情報保護學會論文誌
第 22 卷 第 3 號, 2012. 6

블록 암호 HIGHT에 대한 차분 오류 공격*

이 유 섭,[1†] 김 종 성,[2‡] 홍 석 희[1]
[1]고려대학교, [2]경남대학교

A Differential Fault Attack against Block Cipher HIGHT*

Yuseop Lee,[1†] Jongsung Kim,[2‡] Seokhee Hong[1]
[1]Korea University, [2]Gyungnam University

요 약

HIGHT는 국내에서 개발된 초경량 블록 암호로서, 정보통신단체(TTA) 표준과 국제표준화기구(ISO/IEC) 18033-3 표준으로 제정되었다. 본 논문에서는 블록암호 HIGHT에 대한 차분 오류 주입 공격을 제안한다. 제안하는 공격에서 공격자는 암호화 과정에서 라운드 28의 입력값에 임의의 1-바이트 오류를 주입할 수 있다고 가정한다. 이러한 가정에서 오류 주입을 통해 얻어진 암호문과 정상적으로 얻어진 암호문의 차분 특성을 이용하여 비밀키를 복구한다. 12개의 오류를 주입할 경우에는 88%의 성공 확률, 7개의 오류를 주입하는 경우에는 51%의 성공 확률로 수초 내에 HIGHT의 비밀키를 복구한다.

ABSTRACT

The block cipher HIGHT is designed suitable for low-resource hardware implementation. It established as the TTA standard and ISO/IEC 18033-3 standard. In this paper, we propose a differentail fault attack against the block cipher HIGHT. In the proposed attack, we assume that an attacker is possible to inject a random byte fault in the input value of the 28-th round. This attack can recover the secret key by using the differential property between the original ciphertext and fault cipher text pairs. Using 7 and 12 error, our attack recover secret key within a few second with success probability 87% and 51%, respectively.

Keywords: HIGHT, Differential fault attack, Side-channel

I. 서 론

IT 기술의 발달로 모바일, 센서, RFID와 같이 제한된 자원을 가지는 다양한 컴퓨팅 환경이 대두되고 있다. 이러한 환경에서는 AES[1], SEED[2], ARIA[3]등과 같은 기존의 블록 암호 알고리즘을 사용하는데 어려움이 발생하였다. 이를 해결하기 위해, 2000년대 중반부터 초경량 블록 암호 개발에 대한 연구가 진행되어, HIGHT[4], PRESENT[5], KATAN/KTANTAN[6] 등의 초경량 블록 암호 알고리즘이 제안되었다.

국내에서 개발된 HIGHT는 2006년 정보통신단체(TTA) 표준으로 제정되었으며, 2010년에 국제표준화기구(ISO/IEC) 18033-3의 표준으로 제정되었다. HIGHT는 128-비트 비밀키를 사용하는 64-비트 블록 암호로서 32 라운드 8-branch type Ⅱ generalized Feistel 구조로 설계되었다. 라운드 함수는 2^8 위에서의 법덧셈, XOR 연산과 비트 단위 순환 이

접수일(2011년 10월 17일), 수정일(2012년 1월 2일), 게재확정일(2012년 1월 16일).
* 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No. 2012-0003556).
† 주저자, yusubi@korea.ac.kr
‡ 교신저자, jongsungk@kyungnam.ac.kr

동 연산만을 이용하여 스마트폰, 스마트카드, RFID, 유비쿼터스센서네트워크(USN) 등 제한된 환경에 적합하다.

부채널 분석 방법은 차분 공격이나 선형 공격 같은 기존의 암호 알고리즘 분석 방법이 알고리즘 자체의 취약점을 이용하는 것과 달리 실제 구현된 암호 알고리즘의 동작 중에 발생하는 정보를 이용한다. 대표적으로 암호 알고리즘이 동작하는 동안 발생하는 전력, 시간을 분석하는 수동적인 공격 방법과 암호 알고리즘이 동작하는 동안 물리적으로 오류를 주입하는 능동적인 공격으로 구분된다. 또한, 오류 주입 공격과 차분 공격을 결합한 차분 오류 공격은 Biham 등이 DES에 최초로 적용하였으며[7], AES[8,9,10,11], Triple-DES[12], CLEFIA[13], SEED[14], ARIA[15], SMS4 and MacGiffin[16]등 다양한 블록 암호에 대한 분석이 이루어졌다.

경량 블록 암호인 HIGHT의 특성상 하드웨어로 사용될 것을 감안한다면, 부채널 분석 공격에 대한 고려가 필요하다. 하지만, 현재까지 HIGHT에 대한 차분 오류 주입 공격은 제안되지 않았다. 그 이유로는 차분 오류 공격에서 공격자는 선택한 평문에 대하여 특정 라운드의 입력 레지스터에 임의의 오류를 주입하여 암호문을 얻을 수 있다고 가정하고, 정상적인 암호문과 오류가 주입된 암호문 사이의 차분의 확산 특성을 분석하여 비밀키를 복구한다. 하지만, HIGHT는 8 비트 기반 덧셈, XOR, 로테이션 연산을 사용하여 S-박스와 치환 함수를 사용하는 블록 암호에 비해 많은 수의 오류가 필요하기 때문에 차분 오류 공격을 적용하기에 어려움이 있었다. 본 논문에서는 HIGHT에 대한 차분 오류 공격을 최초로 제안한다. 본 논문에서 제안하는 차분 오류 공격은 7~12개의 오류 주입을 이용하여 수 초내에 HIGHT의 비밀키를 복구한다.

본 논문은 다음과 같이 구성된다. 2절에서 블록 암호 HIGHT와 차분 오류 공격에 대해서 설명한다. 3절에서 공격에 사용하는 오류 주입 가정과 오류에 의한 차분 특성을 설명하고, 4절에서 HIGHT에 대한 차분 오류 주입 공격을 소개한다. 5절에서 구현 결과에 따른 공격 복잡도를 보인 후 결론을 맺는다.

II. 블록 암호 HIGHT와 차분 오류 공격

본 장에서는 블록 암호 HIGHT와 차분 오류 공격에 대해 소개한다.

