情報保護學會論文誌
第 20卷 第 6號, 2010. 12

DES 알고리즘에 대한 새로운 차분오류주입공격 방법*

소 현 동,[1†] 김 성 경,[1] 홍 석 희,[1] 강 은 숙[2‡]
[1]고려대학교 정보경영공학전문대학원, [2]고려대학교 정보수학과

A New Type of Differential Fault Analysis on DES Algorithm*

Hyun-Dong So,[1†] Sung-Kyoung Kim,[1] Seokhie Hong,[1], Eun-Sook Kang[2‡]
[1]Graduate School of Information Management & Security, Korea University,
[2]Department of Information & Mathematics, Korea University

요 약

차분오류주입공격 (Differential Fault Analysis, DFA)은 블록 암호를 분석하는 가장 효과적인 부채널 공격 (Side-Channel Attacks, SCAs)방법 중 하나로 알려져 있다. 본 논문에서는 DES (Data Encryption Standard)에 대한 새로운 DFA 방법에 대하여 제안한다. DES에 대한 DFA 방법은 Biham과 Shamir에 의하여 처음으로 제안된 이후, 2009년 Rivain에 의하여 DES의 중간 라운드 (9~12 라운드)에 대한 DFA 방법에 이르기까지 많은 DFA 방법이 소개되었다. 하지만 기존의 DES에 대한 DFA 방법은 암호화 (또는 복호화)과정에 오류를 주입하는 공격방법임에 비해 본 논문에서 제안하는 DES에 대한 DFA 방법은 DES의 키 생성과정에 오류를 주입하여 분석하는 방법을 처음으로 제안한다. 제안하는 DFA 방법은 기존의 방법들에서 사용하는 오류 모델을 포괄하는 오류 모델을 사용하고 더 적은 오류의 개수로 공격이 가능하다.

ABSTRACT

Differential Fault Analysis (DFA) is widely known for one of the most efficient method analyzing block cipher. In this paper, we propose a new type of DFA on DES (Data Encryption Standard). DFA on DES was first introduced by Biham and Shamir, then Rivain recently introduced DFA on DES middle rounds (9-12 round). However previous attacks on DES can only be applied to the encryption process. Meanwhile, we first propose the DFA on DES key-schedule. In this paper, we proposed a more efficient DFA on DES key schedule with random fault. The proposed DFA method retrieves the key using a more practical fault model and requires fewer faults than the previous DFA on DES.

Keywords: Side-Channel Attacks, Differential Fault Analysis, DES, Key-scheduling

I. 서 론

근래 암호 알고리즘에 대한 분석은 암호 알고리즘에 기반이 되는 수학적 이론에 의한 직접적인 분석 방법뿐만 아니라 알고리즘의 실행에서 얻을 수 있는 부채널 정보를 이용한 부채널 분석 (Side Channel Attacks, SCAs)방법 또한 널리 연구되고 있다. 1996년 SCAs의 개념이 소개된 이후[1], 내장형 장치 (embedded device)의 암호알고리즘의 실행에서 얻을 수 있는 시간 차, 소비전력, 전파 방사량 등의 정보를 이용하여 여러 SCAs 방법이 제안되고 있다. SCAs는 크게 수동적인 분석 (Passive SCAs)방법과 능동적인 분석 (Active SCAs)방법의 두 가지 범주로 나눌 수 있다. 수동적인 분석 방법으로 시간차

접수일(2010년 7월 21일), 게재확정일(2010년 9월 29일)
* 이 연구에 참여한 연구자(의 일부)는 "2단계 BK21 사업"의 지원비를 받았음.
† 주저자, ekurtso@korea.ac.kr
‡ 교신저자, kes@korea.ac.kr

공격 (Timing Attack, TA)[1], 단순전력분석 (Simple Power Analysis, SPA), 차분전력분석 (Differential Power Analysis, DPA)[2], 전자파 분석 (Electromagnetic Analysis, EMA)[3] 등이 있고, 능동적인 분석 방법으로 오류주입공격 (Fault Analysis, FA)([4],[5])과 차분오류주입 공격 (Differential Fault Analysis, DFA)[6]이 있다.

FA를 이용한 첫 번째 분석은 1997년 Boneh 등에 의해 CRT-RSA (Chinese Remaider Theorem에 기반한 RSA) 서명 스킴과 Rabin 서명 스킴에 오류를 주입하여 비밀 정보를 얻는 방법으로 제안되었다[4]. 이후, FA 방법을 대칭키 암호알고리즘에 적용한 DFA 방법이 Biham과 Shamir에 의해 처음 소개 되었다[6]. 대칭키 알고리즘의 DFA 방법은 FA 방법과 차분공격법(Differential Cryptanalysis, DC)을 적용하여 공격하는 방법으로, 블록암호 알고리즘 또는 스트림암호 알고리즘의 데이터 흐름에서 특정한 부분에 오류를 주입하여 얻은 옳은 암호문과 오류를 주입한 암호문의 차분을 이용하여 비밀키를 얻는 방법이다([7], [8], [9], [10]).

가장 널리 쓰이는 블록 암호 중 하나인 DES에 대한 DFA 방법은 Biham 등에 의하여 처음 제안되었고, 2004년 Akkar에 의해 차분공격법을 이용하여 개선된 DFA 방법이 제안되었다[11]. 2009년 Akkar의 방법을 일반화하고 확장하여, Rivain은 중간라운드 (9-12라운드)의 왼쪽 입력 값에 오류를 주입하여 비밀키를 얻는 DFA 방법을 제안하였다. 이로서 암호화 과정과 복호화 과정에 적용하여 DES의 모든 라운드에 DFA 방법을 적용할 수 있게 되었다. 하지만 현재까지 제안된 DES에 대한 DFA는 오직 DES의 암호화 과정 (또는 복호화 과정)의 특정한 라운드의 입력 값에 오류를 발생시켜 비밀키를 얻는 방법만이 고려되었다. 반면 AES의 경우, Giraud에 의해 AES의 암호화 과정이 아닌 AES 키 생성과정에 오류를 주입하여 비밀키를 찾는 DFA 방법[13]이 처음 제안된 이후, AES의 키 생성과정에 오류를 주입하여 DFA 방법을 적용한 여러 분석 방법이 제안되었다([14], [15], [16]).

