 따라서 $E^L_{23,0}$ 과 $E^R_{23,0}$ 의 경우, 서로 구별할 수 없으며 암호문의 차분은 다음과 같은 형태를 가진다. 여기서 α_1와 α_2는 0이 아닌 임의의 8-비트 차분값을 의미하며, '?'는 임의의 16-비트 차분값을 의미한다.

○ $E_{23,0}$: $\Delta C = (\alpha_1 \| 0,\ ?,\ 0 \| \alpha_2, ?)$.

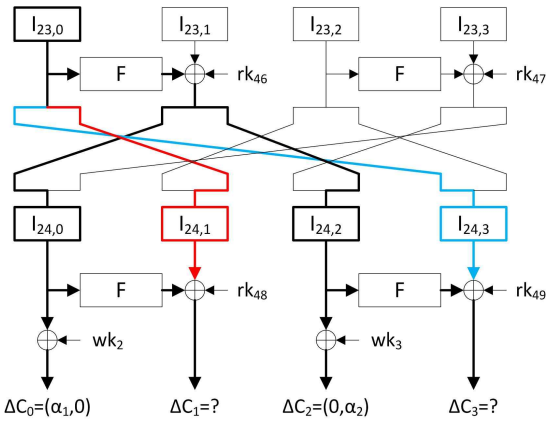

그림 4. $E_{23,0}$
Fig. 4. $E_{23,0}$.

3-2-2 $E^L_{23,1}$ 과 $E^R_{23,1}$

라운드 23의 입력 레지스터 중 $I_{23,1}$에 오류가 주입되었다고 가정한다 (즉, $E^L_{23,1}$ 과 $E^R_{23,1}$). 이 경우의 차분 확산은 그림 5와 같다. 그림에서 파란색 선은 $E^L_{23,1}$ 을 의미하고, 빨간색 선은 $E^R_{23,1}$ 을 의미한다.

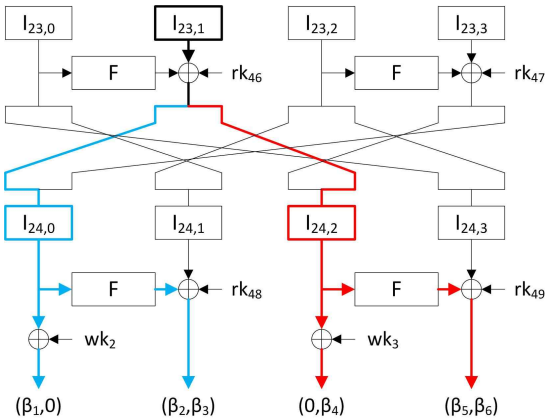

그림 5. $E_{23,1}$
Fig. 5. $E_{23,1}$.

각각의 경우에 대해, 암호문의 차분은 다음과 같은 형태를 가진다. 여기서 β_j는 0이 아닌 임의의 8-비트 차분값을 의미한다 $(j = 1,\cdots,6)$.

○ $E^L_{23,1}$: $\Delta C = (\beta_1 \| 0,\ \beta_2 \| \beta_3,\ 0,0)$.
○ $E^R_{23,1}$: $\Delta C = (0,\ 0,\ 0 \| \beta_4,\ \beta_5 \| \beta_6)$.

3-2-3 $E_{23,2}$

$I_{23,2}$에 오류가 주입되었다고 가정한다 (즉, $E^L_{23,2}$ 과 $E^R_{23,2}$). 이 경우의 차분 확산은 그림 6과 같다. 그림에서 파란색 선은 $E^L_{23,2}$ 을 의미하고, 빨간색 선은 $E^R_{23,2}$ 을 의미한다.

$E_{23,0}$와 유사하게, $E^L_{23,2}$ 과 $E^R_{23,2}$ 도 서로 구별할 수 없으며 다음과 같은 암호문 차분의 형태를 가진다. 여기서 γ_1와 γ_2는 0이 아닌 임의의 8-비트 차분값을 의미하며, '?'는 임의의 16-비트 차분값을 의미한다.

○ $E_{23,2}$: $\Delta C = (0 \| \gamma_1,\ ?,\ \gamma_2 \| 0, ?)$.

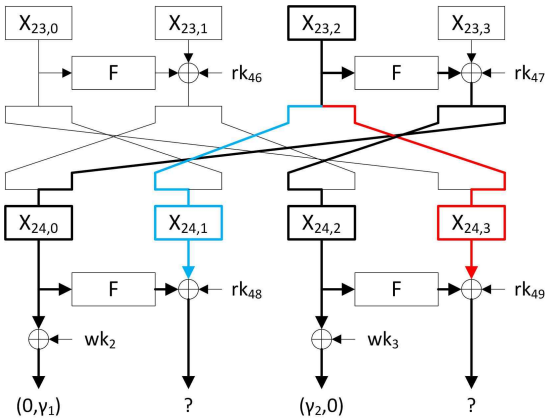

그림 6. $E_{23,2}$
Fig. 6. $E_{23,2}$.