2.1 블록 암호 HIGHT

블록암호 HIGHT는 RFID/USN등과 같이 제한된 하드웨어 환경에 최적화된 초경량/저전력 블록암호이다[4]. 2006년 정보통신단체표준(TTA)으로 제정되었으며, 2010년에 국제표준화기구(ISO/IEC) 18033-3의 표준으로 제정되었다. HIGHT는 128-비트 비밀키를 지원하는 64-비트 블록 암호로서 32-라운드 8-branch type Ⅱ generalized Feistel 구조를 가진다. 라운드 함수는 2^8상의 법덧셈($\boxplus$)과 법뺄셈($\boxminus$), XOR 연산($\oplus$)과 비트 단위 좌측 순환 이동($\ll$)으로 구성된다. 64-비트 평문 P, 암호문 C를 다음과 같이 표시한다.

$$P = P_7 \| P_6 \| P_5 \| P_4 \| P_3 \| P_2 \| P_1 \| P_0,$$
$$C = C_7 \| C_6 \| C_5 \| C_4 \| C_3 \| C_2 \| C_1 \| C_0.,$$

각 라운드의 출력값 $X_i (1 \le X_i \le 32)$를 다음과 표시한다.

$$X_i = X_{i,7} \| X_{i,6} \| X_{i,5} \| X_{i,4} \| X_{i,3} \| X_{i,2} \| X_{i,1} \| X_{i,0}.$$

$WK_{0 \le i \le 7}$은 화이트닝키, $SK_{0 \le i \le 127}$은 라운드키를 나타내고, $F_0(x)$와 $F_1(x)$는 다음과 같다.

$$F_0(x) = (x \ll 1) \oplus (x \ll 2) \oplus (x \ll 7),$$
$$F_1(x) = (x \ll 3) \oplus (x \ll 4) \oplus (x \ll 6).$$

HIGHT의 암호화 과정은 다음과 같다.

1. 초기 변환

- $X_{0,i} = P_i$ for $i = 1,3,5,7$,
  $X_{0,0} = P_0 \boxplus WK_0$,
  $X_{0,2} = P_2 \oplus WK_1$,
  $X_{0,4} = P_4 \boxplus WK_2$,
  $X_{0,6} = P_6 \oplus WK_3$.

2. For i=1 to 31 (라운드 1~31)

- $X_{i,j} = X_{i-1,j-1}$ for $j = 1,3,5,7$,
  $X_{i,0} = X_{i-1,7} \oplus (F_0(X_{i-1,6} \boxplus SK_{4i-1}))$,
  $X_{i,2} = X_{i-1,1} \boxplus (F_0(X_{i-1,0} \oplus SK_{4i-4}))$,
  $X_{i,4} = X_{i-1,3} \oplus (F_0(X_{i-1,2} \boxplus SK_{4i-3}))$,
  $X_{i,6} = X_{i-1,5} \boxplus (F_0(X_{i-1,4} \oplus SK_{4i-2}))$.

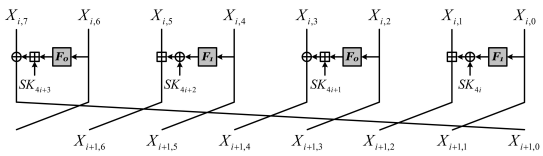
$i=32$ (라운드 32)

- $X_{32,j}=X_{31,j}$ for $j=0,2,4,6$,
  $X_{32,1}=X_{31,1}\boxplus(F_1(X_{31,0}\oplus SK_{124})$,
  $X_{32,3}=X_{31,3}\oplus(F_0(X_{31,2}\boxplus SK_{125})$,
  $X_{32,5}=X_{31,5}\boxplus(F_1(X_{31,4}\oplus SK_{126})$.
  $X_{32,7}=X_{31,7}\oplus(F_0(X_{31,6}\boxplus SK_{127})$.

3. 최종 변환

- $C_i=X_{32,i}$ for $i=1,3,5,7$,
  $C_0=X_{32,0}\boxplus WK_4$,
  $C_2=X_{32,2}\oplus WK_5$,
  $C_4=X_{32,4}\boxplus WK_6$,
  $C_6=X_{32,6}\oplus WK_7$.

〔그림 1〕은 HIGHT의 라운드 i의 라운드 함수를 나타낸다.



〔그림 1〕 i-th 라운드 함수

HIGHT의 키스케줄은 128-비트 비밀키 K를 입력받아 64-비트 화이트닝키 $WK_{0\le i\le 7}$와 1024-비트 라운드 키 $SK_{0\le i\le 255}$를 〔그림 2〕과 같이 생성한다. 여기서, $\delta_{0\le i\le 127}$은 LFSR을 이용하여 생성하는 8-비트 상수이다. 더욱 자세한 사항은 〔4〕를 참조하라.

```
WhiteningKeyGeneration(K, WK)
For i = 0 to 7
    If 0 ≤ i ≤ 3, then WK_i = K_{i+12};
    Else, WK_i = K_{i-4};

SubkeyGeneration(K, SK)
For i = 0 to 7
    For j = 0 to 7
        SK_{16·i+j} ← K_{(j-i mod 8)} ⊞ δ_{16·i+j};
        SK_{16·i+j+8} ← K_{(j-i mod 8)+8} ⊞ δ_{16·i+j+8};
```

〔그림 2〕 키스케줄 알고리즘

2.2 차분 오류 공격

차분 오류 공격은 기존 암호 알고리즘의 분석 방법인 차분 공격과 부채널 공격인 오류 주입 공격을 결합한 공격 방법으로 Biham 등이 DES에 대한 차분 오류 주입 공격을 최조로 제안하였으며, AES, Triple-DES, CLEFIA, IDEA, SEED, ARIA, SMS4 and MacGiffin등 다양한 블록 암호 분석에 사용되었다.

