Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.4, Aug. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.4.739

마스킹 기법이 적용된 SEED 알고리즘에 대한 취약점 분석

김 태 원,[1†] 장 남 수[2‡]
[1]고려대학교, [2]세종사이버대학교

Analysis on Vulnerability of Masked SEED Algorithm

TaeWon Kim,[1†] Nam Su Chang[2‡]
[1]Korea University, [2]Sejong Cyber University

요 약

전력분석의 대응기법으로 가장 널리 알려진 마스킹 기법은 암호 알고리즘 수행 도중 비밀 중간 값을 노출시키지 않게 함으로써 공격자가 필요한 정보를 얻지 못하도록 한다. 마스킹 기법은 대칭키 암호 알고리즘에 적용되어 많은 연구가 진행되었다. 국제표준 알고리즘인 SEED 알고리즘에 대해 마스킹 대응기법 연구가 진행되었다. Cho 등이 제안한 Masked SEED 알고리즘은 1차 전력분석에 안전할 뿐만 아니라 Arithmetic to Boolean 변형 함수의 호출을 줄임으로써 효율성까지 만족시켰다. 본 논문에서는 Cho 등이 제안한 Masked SEED에 대한 취약점을 분석하였다. 효율적인 연산을 위해 추가로 수행되는 사전연산에 의해 마스크 값이 노출되고 이를 이용하여 1차 전력분석 공격으로 비밀키를 복원하였다. 우리는 이론적인 측면과 실험적인 측면을 모두 고려하여 취약점을 분석하였으며 제안한 공격기법은 Cho 등이 제안한 알고리즘이 탑재된 모든 디바이스에서 공통적으로 적용될 수 있음을 예상한다.

ABSTRACT

Masking technique that is most widely known as countermeasure against power analysis attack prevents leakage for sensitive information during the implementations of cryptography algorithm. it have been studied extensively until now applied on block cipher algorithms. Masking countermeasure have been applied to international standard SEED algorithm. Masked SEED algorithm proposed by Cho et al , not only protects against first order power analysis attacks but also efficient by reducing the execution of Arithmetic to Boolean converting function. In this paper, we analyze the vulnerability of Cho's algorithm against first order power analysis attacks. We targeted additional pre-computation to improve the efficiency in order to recover the random mask value being exploited in first order power analysis attacks. We describe weakness by considering both theoretical and practical aspects and are expecting to apply on every device equipped with cho's algorithm using the proposed attack method.

Keywords: Power analysis attacks, Masking scheme, SEED algorithm

I. 서 론

이론적으로 안전한 암호 알고리즘도 실제 구현 단계에서 암호 설계자가 고려하지 못한 부가적인 정보의 노출로 인해 비밀 정보가 노출될 수 있음이 알려졌다. 부채널 공격 (side channel attack)[1]은 오류 주입 공격 (fault injection attack)[2,3], 시간차 공격 (timing attack)[1,4], 전력분석 공격 (power analysis attack)[5-7] 등이 있다.

이 중 전력분석 공격은 가장 현실적이면서 강력한

접수일(2015년 4월 21일), 수정일(2015년 7월 13일), 게재확정일(2015년 7월 14일)
† 주저자, finitefield@korea.ac.kr
‡ 교신저자, nschang@sjcu.ac.kr(Corresponding author)

분석법으로 현재 활발한 연구가 진행되고 있다. 전력분석에 대한 공격기법이 제안됨과 동시에 이를 방어하기 위한 대응기법도 연구되었다. 이 중 가장 널리 사용되고 있는 것은 마스킹(Masking)기법이다[13-18]. 마스킹 기법은 랜덤 마스크 값을 이용하여 전력소모량과 비밀 중간 값과의 상관관계를 제거한다. 이것은 저렴한 비용과 기존 알고리즘에 쉽게 적용시킬 수 있는 장점을 가지고 있다.

국산 블록암호 알고리즘인 SEED[1,2]는 1999년 한국정보보호진흥원(KISA)에서 개발하였고 한국정보통신기술협회(TTA)의 표준으로 채택되었다. 이후, 2005년에는 ISO/IEC 국제표준으로 채택되어 국내·외 산업계 및 학계에 제공되었다. SEED 알고리즘에 대해서도 전력분석에 안전하게 설계하기 위한 대응기법 연구가 진행되었다[10,12]. 2010년 Cho 등은 기존의 방법보다 효율적인 Masked SEED 알고리즘을 제안하였다. Masking 연산 시 많은 연산량을 차지하는 Arithmetic to Boolean 변환연산을 줄임으로서 알고리즘의 효율성을 향상시켰다.

