Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.6, Dec. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.6.1263

함수 생략 오류를 이용하는 AES에 대한 신규 차분 오류 공격*

김 주 환,[1†] 이 종 혁,[2] 한 동 국[3‡]
[1]국민대학교 수학과 (학생), [2,3]국민대학교 금융정보보안학과 (대학원생, 교수)

Novel Differential Fault Attack Using Function-Skipping on AES*

Ju-Hwan Kim,[1†] JongHyeok Lee,[2] Dong-Guk Han[3‡]
[1]Department of Mathematics, Kookmin University (Undergraduate),
[2,3]Department of Financial Information Security, Kookmin University
(Graduate student, Professor)

요 약

차분 오류 공격은 암호화 장비에 인위적인 오작동을 유도하여 발생한 오류 암호문과 정상 암호문의 차분을 이용해 비밀키를 분석하는 기법이다. 일반적으로 차분 오류 공격은 암호 알고리즘의 중간값을 변조시켰을 때의 결과를 이용한다. 반면, 우리는 함수 전체를 생략시켰을 때의 결과를 이용하는 차분 오류 공격을 제안한다. 제안한 기법은 단일 오류 주입 암호문으로 수 초 이내에 전체 비밀키 복구가 가능한 매우 낮은 공격 복잡도를 갖는다. 또한, 우리는 오류가 함수 생략을 유도했는지 검증하는 방안을 제시함으로써 공격자 가정을 현실적으로 낮췄다. 제안한 기법을 실험적으로 검증하기 위해 Riscure 사의 Piñata 보드에 대한 오류 주입 공격을 수행한 결과 함수 생략으로 유도되지 않은 모든 오류 암호문을 거를 수 있었으며, 수 초 내에 비밀키 복구에 성공했다.

ABSTRACT

The differential fault attacks (DFA) are cryptoanalysis methods that reveal the secret key utilizing differences between the normal and faulty ciphertexts, which occurred when artificial faults are injected into an encryption device. The conventional DFA methods use faults to falsify intermediate values. Meanwhile, we propose the novel DFA method that uses a fault to skip a function. The proposed method has a very low attack complexity that reveals the secret key using one fault injected ciphertext within seconds. Also, we proposed a method that filters out ciphertexts where the injected faults did not occur the function-skipping. It makes our method realistic. To demonstrate the proposed method, we performed fault injection on the Riscure's Piñata board. As a result, the proposed method can filter out and reveal the secret key within seconds on a real device.

Keywords: Side-Channel Attack, Differential Fault Attack, AES, Instruction Skipping, Function Skipping

I. 서 론

오류 주입 공격(fault injection attack)이란 암호화 장비에 인위적인 오류를 주입함으로써 발생한 오작동의 결과를 이용해 비밀 정보를 분석하는 방법이다. 공격자는 장비에 강력한 전자파나 레이저를 주

Received(10. 12. 2020), Accepted(11. 10. 2020)
* 이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00520,SCR-Friendly 대칭키 암호 및 응용모드 개발).
† 주저자, zzzz2605@kookmin.ac.kr
‡ 교신저자, christa@kookmin.ac.kr(Corresponding author)

입해 메모리에 저장된 값을 변조시키거나[1] 클락 신호를 변조시키는[2] 등의 오작동을 유도하고, 이때의 결과를 이용해 비밀키를 찾는다. 차분 오류 공격(Differential Fault Attack, DFA)은 오류 주입 공격 중 하나로 오류 암호문과 정상 암호문의 차분을 이용해 비밀키를 분석하는 방법이다[3].

기존의 차분 오류 공격들은 주로 오류로 특정 중간값이 변조되면, 이로부터 파생되는 잘못된 중간값들과 관련된 부분 비밀키를 차분 분석으로 복구할 수 있다는 사실을 기반으로 한다. 이 방법으로 전체 비밀키를 복구하려면 오류를 여러 번 주입해 다양한 중간값을 변조시키거나 오류를 전단부에 주입해 확산시켜야 한다. 따라서 기존 기법은 오류 주입 횟수와 비밀키 추정 횟수가 상충관계에 있으며, 일반적으로 높은 공격 복잡도를 갖는다.