일반적으로 차분 오류 공격에서 공격자는 선택한 평문에 대해 특정 라운드의 입력레지스터에 임의의 오류를 주입하여 암호문을 얻을 수 있다고 가정한다. 이때, 레지스터의 임의의 1-비트에 오류를 주입된다는 가정과 임의의 1-바이트에 오류가 주입된다는 가정을 주로 사용하였다. 실제적으로 오류가 주입될 때, 하나의 비트에만 오류가 발생하기 보다는 여러 비트에 동시에 오류가 발생하기 때문에 임의의 1-바이트 오류 주입 가정이 일반적으로 사용되고 있다.

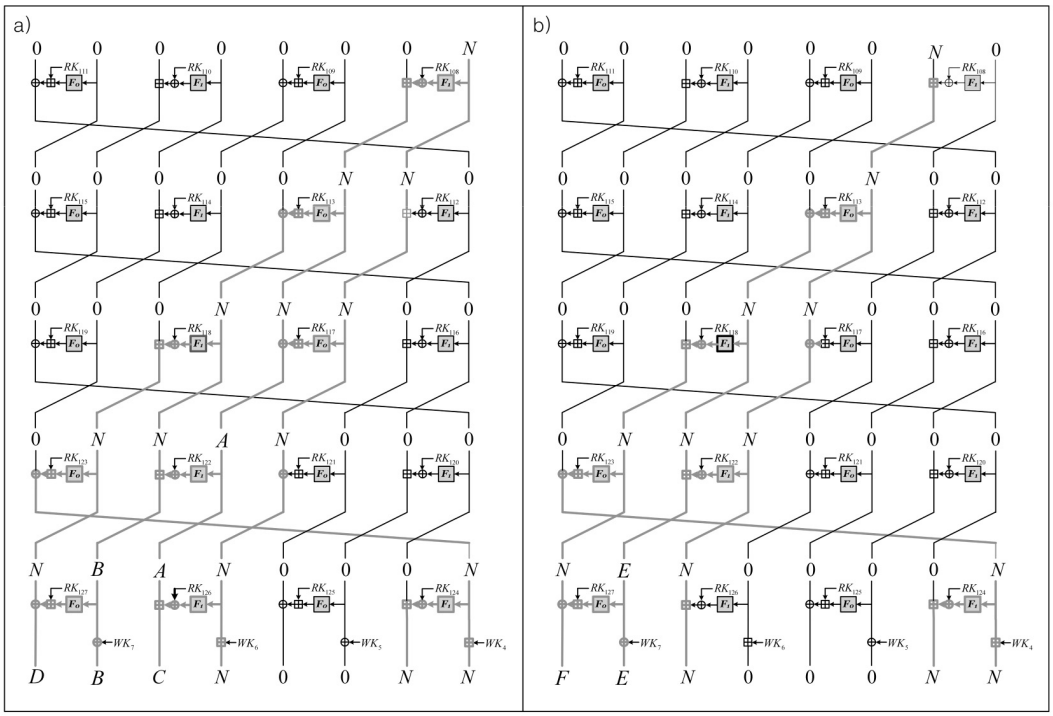
차분 오류 공격의 가장 일반적인 모델로는 블록 암호의 마지막 라운드에 오류를 주입하여 암호문의 차분으로부터 불가능한 차분을 가지는 비밀키 후보를 제거함으로서 마지막 라운드에 사용되는 라운드키를 복구한다. 마지막 라운드키를 복구하고, 이를 이용하여 마지막 두번째 라운드에 오류를 주입하여 라운드키를 복구한 후, 동일한 방법을 반복함으로서 전체 비밀키를 복구한다. 이러한 차분 오류 공격은 계산 복잡도는 매우 낮지만 공격에 필요한 오류 주입의 수가 많아지는 단점이 있다. 이를 극복하기 위해 일정 라운드 위에 차분을 주입하고 차분의 확산 특성을 분석하여 적은 수의 오류 주입을 사용하면서 실제적으로 공격이 가능한 복잡도를 가지는 형태의 차분 오류 공격이 제안되었다. 대표적으로 2009년 1개의 오류를 이용하여 2^{32}의 계산 복잡도를 가지는 AES에 대한 차분 오류 주입 공격을 제안되었다〔11〕. 이 후, ARIA, PRESENT 등에 유사한 기법을 적용하여 적은 수의 오류를 사용하는 차분 오류 주입 공격이 제안되었다.

III. 오류 주입 가정과 차분 확산 특성

본 절에서는 공격에 사용하는 오류 주입 가정과 주입된 오류에 의한 차분 확산 특성을 소개한다.

3.1 오류 주입 가정과 오류의 위치 결정

본 논문에서는 공격자가 라운드 28의 입력 레지스터에 임의의 1-바이트 오류를 주입 할 수 있음을 가정한다. 이러한 가정에서 공격자는 선택한 평문에 대해



〔그림 3〕 a) $X_{27,0}$에 오류가 주입되었을 때 차분 확산, b) $X_{27,1}$에 오류가 주입되었을 때 차분 확산

오류가 주입 되지 않은 정상적인 암호문 C와 오류가 주입된 암호문 C^*를 얻을 수 있다. 임의의 1-비트 오류를 주입할 수 있다고 가정하여도 제안하는 공격은 동일하게 적용되지만, 일반적으로 사용되는 오류 주입 가정인 임의의 1-바이트 오류를 주입할 수 있음을 가정한다.

본 공격에서 공격자는 라운드 28의 입력값에 오류가 주입할 수 있지만, 오류가 주입되는 위치를 선택할 수 없다. 하지만, 차분 확산 특성을 이용하여 암호문 쌍의 차분 형태로부터 오류가 주입된 바이트 위치를 알 수 있다.

〔그림 3〕은 $X_{27,0}$ 혹은 $X_{27,1}$에 오류가 주입되었을 때, 차분이 확산되는 형태를 나타낸다. N은 0이 아닌 차분을 의미하고, A,B,C,D,E,F는 임의의 차분값을 의미한다. 만약, A가 0인 경우, B는 0이 아닌 차분이므로 암호문에서 연속된 2 바이트에 차분이 없는 위치는 2,3번째 바이트이다. 또한, A가 0이 아닌 경우, B가 0인 경우와 B가 0이 아닌 경우로 나누면 연속된 2 바이트에 차분이 없는 위치는 모두 2,3번째 바이트만 존재한다. $X_{27,1}$에 오류가 주입된 경우에는 연속된 3 바이트에 차분이 없는 위치는 2,3,4 번째 바이트가 유일하다. 유사한 방법으로 다른 위치에 오류가 주입된 경우도 위의 2가지 경우처럼 증명가능하다. 〔표 1〕는 암호문의 차분이 존재하지 않는 연속된 바이트에 따라 주입된 오류의 위치를 나타낸다.