본 논문에서는 Cho 등이 제안한 Masked SEED 알고리즘에 대한 취약점을 분석한다. 제안하는 공격은 SEED 알고리즘이 동작하기 전에 수행되는 사전연산을 공격목표로 한다. 마스크 값에 의존하여 연산의 차이가 발생하는 취약점을 이용하여, 전력파형에서 이들의 차이를 확인함으로써 마스크 값을 복원한다. 이러한 작업은 단순한 시각적인 분석을 통해 가능하였다. 수집된 파형에 대해 마스크 값을 실시간으로 복원한 후 알고 있는 평문정보와 키 추측으로 1차 전력분석 공격을 수행한다. 최종적으로 공격자는 1차 전력분석에 안전한 알고리즘에 대해 마스크 복원 공격을 통한 1차 전력분석 공격으로 비밀키를 얻는다.

본 논문의 구성은 다음과 같다. 2장에서 분석 대상 알고리즘을 소개한다. 따라서 SEED 알고리즘과 1차 전력분석에 안전한 Cho 등의 Masked SEED 알고리즘에 대해 서술한다. 3장은 Cho 등의 알고리즘에 대해 취약점을 분석한다. 이론적인 분석뿐만 아니라 실험적인 검증까지도 수행하였다. 3장의 분석 내용을 바탕으로 비밀키를 복원하는 방법을 4장에서 제안하고 5장에서 결론을 맺는다.

II. SEED & Masked SEED

2.1 SEED 알고리즘

SEED-128 알고리즘은 128-bit Key를 이용하여 16라운드를 거쳐 128-bit의 메시지를 암호화한 후 128-bit 암호문을 출력한다. SEED 알고리즘은 Feistel 구조로 되어있는 F함수와 키 스케줄링 함수로 나누어진다. F함수는 비선형 연산인 S-box를 포함하는 G함수가 포함되어 있다. 키 스케줄링 함수는 128-bit 키를 이용하여 64-bit의 16라운드 키를 생성한다. SEED 알고리즘에 대한 전체 구조도는 [**Fig. 1**]을 통해 도식화 하였다. 128-bit 평문이 2개의 64-bit 블록(L_0, R_0)으로 나누어진 뒤 16라운드를 거쳐 암호문 (L_{16}, R_{16})으로 변환되는 구조를 보여준다. SEED 알고리즘에 대한 상세한 설명은 참고문헌 [8],[9]을 참고하기 바란다.

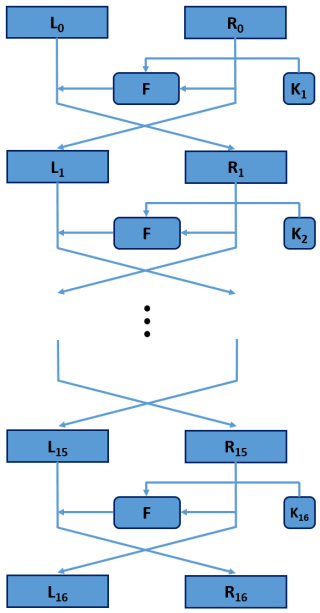

Fig. 1. Structure of SEED

2.2 1차 전력분석에 안전한 SEED

SEED 알고리즘은 Boolean 연산과 Arithmetic 연산을 모두 사용하기 때문에 마스크 값의 형태를 연산에 알맞게 변환을 시켜야한다. 따라서 마스킹 설계에는 Boolean 마스크에서 Arithmetic 마스크로 변

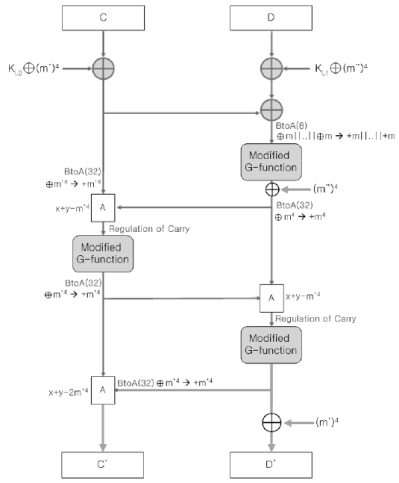

Fig. 2. Cho's masking scheme[10]

환하는 연산(BtoA)과 Arithmetic 마스크에서 Boolean 마스크로 변환하는 연산(AtoB)이 필요하다[19-22]. 이 때, BtoA는 연산이 간단한 반면 AtoB는 다수의 선형 연산으로 구성되어 있기 때문에 많은 연산 소요시간을 요구한다. Cho 등은 이러한 AtoB 변환연산을 줄여 효율적으로 마스킹 기법을 구현하는 방법을 제안하였다.