한편, 오류로 암호 알고리즘의 구조를 변경시킬 수 있음이 알려지면서 특정 명령어를 생략시키거나[4, 5], 반복문을 조기에 종료시켜 암호 알고리즘의 라운드를 축소하는[6] 분석 방안이 제안되었다. 그러나, 라운드를 축소시키는 공격은 표준[7]이 권장하지 않는 취약한 구현을 가정으로 하거나[8], 많은 후보 키를 검증해야 하므로 높은 공격 복잡도를 갖는다.

우리는 오류로 함수를 호출하는 명령어를 생략시키거나 함수 내부 반복문을 생략시킨다면, 단일 연산이 아니라 함수 전체를 생략시킬 수 있다는 사실에 주목했다. 함수 생략은 모든 바이트의 중간값에 영향을 줄 수 있으므로 단일 오류 주입 암호문으로 비밀키 복구가 가능한 장점이 있다. 마지막 라운드의 함수를 생략시키면, 비밀키 복구를 위한 연산 복잡도를 낮출 수 있다. 따라서 함수 생략 기반의 공격은 기존 방법보다 월등히 낮은 공격 복잡도를 갖는다. 한편, 우리는 암호 알고리즘 AES(Advanced Encryption Standard)[7]에 대하여 오류 암호문 중 마지막 SubBytes 함수 생략으로 유도되지 않은 오류 암호문을 거르는 방법을 설립했다. 따라서 제안한 방법은 정확한 시점이나 위치에 오류를 주입하지 않아도 되는 현실적인 공격자 가정에서 적용할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 명령어 생략을 이용하는 기존 차분 오류 공격 연구를 소개한다. 3장에서는 SubBytes 함수가 생략되었을 때의 오류 암호문과 정상 암호문 쌍을 이용해 비밀키를 찾는 방안과 SubBytes 생략으로 유도되지 않은 오류 암호문을 거르는 방안을 제안한다. 4장에서는 어셈블리 코드를 분석함으로써 SubBytes 함수를 생략시키는 두 가지 원인을 제시한다. 5장에서는 Riscure 사의 Piñata 보드에 대한 분석을 수행하여 제안한 방안을 실험적으로 검증한다. 마지막으로 6장에서는 결론을 맺고 향후 연구를 제시한다.

II. 명령어 생략을 이용하는 차분 오류 공격

전통적인 차분 오류 공격은 암호화 수행 중에 오류를 주입해 중간값을 변조시키거나[9], 키 확장 수행 중에 오류를 주입해 라운드 키를 변조시키는[10] 오류 모델을 주로 사용하였다. 즉, 암호 알고리즘의 구조는 정상적이나, 계산되는 값을 변조하여 비밀키를 복구하는 방안이 주로 제안되었다.

최근 오류로 명령어를 생략시킴으로써 암호 알고리즘의 구조를 취약하게 변경했을 때의 암호문을 이용해 비밀키를 복구하는 방안이 활발히 제안되고 있다. Schmidt 등은 오류로 프로그램 카운터를 변경시킴으로써 공개키 암호 RSA의 제곱 연산이 생략됐을 때의 결과를 이용해 비밀키를 찾는 방안을 제안했다[4]. Kumar 등은 스트림 암호 ChaCha의 중간 연산인 덧셈, 비트 회전 등의 연산이 생략되었을 때의 결과를 이용하는 차분 오류 공격을 제안했다[5]. Park 등은 오류로 AES의 반복문을 조기에 종료시킴으로써 AES의 라운드가 축소되었을 때의 결과를 이용하는 차분 오류 공격을 제안했다[6]. 명령어 생략을 이용하는 블록 암호에 대한 차분 오류 공격으로는 라운드를 축소하는 방안이 주로 제안되었다. 그러나 이 방법은 축소된 라운드를 거치며 발생한 혼돈(diffusion) 계층과 확산(confusion) 계층 때문에 많은 비밀키를 추측해야 한다.