〔표 1〕 오류 주입 위치에 따른 암호문의 차분 형태

오류 주입 바이트	암호문의 차분이 존재하지 않는 바이트
$X_{27,0}$	C_3, C_2
$X_{27,1}$	C_4, C_3, C_2
$X_{27,2}$	C_5, C_4
$X_{27,3}$	C_6, C_5, C_4
$X_{27,4}$	C_7, C_6
$X_{27,5}$	C_0, C_7, C_6
$X_{27,6}$	C_1, C_0
$X_{27,7}$	C_2, C_1, C_0

3.2 오류 주입에 따른 차분 확산 특성

차분 오류 공격에서는 오류가 주입되었을 때 가능한 차분의 형태를 이용하여 불가능한 비밀키의 후보를 제거한다. 효율적인 공격을 위해서는 적은 비트의 비밀키를 추측하여 후보키를 제거 할 수 있는 식을 구성

〔표 2〕 오류 주입 위치에 따라 구성되는 식과 키 추측 위치

오류 주입 바이트	식	키 추측 바이트
$X_{27,0}$ 혹은 $X_{27,1}$	$\Delta X_{31,1}=0$	WK_4, RK_{124}
	$\Delta X_{30,7}=0$	$WK_4, WK_7, RK_{123}, RK_{127}$
	$\Delta X_{29,5}=0$	$WK_6, WK_7, RK_{118}, RK_{122}, RK_{126}, RK_{127}$
	$\Delta X_{28,3}=0$	$WK_5, WK_6, WK_7, RK_{113}, RK_{117}, RK_{121}, RK_{122}, RK_{125}, RK_{126}$
$X_{27,2}$ 혹은 $X_{27,3}$	$\Delta X_{31,3}=0$	WK_5, RK_{125}
	$\Delta X_{30,1}=0$	$WK_4, WK_5, RK_{120}, RK_{124}$
	$\Delta X_{29,7}=0$	$WK_4, WK_7, RK_{119}, RK_{123}, RK_{127}, RK_{124}$
	$\Delta X_{28,5}=0$	$WK_4, WK_6, WK_7, RK_{114}, RK_{118}, RK_{122}, RK_{123}, RK_{126}, RK_{127}$
$X_{27,4}$ 혹은 $X_{27,5}$	$\Delta X_{31,5}=0$	WK_6, RK_{126}
	$\Delta X_{30,3}=0$	$WK_5, WK_6, RK_{121}, RK_{125}$
	$\Delta X_{29,1}=0$	$WK_4, WK_5, RK_{116}, RK_{120}, RK_{124}, RK_{125}$
	$\Delta X_{28,7}=0$	$WK_4, WK_5, WK_7, RK_{115}, RK_{119}, RK_{123}, RK_{120}, RK_{127}, RK_{124}$
$X_{27,6}$ 혹은 $X_{27,7}$	$\Delta X_{31,7}=0$	WK_7, RK_{127}
	$\Delta X_{30,5}=0$	$WK_6, WK_7, RK_{122}, RK_{126}$
	$\Delta X_{29,3}=0$	$WK_5, WK_6, RK_{117}, RK_{121}, RK_{125}, RK_{126}$
	$\Delta X_{28,1}=0$	$WK_4, WK_5, WK_6, RK_{112}, RK_{116}, RK_{120}, RK_{121}, RK_{124}, RK_{125}$

하여야 한다. 본 공격에서는 오류 주입 위치에 따라 각각 4개의 식을 구성한다. 오류가 $X_{27,0}$이나 $X_{27,1}$에 주입된 경우, $\Delta X_{31,1}=0$, $\Delta X_{30,7}=0$, $\Delta X_{29,5}=0$, $\Delta X_{28,3}=0$의 4개의 식을 이용한다. 이 식은 〔그림 3〕에서 차분이 0인 위치로서 후보키를 추측하여 복호화 할 때, 비밀키를 옳게 추측하는 경우에는 확률 1로 차분이 0이 되지만, 잘못된 비밀키를 추측한 경우에는 확률적으로 차분이 0이 된다. 유사한 방법으로 오류가 주입되는 위치에 따라 4개의 식을 구성할 수 있다. 〔표 2〕는 구성하는 식과 이 식을 계산하기 위해 필요한 화이트닝키와 라운드 키를 나타낸다. 이러한 식을 고려하는 이유는 암호문으로부터 $\Delta X_{31,1}, \Delta X_{31,3}, \Delta X_{31,5}, \Delta X_{31,7}$을 계산하기 위해서는 각각 8-비트 라운드키와 8-비트 화이트닝키만을 추측하면 가능하기 때문이다. 즉, 2^{16}번의 복호화 연산을 통해 라운드 32의 라운드키와 최종 변환에 사용되는 화이트닝 키에 대한 후보키를 구할 수 있다. 각각의 후보키에 대해, $\Delta X_{30,1}, \Delta X_{30,3}, \Delta X_{30,5}, \Delta X_{30,7}$을 계산하기 위해서 각각의 식마다 8-비트 라운드키만을 추가로 추측하면 가능하다. 유사하게 $\Delta X_{29,1}, \Delta X_{29,3}, \Delta X_{29,5}, \Delta X_{29,7}$과 $\Delta X_{28,1}, \Delta X_{28,3}, \Delta X_{28,5}, \Delta X_{28,7}$을 계산하기 위해서 각각 8-비트 라운드키만을 추가로 추측하면 계산이 가능하다. 즉, 다수의 오류 암호문을 동시에 이용하여 위의 식을 만족하지 않는 후보키를 제거함으로서 후보키의 수를 충분히 줄인다면, 전체적으로 실제 구현 가능한 계산 복잡도로 비밀키를 복구할 수 있다.