[Fig.2]와 같이 알고리즘을 수행하기 전 생성된 2개의 8-bit 난수 m, m'와 두 난수에 XOR 연산으로 생성된 난수 m''을 이용하여 Boolean 또는 Arithmetic 마스킹을 시킨다. 또한 동일한 8-bit 난수 4개를 연접하여 사용함으로써 32-bit 데이터 값의 노출을 방지하였다.

기존 마스킹 기법은 Boolean연산을 수행하기 위

해서는 Boolean 마스킹 형태가 필요했으며, Arithmetic 연산을 수행하기 위해서는 Arithmetic 마스킹 형태가 필요했다. 그러므로 모듈러 덧셈연산이

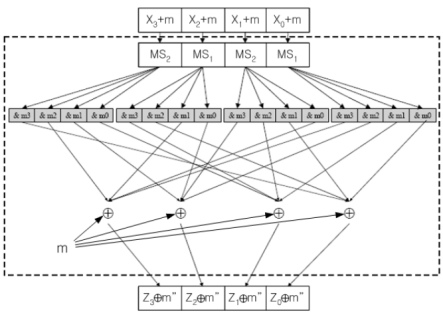

Fig. 3. Modified G-function[10]

동작되기 전 입력 값의 마스크 값 형태를 Arithmetic으로 변환하기 위한 BtoA 연산이 수행되며, G-함수를 수행하기 전 입력 값의 마스

크 값 형태를 Boolean으로 변환하기 위한 AtoB 연산이 진행된다. 이와 달리 Cho 등은 G-함수의 입력 마스크 형태를 Boolean이 아닌 Arithmetic으로 바꿈으로써 Arithmetic 마스킹에서 Boolean

마스킹으로 변환하는 함수의 호출 횟수를 줄였다. 이것을 가능하게 하기 위해 Arithmetic 마스킹 형태의 모듈러 덧셈 결과 값에 대해 캐리 조절 함수와 새롭게 구성된 Masked S-box table을 사용하였다.

[Alg.1]은 알고리즘이 동작하기 전, 생성된 난수 m, m'을 이용하여 Masked S-box을 사전 연산한다. Arithmetic 마스크 형태를 그대로 사용하기 위해 기존의 XOR 연산을 모듈러 덧셈연산으로 대체하였고 Masked S-box의 출력 값은 S-box연산 후 진행된 Boolean 연산을 위해 Boolean 형태가 되도록 설계하였다.

32-bit 단위의 모듈러 덧셈 연산 후 결과 값은 [Fig. 4]의 1과 같이 3개의 32-bit 입력 값 x, y, m^4이 덧셈으로 결합된 형태이다. 이 값이 Masked S-box 연산을 수행하기 위해서는 4개의 8-bit Arithmetic 마스킹 값으로 변환되어야 한다. 이것을 표현한 것이 [Fig. 4]의 2, 4개의 8-bit 블록이다. 여기서 가장 오른쪽 바이트를 제외한 나머지 3개의 바이트는 그 전 바이트의 덧셈 연산 시 발생한 Carry 값이 포함되어 있음을 알 수 있다. 따라서 Masked S-box 테이블을 호출하기 전 반드시 Carry를 제거를 해야 한다. [Fig. 4]의 2,3은 Carry로 인해 두 개의 값이 다름을 표현한 것이다.

32-bit Arithmetic 마스킹 된 중간 값 $z'_3|z'_2|z'_1|z'_0$에서 $z'_i = (z_i + m + carry(z'_{i-1}))$ $i = 0,1,2,3,\ z'_{-1} = 0$의 캐리 발생 유무를 판단해야 한다. 이 때, Carry 발생 여부를 판단하기 위해 수행하는 연산에 의한 중간 값이 노출될 수 있는 문제

Algorithm 1. Generation for Masked S-box

Input : S_2 , m, m'

Output : MS_2

1. **for** i from 0 to 255
2. $\quad MS_2((i+m) \bmod 2^8) = S_2(i) \oplus m'$
3. **Return**(MS_2)

