Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.3, Jun. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.3.477

Square Always 멱승 알고리듬에 대한 부채널 공격*

정 승 교,† 하 재 철‡
호서대학교

Side-Channel Attacks on Square Always Exponentiation Algorithm*

Seung-Gyo Jung,† Jae-Cheol Ha‡
Hoseo University

요 약

정보보호를 위한 암호 시스템을 임베디드 장치에서 개발할 경우 발생할 수 있는 구현상의 문제점을 이용하여 비밀키를 추출하기 위한 여러 부채널 공격들이 시도되어 왔다. 특히, 공개 키 암호 시스템에서 사용하는 멱승(exponentiation) 연산은 기본적으로 곱셈과 자승으로 구현되어 왔으나, 최근 부채널 공격에 대응하기 위한 방법으로 곱셈을 자승 연산으로 대체하는 새로운 Square Always 멱승 알고리듬이 제안되었다. 본 논문에서는 현재까지 부채널 공격에 안전하다고 알려진 Right-to-Left형태의 Square Always 멱승 알고리듬을 공격할 수 있는 기지 전력 충돌 분석(Known Power Collision Analysis) 공격과 변형된 Doubling 공격을 제안한다. 또한, 오류 주입 공격 후 충돌 쌍을 찾아내는 전력 분석 기법을 이용하여 비밀 키를 찾아낼 수 있는 충돌 기반의 조합 공격(Collision-based Combined Attack)을 제안한다. 그리고 Square Always 멱승 알고리듬이 제안한 부채널 공격들에 의해 취약한 특성을 가지고 있음을 컴퓨터 시뮬레이션을 통해 확인하였다.

ABSTRACT

Based on some flaws occurred for implementing a public key cryptosystem in the embedded security device, many side-channel attacks to extract the secret private key have been tried. In spite of the fact that the cryptographic exponentiation is basically composed of a sequence of multiplications and squarings, a new Square Always exponentiation algorithm was recently presented as a countermeasure against side-channel attacks based on trading multiplications for squarings. In this paper, we propose Known Power Collision Analysis and modified Doubling attacks to break the Right-to-Left Square Always exponentiation algorithm which is known resistant to the existing side-channel attacks. And we also present a Collision-based Combined Attack which is a combinational method of fault attack and power collision analysis. Furthermore, we verify that the Square Always algorithm is vulnerable to the proposed side-channel attacks using computer simulation.

Keywords: Side-channel attack, Square Always exponentiation, Collision power analysis, Doubling attack

I. 서 론

RSA(Rivest, Shamir, and Adelman)나 D-H(Diffie-Hellman)과 같은 암호 시스템을 이용

접수일(2014년 4월 5일), 수정일(2014년 5월 23일), 게재확정일(2014년 5월 31일)
* 이 논문은 2013년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임. (2013-0358)
† 주저자, likemuz2@gmail.com
‡ 교신저자, jcha@hoseo.edu(Corresponding author)

하여 디지털 서명이나 데이터 암호화를 실행할 경우 스마트카드와 같이 공개 키 암호 알고리듬이 장착된 임베디드 장치를 사용한다[1, 2]. 그리고 이와 같은 임베디드 디바이스에 공개 키 암호 시스템을 구현할 때에는 고속이면서 구현 효율이 좋은 멱승(exponentiation) 알고리듬을 필요로 한다[3]. 또한, 사용자는 서명이나 암호 등에 필요한 비밀 키를 정보보호 디바이스의 내부에 저장하여 사용하고 있다.

그러나 공개 키 암호 알고리듬을 하드웨어 디바이스에서 개발할 경우 발생할 수 있는 설계상의 실수나 구현 오류로 인해 내장되었던 비밀 키가 노출될 위험이 있는데, 이와 같이 구현 과정에서 발생하는 취약점이나 허점을 이용한 공격을 부채널 공격(Side-Channel Attack)이라 한다. 부채널 공격은 크게 전력 분석(Power Analysis)공격과 같은 수동적 공격[4, 5]과 오류 주입 공격(Fault Injection Attack)과 같은 능동적 공격으로 나누어진다[6, 7]. 디바이스의 소비 전력에 기반한 수동적 공격 방법으로는 단순 전력 분석(Simple Power Analysis, SPA) 공격, 차분 전력 분석(Differential Power Analysis, DPA) 공격, CPA(Correlation Power Analysis) 공격[8] 그리고 전력 충돌 분석(Power Collision Analysis, PCA) 공격[9, 10] 등이 있다. 이 중에서 전력 충돌 분석 공격은 멱승 과정에서 발생하는 중간 값이 충돌하는 과정을 전력 파형을 통해 관측하여 비밀 키를 찾아내는 방법으로서 Doubling 공격[9]과 Relative Doubling 공격[10] 등이 있다. 멱승 알고리듬에 대한 오류 주입 공격으로는 C-safe Error 공격[11]이 대표적인데 이 공격은 연산 알고리듬에 존재하는 더미(dummy) 연산의 취약점을 이용한다.