3-2-4 $E^L_{23,3}$ 과 $E^R_{23,3}$

$E^L_{23,3}$ 과 $E^R_{23,3}$ 경우의 차분 확산은 그림 7과 같다. 그림에서 파란색 선은 $E^L_{23,3}$ 을 의미하고, 빨간색 선은 $E^R_{23,3}$ 을 의미한다.

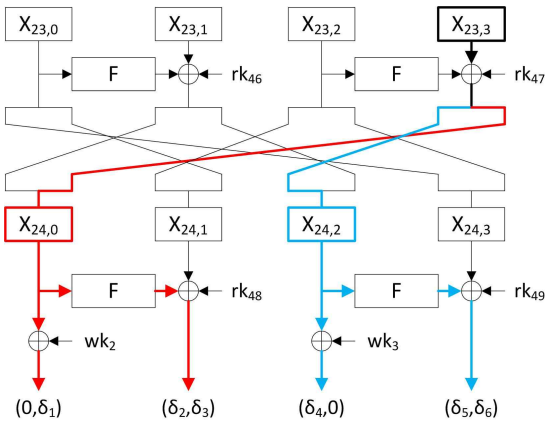

그림 7. $E_{23,3}$
Fig. 7. $E_{23,3}$.

각각의 경우에 대해, 암호문의 차분은 다음과 같은 형태를 가진다. 여기서 δ_j는 0이 아닌 임의의 8-비트 차분값을 의미한다 $(j = 1, \cdots, 6)$.

○ $E^L_{23,3}$: $\Delta C = (0,\ 0,\ \delta_4 \| 0,\ \delta_5 \| \delta_6)$.

○ $E^R_{23,3}$: $\Delta C = (0 \| \beta_1,\ \beta_2 \| \beta_3,\ 0,\ 0)$.

위에서 소개한 암호문의 차분 형태를 통해, 각각의 경우를 쉽게 구별할 수 있음을 알 수 있다.

Ⅳ. Piccolo-80에 대한 차분 오류 공격

본 절에서는 Piccolo-80에 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 비밀키 복구

앞에서 암호문의 차분 형태를 통해 오류가 주입된 위치를 계산하였다. 각각의 경우에 대한 암호문의 차분 형태를 정리하면 표 2와 같다.

표 2. 오류 주입 위치에 따른 암호문 차분의 형태
Table 2. Ciphertext differences for the positions of fault injections.

오류 주입 위치	암호문 차분
$E_{23,0}$	$(\alpha_1 \| 0,\ ?,\ 0 \| \alpha_2, ?)$
$E^L_{23,1}$	$(\beta_1 \| 0,\ \beta_2 \| \beta_3,\ 0, 0)$
$E^R_{23,1}$	$(0,\ 0,\ 0 \| \beta_4,\ \beta_5 \| \beta_6)$
$E_{23,2}$	$(0 \| \gamma_1,\ ?,\ \gamma_2 \| 0, ?)$
$E^L_{23,3}$	$(0,\ 0,\ \delta_4 \| 0,\ \delta_5 \| \delta_6)$
$E^R_{23,3}$	$(0 \| \beta_1,\ \beta_2 \| \beta_3,\ 0,\ 0)$

4-1-1 $E_{23,0}$

랜덤 바이트 오류가 $I_{23,0}$에 주입되었다고 가정한다. 이 경우, 2^{16}개의 후보 48-비트 (k^R_0, k^L_1, k_3, k_4)를 다음과 같은 과정을 통해 얻을 수 있다.

(1) 16-비트 $wk_2 (= k^L_4 \| k^R_3)$을 추측하여, 라운드 24에서의 왼쪽 F 함수의 상위 8-비트 출력 차분이 ΔC_1의 상위 8-비트 값과 동일한지 체크한다 (그림 4 참조). 이 단계를 통과할 확률이 2^{-8}이므

로, 2^8개의 후보 (k_3^R, k_4^L)를 얻을 수 있다.

(2) 16-비트 $wk_3 (= k_3^L \| k_4^R)$을 추측하여, 라운드 24에서의 오른쪽 F 함수의 하위 8-비트 출력 차분이 ΔC_3의 하위 8-비트 값과 동일한지 체크한다 (그림 4 참조). 이 단계를 통과할 확률이 2^{-8}이므로, 2^8개의 후보 (k_3^L, k_4^R)를 얻을 수 있다.