IV. 블록 암호 HIGHT에 대한 차분 오류 공격

본 장에서는 블록 암호 HIGHT에 대한 차분 오류 공격을 소개한다. 제안하는 공격은 다수의 오류 주입 암호문쌍을 동시에 고려하여 비밀키를 복구한다. 제안하는 오류 주입 공격의 공격 과정은 다음과 같다.

1. t개의 평문 $P^i (i=0,1,\cdots t-1)$를 선택하고, 이에 대응되는 정상적인 암호문 C^i와 오류가 주입된 암호문 $C^{*,i}$을 얻고, 다음을 수행한다(총 t개의 오류를 주입).
   1.1. 각각의 암호문 쌍에 대해, 표 1을 이용하여 오류가 주입된 위치를 계산한다.
   1.2. 오류가 주입된 위치에 따라 암호문쌍 집합 $\mathfrak{I}^j (j=0,1,2,3)$을 고려한다. $\mathfrak{I}^j$는 주입된 오류의 위치가 $X_{28,2\cdot j}$ 혹은 $X_{28,2\cdot j+1}$인 암호문쌍들의 집합이다. t_j는 $\mathfrak{I}^j$의 원소의 개수로서 라운드 28의 특정 바이트 위치에 오류가 발생된 오류 암호문의 수를 의미한다.
2. 단계 1에서 생성한 암호문쌍 집합 $\mathfrak{I}^j$에 대해 다음을 수행한다.
   2.1. $\mathfrak{I}^0$에 속하는 암호문 쌍에 대해, RK_{124}와

WK_4를 추측하여 $\Delta X_{31,1}=0$을 만족하는 WK_4, RK_{124}의 후보키들을 테이블 Ω_{32}^0에 저장한다.

2.2. $\mathfrak{I}^1$에 속하는 암호문 쌍에 대해, RK_{125}와 WK_5를 추측하여 $\Delta X_{31,3}=0$을 만족하는 WK_5, RK_{125}의 후보키들을 테이블 Ω_{32}^1에 저장한다.

2.3. $\mathfrak{I}^2$에 속하는 암호문 쌍에 대해, RK_{126}과 WK_6을 추측하여 $\Delta X_{31,5}=0$을 만족하는 WK_6, RK_{126}의 후보키들을 테이블 Ω_{32}^2에 저장한다.

2.4. $\mathfrak{I}^3$에 속하는 암호문 쌍에 대해, RK_{127}와 WK_7를 추측하여 $\Delta X_{31,7}=0$을 만족하는 WK_7, RK_{127}의 후보키들을 테이블 Ω_{32}^3에 저장한다.

3. 테이블 $\Omega_{32}^0, \Omega_{32}^1, \Omega_{32}^2, \Omega_{32}^3$과 $\mathfrak{I}^j$에 대해 다음을 수행한다.

3.1. $\mathfrak{I}^0$에 속하는 암호문 쌍에 대해, RK_{123}을 추측하고, 테이블 $\Omega_{32}^0, \Omega_{32}^3$을 이용하여 $\Delta X_{30,7}=0$을 만족하는 $WK_4, WK_7, RK_{123}, RK_{127}$의 후보키들을 테이블 Ω_{31}^0에 저장한다.

3.2. $\mathfrak{I}^1$에 속하는 암호문 쌍에 대해, RK_{120}을 추측하고, 테이블 $\Omega_{32}^0, \Omega_{32}^1$을 이용하여 $\Delta X_{30,1}=0$을 만족하는 $WK_4, WK_5, RK_{120}, RK_{124}$의 후보키들을 테이블 Ω_{31}^1에 저장한다.

3.3. $\mathfrak{I}^2$에 속하는 암호문 쌍에 대해, RK_{121}을 추측하고, 테이블 $\Omega_{32}^1, \Omega_{32}^0$을 이용하여 $\Delta X_{31,5}=0$을 만족하는 $WK_5, WK_6, RK_{121}, RK_{125}$의 후보키들을 테이블 Ω_{31}^2에 저장한다.

3.4. $\mathfrak{I}^3$에 속하는 암호문 쌍에 대해, RK_{122}를 추측하고, 테이블 $\Omega_{32}^2, \Omega_{32}^3$을 이용하여 $\Delta X_{31,7}=0$을 만족하는 $WK_6, WK_7, RK_{122}, RK_{126}$의 후보키들을 테이블 Ω_{31}^3에 저장한다.

3.5. 테이블 $\Omega_{31}^0, \Omega_{31}^1, \Omega_{31}^2, \Omega_{31}^3$을 동시에 고려하여 불가능한 후보키들을 제거한다. 예를 들어, WK_4의 후보키는 테이블 $\Omega_{31}^0, \Omega_{31}^1$에서 각각 계산된다. 그러므로 두 테이블에 모두 포함되지 않는 WK_4의 후보키를 제거한다. 유사한 방법으로 WK_5, WK_6, WK_7의 불가능한 후보키를 각각의 테이블에서 제거한다.

4. 표 2를 고려하여 단계 3과 유사한 방법으로 $\Omega_{30}^0, \Omega_{30}^1, \Omega_{30}^2, \Omega_{30}^3$을 생성한다. 예를 들어, 테이블 Ω_{30}^0은 다음과 같이 생성한다. $\mathfrak{I}^0$에 속하는 암호문 쌍에 대해, RK_{118}을 추측하고, 테이블 $\Omega_{31}^0, \Omega_{31}^3$을 이용하여 $\Delta X_{29,5}=0$을 만족하는 $WK_6, WK_7, RK_{118}, RK_{122}, RK_{126}, RK_{127}$의 후보키들을 테이블 Ω_{30}^0에 저장한다. 이후, 단계 3.5와 유사하게 $\Omega_{30}^0, \Omega_{30}^1, \Omega_{30}^2, \Omega_{30}^3$에서 불가능한 후보키들을 제거하여 테이블을 갱신한다.