이외에도 최근에는 능동적인 오류 주입 공격과 수동적인 SPA 공격을 결합한 조합 공격(Combined Attack, CA) 기법이 제안되기도 하였다[12, 13]. 한편, 멱승 알고리듬에 대한 부채널 공격이 다양해짐에 따라 이를 막기 위한 대응책도 알고리듬 특성과 공격 모델을 고려하여 연구되어 왔다. 특히, 전력 분석 공격과 오류 주입 공격을 동시에 방어할 수 있으면서도 구현 효율이 높은 이진(binary) 멱승 알고리듬에 대한 연구가 많았다.

이러한 과정에서 2011년 INDOCRYPT에서 Clavier 등은 다양한 부채널 공격을 방어하기 위한 대응책으로서 모듈라 곱셈(multiplication) 연산 없이 자승(square) 연산만으로도 멱승을 실현할 수 있는 새로운 방법을 제시하였다[14]. 제안된 Square Always 멱승 연산은 비밀 키를 탐색하는 방향에 따라 Left-to-Right 형태와 Right-to-Left 형태로 개발하였으나 Left-to-Right 방식은 Doubling 공격과 조합 공격에 취약할 수 있어 Right-to-Left형 멱승 알고리듬의 사용을 권고하였다.

본 논문에서는 Right-to-Left 형태의 Square Always 멱승 알고리듬에 대해서 비밀 키를 추출할 수 있는 기지 전력 충돌 분석(Known Power Collision Analysis, KPCA) 공격과 변형된 Doubling 공격을 제안한다. 또한 이 멱승 알고리듬이 오류 주입 공격과 충돌 쌍 분석 공격을 결합한 조합 공격에 의해서 비밀 키가 노출될 수 있음을 밝히고자 한다. 제안하는 부채널 공격 방법들의 핵심은 멱승 연산 중 동일한 자승 연산 값을 출력하는 충돌 쌍이 있는지를 전력 분석을 통해 찾아내는 것인데, 논문에서는 컴퓨터 시뮬레이션을 통해 Square Always 알고리듬의 연산 과정에서 비밀 키 추출에 필요한 충돌 쌍을 찾을 수 있음을 확인하였다.

II. 임베디드 장치에서의 멱승과 부채널 공격

임베디드 형태의 RSA나 D-H 암호 시스템에서 정보보호 서비스를 제공하기 위해서는 아래와 같이 메시지 M을 입력으로 모듈러스 N에 대한 멱승 연산이 필요하다.

$$M^d \bmod N \tag{1}$$

여기서 주목할 것은 디지털 서명, 데이터 복호, 그리고 임시 비밀 키 공유 등을 위한 멱승 연산 시에는 사용자의 비밀 키 d가 지수(exponent)로 사용된다는 점이다. 암호시스템의 비밀 키 d를 l비트라 가정하고 이를 이진수로 표기하면 아래와 같다.

$$d=\sum_{i=0}^{l-1} d_i 2^i = d_{l-1}\cdot 2^{l-1} + \ldots + d_1\cdot 2 + d_0 \tag{2}$$

멱승 연산을 수행할 경우 비밀 키 d를 정해진 비트 크기만큼 순차적으로 탐색하게 되는데 탐색하는 방향에 따라 Right-to-Left 방식과 Left-to-Right 방식으로 나누어진다. 또한, 멱승 알고리듬은 비밀 키와 관련한 연산을 순차적으로 처리하는 단위에 따라 이진 방식(binary method), m진 방식(m-ary

method) 그리고 윈도우 방식(window method) 등이 있는데 저용량, 저성능의 임베디드 환경과 구현 효율을 고려하여 이진 멱승 방식을 많이 개발하고 있다[3].

Fig. 1은 멱승 연산 중 Right-to-Left형 이진 방법을 나타낸 것이다. 이 알고리듬에서 변수 S는 메시지 M에 대한 순차적인 자승 값($M^2, M^4, M^8, ..$)을 저장하게 된다. 또한, 변수 R은 비밀 키 비트 d_i가 1인 경우 이전 값과 저장된 S값을 곱하여 새로운 값으로 갱신하게 된다. 이와 같이 반복적인 루프(loop) 연산을 통해 계산된 R값이 최종적으로 멱승의 결과가 된다.