본 논문은 DES의 마지막 세 라운드 (14, 15, 16 라운드)의 키 생성과정에 오류를 주입하여 DES의 비밀키를 복구하는 DFA 방법을 제안한다. AES의 키 생성과정은 Permutation 과정이 없으므로 하나의 워드에 오류가 발생하면 그 오류가 다른 워드에 확산되지 않아 분석이 쉬운 반면, DES의 키 생성과정의 경우 하위 라운드 키에 오류가 발생하면, Permutation 과정을 거쳐 상위 라운드 키의 여러 워드에 오류의 영향을 받아 분석이 어렵다. 또한, 처음으로 오류가 발생한 라운드의 데이터는 상위 라운드를 거치면서 또 다른 오류와 데이터 암호화 과정에 영향을 받아 여러 개의 오류를 주입하는 효과가 나타난다. 따라서 본 논문에서 제안하는 DFA 방법에서 주입된 오류는 각 라운드에 중첩적으로 발생하여 기존의 DES에 대한 DFA 방법의 오류 모델에 비하여 오류의 확산을 분석하기 어렵다.

본 논문에서 제안하는 DFA 방법은 기존의 DES 암호화 과정에 대한 DFA가 1-비트 또는 1-바이트 랜덤 오류 모델을 이용하여 분석하는 것과는 달리, DES의 키 생성과정의 1-비트 또는 1-바이트 랜덤 오류 모델을 포함하는 28-비트 레지스터에 랜덤 오류 모델을 이용하여 분석한다. 그러므로 제안하는 DFA 방법은 키 생성과정에 제약이 적은 오류를 주입하여 DFA 방법을 적용할 수 있고, 그 결과 기존에 제안된 DES에 대한 DFA 방법에 비하여 적은 개수의 오류를 이용한 DFA 방법을 적용하여 비밀키를 찾아 낼 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 DES 블록암호 알고리즘과 기존의 연구에 대해 설명하고 3절에서는 오류주입 가정과 제안하는 DES의 키 생성과정에 대한 DFA 방법을 설명한다. 그리고 4절에서는 제안하는 방법에 대한 실험결과를 제시하고 마지막으로 5절에서 결론을 내린다.

II. DES 블록 알고리즘과 기존의 차분오류주입 공격

본 장에서는 본 논문에서 사용하는 표기법을 정리하고 DES 블록 알고리즘 (Data Encryption Standard)과 지금까지 제안된 DES에 대한 차분오류주입공격 (DFA)방법에 대해 설명한다.

2.1 표기법

$A[n]$: 데이터 A의 n번째 비트
$A[n:m]$: A의 n번째 비트에서 m번째 비트의 (m-n+1)-비트
$A \oplus B$: A와 B의 배타적 논리합 (XOR)
P, C: 평문, 옳은 암호문 (각 64-비트)

K: 비밀키 (64-비트)
RK_i: i번째 라운드 키
reg_i^L: i번째 라운드 키 생성과정의 Compression Permutation의 좌측 입력 값 레지스터
reg_i^R: i번째 라운드 키 생성과정의 Compression Permutation의 우측 입력 값 레지스터
$C_L^*(C_R^*)$: $reg_i^L(reg_i^R)$에 오류가 주입된 암호문
$L_i(R_i)$: i번째 라운드의 좌측 (우측) 32-비트 옳은 출력 값
$L_i^*(R_i^*)$: 오류가 주입된 i번째 라운드의 좌측 (우측) 32-비트 출력 값

2.2 DES 블록 알고리즘 설명

DES는 1976년 미국 NBS (National Bureau of Standards, 현재 NIST)에서 데이터의 암호를 위하여 미국 암호표준으로 정한 블록암호 알고리즘이다[17]. DES는 64-비트 블록크기와 패리티 체크 비트 8비트를 포함하는 64-비트 키 길이를 가지고, F-함수를 포함하는 16라운드의 일반적인 Feistel 네트워크 (Generalized Feistel Network, GFN)구조이다. 입력 값 64-비트의 P는 Initial Permutation (IP)과정을 거쳐 2개의 32-비트 블록 L_1과 R_1로 나눠져, 16라운드 암호화 과정을 거친 후 Final Permutation (IP^{-1})과정을 거쳐 64-비트 C을 출력한다.

2.2.1 F-함수

F-함수는 우측 32-비트 입력 값 (R_{i-1})과 48-비트 라운드 키 (RK_i)를 입력 받고, 32-비트 $F(R_{i-1}, RK_i)$를 출력한다. [그림 1 (A)]는 F-함수를 나타낸 것이며, F-함수는 Expansion Permutation (EP)과정, 라운드 키 XOR 연산 (Add round-key)과정 , S-box (S1,...,S8) 연산과정, Straight Permutation (SP)과정의 4개의 과정으로 이루어진다.

2.2.2 라운드 키 생성 (Round-key Scheduling)과정

DES의 비밀키는 56-비트 키와 패리티 체크 비트 8-비트를 포함하여 64-비트이다. DES의 라운드 키 생성과정은 [그림 1 (B)]와 같고, 56-비트의 비밀키를 이용하여 16개의 48-비트 라운드 키를 생성한다. 라운드 키 생성과정은 아래와 같이 크게 3단계로 구성된다.

단계 1. 패리티 드롭 (Parity drop)과정
비밀키 64-비트에서 8-비트 패리티 비트를 제외한 56-비트 암호화 키(Cipher key)를 생성하는 과정.

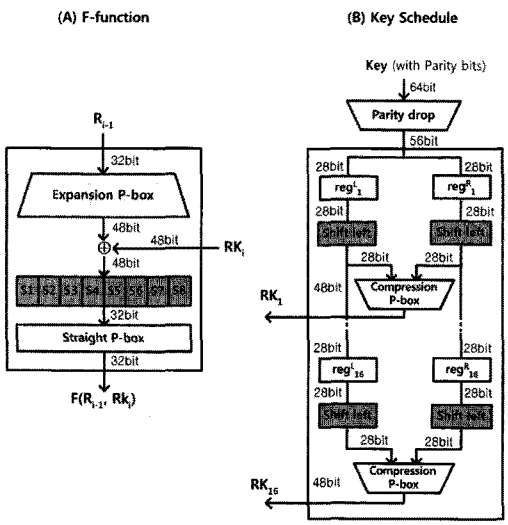
단계 2. Left circular- shift 과정
좌측 레지스터 (reg_i^L)와 우측 레지스터 (reg_i^R)에 일시적으로 저장된 28-비트를 2비트 (1, 2, 9, 16라운드는 1비트) Left circular-shift 연산하는 과정.

단계 3. Compression Permutation 과정
56-비트 입력 값을 치환하여 48-비트 출력 값으로 출력하는 과정.