Breier 등은 오류로 AES 마지막 라운드의 AddRoundKey 계층을 생략시킴으로써 하나의 오류 및 정상 암호문으로 비밀키를 복구하는 방안을 제안했다[11]. 그러나, 이 방법을 적용하기 위해서는 사전에 원하는 오류를 유도하기 위한 파라미터를 찾는 과정이 필요하며, 정확한 시점에 오류를 주입하기 위해 암호 알고리즘 내부에 인위적인 트리거를 발생시켜야 하므로 현실적으로 적용하기 어렵다.

III. 함수 생략 오류를 이용하는 차분 오류 공격

본 장에서는 오류로 AES 마지막 라운드의 SubBytes 함수가 생략되었을 때의 결과를 이용해 비밀키를 복구하는 방안과 SubBytes 함수 생략으로 유도

Table 1. Notations

Notation	Description
C	Normal ciphertext
C^*	Faulty ciphertext
K	10^{th} round key
I	10^{th} round input
SBox	AES S-Box table
$\oplus$	Exclusive-or (XOR)
$\|S\|$	Cardinality of a set S

되지 않은 오류 암호문을 거르는 방안을 제안한다.

본 논문에서 사용하는 기호는 Table 1.과 같다.

3.1 차분 오류 공격 방안

AES의 마지막 라운드는 SubBytes, ShiftRows, AddRoundKey 세 함수로 이루어져 있다. SubBytes와 AddRoundKey는 바이트 단위 연산을 수행하고, ShiftRows는 바이트의 순서만 바꾼다. 따라서 논의의 편의를 위해 ShiftRows는 무시한다. 마지막 라운드의 모든 함수는 바이트 단위 연산을 수행하므로 바이트 단위 비밀키 추정을 할 수 있다. 즉, 분할 정복 기법(divide-and-conquer)을 적용하여 낮은 공격 복잡도로 비밀키를 복구할 수 있다.

10 라운드 입력 I에 대해 정상 암호문 C와 오류로 SubBytes 함수가 생략되었을 때의 암호문 C^*는 수식 (1)과 같다.

$$\begin{cases} C &= \mathrm{SBox}(I) \oplus K \\ C^* &= I \oplus K \end{cases} \tag{1}$$

따라서 정상 암호문과 오류 암호문의 차분은 수식 (2)와 같다.

$$C \oplus C^* = \mathrm{SBox}(I) \oplus I \tag{2}$$

공격자는 암호문 쌍의 차분에서 10라운드 입력 I로 가능한 값의 집합을 얻을 수 있다. 즉, 수식 (3)과 같이 집합 S_i를 정의했을 때, 오류 암호문의 차분 $C \oplus C^*$로부터 후보 중간값 I의 집합 $S_{C \oplus C^*}$를 얻을 수 있다.

$$S_i := \{I : \mathrm{SBox}(I) \oplus I = i\},\ i \in [0, 255] \tag{3}$$

한편, 수식 (1)에서 $C^* = I \oplus K$이므로 수식 (4)와 같이 후보 중간값 I로부터 후보 키 K를 계산할 수 있다.

$$K = C^* \oplus I \tag{4}$$

후보 키 중 올바른 키를 찾기 위해 각 후보 키에 대해 암호화를 수행하여 암호화 결과가 올바른지 판정한다.

AES의 마지막 라운드는 바이트 단위 연산만 수행하므로, 수식 (2)에서 가능한 I의 수는 일반적으로 매우 작다. 따라서 제안한 공격은 평균적으로 수 초 이내에 비밀키를 복구할 수 있다. 다음 절에서 제안한 공격의 복잡도를 분석한다.

3.2 공격 복잡도 분석

제안한 공격의 복잡도는 수식 (2)를 만족하는 I의 수, 즉 집합 $S_{C \oplus C^*}$의 원소의 수에 의해 결정된다. $|S_i|$의 분포를 확인하기 위해 집합 N_x를 수식 (5)와 같이 정의한다. 255 이하의 모든 i에 대해 $|S_i|$를 계산한 결과 $|S_i|$의 최댓값이 4임을 확인했다.

$$N_x := \{i : |S_i| = x\},\ x \in [0, 4] \tag{5}$$

x에 따른 $|N_x|$의 값이 Fig. 1과 같다. 예를 들어 $|N_1| = 91$은 $C \oplus C^*$로부터 I를 유일하게 결정할 수 있는 경우가 91가지임을 의미한다.