Classical Binary Exponentiation(R-to-L)
1. $R=1$; $S=M$ 2. for $i=0$ to $l-1$ { 2.1 if($d_i=1$) $R=R\cdot S \bmod N$ 2.2 $S=S\cdot S \bmod N$ } 3. Return(R)

Fig. 1. Classical R-to-L binary algorithm

2.1 단순 전력 분석 공격

상기한 Fig. 1과 같은 이진 멱승 알고리듬을 그대로 임베디드 장치에 구현하여 사용한다면 공격자는 이 장치를 구동하여 수집한 전력 파형을 분석만하여도 쉽게 사용자의 비밀 키를 추출할 수 있다. 이와 같은 전력 분석 공격을 SPA 공격이라 하는데 측정한 한 개의 전력 파형만으로도 전체 비밀 키를 찾아낼 수 있다. SPA 공격은 공격자가 멱승 연산 시 소비된 전력 파형을 통해 곱셈 연산과 자승 연산을 구별할 수 있으면 곱셈이 사용된 위치를 찾아 비밀 키 비트 d_i를 결정하는 공격 방법이다.

SPA 공격에 대응하기 위한 멱승 방법으로서 Square-Multiply Always 이진 방식[15]이나 Montgomery Ladder 방식[16] 그리고 Atomic Multiply Always 방식 등이 제안되었는데 Fig. 2는 Right-to-Left형 Atomic Multiply Always 이진 방식을 나타낸 것이다[17].

이 멱승 알고리듬에서는 다른 이진 방식에서 사용되었던 자승 연산이 없으며 오직 곱셈 연산만으로 멱승을 수행하고 있다. 또한, SPA를 방어하는 다른 방식들이 대부분 지수 한 비트당 평균 2번씩의 곱셈(혹은 자승) 연산이 필요한 반면 이 알고리듬은 한 비트당 평균 1.5번의 곱셈만 필요하므로 구현 성능 면에서도 우수한 것으로 알려져 왔다.

Multiply Always Algorithm(R-to-L)
1. $R[0]=1$; $R[1]=M$ 2. $i=0$; $k=1$ 3. while($i \le l-1$) do{ 4. $k=k\oplus d_i$ 5. $R[k]=R[k]\cdot R[1]$ 6. $i=i+k$ } 7. Return(S)

Fig. 2. Multiply Always binary algorithm

그러나 이 알고리듬은 Amiel 등에 의해 비밀 키가 노출될 수 있음이 증명되었다[18]. 그 이유는 자승과 관련한 연산을 곱셈기에 의해 처리는 할 수 있지만 실제로 그 연산이 곱셈 값을 구하는 연산인지 자승 값을 구하는 연산인지를 전력 분석을 통해 구별해 낼 수 있기 때문이다. 따라서 공격자는 Multiply Always 멱승 연산 시 소비되는 전력 분석을 통해 곱셈과 자승 결과를 구별해 낼 수 있고, 결국 이 알고리듬은 부채널 공격에 취약한 것으로 밝혀졌다.

2.2 Doubling 공격

멱승 알고리듬에 대한 Doubling 공격은 공격자가 의도적으로 수학적 연관성을 가진 두 메시지를 입력으로 멱승 연산을 수행한 후 얻은 전력 파형을 분석하여 비밀 키를 추출하는 공격이다[9]. 여기서 사용되는 두 메시지는 M과 M^2의 관계를 가지고 있는데 이 메시지들을 입력으로 각각의 멱승을 수행하게 되면 연산 도중 서로 동일한 중간 값을 가지는 경우가 발생하게 된다. 이를 연산 값의 충돌(collision)이라 하는데 Doubling 공격에서는 두 전력 파형을 분석하여 이러한 충돌이 일어나는 부분을 찾아내게 된다. 결국, 두 배 관계를 가진 메시지를 입력으로 하는 두 전력 파형에서 중간에 동일한 값을 가지는 충돌 파형을 찾아냄으로써 전체 비밀 키를 추출할 수 있다. Doubling 공격은 Left-to-Right형 멱승 방식에 적용되므로 알고리듬 개발자들은 Right-to-Left 형태의 연산 기법을 선호하고 있다.

한편, Montgomery Ladder 멱승 알고리듬은 처

음 SPA에 대응 기법으로 제안되었지만 그 후 Relative Doubling 공격[19]에 의해 안전하지 않음이 밝혀졌다. Relative Doubling 공격에서도 공격에 필요한 전력 파형 수집을 위해 Doubling 공격과 같이 M과 M^2을 입력으로 사용한다. 그러나 Doubling 공격이 전력 파형 분석을 통해 바로 비밀 키 비트가 0인지 1인지 판별하는 공격인 반면, Relative Doubling 공격은 인접한 비밀 키 비트간의 연관성을 찾아내어 전체 비밀 키를 추출하는 전력 분석 공격 방법이다.

2.3 오류 주입 공격 및 조합 공격

멱승 알고리듬에 대한 C-safe Error 공격은 연산이 수행되는 중간에 오류를 주입 후 그 결과가 정상인지 그렇지 않은지 여부를 판별하여 비밀 키 비트를 찾아내는 공격이다. 이 공격은 Square-Multiply Always 이진 방식과 같이 연산 과정에서 결과 값에 영향을 미치지 않는 더미 연산이 있는 경우에 적용할 수 있다. C-safe Error 공격에 대응하기 위해 Montgomery Ladder 방식이나 Atomic Multiply Always 방식을 구현하여 사용할 수 있다.