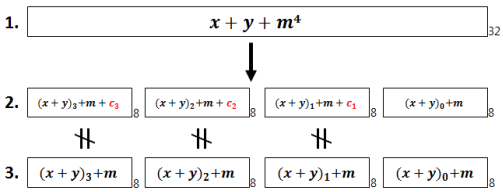

Fig. 4. Figure description for regulation of carry

Algorithm 2. Generation for Carry tables C_1, C_2

Input : 8-bit Random number m, λ
Output : Carry table C_1, C_2

1. **for** i from 0 to 15
2. if $i < (m \,\&\, 0xf0)^{\gg 4}$, $C_1[i] = \lambda + 1$;
3. else if $i = (m \,\&\, 0xf0)^{\gg 4}$, $C_1[i] = \lambda + 2$;
4. else $C_1[i] = \lambda$;
5. if $i < (m \,\&\, 0x0f)$, $C_2[i] = \lambda + 1$;
6. else $C_2[i] = \lambda$;

Return(C_1, C_2)

를 피하기 위해 〔**Alg. 2**〕와 같이 8-bit 랜덤 수 λ를 이용한 Carry 테이블을 생성한다. 〔**Alg. 2**〕에서 8-bit 마스크 값 m에 의한 Carry 생성 여부는 4-bit씩 2개로 나누어서 판단한다. C_1 테이블은 8-bit 중 상위 4-bit에 대한 Carry 발생 여부를 확인하며, C_2 테이블은 하위4-bit에 대한 Carry 발생 여부를 판단한다. C_1, C_2를 이용하여 발생 된 Carry를 제거한 후 최종적으로 G-함수의 입력 값으로 들어간다(〔**Fig. 3**〕을 참조). 〔**Alg. 3**〕는 최

Algorithm 3. Regulation of Carry

Input : 32-bit masked intermediate value
$z'_3|z'_2|z'_1|z'_0 = (z_3|z_2|z_1|z_0) +_{32} (m|m|m|m)$
Output : 8-bit masked intermediate value
$z'_3|z'_2|z'_1|z'_0 = (z_3+m)|(z_2+m)|(z_1+m)|(z_0+m)$

1. **for** i from 0 to 2
2. Carry$= C_1[(z_i \,\&\, 0xf0)^{\gg 4}]$
3. if Carry$=\lambda+2$, Carry$= C_2[z_i \,\&\, 0x0f]$
4. $(z'_3|z'_2|\cdots|z'_{i+1}) -=$ Carry
5. $(z'_3|z'_2|\cdots|z'_{i+1}) += \lambda$

Return$(z'_3|z'_2|z'_1|z'_0)$

하위 2번째 바이트부터 최상위 바이트까지 3개의 바이트에 대하여 순차적으로 Carry를 발생 여부를 확인한 후, 발생했다면 제거시킨다.

III. Cho 등의 Masked SEED 취약점 분석

우리는 2장에서 Cho 등이 제안한 1차 전력분석 안전한 효율적인 Masked SEED에 대해 살펴보았다. 효율적인 Masked SEED 수행을 위해 AtoB 연산을 줄였고 이로 인해 Carry 테이블 생성 함수와 Carry 제어 함수를 추가적으로 수행하였다.

본 장에서는 위에서 소개한 추가적인 연산에서 발생하는 취약점을 분석하였다. SEED 알고리즘 연산 전 수행하는 Carry 테이블 생성연산은 마스크 값에 의존함을 확인하였고 이 정보를 통해 마스크 정보를 복원할 수 있었다. 이러한 취약점에 대해 이론적인 분석뿐만 아니라 실제적인 실험을 통해 이를 검증하였다.

3.1 이론적 분석

취약점 분석을 위해 〔**Alg. 2**〕을 다시 한 번 살펴볼 필요가 있다. Carry 발생 여부를 판단하기 위해 사용되는 Carry 테이블을 생성하는 알고리즘이다. SEED 알고리즘이 호출되면 암호화 연산이 수행되기 전 마스킹 연산에 사용되는 8-bit 마스크 값이 생성된다. 마스크 값 m에 대하여 0부터 15까지 순차적으로 증가하는 루프 값과 크기를 비교하여 테이블에 값을 저장한다. 여기서 마스크 값 m이 공개된 정보 0-15와의 크기비교에 따라 분기처리 된다는 사실에 주목해야 한다. 마스크 값 m의 상위 4비트 값 m_h의 경우, i값이 m_h보다 크기가 작다면 〔**Alg. 2**〕-Step2에서 분기가 처리되어 테이블에 $\lambda+1$을 저장한다. 또한 i값이 m_h와 크기가 같다면 〔**Alg. 2**〕-Step 2의 분기에서 조건을 비교한 후 Step 3로 분기가 처리되어 총 2번의 조건 비교를 수행한다. 조건에 맞는 분기처리를 위해 수행하는 조건비교 연산이 m_h값에 따라 1번 또는 2번이다. 공격자는 이러한 수행 횟수의 차이를 전력소모량을 통해 확인할 수 있다. 즉, m_h과 같은 값이면 m_h보다 작은 값일 경우와 달리 추가적인 조건비교 연산이 더 필요하다. 따라서 수집한 파형을 통해 분기가 처리되는 시점에 대해 차이를 구별하여 중간 값 노출을 방지하기 위한

마스크 값 m_h을 복원한다.