2.3 DES에 대한 기존 DFA 방법

DES에 대한 DFA 방법은 1997년 Biham과 Shamir에 의해 처음 제안되었다[6]. Biham 등은 13라운드 - 16라운드의 왼쪽 입력 값에 오류를 발생시켜 얻은 암호문의 차분에 대하여 차분공격법을 적용하여 비밀키를 찾아내었다. 이때, 약 10개의 1-비트 오류를 사용하여 16라운드 키를 찾고 비밀키의 남은 8-비트를 전수조사 (Exhaustive search)하여 비밀키를 획득하는 방법을 사용하였다. 이후 Akkar은 차분공격법을 DFA에 적용하여 Biham의 DFA 방법을 일반화하였으며, Rivain은 구별자 (Distinguisher) 개념을 이용하여 중간라운드 (9라운드~12라운드)에 대한 DFA 방법을 제안하였다[12]. 특히 Rivain은 1-비트 또는 1-바이트 오류를 주입하고 공격자의 오류의 인지 정도에 따라 구별자를 선택하여 DES에 대한 DFA공격의 성공확률을 높이는 방법을 사용하였다. 이로서 DES에 대한 DFA 방법을 암호화와 복호화 과정에 적용하여 모든 라운드에서 DFA가 가능하게 되었다.

하지만 현재까지 제안된 DES에 대한 DFA 방법은 암호화 과정에서 오류를 주입하는 방법으로 제안되었고, DES의 키 생성과정에 대한 DFA 방법은 존재하지 않는다. 본 논문에서는 DES의 키 생성과정에 오류를 주입하는 DFA 방법에 대하여 제안한다.



〔그림 1〕 (A)F-함수와 (B)라운드 키 생성과정

III. 제안하는 차분오류주입공격

본 장에서는 제안하는 DES의 키 생성과정에 대한 DFA 방법의 오류주입 모델에 대한 정의와 공격 방법에 대하여 설명한다. 14라운드와 15라운드에서의 공격 방법과는 달리, 오류확산에 대한 분석이 필요 없는 16라운드 공격 방법은 〔Appendix 1〕에서 따로 설명한다.

3.1 오류주입 모델

본 절에서는 본 논문에서 제안하는 라운드 키 생성과정에 대한 DFA 방법에서 고려하는 오류주입 모델에 대하여 설명한다. 제안하는 DFA 방법은 공격자가 고정된 P에 대하여 옳은 C를 얻을 수 있고, reg_i^L (또는 reg_i^R, $i=14, 15$)에 랜덤 오류를 주입하여 C_L^* (또는 C_R^*)을 얻을 수 있다고 가정한다. 본 논문에서 가정하는 오류주입에 대한 표현은 〔Definition 1〕과 같이 정의한다.

Definition 1. (Random bit Fault Notation) X를 오류가 주입될 28-비트 레지스터라고 하면, 오류가 주입된 X는 X^*라고 표현하고 다음과 같이 정의된다.

$$X^* = X + e(X) \bmod 2^{28} \quad (1)$$

이때, $e(X) \in \{1, 2, \cdots, 2^{28}-1\}$는 공격자가 모르는 값이고 $X^* \in [0, 2^{28}-1]$을 만족한다.

공격자는 주입한 랜덤 오류의 값을 모르고 주입된 오류가 암호화 과정에서 어떻게 영향을 미치는지 알 수 없지만, 오류를 주입할 위치를 선택하여 주입할 수 있다고 가정한다. 즉, 공격자는 (reg_{14}^L, reg_{14}^R, reg_{15}^L, reg_{15}^R) 중 하나의 레지스터에 랜덤 오류를 주입하여 대응하는 암호문 쌍을 얻을 수 있다.

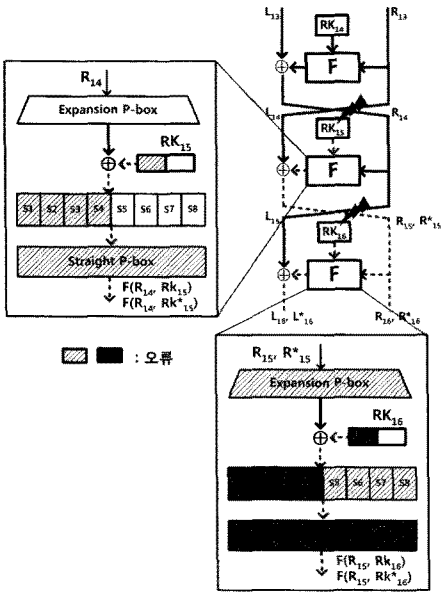
지금까지 제안된 DES에 대한 DFA 방법에서의 오류주입 모델은 32-비트 암호화 (또는 복호화) 연산의 한 시점에 1-비트 또는 1-바이트 랜덤 오류가 주입되는 것으로 가정하고 있다. 이때 1-비트 랜덤 오류주입 모델은 타겟 32-비트의 임의의 1-비트 값이 0이면 1로 변화되고, 1이면 0으로 변화되는 오류주입 모델이다. 즉, 타겟이 되는 32-비트에서 정확히 1-비트만이 원래의 값과 다른 값으로 바뀌고 그 값이 여러 라운드를 거치면서 다른 암호문으로 출력된다. 비슷한 방법으로 1-바이트 랜덤 오류모델은 L_i의 임의의 1-바이트를 다른 값으로 변화되는 오류주입 모델이다. 따라서 1-바이트 랜덤 오류모델을 이용한 DFA 방법은 1-비트 랜덤 오류모델을 이용한 DFA 방법을 포함하는 더 일반적인 랜덤 오류주입 모델이다.

하지만 본 논문에서 제안하는 오류주입 모델은 마지막 3 라운드 (14, 15, 16라운드)에 대한 키 생성과정의 28-비트 레지스터에서의 랜덤 오류를 고려한다. 하드웨어 실행 관점에서 28-비트 레지스터 랜덤 오류주입 모델은 DES의 키 생성과정에서 저장된 레지스터에 임의의 오류를 주입하여 실제적으로 공격가능하다. 이는 기존의 1-비트 랜덤 오류와 1-바이트 (8-비트) 랜덤 오류주입 모델을 포함하여 기존의 오류주입 모델에 비하여 더 일반적인 오류주입 모델이다. 따라서 제안하는 오류주입 모델은 이전 오류주입 모델에 비하여 더 실제적인 방법으로 오류를 주입하여 비밀키를 얻을 수 있다.