한편, 그 동안 전력 분석 공격과 오류 주입 공격은 서로 독립적으로 연구되어 왔으나 최근에는 오류 주입 공격과 전력 분석 공격을 결합한 조합 공격이 시도되었다[12, 13]. 여기에 사용된 조합 공격은 비밀 키와 연관된 정보를 저장할 수 있는 레지스터를 0으로 세팅하는 오류를 주입한 후 단순 전력 파형을 분석하는 방법이다 즉, 멱승 연산 시 오류를 주입하여 명령어를 스킵(skip)하는 방식으로 레지스터 값을 0으로 세팅하면 이 레지스터 값과 곱하는 연산 결과는 모두 0으로 출력되므로 다른 연산과 쉽게 구별이 가능하다는 점을 이용하고 있다. 최근 발표된 연구 결과에 의하면 SPA나 오류 주입 공격을 방어하면서 CRT-RSA(Chinese Remainder Theorem based RSA) 멱승[19]에 매우 효과적인 것으로 알려진 BNP(Boscher, Naciri, and Prouff) 알고리듬[20]도 이 조합 공격에 의해 비밀 키를 노출할 가능성이 있음이 지적되었다[13].

III. Square Always 멱승 알고리듬에 대한 부채널 공격

지금까지 기술한 부채널 공격에 대응하기 위해 최근 Square Always 멱승 알고리듬이 제안되었다[14]. 본 절에서는 이 알고리듬을 살펴보고 전력 소비 파형과의 충돌 분석을 통해 비밀 키를 추출할 수 있는 새로운 부채널 공격 방법들을 제안한다.

3.1 Square Always 멱승 연산

Square Always 멱승 알고리듬은 멱승 연산 시 곱셈을 다음과 같이 자승 연산만으로 대체할 수 있다는 성질에 기반하여 제안되었다.

$$X\times Y=\frac{(X+Y)^2-X^2-Y^2}{2} \tag{3}$$

$$X\times Y=\left(\frac{X+Y}{2}\right)^2-\left(\frac{X-Y}{2}\right)^2 \tag{4}$$

즉, 식(3)에서 보면 한 번의 곱셈은 세 번의 자승 연산으로 대체할 수 있음을 알 수 있다. 그렇지만 한 번의 자승(Y^2)은 멱승 알고리듬 구현 시 사전에 처리할 수 있으므로 실질적으로는 두 번의 자승 연산만 필요하다. 예를 들어 Fig. 1의 알고리듬에서 $R\cdot S \bmod N$ 곱셈을 식 (3)의 연산 기법에 적용할 경우, 단계 2.2의 $S^2 \bmod N$은 곱셈 연산을 처리하는 과정에서 미리 계산되어지기 때문에 이를 무시할 수 있다. 또한, 식 (4)는 사전 계산과 관계없이 단지 두 번의 자승으로 곱셈을 대체 처리할 수 있음을 알 수 있다. 하지만 이와 같이 곱셈을 자승으로 처리한다는 개념을 단순하게 Fig. 1에 적용한다면 비밀 키 비트가 0일 때와 1일 경우에는 다른 연산 과정을 가진다는 부채널 정보가 노출되게 된다. 따라서 식 (3)이나 (4)의 개념을Atomicity에 기반한 멱승 알고리듬에 적용하게 되었다. 다음 Fig. 3은 Atomicity에 기반한 Right-to-Left 형태의 Square Always 멱승 알고리듬으로서 위의 식 (4)을 이용하여 한 번의 곱셈을 두 번의 자승으로 처리하였다.

원 논문에서는 이 멱승 알고리듬의 효과적인 구현을 위해 다음과 같은 행렬식을 이용하였다[17].

$$M=\begin{pmatrix}0&0&2&0&0&0&2&1\\2&1&2&2&1&0&1&0\\0&2&1&1&0&0&2&0\\0&0&0&0&1&2&1&1\end{pmatrix} \tag{5}$$

Square Always Exponentiation(R-to-L)

1. $R[0]=M$: $R[1]=1$; $R[2]=1$
2. $i=0$; $j=0$
3. while($i \le l-1$) do{
4. $\quad j=d_i\,(1+(j \bmod 3))$
5. $\quad R[M_{j,0}]=R[M_{j,1}]+R_0 \bmod N$
6. $\quad R[M_{j,2}]=R[M_{j,3}]/2 \bmod N$
7. $\quad R[M_{j,4}]=R[M_{j,5}]-R[M_{j,6}] \bmod N$
8. $\quad R[M_{j,3}]=R[M_{j,3}]^2 \bmod N$
9. $\quad i=i+M_{j,7}$ }
10. Return(R_1)

Fig. 3. Right-to-Left Square Always Exponentiation

Fig. 3에 기술한 알고리듬의 각 루프에서는 행렬 M의 행(row)값 j에 따라 Atomic 형식을 결정하게 되는데 Fig. 4와 같은 4가지의 연산 상태를 갖게 된다. 여기서 d_i가 0이면 $j=0$인 상태를 수행하고 d_i가 1인 경우에는 $j=1$, $j=2$ 그리고 $j=3$일 때의 연산을 수행하게 된다. 각 상태에서 ★ 표시는 연산 결과에 영향을 미치지 않고 SPA를 방어하기 위해 사용하는 더미 연산을 의미한다.