마스크 값 m의 하위 4비트 값 m_l은 2가지 경우의 분기로 구성되어 있다. [**Alg. 2**]-Step 5에서 i 값과 m_l의 값을 비교한 후 작으면 $\lambda+1$ 값을 C_2 테이블에 저장하고 같거나 크면 λ 값을 C_2 테이블에 저장한다. 여기서 m_l 값에 따라서 비교되는 조건은 값이 작거나 같거나 크거나 [**Alg. 2**]-Step 5에서 수행하는 조건비교 연산 1개로 모두 동일하다. 따라서 공격자는 m_h를 복원했던 방법과는 달리 [**Alg. 2**]-Step 5와 6에서 수행되는 연산의 차이를 이용한다. C_2 테이블에 저장되는 값은 $\lambda+1$ 또는 λ이다. λ 값은 C_2 테이블의 해당하는 주소 값에 바로 저장되는 반면 $\lambda+1$은 λ와 1의 덧셈연산(또는 증가연산)을 수행한 후 이를 레지스터 저장한 후 C_2 테이블에 저장된다. 따라서 m_l값에 따라 연산량의 차이가 발생하고 공격자는 이러한 차이를 전력소모량을 통해 구별해 낼 수 있다.

위에서 설명한 방법을 이용하여 공격자는 m_h과 m_l를 각각 복원하여 최종적으로 8-bit 마스크 값 m을 복원한다.

3.2 실험적 분석

Carry 테이블을 생성 시 $i=0$부터 15까지, 16번의 반복문이 동작하면서 마스크 값 m에 상위/하위 4-bit 값에 따라 연산의 차이가 발생함을 확인하였다.

본 절은 마스크 값에 의한 연산의 차이가 전력 소모량의 차이로 나타나, 8-bit 마스크 값 정보를 단순한 시각적인 검사로 구별할 수 있음을 실제적인 실험을 통하여 확인한다. 검증을 위해 KLA-SCARF 시스템의 MSP430 부채널 분석 보드를 이용하였다[11].

[**Fig. 5**]는 마스크 값 0x58에 대한 Carry 테이블 생성 알고리즘 수행 시 수집한 전력파형이다. [**Alg. 2**]은 16번의 반복적인 연산이 진행되기 때문에 16개의 비슷한 패턴이 전력파형을 통해 나타날 것이다. 따라서 파형을 살펴보면 seg1부터 seg16까지 반복문에 의한 16개의 구간으로 파형을 나눌 수 있다. 즉, 각각의 seg는 1개의 반복문 수행에 대한 전력소모량을 나타낸다.

마스크 값에 의한 전력소모량 차이를 확인하기 위하여 알고 있는 마스크 정보 0x58를 이용한다. 우선

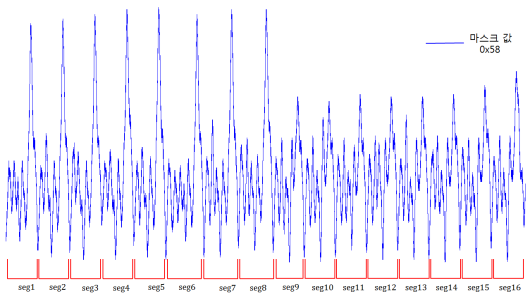

Fig. 5. Power trace for generation of carry table

상위 마스크 값 $M_h=5$에 대해 반복문 $i=0$부터 4까지의 전력파형(seg1 - seg5)과 이 후의 파형(seg6 - seg16)의 차이를 살펴본다.

[**Fig. 6**]은 반복문 $i=4$(seg5)와 $i=5$(seg5)일 때의 전력소모량을 나타낸 그림이다. seg5는 3개의 작은 피크와 1개의 큰 피크로 구성되어 있는 반면 seg6의 경우는 작은 피크 4개와 큰 피크 1개로 이루어져 있다. 따라서 작은 피크의 개수가 3개와 4개로 서로 다름을 확인하였다.