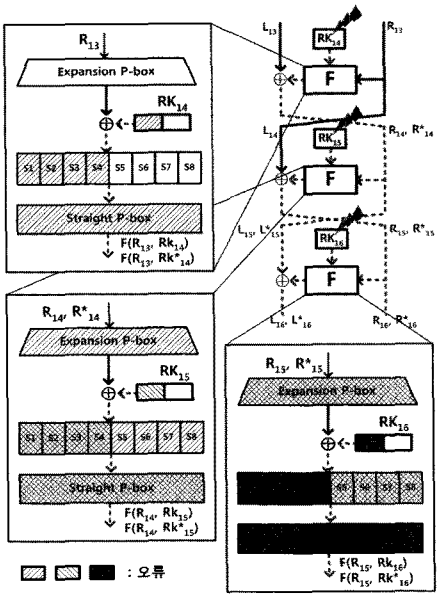
3.2 키 생성과정에 주입한 오류의 확산 분석

본 논문에서 제안하는 DES의 키 생성과정에 대한 DFA 방법을 설명하기 전에 제안하는 오류주입 모델에 따른 오류가 암호화 과정에서 확산되는 성질을 본 절에서 분석한다.

다음에 제시하는 〔Fact 1〕과 〔Fact 2〕는 공격자가 옳은 암호문과 오류가 주입된 암호문을 얻었을 때,



〔그림 2〕 15라운드의 오류확산



〔그림 3〕 14라운드의 오류확산

16라운드의 특정 S-box에 대한 입력 차분을 알 수 있음을 보여준다.

Fact 1. 〔표 1〕은 *CP*의 입력 값과 출력 값의 관계를 나타낸다. *CP*의 입력 값을 28-비트로 나누었을 때, 28-비트 데이터는 각각 독립적으로 치환된다. 따라서 오류가 reg_i^L (또는 reg_i^R, $i=14,15$)에 주입되면, $RK_j(j=i+1,...,16)$의 $RK_j[1:24]$ (또는 $RK_j[25:48]$)에만 오류가 주입된다.

Fact 2. $x_i\,(i=1,...,8)$를 각각 16라운드의 i번째 S-box에 대응하는 옳은 입력 값이라 하고, $x_i^*(i=1,...,8)$를 $reg_j^L(j=14,15)$에 오류가 발생하여 얻은 S-box의 입력 값이라 하면, 공격자는 $x_5\oplus x_5^*,...,x_8\oplus x_8^*$를 알 수 있다.

유사하게 $x_i^*(i=1,...,8)$가 $reg_j^R(j=14,15)$에 오류가 발생하여 얻은 입력 값이면, 공격자는 $x_1\oplus x_1^*,...,x_4\oplus x_4^*$를 알 수 있다.

15라운드 키 생성과정에서 오류가 발생하면, 〔그림 2〕와 같이 16라운드 F-함수의 출력 값에 대한 차분 $L_{16}\oplus L_{16}^*$을 알 수 있다. 반면 14라운드 키 생성과정에서 오류가 발생하면, 〔그림 3〕과 같이 16라운드 F-함수의 출력 값에 대한 차분은 $L_{16}\oplus L_{16}^*\oplus L_{15}\oplus L_{15}^*$이고, 공격자가 알 수 없는 값이다. 다음에 제시하는 〔Fact 3〕과 〔Proposition 1〕은 $L_{16}\oplus L_{16}^*\oplus L_{15}\oplus L_{15}^*$의 일부 비트 값을 알 수 있음을 보여준다.

Fact 3. reg_{14}^L에 오류가 발생하면, 14라운드 키의 왼쪽 24-비트에 오류가 주입된다. 〔그림 3〕과 같이 오류는 14라운드의 S1-box, ..., S4-box에 의해 *SP*과정 입력 값의 1-비트~16-비트에 영향을 미친다. 그 결과 *SP*과정을 거치면, 〔표 2〕와 같이 16비트에 오류가 발생한다. 따라서 F-함수의 출력 차분은 $\Delta L_{15}=L_{15}\oplus L_{15}^*$와 같고 $\Delta L_{15}[n]=0$이다. 단, $n=3,4,5,7,8,11,12,14,15,19,21,22,25,27,29,32$이다. 같은 방법으로 reg_{14}^R에 오류가 발생하면, $\Delta L_{15}[n]=0$이다. 단, $n=1,2,6,9,10,13,16,17,18,20,23,24,26,28,30,31$이다.

〔표 1〕 키 생성과정의 *CP* 테이블

좌측 28-비트	14 17 11 24 01 05 03 28 15 06 21 10 23 19 12 04 26 08 16 07 27 20 13 02
우측 28-비트	41 52 31 37 47 55 30 40 51 45 33 48 44 49 39 56 34 53 46 42 50 36 29 32

〔표 2〕 14라운드 F-함수의 SP 테이블 오류 확산
(ⓝ : 오류 발생 비트)

reg_{14}^{L}에 오류 발생	⑯ ⑦ 20 21 29 ⑫ 28 17 ① ⑮ 23 26 ⑤ 18 31 ⑩ ② ⑧ 24 ⑭ 32 27 ③ ⑨ 19 ⑬ 30 ⑥ 22 ⑪ ④ 25
reg_{14}^{R}에 오류 발생	16 07 ⑳ ㉑ ㉙ 12 ㉘ ⑰ 01 15 ㉓ ㉖ 05 ⑱ ㉛ 10 02 08 ㉔ 14 ㉜ ㉗ 03 09 ⑲ 13 ㉚ 06 ㉒ 11 04 ㉕

Proposition 1. reg_{14}^{L}(또는 reg_{14}^{R})에 오류가 주입되면, 공격자는 16라운드 F-함수의 S5-box, ..., S8-box(또는 S1-box, ..., S4-box)에 대한 출력 차분 값을 알 수 있다.

증명. reg_{14}^{L}에 오류가 주입되었다고 가정하면, 〔Fact 3〕에 의하여 $\Delta L_{15} = L_{15} \oplus L_{15}^{*}$의 $\Delta L_{15}[n]$에 오류가 주입될 수 있다. 단, $n = 1, 2, 6, 9, 10, 13, 16, 17, 18, 20, 23, 24, 26, 28, 30, 31$이다. 따라서 16라운드 F-함수의 출력 차분 $\Delta F_{16}(= L_{16} \oplus L_{16}^{*} \oplus L_{15} \oplus L_{15}^{*})$는 다음을 만족한다.

$$\Delta F_{16}[n] = \Delta L_{16}[n] (= L_{16} \oplus L_{16}^{*}) \quad (2)$$

이때, $n = 3, 4, 5, 7, 8, 11, 12, 14, 15, 19, 21, 22, 25, 27, 29, 32$이다.