이 방식은 비밀 키 한 비트당 평균 2번의 자승 연산이 필요하므로 Montgomery Ladder 방식보다 고속 연산이 가능하다. 즉, 한 번의 자승은 한 번의 곱셈에 비해 0.8배의 연산량을 가진다고 가정하면 Square Always 알고리듬은 비트 당 약 1.6번의 곱셈이 필요하며 반면 Montgomery Ladder 방식은 1.8번의 곱셈이 필요하므로 약 11.1%를 더 고속으로 처리할 수 있다[14].

Square Always 멱승 알고리듬은 Atomicity 원리에 의해 개발된 것으로서 곱셈 연산과 자승 연산을 구별할 수 없어 SPA 공격은 방어할 수 있다. 그리고 Left-to-Right 방식은 Doubling 공격에 취약할 가능성이 있으므로 저자들은 Right-to-Left 방식을 사용하도록 권고한 바 있다. 또한, Square Always 멱승 알고리듬은 더미 연산을 가지고 있지만 더미 연산에 오류를 주입해도 최종적으로는 오류 결과 값을 출력하므로 공격자는 유용한 부채널 정보를 얻을 수 없다. 따라서 C-safe Error 공격에도 대응할 수 있다.

$j=0$	[★ if j was 0]
$R[0]=R[0]+R[0] \bmod N$	★
$R[2]=R[0]/2 \bmod N$	★
$R[0]=R[0]-R[2] \bmod N$	★
$R[0]=R[0]^2 \bmod N$	
$i=i+1$	★

(a) In case $j=0(d_i=0)$

$j=1$	
$R[2]=R[1]+R[0] \bmod N$	
$R[2]=R[2]/2 \bmod N$	
$R[1]=R[0]-R[1] \bmod N$	
$R[2]=R[2]^2 \bmod N$	
$i=i$	★

(b) In case $j=1(d_i=1)$

$j=2$	
$R[0]=R[2]+R[0] \bmod N$	★
$R[1]=R[1]/2 \bmod N$	
$R[0]=R[0]-R[2] \bmod N$	★
$R[1]=R[1]^2 \bmod N$	
$i=i$	★

(c) In case $j=2(d_i=1)$

$j=3$	
$R[0]=R[0]+R[0] \bmod N$	★
$R[0]=R[0]/2 \bmod N$	★
$R[1]=R[2]-R[1] \bmod N$	
$R[0]=R[0]^2 \bmod N$	
$i=i+1$	

(d) In case $j=3(d_i=1)$

Fig. 4. Four states of main loop in Square Always exponentiation algorithm

3.2 기지 전력 충돌 분석 공격

본 논문에서는 Square Always 멱승 알고리듬을 구현하여 사용할 경우 비밀 키를 추출할 수 있는 새로운 전력 분석 방법을 제안하고자 한다. 연산 과정 분석을 위해 멱승 알고리듬의 루프 연산이 진행되는 동안의 천이 순서를 나타낸 상태도가 Fig. 5이다.

그림에서 보듯이 $d_i=0$일 때는 j가 0인 경우만 수행하고 $d_i=1$일 때는 $j=1,2,3$인 경우를 연속적으로 수행하게 된다. 또한, j가 1일 때와 2일 때는 한 번의 곱셈을 두 번의 자승으로 대체하는 과정을 나타낸다. 그 곱셈의 결과는 최종적으로 Fig. 4의 (c)에 있는 $R[1]$ 레지스터에 저장된다. 상태도에서 화살표는