하위 4-bit 마스크 값 $M_l=8$에 의한 전력소모량 차이를 확인하기 위하여 [**Fig.7**]와 같이 $i=7$(seg8)과 $i=8$(seg9)의 반복문에 대한 전력소모량을 비교하였다. 각 seg를 살펴보면 확연한 차이를 나타내는 2개의 피크([**Fig.7**]에서 빨간박스로 표시해 놓은 부분)를 확인할 수 있다.

공격자는 상위 4-bit 마스크 값을 복원하기 위하여 16개의 Carry 테이블 연산 파형 중 하나의 반복문을 구성하는 피크의 개수가 1개 추가되는 i 값을 확인해야 한다. 또한 하위 4-bit 마스크 값 복원을 위해 각 seg에 대해 최대피크가 작아지는 구간을 확인해야 한다.

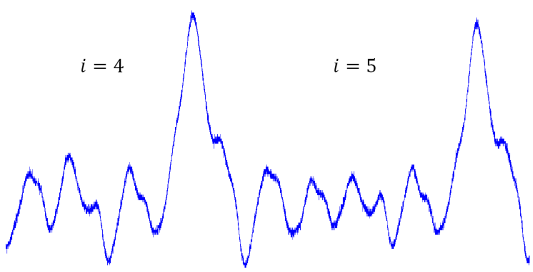

Fig. 6. Difference of power consumption by the M_h

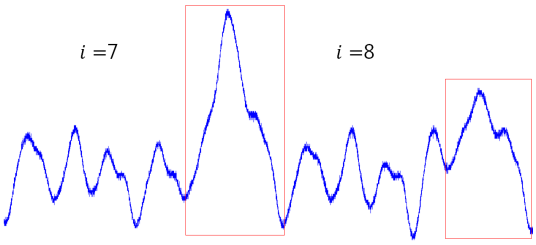

Fig. 7. Difference of power consumption by the M_l

3.3 마스크 값에 의한 특징화

실제로 공격자는 마스크 값을 모르는 상태에서 M_h와 M_l를 구별할 수 있는 특징을 찾아야 한다. 이것은 3.1절에서 소개한 것처럼 Carry 테이블 생성 시 마스크 값에 의해 발생하는 알고리즘 상의 연산차이로 쉽게 확인할 수 있다. M_h값은 분기문에 의한 비교연사의 차이가 파형에서 나타날 것이고, M_l은 덧셈연산(혹은 증가연산)의 유·무로 발생하는 전력 소모량의 차이를 확인하면 된다. 이러한 차이는 각 디바이스마다 다르지만 앞서 소개한 방법과 유사하게 특징화를 시켜 마스크 값을 복원할 수 있다.

IV. Masked SEED에 대한 1차 전력분석 공격

앞에서 소개한 취약점을 이용하여 SEED 알고리즘에서 사용된 비밀정보를 복원하는 방법을 서술한다. 1차 전력분석에 안전한 마스킹 기법을 사용했지만 마스크 값 복원을 통해 1차 전력분석으로 비밀키를 찾을 수 있다.

4.1 공격 흐름

공격자는 Cho 등이 제안한 Masked SEED 알고리즘이 동작했을 때 소모한 전력파형을 n개 수집했다고 가정하자. 그러면 공격자는 다음과 같은 과정을 통해 분석을 진행한다.

Step 1. Carry 테이블 생성 연산시점 확인

최초로 공격자는 Masked SEED 알고리즘이 동작한 파형 중 〔Alg. 2〕가 수행된 부분을 찾아내야 한다. 우선 SEED 알고리즘은 16라운드로 구성 되었으므로 16개의 동일한 패턴을 관찰하여 16라운드 동작 부분을 확인한다. 따라서 16라운드 함수 전에 나타나는 파형을 관찰하면 사전연산이 동작되며 소모된 전력소모량을 확인할 수 있다. Cho 등의 마스킹 스킴은 Carry 테이블 생성,마스킹 S-box 생성,키 스케줄링, 랜덤수 선택과 같은 사전연산이 있다. 최종적으로 공격자는 16번의 반복문이 수행되는 Carry 테이블 생성 부분을 추출하기 위해 16개의 비슷한 패턴이 관찰되는 구간을 선택한다. 이 모든 작업은 단순히 시각적인 분석을 통해 가능하다. 또한 단 하나의 파형을 이용하여 공격자가 원하는 연산 시점을 선택할 수 있으며 이러한 시점 정보를 나머지 파형에 모두 동일하게 적용 가능하다.