바이트별 후보 키의 수의 평균은 수식 (6)에서 약 1.57055이므로, 전체 후보 키의 수의 평균은 약 $1.57055^{16} \approx 1370$이다. 이는 하나의 오류 및 정상 암

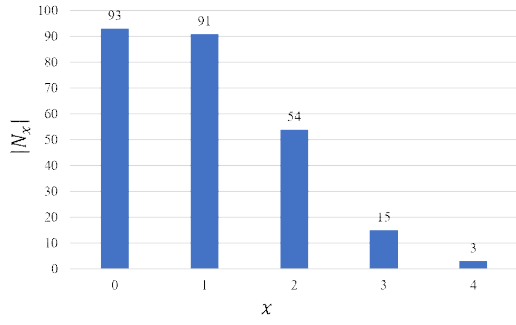


Fig. 1. The cardinality of N_x

호문 쌍만으로 후보 키를 평균 1370개로 줄일 수 있음을 의미하므로 제안한 공격 기법은 하나의 암호문 쌍만으로도 수 초 이내에 비밀키를 복구할 수 있다.

$$\frac{\sum_{x=1}^{3}(|N_x|\times x)}{\sum_{x=1}^{3}|N_x|}\approx 1.57055 \quad (6)$$

3.3 SubBytes 생략 검증 방안

대부분의 오류 주입 공격은 공격자가 원하는 위치나 시점에 오작동을 유도할 수 있다는 가정을 기반으로 한다. 공격자가 원하는 오작동을 유발하기 위해서는 프로파일링을 수행하고, 대상 장비를 직접 조작하여 암호 내부에 인위적인 트리거를 발생시킬 수 있어야 한다. 따라서 이러한 공격은 현실적으로 적용하기 어렵다. 우리는 이러한 공격자 가정을 낮추기 위해 정상 및 오류 암호문 쌍을 이용해 오류가 SubBytes 함수 생략을 유도했는지 판정하는 방안을 제안한다. 오류 암호문 중 원하는 오류로 유도된 것만 선택하는 방법이 있다면, 대략적인 위치와 시점에 오류를 주입하고 원하는 결과만을 취해 공격을 수행할 수 있으므로 공격자 가정을 현실적으로 낮출 수 있다.

제안한 검증 방안은 S_i 중 공집합이 있음을 이용한다. 즉, $\mathrm{SBox}(I)\oplus I$의 값으로 불가능한 것이 존재함을 이용한다. 수식 (7)과 같이 집합 P를 정의했을 때, 집합 P는 집합 $[0,\ 255]$의 진부분집합이다.

$$P := \{\mathrm{SBox}(I)\oplus I : I\in[0,\ 255]\} = \bigcup_{x=1}^{3} N_x \quad (7)$$

수식 (2)의 관계에서 암호문의 차분 $C\oplus C^*$ 16바이트가 모두 집합 P의 원소라면 오류 암호문이 SubBytes 생략으로 유도되었다고 판정할 수 있다.

제안한 방안은 높은 확률로 SubBytes로 생략되지 않은 오류 암호문을 거를 수 있다. Fig. 1에서 $|P|=163$이므로, 오류 암호문의 차분 16바이트가 모두 집합 P의 원소일 때, SubBytes 생략이 유도되었을 확률은 $1-\left(\frac{163}{256}\right)^{16}\approx 0.99927$이다. 즉, 제안한 검증 방안을 통과하면, 99.927%의 확률로 SubBytes 생략이 유도되었다 판정할 수 있다.

IV. 함수 생략 원인 분석

본 장에서는 Riscure 사에서 C 언어로 구현되고 CoIDE로 컴파일된 AES-128 암호 알고리즘을 분석해 실제 장비에서 함수 생략이 발생하는 원인을 해석한다.

Riscure의 SubBytes 함수 구현은 Fig. 2와 같다. AES의 상태배열은 전역 변수 2차원 배열로 선언되어 있으며, S-Box 참조는 별도의 함수 getSBoxValue를 호출하여 처리한다.