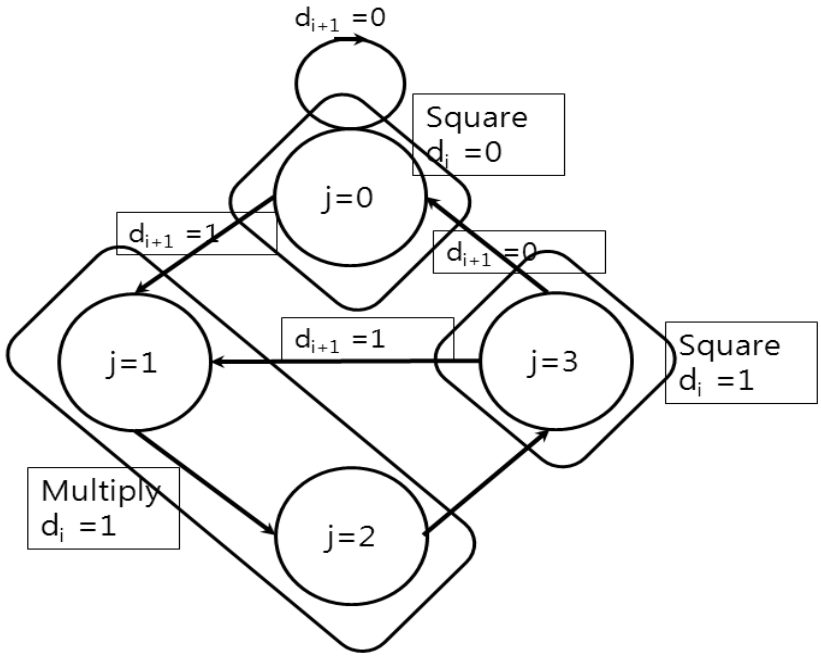


Fig. 5. State diagram of Square Always exponentiation algorithm

상태의 천이 조건을 나타내는 것이며 $j=0$인 상태에서 d_{i+1}이 0이면 다시 그 상태로 되며 d_{i+1}이 1이면 $j=1$상태로 천이한다. 또한, $j=3$인 상태에서 d_{i+1}이 0이면 $j=0$ 상태로 되며 d_{i+1}이 1이면 $j=1$상태로 천이함을 알 수 있다. 따라서 $j=0$ 상태는 d_i가 0인 경우의 자승 연산을 하는 과정이며 $j=3$ 상태는 d_i가 1인 경우의 자승 연산을 하는 과정이다. 물론, 자승한 최종 결과는 모두 $R[0]$ 레지스터에 저장된다.

위에서 살펴본 바와 같이 $j=0$와 $j=3$상태는 서로 다른 비밀 키 비트에 대한 연산 과정으로서 해당 자승 값을 저장하게 된다. 즉, M^2, M^4, M^8과 같이 연속적인 자승 값을 저장하게 된다. 따라서 공격자는 각 자승 연산에 대한 전력 파형을 알 수 있다면 이러한 자승 결과가 $j=0$인 상태에서 발생하는지 $j=3$인 상태에서 발생하는지 분석하여 비밀 키 비트를 알아낼 수 있다.

Fig. 6은 비밀 키 d가 $75=(1001011)_2$인 경우 Square Always 멱승 알고리듬을 처리하는 과정과 공격 방법을 나타낸 것이다. 예에서 보듯이 d_i가 0일 때는 $j=0$인 상태에서 $R[0]$ 값에 대한 자승만 수행한다. 그리고 d_i가 1일 때는 $j=1$인 상태에서 $j=3$인 상태까지 진행하면서 $R[1]$에는 식 (4)에 따라 곱셈한 결과를 저장하며 $R[0]$에는 기존의 $R[0]$ 값에 대한 자승을 수행한다. 여기서 메시지 입력에 대한 자승의 연속 값, 즉 M^2, M^4, M^8 등이 계산되는데 $j=0$인 경우와 $j=3$인 경우에만 계산이 된다. Fig. 6에서 바탕색이 표시된 부분은 자승을 계산하는 지점을 나타낸 것이다. 따라서 공격자가 연속되는 자승 값에 대한 전력 파형을 사전에 가지고 있다면 $j=0$인 상태에서 수행한 것인지 $j=3$인 상태에서 수행한 것인지 충돌 지점을 찾을 수 있다. 결국, 공격자는 자승 값의 충돌 지점을 연속해서 찾아 비밀 키를 모두 추출할 수 있다.

이를 일반화하기 위해 $(i-1)$번째 루프에서의 자승 값 M^k에 대한 충돌 지점을 찾았다고 가정하자. 그리고 i번째 루프에서 $(M^k)^2$ 연산을 수행하는 충돌 지점이 다음 자승과 연접(adjacent)하여 발견된다면 $(j=0)$ d_i가 0이라고 판단할 수 있으며 충돌 지점이 3번째 자승 연산 지점$(j=3)$에서 발견되면 d_i가 1이라고 판단한다. 결국, Fig. 6에서 보는 바와 같이 연속하는 자승 값에 대한 전력 파형을 알고 있는 공격자는 이전 충돌 지점과 다음 충돌 지점이 인접해 있으면 $d_i=0$로 판단하며 다음 충돌 지점이 두 번의 자승을 건너뛰어 세 번째 자승 부분에서 충돌하면 $d_i=1$이라고 판단한다.