Step 2. 특징화

M_h,M_l값에 따른 파형 간 차이를 특징화시키는 단계이다. 〔**Alg. 2**〕에 대한 파형은 반복문에 의한 16개의 seg로 나뉠 수 있다. 따라서 단순한 시각적인 비교로 각 seg를 특징화 시킬 수 있다. 실험에 사용했던 〔**Fig. 5**〕파형의 경우 M_l에 대한 특징은 각각의 seg대한 최대피크 크기이다. 따라서 공격자는 seg1부터 seg8까지 각 seg에서 가장 큰 값을 추출한 후 이 값들에 대한 평균을 계산한다. 또한 seg9부터 seg16에 대해서도 동일한 작업을 수행한다. 2개의 평균값에 대해서 다시 한 번 평균을 계산하여 최종적인 특징 값 ch_M_l을 얻는다.

M_h는 각각의 seg간 수행시간(파형의 길이)을 이용한다. 우선 seg1부터 seg5까지의 각 seg에 대한 길이를 구한 후 이들의 평균값을 구한다. 또한 seg6부터 seg16에 대해서도 동일한 작업을 수행한다. 2개의 평균값의 평균을 구함으로써 M_h에 대한 특징값 ch_M_h를 얻는다.

Step 3. 파형분리

마스크 값을 복원하기 위하여 n개 파형의 Carry 테이블 생성에 대한 전력파형 부분을 〔**Fig. 5**〕과 같이 16개의 seg로 분리시켜야 한다. 각 16개의 분기문의 시작과 끝 부분에서 전력소모량이 급격히 떨어지는 부분을 관찰할 수 있으므로 이에 대한 주기를 기준으로 분리작업을 수행할 수 있다.

Step 4. M_h값 복원

공격자는 순차적으로 각 seg에 대해 길이를 구한

후, 최초로ch_M_h보다 큰 값을 갖는 seg를 찾는다. 최종적으로 찾은 seg의 인덱스 정보를 확인하면 M_h 값을 복원할 수 있다.

Step 5. M_l값 복원

공격자는 순차적으로 각 seg에 대해 가장 높은 값을 구한 후, 최초로ch_M_l보다 작은 값을 갖는 seg를 찾는다. 최종적으로 찾은 seg의 인덱스 정보를 확인하면 M_l 값을 복원할 수 있다.

Step 6. 비밀정보 복원

공격자는 n번의 암호 알고리즘 동작에서 사용된 각 마스크 값 $m_i, (i=1,\cdots,n)$을 복원한 후 알고 있는 평문정보를 이용하여 키를 추측한 후 비밀 중간 값에 대해 1차 상관계수 전력분석을 수행한다. 마스크 값, 평문, 추측 키를 이용하여 Modified G-function 의 입력 값을 추측할 수 있고, 옳은 키를 추측한다면 이와 관련된 연산시점에서 피크 값을 관찰할 수 있다.

V. 결 론

1차 전력분석에 안전하게 설계된 알고리즘이라도 구현상에서 발생할 수 있는 취약점을 이용하여 1차 전력분석으로 비밀 키를 복원할 수 있음을 확인하였다.

Cho 등이 제안한 알고리즘에서 Carry 테이블 생성은 마스크 값에 의존하여 연산이 순차적으로 진행된다. 0부터 15까지의 16번의 반복문 수행에 따라 상위 4-bit 마스크 값과 하위 4-bit의 마스크 값에 따라 연산의 차이가 발생하였다. 이런 차이로 인해 마스크 값을 복원할 수 있음을 이론적으로 분석하였다. 또한 각각의 차이점을 파형을 통해 확인함으로써 실제적인 공격 가능성을 확인하였다. 분석결과 단순한 시각적인 관찰만으로도 차이점을 확인할 수 있으며, 이를 특징화 시켜 모든 파형에 실시간으로 적용시켜 마스크 값 복원이 가능했다.

본 논문에서 제안한 분석기법은 실험에 사용된 MSP430칩에 한정되지 않는다. 각 디바이스 마다 〔**Alg.2**〕수행에 대한 전력파형은 조금씩 다를 것이다. 하지만 마스크 값에 의하여 연산 차이가 발생한다는 사실은 모두 공통적으로 적용된다. 따라서 각 디바이스에 알맞은 특징화를 통해 제안한 공격과정을 수행한다면 비밀키 정보를 얻을 수 있을 것이다.

References

〔1〕 P. Kocher, J. Jaffe, and B. Jun, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS,and Others Systems," CRYPTO '96, LNCS 1109, pp. 104-113, Aug. 1996.

〔2〕 E. Biham, A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," CRYPTO '97, LNCS 1294, pp. 513-525, Aug. 1997.