위의 구현에서 SubBytes 함수가 생략되는 원인을 확인하기 위해 어셈블리 코드를 분석한 결과 다음 세 가지 원인을 파악했다.

첫 번째 원인은 AES 함수에서 SubBytes 함수를 호출하는 명령어가 생략되는 것이다. Fig. 3은 AES 함수에 대한 어셈블리 코드 중 마지막 라운드 부분을 나타낸 것이다. 특정 함수를 호출하기 위해서는 해당 함수의 명령어가 저장된 곳으로 분기하는 bl(branch with link) 명령어가 수행되어야 한다. 만약 오류로 인해 이 명령어가 NOP(no operation)로 바뀌거나, PC(program counter)가 변경되어 해당 명령어가 수행되지 않는다면 SubBytes 함수가 생략될 수 있다.

두 번째와 세 번째 경우는 SubBytes 함수 내부

```
01 static uint8_t getSBoxValue(uint8_t num)
02 {
03     return sbox[num];
04 }
05
```

```
01 static void SubBytes()
02 {
03     uint8_t i, j;
04     for (i = 0; i < 4; ++i)
05     {
06         for (j = 0; j < 4; ++j)
07         {
08             state[i][j] = getSBoxValue(state[i][j]);
09         }
10     }
11 }
```

Fig. 2. The Riscure's SubBytes implementation in C

```
⋮
01 bl      80086ec <AddRoundKey>
02 bl      800875c <SubBytes>
03 bl      80087b8 <ShiftRows>
04 bl      80086ec <AddRoundKey>
⋮
```

Fig. 3. Last round of the AES assembly code

의 반복문이 수행되지 않고 종료되는 경우이다. Fig. 2의 SubBytes 함수 중 04-10번 줄의 반복문이 수행되지 않고 종료된다면, SubBytes 함수 생략을 유도할 수 있다.

Fig. 4는 SubBytes 함수에 대한 어셈블리 코드 중 첫 번째 반복문 관련 부분을 나타낸 것이다. 01, 02번 줄은 반복문의 변수(C 코드의 변수 i)를 0으로 초기화하는 부분이며, 03번 줄은 조건을 확인하기 위한 명령어가 저장된 위치인 09번 줄로 분기하는 부분이다. 05-08번 줄은 반복문의 변수를 1 증가하는 부분이다. 마지막으로 09번 줄에서 반복문의 변수와 3을 비교하고, 10번 줄에서 비교 결과 변수가 3 이하이면 반복문에서 수행할 명령이 저장된 04번 줄로 분기, 3보다 크면 11번 줄 명령을 수행함으로써 반복문 종료 조건을 판정한다.

이상의 명령어에서 반복문 내부 명령어가 수행되지 않고 종료되는 경우는 다음 두 가지이다. 첫 번째 경우는 오류로 인해 반복문의 변수가 큰 값으로 초기화되는 경우이다. 변수를 초기화하는 01번 줄 수행 중에 오류로 레지스터 r3의 값이 3보다 큰 값으로 변경되면, 반복문의 조건을 만족하지 못하므로 내부 명령어가 수행되지 않고 반복문이 종료된다. 두 번째 경우는 오류로 10번 줄의 명령어가 생략되는 경우이다. 09번 줄의 비교 결과에서 r3이 3보다 작더라도 10번 줄의 명령어가 생략되면, 반복문 내부 명령어가 저장된 04번 줄로 분기할 수 없다. 이상의 두 가지 경우에서 반복문 내부 명령어가 수행되지 못하면, SubBytes 변환을 수행하지 못하므로 함수 전체를 생략시킬 수 있다.

우리는 실제 장비에 대한 오류 주입 결과

```
                    ⋮
01  movs    r3, #0
02  strb    r3, [r7, #7]
03  b.n     80087a6 <SubBytes+0x4a>
04  movs    r3, #0
                    ⋮
05  ldrb    r3, [r7, #7]
06  adds    r3, #1
07  strb    r3, [r7, #7]
08  ldrb    r3, [r7, #7]
09  cmp     r3, #3
10  bls.n   8008768 <SubBytes+0xc>
                    ⋮
```

Fig. 4. First loop of the SubBytes assembly code

SubBytes 생략이 유도되는 시점이 세 구간 존재함을 보임으로써 어셈블리 분석 결과를 실험적으로 입증한다.