제안 공격은 공격자가 특정한 메시지를 선택하고

i	d_i	$j=0$	$j=1$	$j=2$	$j=3$	Remarks
0	1		$R[2]=\left(\frac{M+1}{2}\right)^2$	$R[1]=\left(\frac{M-1}{2}\right)^2$	$R[1]=M$ $R[0]=M^2$	
1	1		$R[2]=\left(\frac{M^2+M}{2}\right)^2$	$R[1]=\left(\frac{M^2-M}{2}\right)^2$	$R[1]=M^3$ $R[0]=M^4$	
2	0	$R[0]=M^8$				Adjacent Collision
3	1		$R[2]=\left(\frac{M^8+M^3}{2}\right)^2$	$R[1]=\left(\frac{M^8-M^3}{2}\right)^2$	$R[1]=M^{11}$ $R[0]=M^{16}$	
4	0	$R[0]=M^{32}$				Adjacent Collision
5	0	$R[0]=M^{64}$				Adjacent Collision
6	1		$R[2]=\left(\frac{M^{64}+M^{11}}{2}\right)^2$	$R[1]=\left(\frac{M^{64}-M^{11}}{2}\right)^2$	$R[1]=M^{75}$ $R[0]=M^{128}$	
Return		$R[1]=M^{75}$				

Fig. 6. An example of proposed Known Power Collision Analysis attack

연속되는 자승 연산에 대한 전력 파형을 미리 알고 있다면 이 값들에 대해 충돌 지점을 분석함으로써 비밀 키를 찾아낼 수 있다. 이러한 의미에서 논문에서 제안하는 공격을 기지 전력 충돌 분석 (Known Power Collision Analysis, KPCA) 공격이라고 할 수 있다. 다른 충돌 기반 공격 방법들은 충돌이 예상되는 값을 예측할 수 없는데 반해, Square Always 멱승 알고리듬에서는 분석에 필요한 충돌 전력 정보를 미리 수집해 둘 수 있다는 것이 특징이다.

3.3 변형된 Doubling 공격

상기한 KPCA 공격을 수행하기 위해서는 공격자는 연속한 자승 값을 계산하는 전력 파형을 미리 저장한 후 충돌 지점을 찾아야 한다. 이러한 공격 목적을 달성하기 위해서는 공격자는 동일한 종류의 다른 칩에서 일정한 메시지 M에 대한 연속되는 자승 값을 계산하는데 소비된 전력 파형을 가지고 있어야 한다. 이를 위해 공격자는 공격 대상이 되는 디바이스 이외의 자신이 제어할 수 있는 다른 디바이스를 가져야만 한다.

그러나 위와 같은 분석 환경이 아니라 공격자는 자신이 목표로 하는 디바이스 하나만 가지고 있을 경우를 가정해 보자. 이 경우 공격자는 목표 디바이스에 대해 두 개의 입력에 대한 파형을 분석함으로써 공격을 시도할 수 있다. 즉, 공격자는 메시지 M과 M^2대해 Square Always 알고리듬을 수행한 두 전력 파형을 수집한다. 이 경우 Doubling 공격과 같은 입력 조건이 된다.

본 논문에서는 상기한 Right-to-Left Square Always 멱승 알고리듬을 사용할 경우 비밀 키를 추출할 수 있는 변형된 Doubling 공격을 제안하고자 한다. Fig. 7은 메시지 M과 M^2에 대한 멱승 과정에서 변형된 Doubling 공격이 수행되는 과정을 설명한 것이다. 그림에서 보는 바와 같이 두 파형의 각 자승을 연산하는 과정에서 비밀 키 비트가 0일 경우에는 충돌 쌍이 바로 인접하여 발생하며($j=0$), 1인 경우에는 인접한 구간에서 충돌 쌍이 발생하지 않고 세 번째 자승 연산 구간($j=3$)에서 충돌 쌍이 발생하게 된다. 따라서 공격자는 각 자승 연산에 대한 전력 파형을 별도로 가지 있지 않아도 공격 목표 디바이스에 대해 두 번의 멱승을 수행한 파형에서 충돌 쌍을 찾는 방법으로 비밀 키를 찾을 수 있다. 기존의 Doubling 공격은 공격하고자 하는 지점을 정해 두고 충돌 쌍이 있는지 없는지 존재 여부를 판별하여 비밀 키를 추출했다. 반면 본 논문에서는 충돌 쌍은 반드시 존재하는데 어느 위치에 존재하는가를 가지고 비밀 키를 찾아내는 것이 차이점이라 할 수 있다.