5. 단계 4와 유사한 방법으로 $\Omega_{29}^0, \Omega_{29}^1, \Omega_{29}^2, \Omega_{29}^3$을 생성한 후, 키스케줄을 고려하여 불가능한 후보키들을 제거한다. 즉, 테이블 $\Omega_{29}^0, \Omega_{29}^1, \Omega_{29}^2, \Omega_{29}^3$에 저장되어 있는 128-비트 라운드키와 32-비트 화이트닝키의 후보키들 중, 키스케줄에 의해 모순이 발생되는 후보키들을 제거한다.

6. 단계 5에서 생성한 $\Omega_{29}^0, \Omega_{29}^1, \Omega_{29}^2, \Omega_{29}^3$로부터 키스케줄을 이용하여 가능한 비밀키 후보에 대해 전수조사를 통해 비밀키를 복구한다.

제안하는 공격의 데이터 복잡도는 t개의 선택 평문과 이에 대응되는 t개의 오류가 암호문이다. 계산 복잡도와 메모리 복잡도의 경우, 사용된 오류 주입 개수에 따라 달라진다.

계산 복잡도를 구하기 위해서 각각의 단계에서 추측하는 후보키의 개수와 후보키 중 틀린키가 제거될 확률이 필요하다. 단계 2에서는 각각의 단계(2.1, 2.2, 2.3. 2.4)에서는 16-비트 후보키를 추측한다. 각각의 식에 대해 잘못된 키가 제거 되지 않을 확률이 2^{-8}이라면, 각각의 t_i가 2이면 각 단계에서 남는 후보키의 수는 $2 \simeq (1+(2^8-1)/2^8)$이다. 그리고 단계 3과 단계 4를 통하여 비밀키가 유일하게 남을 것으로 생각할 수 있다. 하지만, 실제 구현 결과는 각 단계에서 더욱 많은 후보키가 남았다. 그 이유는 AES와 같이 차분 특성이 좋은 S-박스를 사용하는 블록 암호에 대한 공격에서는 하나의 식에 의해 틀린키가 제거되지 않을 확률이 2^{-8}에 가까운 값을 가진다. 하지만 HIGHT같이 덧셈을 사용하는 경우에는 틀린키가 제거 되지 확률이 훨씬 높을 수 있다. 예를 들어, 단계 2의 경우에

사용된 비선형 함수는 법덧셈이 1번 혹은 2번만 사용되어 평균적으로 틀린키가 제거 될 확률이 2^{-8}에 비해 높은 값을 가진다. 실제로, 가능한 모든 값에 대한 전수조사를 통해 틀린키가 제거될 확률을 계산할 결과, $\Delta X_{31,3}=0$과 $\Delta X_{31,7}=0$의 경우, 틀린 키가 제거 되지 않을 확률은 $2^{-3.59}$이었고, $\Delta X_{31,5}=0$과 $\Delta X_{31,1}=0$에 의해 틀린 키가 제거 되지 않을 확률은 $2^{-5.78}$이었다. 단계 3과 단계 4에서 각각의 식에 관여하는 덧셈 연산의 수가 단계 2에 비해 많아 틀린 키가 제거 되지 않을 확률이 2^{-8}에 가까울 것으로 예상되지만, 고려해야 한 변수의 비트의 수가 크기 때문에 정확한 값을 계산할 수 없었다.

단계 3 이후, 각각의 식에 의해 잘못 추측된 비밀키가 필터링 될 확률을 2^{-8}으로 계산하여도 실제 구현시 필요한 계산 복잡도와 메모리 복잡도는 이론적인 결과와 큰 차이가 발생하였다. 이에 따라, 5절에서 실험결과를 바탕으로 계산 복잡도와 메모리 복잡도를 제시한다. 본 논문에서 제안하는 공격은 적은 수의 오류를 사용하여 실제로 구현 가능한 공격이므로 구현 결과를 바탕으로 한 계산 복잡도분석이 의미가 있을 것으로 생각된다.

V. 구현 결과와 공격 복잡도

본 장에서는 제안하는 공격에 대한 구현 결과를 소개한다. 실제 오류 주입의 공격 모델은 하드웨어 환경이지만, 소프트웨어 환경에서 오류 가입 가정을 이용하여 공격 과정을 구현하였다. 구현 환경은 Intel(R) Core(TM) i7 2.80GHZ CPU, 6 GB Ram, Visual studio 2008이고, 테이블을 저장하는데 사용되는 메모리는 최대 2^{27} 바이트로 설정하였다.

첫 번째 실험은 실제 공격 가정과 달리 공격자가 오류를 주입할 수 있는 위치를 지정할 수 있도록 하여 $t_0=t_1=t_2=t_3=2$(총 8개의 오류 주입)로 실험하였다. 즉, $X_{27,0}$이나 $X_{27,1}$에 오류가 주입된 2개의 암호문, $X_{27,2}$이나 $X_{27,3}$, $X_{27,4}$이나 $X_{27,5}$, $X_{27,6}$이나 $X_{27,7}$에 오류가 주입된 암호문을 각각 2개씩을 획득하여 총 8개의 오류 암호문쌍을 동시에 이용할 수 있다고 가정하여 공격을 수행하였다.

총 100회의 키 복구 공격을 수행하여 각 단계에서 추측하는 비밀키 후보의 수와 이전 단계에 남아있는 테이블에 남아있는 비밀키의 수를 통하여 각 단계에 필요한 연산량을 HIGHT 암호화 연산의 수로 측정하였고, 각 단계를 마친 후 저장하는 비밀키 후보의 수에 따라 생성되는 테이블의 크기를 측정하하여 필요한 메모리 복잡도를 계산하였다. 각 단계의 계산 복잡도와 메모리 복잡도를 합하여 전체 계산 복잡도와 메모리 복잡도를 계산하였다. 〔표 3〕은 이러한 가정에서 100회 공격 결과에 대한 평균값이다.