〔3〕 D. Boneh, R. A. DeMillo and R. J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," EUROCRYPT '97, LNCS 1233, pp. 37-51, May. 1997.

〔4〕 P. Kocher, J. Jaffe, and B. Jun, "Introduction to differential power analysis and related attacks," White Paper, Cryptography Research, 1998.

〔5〕 P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," CRYPTO '99, LNCS 1666, pp. 388-397, Aug. 1999.

〔6〕 S. Chari, J.R. Rao, and P. Rohatgi. "Template Attacks," CHES 2002, LNCS 2523, pp. 13 - 28, Aug. 2002.

〔7〕 E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," CHES, LNCS vol. 3156, pp. 16-29, Aug. 2004.

〔8〕 Korea Information & Security Agency, "Block Cipher Algorithm SEED," Available at http://www.kisa.or.kr

〔9〕 "Information Technology - Security Techniques - Encryption Algorithms - Part 3: Block Ciphers," ISO/IEC 18033-3:2005, 2005.

〔10〕 YoungIn Cho, HeeSeok Kim, Dooho Choi, Dong-Guk Han, Seokhie Hong, and Okyeon Yi, "Efficient Masking Method to Protect SEED Against Power Analysis Attack," Korea Information Processing Society, 17-C(3), pp. 233-242, Jun. 2010.

〔11〕 YongJe Choi, DooHo Cho, and JaeCheol

Ryou, "Implementing Side Channel Analysis Evaluation Boards of KLA-SCARF system," Journal of The Korea Institude of Information Security & Cryptology, 24(1), pp. 229-240, Feb. 2014.

[12] Y.Lu, K.-H. Boey, P.Hodgers, and M.O'Neill. "SEED Masking Implementations against Power Analysis Attacks," IEEE Asia Pacific Conference on Circuit and Systems, pp.1199-1202, Dec. 2010.

[13] S. Mangard, E. Oswald, and T. Popp, Power Analysis Attacks: Revealing the Secrets of Smart Cards., (Advances in Information Security), Springer, 2007.

[14] S. Chari, C. Jutla, J. Rao, and P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks," CPYPTO '99, LNCS 1666, pp. 398 - 412, Aug. 1999.

[15] L. Goubin and J. Patarin, "DES and Differential Power Analysis - The "Duplication" Method," CHES 1999, LNCS 1717, pp. 158 - 172, Aug. 1999.

[16] T. Messerges, "Power Analysis Attacks and Countermeasures for Cryptographic Algorithms," Ph.D. Thesis, University of Illinois, Oct. 2000.

[17] J-S. Coron and L. Goubin, "On boolean and arithmetic masking against differential power analysis," CHES 2000. LNCS 1965, pp. 231 - 237. Aug. 2000.

[18] M. Nassar, Y. Souissi, S. Guilley, and J-L, Danger, "RSM: a small and fast countermeasure for AES, secure against 1st and 2nd-order Zero-Offset SCAs," Design, Automation & Test in Europe Conference & Exhibition (DATE) 2012, IEEE, pp. 1173 - 1178, Mar. 2012.

[19] L. Goubin, "A sound method for switching between boolean and arithmetic masking," CHES 2001, LNCS 2162, pp. 3-15, May. 2001.

[20] J-S. Coron and A. Tchulkine, "A new algorithm for switching from arithmetic to boolean masking," CHES 2003, LNCS 2779, pp. 89-97, Sep. 2003.

[21] O. Neibe and J. Pulkus, "Switching blindings with a view towards idea," CHES 2004, LNCS 3156, pp. 230-239, Aug. 2004.

[22] J-S. Coron, J. Groschadl, and PK. Vadnala, "Secure conversion between boolean and arithmetic masking of any order," CHES 2014, LNCS 3156 pp. 188-205, Sep. 2014.

〈저자소개〉

김 태 원 (TaeWon Kim) 학생회원
2010년 2월: 광운대학교 수학과 학사
2012년 8월: 고려대학교 정보보호대학원 석사
2012년 8월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 부채널 공격, 스마트 카드 보안, 암호시스템 안전성 분석 및 고속구현

장 남 수 (Nam Su Chang) 종신회원
2002년 2월: 서울 시립대학교 수학과 이학사
2004년 8월: 고려대학교 정보보호대학원 공학석사
2010년 2월: 고려대학교 정보경영공학전문대학원 공학박사
2010년 7월~현재: 세종사이버대학교 정보보호학과 조교수
〈관심분야〉 암호칩 설계 기술, 부채널 공격, 공개키 암호 알고리즘, 공개키 암호 암호분석