V. 실험 결과

실험을 위해 Riscure 사의 Piñata 보드에서 소프트웨어로 구현된 AES-128 암호 알고리즘이 동작할 때 전자파 오류 주입을 수행했다. 오류 주입을 위해 EM-FI Transient Probe, EM Probe Station, Spider 장비를 이용했으며, EM Probe Tip은 지름이 1.5mm이고 극성은 (+)인 것을 사용했다. 암호화 수행 시 전자파를 확인하기 위해 Langer사의 EM Probe와 Oscilloscope를 이용했다. Fig. 5는 이상의 실험 환경을 나타낸 것이다.

제안한 공격 기법을 검증하기 위해 두 가지 실험을 수행했다. 첫 번째는 4장의 분석 결과를 검증하기 위해 미리 적합한 오류 주입 위치와 시점을 프로파일링하고, 정확한 시점에 오류를 주입하기 위해 암호화 함수 내부에 인위적인 트리거를 발생시키는 강력한 공격자 가정에서의 실험이다. 두 번째는 제안한 SubBytes 생략 검증 기법을 적용하면 현실적인 공격자 가정에서 비밀키 복구가 가능함을 보이기 위해 암호화 함수 외부에 트리거를 발생시키고, 전자파 파형을 기반으로 추정한 SubBytes 호출 시점에 대해 오류를 주입하는 현실적인 공격자 가정에서의 실험이다.

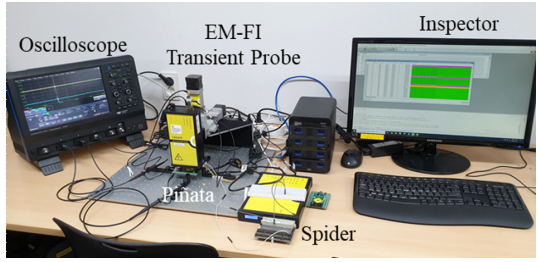


Fig. 5. Experimental environment

5.1 강력한 공격자 가정에서의 오류 주입

본 절에서는 넓은 시점에 오류를 주입하고, SubBytes 생략이 발생하는 세 구간이 존재함을 보임으로써 4장의 분석 결과를 실험적으로 입증한다. 정확한 시점에 오류를 주입하기 위해 암호 알고리즘 내부에 인위적인 트리거를 발생시킬 수 있다는 가정에서의

실험을 수행했다. 10라운드 SubBytes 함수를 생략시키는 것이 목표이므로 하드웨어적 지연을 고려하여 9라운드 AddRoundKey 시작 지점에 트리거를 설정하였다.

SubBytes 함수 생략을 유도하기 위하여 함수 호출 직전부터 첫 번째 바이트 연산까지의 시점인 트리거로부터 4700-5100ns 지점에 오류를 194425번 주입했다. 그 결과 874개의 오류 암호문이 도출되었으며, 이 중 SubBytes 생략으로 유도된 오류 암호문은 37개였다. 이 중 SubBytes 생략이 발생한 시점을 도식화한 것이 Fig. 6과 같다. 그림에서 SubBytes 생략이 발생하는 시점을 4732-4736ns, 4780-4820ns, 4852-4876ns 세 구간으로 나눌 수 있다. 각 구간은 4장의 어셈블리어 분석에서 각각 SubBytes 함수 호출 (Fig. 3.의 02번 줄), 반복문 변수 초기화 (Fig. 4의 01번 줄), 반복문 종료 조건 검사 후 분기 (Fig. 4.의 10번 줄) 수행 부분이라 추정할 수 있다. 이는 4장의 원인 분석 결과가 정확함을 시사한다. 한편, 각 구간의 길이가 넓은 이유는 하드웨어적 잡음으로 인해 정확한 시점에 오류를 주입할 수 없기 때문이다.