두 번째 구현 결과는 본 논문에서 제안한 공격에서 가정했던 공격 모델로 오류가 주입되는 바이트 위치를 공격자가 선택할 수 없도록 하여 8개의 오류 주입을 이용한 공격을 100회 실험하였다. 결과적으로 62%의 성공확률을 보여 100회 중 38회가 공격에 실패하였다. 공격이 실패 경우는 모두 t_0,t_1,t_2,t_3중 하나가 0이 되는 경우였다. 예를 들어, t_0가 0인 경우, 단계 2.1, 단계 3.1, 단계 4.1, 단계 5.1가 수행되지 않아 비밀키에 대한 필터링이 부족하여 공격이 실패하였다. 〔표 4〕는 〔표 3〕과 유사한 방법으로 8개의 오류를 주입하였을 때, 100회 실험 중 성공한 공격 결과의 평균값을 구한 결과이다.

세번째 구현 결과는 오류 주입의 수를 변화하면서 각각 100회의 공격 실험을 통해 공격의 성공 확률과 계산 복잡도를 측정하였다. 표 5는 이 결과를 나타낸

〔표 3〕 $t_0=t_1=t_2=t_3=2$일때, 100회 구현 결과

	계산 복잡도 (HIGHT 암호화 연산)				각 단계에서 생성된 테이블 Ω의 크기(byte)			
	단계 X.1	단계 X.2	단계 X.3	단계 X.4	단계 X.1	단계 X.2	단계 X.3	단계 X.4
단계 2	2^{10}	2^{10}	2^{10}	2^{10}	$2\cdot2^{5.78}$	$2\cdot2^{9.60}$	$2\cdot2^{5.90}$	$2\cdot2^{9.65}$
단계 3	$2^{15.27}$	$2^{15.59}$	$2^{15.41}$	$2^{15.28}$	$4\cdot2^{8.29}$	$4\cdot2^{8.46}$	$4.\cdot2^{7.80}$	$4.\cdot2^{8.12}$
단계 4	$2^{17.70}$	$2^{14.90}$	$2^{18.36}$	$2^{14.20}$	$6\cdot2^{6.49}$	$6\cdot2^{5.98}$	$6\cdot2^{6.63}$	$6\cdot2^{6.02}$
단계 5	$2^{14.04}$	$2^{12.36}$	$2^{12.17}$	$2^{13.15}$	$9\cdot2^{3.63}$	$9\cdot2^{4.06}$	$9\cdot2^{5.23}$	$9\cdot2^{3.55}$
단계 6	$2^{12.09}$				.			

전체 계산 복잡도	$2^{19.66}$	성공 확률(%)	100
전체 메모리 복잡도	$2^{13.36}$	공격 시간(초)	0.23

〔표 4〕 $t=8$일때, 100회 구현 결과(성공한 공격 결과에 대한 평균값)

	계산 복잡도 (HIGHT 암호화 연산)				각 단계에서 생성된 테이블 Ω의 크기(byte)			
	단계 X.1	단계 X.2	단계 X.3	단계 X.4	단계 X.1	단계 X.2	단계 X.3	단계 X.4
단계 2	$2^{9.54}$	$2^{9.23}$	$2^{9.26}$	$2^{9.38}$	$2\cdot 2^{8.47}$	$2\cdot 2^{10.05}$	$2\cdot 2^{7.97}$	$2\cdot 2^{10.50}$
단계 3	$2^{16.14}$	$2^{17.94}$	$2^{16.41}$	$2^{16.65}$	$4\cdot 2^{10.20}$	$4\cdot 2^{11.37}$	$4\cdot 2^{12.82}$	$4\cdot 2^{11.17}$
단계 4	$2^{22.41}$	$2^{17.14}$	$2^{22.26}$	$2^{19.37}$	$6\cdot 2^{7.68}$	$6\cdot 2^{7.62}$	$6\cdot 2^{8.99}$	$6\cdot 2^{8.15}$
단계 5	$2^{17.83}$	$2^{14.58}$	$2^{17.62}$	$2^{21.12}$	$9\cdot 2^{3.71}$	$9\cdot 2^{3.58}$	$9\cdot 2^{3.87}$	$9\cdot 2^{3.29}$
단계 6	$2^{13.43}$				.			
전체 계산 복잡도	$2^{23.49}$			성공 확률(%)	61			
전체 메모리 복잡도	$2^{14.56}$			공격 시간(초)	6.03			

〔표 5〕 t에 따른 공격 복잡도 및 성공 확률의 실험 결과

t(오류 암호문 수)	6	7	8	9	10	11	12
계산 복잡도	$2^{25.82}$	$2^{24.52}$	$2^{23.49}$	$2^{23.12}$	$2^{22.45}$	$2^{19.23}$	$2^{18.57}$
메모리 복잡도(바이트)	$2^{14.56}$	$2^{13.23}$	$2^{13.21}$	$2^{13.43}$	$2^{12.34}$	$2^{11.23}$	$2^{10.17}$
성공 확률(%)	0.40	0.53	0.61	0.74	0.77	0.85	0.90
공격 시간(초)	15.73	7.08	6.03	4.32	2.81	0.52	0.42

다. t의 값에 상관없이 t_0,t_1,t_2,t_3가 모두 0이 아닌 경우에는 공격이 성공하였다. t가 작을수록 평균적인 계산 복잡도가 작아지고 메모리 복잡도는 증가함을 알 수 있다. $t=8,9,10$인 경우의 공격 복잡도는 거의 유사하지만 t가 11보다 커지면 공격 시간이 크게 감소하였다. 이는 t가 11보다 큰 경우에는 t_0,t_1,t_2,t_3가 모두 2 이상이 될 확률이 증가하기 때문이다(〔표 2〕의 구현 결과와 유사).

이러한 구현 결과를 바탕으로 공격의 성공 확률은 t_0,t_1,t_2,t_3가 모두 0이 아닌 확률로 예상할 수 있다.그러므로 오류 주입 개수에 대한 제안하는 공격의 이론적인 성공확률은 〔표 6〕과 같다.

〔표 6〕 t에 따른 성공 확률

t	6	7	8	9	10	11	12
성공 확률(%)	38	51	62	71	78	83	88

결론적으로 12개의 오류를 주입하면 성공 확률 88%의 성공 확률로 HIGHT의 비밀키를 수초내로 복구할 수 있으며, 7개의 오류를 주입하면 50%이상의 확률로 HIGHT의 비밀키를 수초내로 복구할 수 있을 것으로 예측된다.