그러므로 $SP^{-1}(L_{16} \oplus L_{16}^{*})$를 이용하여 S5-box, ..., S8-box의 출력 값을 얻을 수 있고, 동일한 방법으로 reg_{14}^{R}에 오류가 주입되었을 경우 S1-box, ..., S4-box의 출력 값을 얻을 수 있다.

3.3 일반적인 DES 키 생성과정에 대한 DFA 방법

〔3.2〕에서 키 생성과정의 오류가 발생한 레지스터의 위치에 따른 오류의 확산을 살펴보았다. 본 절에서는 이를 이용하여 일반적인 DES의 키 생성과정에 대한 DFA 방법에 대하여 서술한다. 먼저 랜덤하게 선택한 P에 대응하는 C를 얻는다. 〔3.1〕에서 제안하는 오류 모델에 부합하는 오류를 reg_i^{L} (또는 reg_i^{R}, $i = 14, 15$)에 주입하여 C_L^{*}(또는 C_R^{*})를 충분히 모으고, FP의 역 과정을 적용하여 16라운드의 옳은 출력 값 (L_{16}, R_{16})과 오류가 주입된 출력 값 (L_{16}^{*}, R_{16}^{*})을 얻는다. 16라운드 F-함수의 출력 차분은 $\Delta F_{16} = L_{16} \oplus L_{16}^{*} \oplus L_{15} \oplus L_{15}^{*}$이고, 16라운드 F-함수의 옳은 입력 값과 오류가 주입된 입력 값은 각각 R_{16}, R_{16}^{*}이다. 〔Proposition 1〕을 적용하면 16라운드의 모든 (또는 일부) S-box 출력 차분 값을 얻을 수 있다. 그리고 얻은 S-box 출력 차분 값에 대응하는 옳은 S-box 입력 값과 오류가 주입된 S-box 입력 값에 대한 후보쌍을 구한다. 한편 〔Fact 2〕에 의하여 16라운드 S5-box, ..., S8-box (또는 S1-box, ..., S4-box)의 입력 차분은 알고 있는 값이다. 이러한 정보를 이용하여 후보쌍의 차분과 입력 차분 값이 같지 않은 후보쌍을 제거 할 수 있다. 후보쌍이 1개로 결정될 때까지 이러한 과정을 반복하여 옳은 S-box 차분과 $EP(R_{16})$을 XOR연산하여 옳은 16라운드 키를 계산한다. 비슷한 방법으로 15라운드 키를 구하고, 키 생성과정의 역 과정을 적용하여 비밀키를 찾아 낼 수 있다.

3.4 15라운드 키 생성과정에 대한 DFA 방법

본 절에서는 앞 절의 내용을 바탕으로 15라운드 키 생성과정에 대한 DFA 방법을 상세히 서술한다.

■ Step 1. 옳은 암호문과 오류가 주입된 암호문 획득

랜덤하게 선택한 임의의 P (공격자가 모르는 값)에 대응하는 C를 얻는다. 그리고 제안하는 오류주입 모델에 따라 15라운드 키 생성과정의 reg_{15}^{L} (또는 reg_{15}^{R})에 오류를 주입하여 P에 대응하는 C_L^{*}(또는 C_R^{*})을 얻는다.

■ Step 2. S-box 출력 값의 후보쌍 리스트 생성

- Step 2-1. S-box 출력 값 리스트 $O_5, ..., O_8$ 생성

C와 C_L^{*}에서 16라운드의 옳은 출력 값 L_{16}, R_{16}과 오류가 주입된 출력 값 L_{16}^{*}, R_{16}^{*}을 얻는다. 이때, 16라운드 F-함수의 입력 값 쌍 (R_{16}, R_{16}^{*})과 F-함수의 출력 차분 $L_{16} \oplus L_{16}^{*}$는 알고 있는 값이므로 $SP^{-1}(L_{16} \oplus L_{16}^{*})$을 구할 수 있다. $SP_i^{-1}(i = 1, ..., 8)$을 SP 역 과정의 i번째 4-비트라 하면, $SP_i^{-1}(L_{16} \oplus L_{16}^{*})(i = 5, ..., 8)$은 각각 S5-box, ..., S8-box의 출력 차분이고 각 S-box는 옳은 출력 값과 오류가 주입된 출력 값에 대한 16개의 후보쌍이 존재한다. 각 S5-box, ..., S8-box에서

구한 16개의 후보쌍을 리스트 $O_5, ..., O_8$에 각각 저장한다.

- Step 2-2. S-box 출력 값 리스트 $O_1, ..., O_4$ 생성

C와 C_R^*에서 16라운드의 옳은 출력 값 L_{16}, R_{16}과 오류가 주입된 출력 값 L_{16}^*, R_{16}^*을 얻는다. Step 2-1과 같은 방법으로 S1-box, ..., S4-box의 출력 값 후보쌍을 리스트 $O_1, ..., O_4$에 저장한다.

■ Step 3. S-box 입력 값의 후보쌍 리스트 생성

- Step 3-1. S-box 입력 값 리스트 $I_5, ..., I_8$ 생성

DES의 S-box의 특성상 4개의 S-box 입력 값이 같은 출력 값을 갖는다. 따라서 리스트 O_i의 각 원소는 16개의 옳은 입력 값과 오류가 주입된 입력 값 후보쌍을 갖는다. 즉, i번째 S-box $(i=5, ..., 8)$는 $16\times16(=256)$개의 후보쌍을 갖는다. 각 S-box에서 생성된 S-box 입력 값 후보쌍을 리스트 $I_5, ..., I_8$에 저장한다.

- Step 3-2. S-box 입력 값 리스트 $I_1, ..., I_4$ 생성

Step 3-1과 같은 방법으로 S1-box, ..., S4-box의 입력 값 후보쌍을 저장한 리스트 $I_1, ..., I_4$를 생성한다.

■ Step 4. 16라운드 키 후보쌍 리스트 생성

- Step 4-1. 리스트 $< RK_{16,i} > (i=5, ..., 8)$ 생성

[Fact 2]에 의하여 S5-box, ..., S8-box의 입력 값 차분의 값은 알고 있는 값이다. 따라서 [Algorithm 1]과 같이 $EP(R_{16}) \oplus EP(R_{16}^*) \oplus RK_{16} \oplus RK_{16}^*$와 $I_i (i=5,...,8)$의 각각의 후보쌍의 차분을 비교하여 16라운드 키에 대한 후보쌍을 얻는다. 16라운드 키의 i번째 $(i=5,...,8)$ 6-비트에 대한 후보쌍을 리스트 $< RK_{16,5} >, ..., < RK_{16,8} >$에 각각 저장한다.