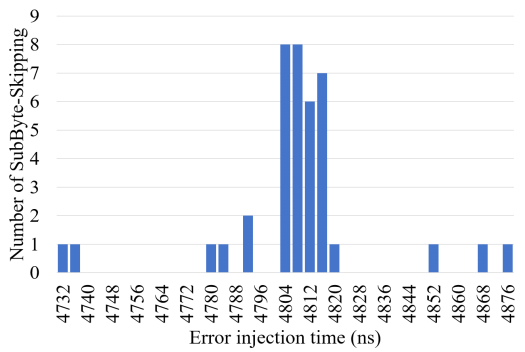


Fig. 6. The number of SubBytes-skipping according to the fault injection time

5.2 현실적인 공격자 가정에서의 오류 주입

본 절에서는 SubBytes 생략 검증 기법을 이용하면 정확한 위치나 시점에 오류를 주입하지 못하는 현실적인 공격자 가정에서 제안한 공격 기법이 적용 가능함을 보인다. 프로파일링이 선행되지 않았다는 가정에서 실험하기 위해 전자파 파형을 이용해 대략적인 오류 주입 시점을 판정하고 넓은 범위에 오류를 주입한다.

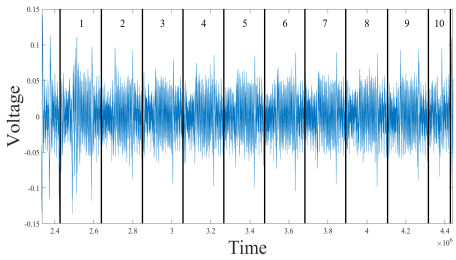


Fig. 7. Electromagnetic trace of the AES

장비에서 AES가 동작할 때의 전자파 파형이 Fig. 7과 같다. 그림에서 9개의 동일한 패턴이 나타나고, 마지막에 짧은 패턴이 나타나므로 10라운드 수행 시점을 판정할 수 있다.

AES에서 아홉번만 동작하는 함수는 MixColumns가 유일하므로 Fig. 7에서 MixColumns 함수가 동작할 때의 전자파 개형을 판단할 수 있다. 이를 이용해 9라운드 AddRoundKey부터 10라운드 AddRoundKey까지의 파형을 판단할 수 있다. Fig. 8은 이때의 파형을 나타낸 것이다. 녹색으로 표현한 것처럼 전단부와 후단부의 동일한 패턴이 나타나므로 AddRoundKey 함수 수행 위치를 판정할 수 있다. 따라서 대략적인 SubBytes 함수 호출 위치는 붉은 직선으로 나타낸 시점임을 유추할 수 있다.

SubBytes 함수 생략을 유도하기 위해 유추한 함수 호출 시점을 중심으로 좌우 500ns 범위에 오류를 주입했으며, 암호화 장비의 칩을 15×15 등분하여 넓은 범위에 오류를 주입했다. 그 결과 전체 64000번 오류 주입으로 27개의 오류 암호문을 얻었으며, 이 중 SubBytes 생략으로 유도된 오류 암호문은 1개였다. 제안한 검증 기법을 사용하면 26개의 오류 암호문이 SubBytes 생략으로 유도되지 않았음을 판정할 수 있었으며, 남은 1개의 오류 암호문쌍으로 분

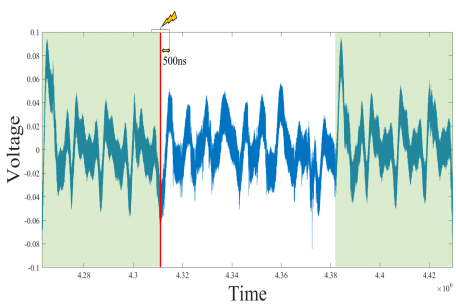


Fig. 8. Electromagnetic trace of part of 9th-10th round AES

석한 결과 후보 키를 2304개로 줄일 수 있었다. 따라서 제안한 기법을 이용하면 현실적인 공격자 가정에서 낮은 공격 복잡도로 비밀키 분석을 수행할 수 있다.

Ⅵ. 결 론

본 논문에서는 기존 기법과 달리 SubBytes 함수 생략 오류를 이용하여 비밀키를 분석하는 방안과 SubBytes 생략이 발생했는지 검증하는 방안을 제안하였다. 제안한 공격기법은 하나의 정상 및 오류 암호문 쌍만으로 후보 키를 평균 1370개로 줄일 수 있으며, 검증 기법을 사용해 현실적인 공격자 가정에서 제안한 방법을 적용할 수 있다.