(3) $rk_{48} (= con_{48}^{80} \oplus k_0)$의 하위 8-비트 값과 rk_{49} $(= con_{49}^{80} \oplus k_1)$의 상위 8-비트 값을 추측한 후, 단계 (1)과 (2)를 통과한 2^{16}개의 후보 (k_3, k_4)를 이용하여 라운드 23에서의 왼쪽 F 함수를 만족하는지 체크한다 (그림 4 참조). 이 단계를 통과할 확률이 2^{-16}이므로, 2^{16}개의 후보 (k_0^R, k_1^L, k_3, k_4)를 얻을 수 있다.

위의 공격 과정을 통해, $I_{23,0}$에 주입된 1개의 랜덤 바이트 오류를 이용하여 2^{16}개의 후보 48-비트 (k_0^R, k_1^L, k_3, k_4)를 계산할 수 있다.

4-1-2 $E_{23,1}$

랜덤 바이트 오류가 $I_{23,1}$에 주입되었다고 가정한다. 이 경우, 다음과 같은 과정을 통해 옳은 16-비트 (k_3^R, k_4^L) $(E_{23,1}^L)$ 또는 (k_3^L, k_4^R) $(E_{23,1}^R)$을 각각 얻을 수 있다.

○ $E_{23,1}^L$

- 16-비트 $wk_2 (= k_4^L \| k_3^R)$을 추측하여, 라운드 24에서의 왼쪽 F 함수의 출력 차분이 ΔC_1과 동일한지 체크한다 (그림 5 참조). 이 단계를 통과할 확률이 2^{-16}이므로, 옳은 (k_3^R, k_4^L)를 얻을 수 있다.

○ $E_{23,1}^R$

- 16-비트 $wk_3 (= k_3^L \| k_4^R)$을 추측하여, 라운드 24에서의 오른쪽 F 함수의 출력 차분이 ΔC_3과 동일한지 체크한다 (그림 5 참조). 이 단계를 통과할 확률이 2^{-16}이므로, 옳은 (k_3^L, k_4^R)를 얻을 수 있다.

4-1-3 $E_{23,2}$

랜덤 바이트 오류가 $I_{23,2}$에 주입되었을 경우, 2^{16}개의 후보 48-비트 (k_0^L, k_1^R, k_3, k_4)를 다음과 같은 과정을 통해 얻을 수 있다.

(1) 16-비트 $wk_2 (= k_4^L \| k_3^R)$을 추측하여, 라운드 24에서의 왼쪽 F 함수의 하위 8-비트 출력 차분이 ΔC_1의 하위 8-비트 값과 동일한지 체크한다 (그림 6 참조). 이 단계를 통과할 확률이 2^{-8}이므로, 2^8개의 후보 (k_3^R, k_4^L)를 얻을 수 있다.

(2) 16-비트 $wk_3 (= k_3^L \| k_4^R)$을 추측하여, 라운드 24에서의 오른쪽 F 함수의 상위 8-비트 출력 차분이 ΔC_3의 상위 8-비트 값과 동일한지 체크한다 (그림 6 참조). 이 단계를 통과할 확률이 2^{-8}이므로, 2^8개의 후보 (k_3^L, k_4^R)를 얻을 수 있다.

(3) $rk_{48} (= con_{48}^{80} \oplus k_0)$의 상위 8-비트 값과 rk_{49} $(= con_{49}^{80} \oplus k_1)$의 하위 8-비트 값을 추측한 후, 단계 (1)과 (2)를 통과한 2^{16}개의 후보 (k_3, k_4)를 이용하여 라운드 23에서의 오른쪽 F 함수를 만족하는지 체크한다 (그림 6 참조). 이 단계를 통과할 확률이 2^{-16}이므로, 2^{16}개의 후보 (k_0^L, k_1^R, k_3, k_4)를 얻을 수 있다.

4-1-4 $E_{23,3}$

마지막으로 랜덤 바이트 오류가 $I_{23,3}$에 주입되었다고 가정한다. 이 경우, 다음과 같은 과정을 통해 옳은 16-비트 (k_3^L, k_4^R) $(E_{23,3}^L)$ 또는 (k_3^R, k_4^L) $(E_{23,3}^R)$을 각각 얻을 수 있다.

○ $E_{23,3}^L$

- 16-비트 $wk_3 (= k_3^L \| k_4^R)$을 추측하여, 라운드 24에서의 오른쪽 F 함수의 출력 차분이 ΔC_3과 동일한지 체크한다 (그림 7 참조). 이 단계를 통과할 확률이 2^{-16}이므로, 옳은 (k_3^L, k_4^R)를 얻을 수 있다.

○ $E_{23,3}^{R}$

- 16-비트 $wk_2(=k_4^L \| k_3^R)$을 추측하여, 라운드 24에서의 왼쪽 F 함수의 출력 차분이 ΔC_1과 동일한지 체크한다 (그림 7 참조). 이 단계를 통과할 확률이 2^{-16}이므로, 옳은 (k_3^R, k_4^L)를 얻을 수 있다.