- Step 4-2. 리스트 $< RK_{16,i} > (i=1, ..., 4)$ 생성

Step 4-1과 같은 방법으로 [Algorithm 1]에 $I_i (i=1,...,4)$을 적용하여 16라운드 키 후보값 리스트 $< RK_{16,1} >, ..., < RK_{16,4} >$을 생성한다.

■ Step 5. 16라운드 키 획득

옳은 암호문과 오류가 주입된 암호문 쌍 (C, C_L^*)에 대하여 Step 2-1~Step 4-1과정을 적용하여 16라운드 키 후보쌍 리스트 $< RK_{16,i} > (i=5,6,7,8)$을 생성한다. 다른 오류에 대하여 같은 방법을 적용한 16라운드 키 후보쌍의 리스트 $< RK_{16,i} >^* (i=5,6,7,8)$를 생성하면 다음과 같이 나타낼 수 있다.

$$< RK_{16,i} > = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$
$$< RK_{16,i} >^* = \{(x_1^*, y_1^*), (x_2^*, y_2^*), \cdots, (x_m^*, y_m^*)\}$$

여기서 x_j, x_k^*는 옳은 라운드 키 후보값 $(j \in \{1, \cdots, n\}, k \in \{1, \cdots, m\})$, y_j, y_k^*는 오류가 주입된 라운드 키 후보값이다. $(j \in \{1, \cdots, n\},\ k \in \{1, \cdots, m\})$

각 x_j에 대하여 $x_j = x_{j'}^*$인 $x_{j'}^*$가 존재하면 $< RK_{16,i} >$와 $< RK_{16,i} >^*$에서 각각 (x_j, y_j)와 $(x_{j'}^*, y_{j'}^*)$를 남기고 그렇지 않으면 제거한다. 즉, 옳은 라운드 키 후보값을 비교하여 두 리스트에 모두 존재하는 옳은 라운드 키를 가지는 라운드 키 쌍만 남긴다. 또 다른 오류를 주입한 리스트를 생성하고 생성된 모든 리스트에 같은 방법으로 원소를 제거하여 각 리스트에 원소가 한 개 남을 때 까지 반복한다. 그 결과 하나의 옳은 라운드 키 원소를 포함하는 여러 개의 16라운드 키 쌍 리스트를 얻을 수 있다. 그 원소를 좌측 16라운드 키 쌍에 대한 리스트 $< RK_{16,i}^L > (i=5, \cdots, 8)$에 저장한다. 리스트 $< RK_{16,i}^L >$은 다음과 같이 나타낼 수 있다.

$$< RK_{16,i}^L > = \{(rk_i, rk_1^*), (rk_i, rk_2^*), ..., (rk_i, rk_n^*)\}$$

여기서 rk_i는 i번째 6-비트 옳은 16라운드 키이고 rk_j^*는 j번째 오류가 주입되어 생성된 6-비트 16라운드 키이며 n은 오류의 개수이다.

이러한 과정은 각 i에 대하여 병렬적으로 수행되고 모든 $i=5, \cdots, 8$에 대하여 $< RK_{16,i}^L >$의 옳은 라운드 키 원소가 하나로 결정될 때까지 반복한다.

같은 방법으로 여러 개의 옳은 암호문과 오류가 주입된 암호문 쌍 (C, C_R^*)에 대하여 우측 16라운드 키 쌍에 대한 리스트 $< RK_{16,i}^R > (i=1, \cdots, 4)$를 생성한다. 그 결과 옳은 16라운드 키 RK_{16}를 얻을 수 있고 주입한 오류에 따른 오류가 주입된 6-비트 16라운드 키를 얻을 수 있다.

■ Step 6. 15라운드 키와 비밀키 획득

Step 5에서 얻은 16라운드 키를 DES의 키 생성

Algorithm 1 : $< RK_{16,i} >$ $(i=1,...,8)$**생성 알고리즘**
Input : 16라운드 F-함수 입력 값 R_{16}, R_{16}^{*}, 16라운드 S-box 입력 값의 후보쌍 리스트 $I_i (i=1,...,8)$
Output : 리스트 $< RK_{16,i} >$ $(i=1,...,8)$

1: $a \leftarrow EP(R_{16}) \oplus EP(R_{16}^{*})$
2: for i from 0 to 7 do $b_i \leftarrow a[6i+1:6i+6]$
3: if fault is injected into $reg_j^L (j=14,15)$
  3.1 for i from 4 to 7 do
    3.1.1 for all possible value of $(x,y) \in I_i$ compute $x \oplus y \oplus b_i$
    3.1.2 if $x \oplus y \oplus b_i = 0$
      then add $(x \oplus EP(R_{16})[6i+1:6i+6]$, $y \oplus EP(R_{16}^{*})[6i+1:6i+6])$to the list $< RK_{16,i} >$
      else remove (x,y)
4: else fault is injected into $reg_j^R (j=14,15)$
  4.1 for i from 0 to 3 do
    4.1.1 for all possible value of $(x,y) \in I_i$ compute $x \oplus y \oplus b_i$
    4.1.2 if $x \oplus y \oplus b_i = 0$
      then add $(x \oplus EP(R_{16})[6i+1:6i+6], y \oplus EP(R_{16}^{*})[6i+1:6i+6])$ to the list $< RK_{16,i} >$
      else remove (x,y)
5: return $< RK_{16,i} >$ $(i=1,...,8)$

과정의 역 과정에 적용하면, 15라운드 키의 8비트 $(RK_{15}[n],\ n=12,13,14,17,30,37,39,46)$를 제외한 모든 비트를 알 수 있다. RK_{16}와 L_{16}를 알고 있으므로 $L_{15} = L_{16} \oplus F(R_{16}, RK_{16})$을 알 수 있다.($F(R_{16}, RK_{16})$은 16라운드 F-함수 출력 값) 그리고 $R_{16} \oplus R_{16}^{*}$은 15라운드 F-함수의 출력 차분이므로 15라운드의 입력 값과 출력 값의 차분을 알 수 있다. 따라서 $< RK_{16,i}^{L} >$와 $< RK_{16,i}^{R} >$에서 8비트를 제외한 15라운드 키 차분을 구하여 16라운드 키를 획득하는 방법과 비슷한 방법으로 15라운드 키를 얻을 수 있다. 그 결과 15라운드 키와 16라운드 키를 DES의 키 생성과정의 역 과정에 적용하여 비밀키를 구할 수 있다.