본 논문에서는 AES에 대한 키 복구 논리만을 제시하였으나, 제안한 방법은 비선형 함수가 존재하는 대부분의 블록 암호에 적용 가능할 것으로 예상한다. 따라서 향후 다양한 암호 알고리즘에 대한 키 복구 방안을 연구할 예정이다.

References

[1] D. Boneh, R. DeMillo, and R. Lipton., "On the importance of checking cryptographic protocols for faults," Advances in Cryptology, EUROCRYPT'97, LNCS 1233, pp. 37-51, May. 1997.

[2] O. Kommerling and M. Kuhn, "Design principles for tamper-resistant smartcard processors," Proceedings of the 1st Workshop on Smartcard Technology, pp. 9 - 20, May. 1999.

[3] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," Advances in Cryptology, CRYPTO'97, LNCS 1294, pp. 513-525, Aug, 1997.

[4] J. Schmidt and C. Herbst, "A practical fault attack on square and multiply," Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 53-58, Aug. 2008.

[5] D. Kumar, S. Patranabis, J. Breier, D. Mukhopadhyay, S. Bhasin, A. Chattopadhyay, and A. Baksi "A practical fault attack on ARX-like ciphers with a case study on ChaCha20," Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 33-40, Sep. 2017.

[6] J. Park, S. Moon, D. Choi, Y. Kang, and J. Ha, "Differential fault analysis for round-reduced AES by fault injection," ETRI Journal, 33(3), pp. 434-442, Jun. 2011.

[7] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray Jr., "Advanced Encryption Standard (AES)," NIST FIPS 197, Nov. 2001.

[8] H. Choukri and M. Tunstall, "Round reduction using faults," Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 13-24, Sep. 2005.

[9] J. Blömer and JP. Seifert, "Fault based cryptanalysis of the Advanced Encryption Standard (AES)," financial cryptography, LNCS 2742, pp. 162-181, Jan. 2003.

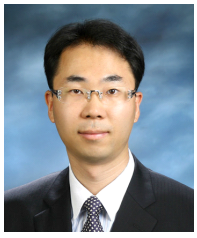
[10] C. Chen and S. Yen, "Differential fault analysis on AES key schedule and Some countermeasures," Australasian Conference on Information Security and Privacy, LNCS 2727. pp. 118-129, Jul. 2003.

[11] J. Breier, D. Jap, and C. Chen, "Laser profiling for the back-side fault attacks: with a practical laser skip Instruction attack on AES," Worksohp on Cyber-Physical System Security, pp. 99-103, Apr. 2015.

〈저자소개〉

김 주 환 (Ju-Hwan Kim) 학생회원
2016년 3월~현재: 국민대학교 수학과 학사과정
〈관심분야〉 부채널 분석 및 대응법 설계, 딥러닝, 오류 주입 공격

이 종 혁 (JongHyeok Lee) 학생회원
2017년 2월: 국민대학교 수학과 학사
2017년 3월~현재: 국민대학교 금융정보보안학과 석박사통합과정
〈관심분야〉 부채널 분석 및 대응법 설계, 대칭키 암호 알고리즘, 스마트 카드 보안, 오류 주입 공격



한 동 국 (Dong-Guk Han) 종신회원
1999년 2월: 고려대학교 수학과 학사
2002년 2월: 고려대학교 수학과 이학석사
2005년 2월: 고려대학교 정보보호대학원 공학박사
2004년 4월~2005년 4월: 일본 Kyushu Univ., 방문연구원
2005년 4월~2006년 4월: 일본 Future Univ.-Hakodate, Post.Doc.
2006년 6월~2009년 2월: 한국전자통신연구원 정보보호연구단 선임연구원
2009년 3월~현재: 국민대학교 정보보안암호수학과 정교수
〈관심분야〉 공개키 암호시스템 안전성 분석 및 고속 구현, 부채널 분석 및 대응법 설계, IoT 정보보호 기술