VI. 결 론

본 논문에서는 블록암호 HIGHT에 대한 차분 오류 주입 공격을 제안하였다. 제안한 공격은 라운드 28의 입력값에 임의의 1-바이트 오류를 주입하여 오류 주입을 통해 얻어진 암호문과 정상적으로 얻어진 암호문의 차분 특성을 이용하여 비밀키를 복구하였다. 12개의 오류를 주입하여 87.5%의 성공 확률로 수초 내에 HIGHT의 비밀키를 복구할 수 있으며 7개의 오류를 주입하여 51.3%의 성공 확률로 HIGHT의 비밀키를 복구하였다.

이 후 연구 목표로는 t_0,t_1,t_2,t_3 중 하나의 값이 0 인 경우에, 가능한 공격 방법과 보다 적은 수의 오류를 사용하면서도 성공 확률을 높이는 방법을 연구할 계획이다. 또한, 각 단계에서 각각의 식에 의해 틀린키가 필터링 되는 확률을 분석함으로서 공격 복잡도에 대한 이론적인 분석을 수행할 계획이다.

참고문헌

〔1〕 NIST, “Advanced Encryption Standard,” FIPS-197, Nov. 2001,

〔2〕 Korea Information Security Agency, “SEED Algorithm Specification”. Available at http://seed.kisa.or.kr/seed/down/SEED_Specification_english.pdf.

〔3〕 D. Kwon, J. Kim, S. Park, S. Sung, Y. Sohn, J. Song, Y. Yeom, E. Yoon, S. Lee, J. Lee, S. Chee, D. Han and J. Hong, "New Block Cipher: ARIA," ICISC'03, LNCS 2971, pp. 443-456, Springer-Verlag, 2003.

〔4〕 D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim and S. Chee, "HIGHT: a new block cipher suitable for low-resource device," CHES 2006, LNCS 4249, pp. 46-59, Springer-Verlag, 2006.

〔5〕 A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin and C. Vikkelsoe "PRESENT: An Ultra-Lightweight Block Cipher," CHES 2007, LNCS 4727, pp. 450-466, Springer-Verlag, 2007.

〔6〕 C. Cannière, O. Dunkelman and M. Knezevic, "KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers," CHES 2009, LNCS 5747, pp. 272-288, Springer-Verlag, 2009.

〔7〕 E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," Crypto 1997, LNCS 1294, pp. 513-525, 1997.

〔8〕 J. Blömer and J.-P. Seifert. "Fault based cryptanalysis of the advanced encryption standard (AES)." In Financial Cryptography, FC 2003, LNCS 2742, pp. 162-181, 2003.

〔9〕 P. Dusart, G. Letourneux, and O. Vivolo, "Differential fault analysis on AES," ACNS 2003, LNCS 2846, pp. 293-306, 2003.

〔10〕 C. Giraud, "DFA on AES," 4th International Conference, AES 2004, LNCS 3373, pp. 27-41, 2005.

〔11〕 D. Mukhopadhyay, "An improved fault based attack of the advanced encryption standard," AFRICACRYPT 2009, LNCS 5580, pp. 421-434, 2009.

〔12〕 L. Hemme, "A differential fault analysis against early rounds of (Triple-) DES," CHES 2004, LNCS 3156, pp. 254-267, 2006.

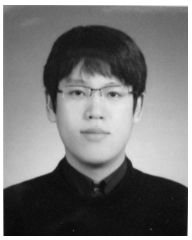
〔13〕 H. Chen, W. Wu, and D. Feng, "Differential fault analysis on CLEFIA," ICICS 2007, LNCS 4861 pp. 284-295, 2007.

〔14〕 K. Jeong, Y. Lee, J. Sung, and S. Hong, "Differential fault analysis on block cipher SEED," Mathematical and Computer Modelling, Vol. 55, No. 1-2, pp. 26-34, Jan. 2012.

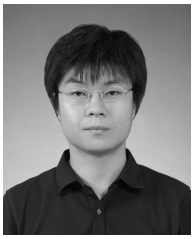
〔15〕 W. Li, D. Gu and J. Li, "Differential fault analysis on the ARIA algorithm," Information Sciences, Vol. 178, No. 19, pp. 3727-3737, Oct. 2008.

〔16〕 W. Li, D. Gu, and Y. Wang. "Dierential fault analysis on the contracting UFN structure, with application to SMS4 and Macguffin." Journal of Systems and Software, Vol. 82, No. 2, pp. 346-354, Feb. 2009.

〈著者紹介〉



이 유 섭 (Yuseop Lee) 학생회원
2007년 2월 : 서울시립대학교 수학과 학사
2007년 3월~현재 : 고려대학교 정보경영공학전문대학원 석박사 통합과정
〈관심분야〉 스트림 암호 및 해쉬 함수의 분석 및 설계



김 종 성 (Jong-sung Kim) 종신회원
2000년 8월: 고려대학교 수학과 (이학사)
2002년 8월: 고려대학교 수학과(이학석사)
2006년 11월: K.U.Leuven, ESAT/SCD-COSIC (공학박사)
2007년 2월: 고려대학교 정보보호대학원 (공학박사)
2007년 3월~2009년 8월: 고려대학교 정보보호기술연구센터 연구교수
2009년 9월~2011년8월: 경남대학교 e-비즈니스학부 전임강사
2011년 9월~현재: 경남대학교 e-비즈니스학부 조교수
〈관심분야〉 정보보호, 블록암호 분석 및 설계

홍 석 희 (Seokhie Hong) 종신회원
1995년 2월: 고려대학교 수학과 학사
1997년 2월: 고려대학교 수학과 석사
2001년 2월: 고려대학교 수학과 박사
1999년 8월~2004년 2월: (주) 시큐리티 테크놀로지스 선임연구원
2004년 4월~2005년 2월: K.U. Leuven, ESAT/SCD-COSIC 박사후연구원
2005년 3월~2008년 8월: 고려대학교 정보보호대학원 조교수
2008년 9월~현재: 고려대학교 정보보호대학원 부교수
〈관심분야〉 암호 알고리즘 설계 및 분석, 컴퓨터 포렌식