3.5 14라운드 키 생성과정에 대한 DFA 방법

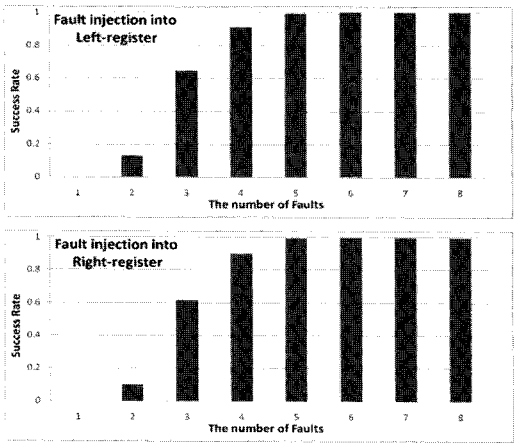
[3.4]절의 15라운드 키 생성과정에 대한 DFA 방법에서 16라운드 F-함수의 입력 값 쌍 (R_{16}, R_{16}^{*})과 S-box의 출력 차분 값을 이용하여 비밀키를 복구 할 수 있다. 이때 S-box의 출력 차분의 경우, 모든 S-box의 출력 차분이 아닌 일부의 값만을 이용하여 공격한다. 즉, 오류가 reg_{15}^{L}에 주입되면 S5-box, ..., S8-box의 출력 차분 값만 이용하고 reg_{15}^{R}에 주입되었을 경우 S1-box, ..., S4-box의 출력 차분 값만을 이용한다. 14라운드 키 생성과정에 오류를 주입할 경우, 16라운드 F-함수의 입력 값 쌍 (R_{16}, R_{16}^{*})은 알고 있고 [3.3. Proposition 1]에 의해 14라운드 키 생성과정에서 오류가 발생할 경우 필요한 S-box의 출력 차분을 구할 수 있다. 따라서 15라운드 키 생성과정에 대한 DFA 방법과 같은 방법으로 14라운드 키 생성과정에 오류를 주입하여 15라운드 키와 16라운드 키를 복구하고 비밀키를 획득할 수 있다.

IV. 실험결과

본 절에서는 DES의 키 생성과정에 대한 DFA 방법에 대한 실험결과를 제시한다. 본 실험은 Intel Pentium D 3.4GHz PC를 이용하여 C코드로 시뮬레이션을 실행 하였다. 14라운드 키 생성과정과 15라운드 키 생성과정에 오류가 주입된 DFA방법은 동일하므로 랜덤하게 선택한 P와 K에 대하여 랜덤 오류를 $reg_{14}^{L}, reg_{14}^{R}$에 주입하여 14라운드 키 생성과정에 대한 DFA 방법을 이용하여 비밀키를 획득하는 실험을 하였다. 총 100,000개의 오류를 주입하여 본 논문에서 제안하는 14라운드 키 생성과정에 오류를 주입한 DFA 방법을 이용하여 비밀키를 얻는데 필요한 오류의 개수는 [그림 4]와 같다. 실험결과에서 5개의 오류를 주입하면 99.52% (reg_i^{L}에 오류를 주입)와 99.54% (reg_i^{R}에 오류를 주입)의 확률로 비밀키를 얻

〔표 3〕 DES에 대한 DFA

DES에 대한 DFA			
DFA 방법	오류모델	공격 라운드	오류개수
〔6〕	1-비트	13~16	약 10개
〔12〕	1-비트, 1-바이트	9~12	20~10^8개
제안하는 방법	28-비트	14~16	약 10개



〔그림 4〕 DES의 14라운드 키 생성과정에 대한 DFA 실험결과

을 수 있었다. 따라서 본 실험결과는 5개의 오류를 각각의 레지스터에 오류를 주입하면 99%이상의 확률로 옳은 비밀키를 획득 할 수 있음을 보여준다.

V. 결 론

본 논문은 DES의 키 생성과정에 오류를 주입하여 DFA 방법을 처음으로 제안한 논문이다. 제안하는 DFA 방법은 마지막 3라운드 (14, 15, 16라운드)의 28-비트 레지스터에 랜덤 오류를 주입하여 비밀키를 찾는다. 제안하는 오류 모델에서 기존 DFA 방법이 1-비트 또는 1-바이트 오류를 주입하는 것에 비하여 제안하는 오류 모델은 28-비트 레지스터에 랜덤 오류를 사용함으로서 더욱 실제적으로 쉽고 효과적으로 오류를 주입하여 공격할 수 있다. 또한 제안하는 DFA 방법에서 라운드 키의 후보를 추측하고 후보를 줄여가는 방법을 사용하여 비밀키를 찾는 방법을 이용한다. 제안한 DFA 방법을 확인하기 위한 실험에서 10개 이하의 오류를 사용하여 99%이상의 확률로 공격을 성공할 수 있으므로 기존의 DFA 방법에 비하여 더 효율적으로 공격할 수 있다 (〔표 3〕 참조). 결론적으로 DES의 암호화 과정 (또는 복호화 과정)에 대한 DFA뿐만 아니라 키 생성과정에 대한 DFA 방법을 효율적으로 적용할 수 있음을 알 수 있다.

참고문헌

〔1〕 P. Kocher, "Timing Attacks on Implementations of Diffie Hellman, RSA, DSS, and Other Systems," *CRYPTO 1996, LNCS 1109*, pp. 104-113, Springer-Verlag, 1996.

〔2〕 P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *CRYPTO 1999, LNCS 1666*, pp. 388-397, Springer-Verlag, 1999.

〔3〕 Ç. K. Koç, D. Naccache, and C. Paar, "Electromagnetic Analysis:Concrete Results," *CHES 2001, LNCS 2162*, pp. 251-261, Springer-Verlag, 2001.

〔4〕 D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for fault," *EUROCRYPTO 1997, LNCS 1233*, pp. 37-51, Springer-Verlag, 1997.

〔5〕 T. C. May, M. H. Woods, "A New Physical Mechanism for Soft Errors in Dynamic Memories," *Proceedings 16 Int'l Reliability Physics Symposium*, pp. 33-40, Apr. 1978.

〔6〕 E. Biham, A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystem," *CRYPTO 1997, LNCS 1294*, pp. 513-525, Springer-Verlag, 1997.

〔7〕 H. Chen, W. Wu, and D. Feng, "Differential Fault Analysis on CLEFIA," *ICISC 2007, LNCS 4861*, pp. 284-295, Springer-Verlag, 2007.

〔8〕 J. Takahashi, T. Fukunaga, "Improved Differential Fault Analysis on CLEFIA," *FDTC 2007*, pp. 62-72, Aug. 2007.