4-2 Piccolo-80에 대한 DFA

본 논문에서 제안하는 Piccolo-80에 대한 차분 오류 공격은 크게 두 단계로 구성된다. 먼저, 암호문의 차분 형태를 통해 오류가 주입된 위치를 계산한다. 그리고 계산된 위치에 따라 비밀키의 부분 정보를 복구한다.

Piccolo-80에 대한 DFA 공격 과정은 다음과 같다.

(1) [오류가 발생하지 않은 데이터 수집] 오류가 발생하지 않은 알고리즘을 이용하여 평문 P에 대한 암호문 $C=(C_0, C_1, C_2, C_3)$을 얻는다.
(2) [오류가 발생한 데이터 수집] 라운드 23의 입력 레지스터 $I_{23}=(I_{23,0}, I_{23,1}, I_{23,2}, I_{23,3})$에 랜덤 바이트 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_0, k_1, k_3, k_4) 계산] 단계 (3)에서 계산한 오류의 위치에 따라, 4-1절에서 소개한 방법을 이용하여 후보 (k_0, k_1, k_3, k_4)를 모든 Δ^i에 대해 계산한다.
(5) [Piccolo-80의 80-비트 비밀키 복구] 16-비트 k_2를 추측한 후, 단계 (4)를 통과한 후보 (k_0, k_1, k_3, k_4)를 이용하여 한 개의 평문/암호문 쌍에 대해 전수조사를 수행한다. 이 단계를 통과한 $(k_0, k_1, k_2, k_3, k_4)$를 Piccolo-80의 80-비트 비밀키로 출력한다.

위의 공격 과정을 일반적인 PC에서 구현한 결과, 평균 6개의 랜덤 바이트 오류를 주입하면 약 2^8개의 후보 (k_0, k_1, k_3, k_4)를 얻을 수 있다. 따라서 총 2^{24} $(=2^8 \cdot 2^{16})$개의 후보 $(k_0, k_1, k_2, k_3, k_4)$에 대해 전수조사를 수행한다. 이 단계를 통과할 확률은 2^{-64}이므로, 틀린 비밀키가 이 단계를 통과할 확률은 $2^{-40}(=2^{24} \cdot 2^{-64})$이다. 이는 본 논문에서 제안하는 공격이 항상 옳은 80-비트 비밀키를 복구할 수 있음을 의미한다. 일반적인 PC에서 구현한 결과, 수 초 내에 항상 옳은 80-비트 비밀키를 복구할 수 있었다.

Ⅴ. 결 론

본 논문에서는 Piccolo-80에 대한 첫 번째 부채널 분석 결과를 제안하였다. 본 논문에서 제안한 공격을 이용하여, 랜덤 바이트 오류 가정 하에서, 라운드 23의 입력 레지스터에 랜덤 바이트 오류를 평균 6번 주입하여 비밀키를 복구할 수 있음을 보였다. 일반적인 PC에서 이 공격을 구현한 결과, 2^{24}의 전수 조사를 이용하여 수 초 내에 Piccolo-80의 비밀키를 복구할 수 있었다.

Piccolo-80이 제한된 환경에 적합한 경량 블록 암호 알고리즘임을 고려할 때, 본 논문의 결과를 통해 Piccolo-80의 안전성에 문제가 있음을 알 수 있다. 본 공격을 확장하여 Piccolo-128에 대한 DFA 분석은 향후 연구 과제이다.

감사의 글

본 논문은 지식경제부 IT R&D 사업의 일환으로 수행하였음 (유비쿼터스 환경에서의 정보보호 서비스를 위한 프라이버시 강화 암호 기술 개발)

참 고 문 헌

[1] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita and T. Shirai, “Piccolo: An Ultra-

Lightweight Blockcipher", *CHES 2011, LNCS 6917,* pp. 342-357, Springer-Verlag, 2011.

[2] Y. Wang, W. Wu and X. Yu, "Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher", *ISPEC 2012, LNCS 7232,* pp. 337-352, Springer-Verlag, 2012.

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

[4] 정기태, 성재철, 홍석희, "블록 암호 SEED에 대한 차분 오류 공격", *정보보호학회논문지,* 제 20권, 제 4호, pp. 17-24, 2010.

[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 센서 네트워크 환경에 적합한 블록 암호 LED-64에 대한 안전성 분석", *한국항행학회논문지,* 제 16권, 제 1호, pp. 70-75, 2012.

정 기 태 (鄭基台)

2004년 2월 : 고려대학교 수학과 이학사
2006년 2월 : 고려대학교 정보보호 대학원 공학석사
2011년 8월 : 고려대학교 정보보호 대학원 공학박사
2011년 9월~현재 : 고려대학교 정보보호연구원 박사후연구원
관심분야 : 대칭키 암호의 분석 및 설계