〔9〕 L. Hemme, "A Differental Fault Attack Against Early Round of (Triple-)DES," *CHES 2004, LNCS 3156*, pp. 254-267, Springer-Verlag, 2004.

〔10〕 L. Wei, G. Dawu, and L. Juanru, "Differ-

ential Fualt Analysis on the ARIA Algorithm," *ELSEVIER Information Sciences 178 (2008)*, pp. 3727-3737, Oct. 2008.

[11] M.-L. Akkar, "Attaques et méthodes de protections de systémes crypto- graphiques embarqués," *PhD thesis, Université de Versailles Saint-Quentin*, Jan. 2004.

[12] M. Rivain, "Differential Fault Analysis on DES Middle Rounds," *CHES 2009, LNCS 5747*, pp. 457-469, Springer-Verlag, 2009.

[13] C. Giraud, "DFA on AES," *AES 2004, LNCS 3373*, pp. 27-41, Springer- Verlag, 2005.

[14] D. Peacham, B. Thomas, "A DFA attack against the AES key schedule," SiVenture White Paper 001 (26 Oct. 2006), http://www.siventure.com/pdfs/AES KeySchedule DFA whitepaper.pdf.

[15] J. Takahashi, T. Fukunaga, and K. Yamakoshi, "DFA mechanism on the AES key schedule," *FDTC 2007*, pp. 62-72, Sep. 2007.

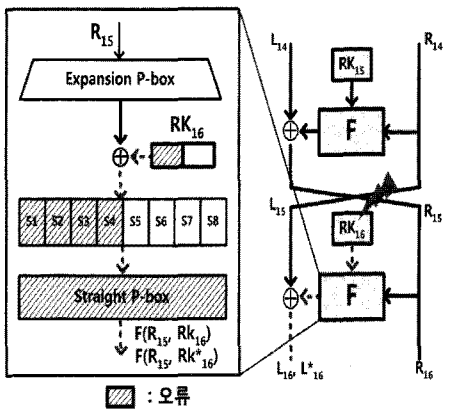
[16] C. H. Kim, J.-J. Quisquarter, "New Differential Fault Analysis on AES Key Schedule : Two Fault Are Enough," *CARDIS 2008, LNCS 5189*, pp. 48-60, Springer-Verlag, 2008.

[17] NIST, "Data Encryption Standard (DES)," *FIPS PUB 46-2*, Dec. 1993.

Appendix 1.

DES의 16라운드 키 생성과정에 대한 DFA

랜덤하게 선택한 P에 대응하는 C를 얻는다. 그리고 reg_{16}^{L}에 오류를 주입하여 암호문 C_L^*와 reg_{16}^{R}에 오류를 주입하여 암호문 C_R^*를 충분히 모은다. reg_{16}^{L}에 오류가 주입되었다고 가정하면, [그림 5]와 같이 오류가 확산되고 F-함수 입력 값 R_{16}과 F-함수 출력 차분



[그림 5] 16라운드의 오류확산

$L_{16} \oplus L_{16}^{*}$은 알고 있는 값이다. [Ⅲ-4]의 Step 2과 Step 3을 적용하면, i번째 S-box($i=1,2,3,4$)의 옳은 입력 값 후보 256개를 구할 수 있다. x_i를 i번째 S-box의 입력 값이라 하고 $EP_i(R_{16})$과 $RK_{16,i}$를 $EP(R_{16})$과 RK_{16}의 6-비트로 나눈 i번째 블록이라 하면, 다음을 만족한다.

$$x_i \oplus EP_i(R_{16}) = RK_{16,i} \ (i=1,2,3,4) \tag{3}$$

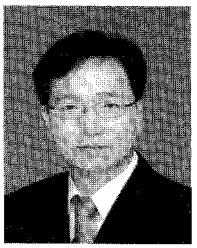
따라서 하나의 오류에 대하여 $RK_{16,i}$의 256개 후보를 구할 수 있고 이 후보를 리스트$< RK_{16,i} > (i=1,2,3,4)$에 저장한다. 이와 같은 과정을 반복하여 생성한 16라운드 후보 중 중복 되지 않는 라운드 키 후보를 제거하고 하나의 좌측 라운드 키 (24-비트)가 남을 때 까지 반복한다.

같은 방법으로 reg_{16}^{R}에 오류가 주입하여 $< RK_{16,i} > (i=5,6,7,8)$를 생성하고 우측 라운드 키 (24-비트)를 구하여 48-비트 16라운드 키를 얻을 수 있다. 하지만 14라운드와 15라운드 키 생성과정에 대한 DFA 방법과 달리 16라운드 키 생성과정에 대한 DFA 방법은 15라운드에 오류가 주입되지 않는다. 즉, 15라운드에 대한 비교 대상이 존재하지 않는다. 따라서 16라운드 키로 찾을 수 없는 15라운드의 8-비트에 대한 추측을 통해 전수조사를 해야 한다. 즉, $2^8 = 256$개의 전수조사를 통해 15라운드의 8-비트를 찾고 키 생성과정의 역 과정을 통해 비밀키를 얻을 수 있다.

〈著者紹介〉

소 현 동 (Hyun-Dong So) 학생회원
2007년 8월: 부경대학교 응용수학과 학사
2009년 2월~현재: 고려대학교 정보경영공학전문대학원 석사과정
〈관심분야〉 부채널 분석, 공개키 암호

김 성 경 (Sung-Kyoung Kim) 학생회원
2005년 2월: 동의대학교 수학과 학사
2007년 8월: 고려대학교 정보경영공학전문대학원 석사
2007년 8월~현재: 고려대학교 정보경영공학전문대학원 박사과정
〈관심분야〉 부채널 분석, 공개키 암호, 암호칩 설계 기술



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
2003년 2월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U.Leuven, ESAT/SCD-COSIC 박사후연구원
2005년 3월~2008년 8월: 고려대학교 정보보호대학원 조교수
2008년 9월~현재: 고려대학교 정보경영공학전문대학원 부교수
〈관심분야〉 대칭키 암호의 분석 및 설계, 컴퓨터 포렌식

강 은 숙 (Eun-Sook Kang) 정회원
1981년 2월: 고려대학교 수학과 학사
1983년 2월: 고려대학교 수학과 석사
1987년 2월: 고려대학교 수학과 박사
1988년 3월~1992년 2월: 고려대학교 과학기술대학 정보수학과 조교수
1993년 3월~1997년 2월: 고려대학교 과학기술대학 정보수학과 부교수
1998년 3월~현재 :고려대학교 과학기술대학 정보수학과 교수
〈관심분야〉 정보보호