3.4 충돌 쌍에 기반한 조합 공격

본 논문에서는 Square Always 멱승 알고리듬이 최근에 시도된 조합 공격에도 취약함을 밝히고자 한다. 조합 공격은 오류 주입 공격과 SPA를 결합한 공격으로서 먼저 멱승 연산이 수행되는 동안 오류를 넣

i	d_i	Input : M				Input : M^2				Remarks
		$j=0$	$j=1$	$j=2$	$j=3$	$j=0$	$j=1$	$j=2$	$j=3$	
0	1		$R[2]$		$R[1]=M$ $R[0]=M^2$		$R[2]$		$R[1]=M^2$ $R[0]=M^4$	
1	1		$R[2]$	$R[1]$	$R[1]=M^3$ $R[0]=M^4$		$R[2]$	$R[1]$	$R[1]=M^6$ $R[0]=M^8$	
2	0	$R[0]=M^8$				$R[0]=M^{16}$				Adjacent Collision
3	1		$R[2]$	$R[1]$	$R[1]=M^{11}$ $R[0]=M^{16}$		$R[2]$	$R[1]$	$R[1]=M^{22}$ $R[0]=M^{32}$	
4	0	$R[0]=M^{32}$				$R[0]=M^{64}$				Adjacent Collision
5	0	$R[0]=M^{64}$				$R[0]=M^{128}$				Adjacent Collision
6	1		$R[2]$	$R[1]$	$R[1]=M^{75}$ $R[0]=M^{128}$		$R[2]$	$R[1]$	$R[1]=M^{150}$ $R[0]=M^{256}$	
Return		$R[1]=M^{75}$				$R[1]=M^{150}$				

Fig. 7 An example of modified Doubling attack

고 최종적인 결과가 나오기 전까지 연산을 수행하는 과정에서 전력 소비량을 측정하여 비밀 키를 추출하는 방법이다.

본 논문에서 분석하고자 하는 조합 공격의 공격 모델은 논문 [12]에서 사용한 오류 주입 모델과 동일하지만 전력 파형을 분석하는 방법은 충돌 쌍의 존재성을 판단하는 충돌 기반 조합 공격(Collision-based Combined Attack)이다. 즉, 논문 [12]에서는 전력 파형을 분석하여 0으로 세팅된 레지스터와 곱셈 연산을 수행하는 파형을 찾음으로써 비밀 키 비트를 구별해 내었다. 그러나 본 논문에서는 오류 주입 후 전력 파형의 충돌 쌍이 있는지 여부를 판별하여 비밀 키 비트를 추출하고자 한다. 조합 공격을 위해 공격자는 먼저 Fig. 3의 Square Always 알고리듬에 오류를 주입하여 사용하는 레지스터를 0으로 세팅하게 된다. 레지스터 $R[1]$을 0으로 세팅하는 오류 주입은 Fig. 3의 1단계에서 초기화 명령어를 스킵하는 것으로 가정하였다. 이 과정은 이전의 조합 공격에서 사용한 오류 주입 모델과 동일하다.

구체적으로 살펴보면, Fig. 3의 1단계에서 오류 주입 공격에 의해 $R[1]$을 초기화 하는 과정을 생략하고 $R[1]$이 0으로 세팅되었다고 가정한다. 이 경우 레지스터 $R[1]$과 관련한 연산은 d_i가 1일 때에만 발생한다. 즉, Fig. 4에서 $R[1]$이 0일 때, $j=1$일 때부터 $j=3$일 때의 연산 과정을 보면 다음과 같은 연산이 수행된다.

1) $j=1$일 때

$$R[2]=\left(\frac{R[0]+R[1]}{2}\right)^2=\left(\frac{R[0]}{2}\right)^2$$

2) $j=2$일 때

$$R[1]=\left(\frac{R[0]-R[1]}{2}\right)^2=\left(\frac{R[0]}{2}\right)^2$$

3) $j=3$일 때

$$R[1]=R[2]-R[1]=0$$

따라서 한 번 0으로 세팅된 $R[1]$값은 연속적으로 0의 값을 유지한다. 그 결과 $j=1$일 때의 자승 연산과 $j=2$일 때의 자승 연산이 동일한 결과 값을 가진다. 따라서 이 두 연산은 서로 충돌 값을 가지게 되며 소비되는 전력량도 특별한 상관 관계를 가지고 있다. 따라서 공격자는 전력 파형 중 바로 다음 루프에서 충돌 쌍을 찾으면 해당하는 비밀 키 비트 d_i는 1이라고 판단하고 그렇지 않은 경우에는 0이라고 판단한다.

Fig. 8은 조합 공격에 의해 $R[1]$값이 0으로 되었을 경우 멱승 연산을 처리하는 과정을 나타낸 것이다. 그림에서 보는 바와 같이 처음 비밀 키 비트와 관련한 자승 연산 후 다음 자승 연산이 동일한 값을 가짐으로써 충돌이 발생한다면 비밀 키 비트는 1이 되고 그렇지 않으면 0임을 알 수 있다. 따라서 Square Always 멱승 알고리듬이 수행되는 동안 $R[1]$을 0으로 세팅만 하고 이후의 전력 파형을 얻을 수 있다면 연속한 루프에 충돌 쌍이 있는지를 판단하여 모든 비밀 키를 추출할 수 있다.

i	d_i	$j=0$	$j=1$	$j=2$	$j=3$	Remarks
0	1		$R[2]=\left(\frac{M}{2}\right)^2$	$R[1]=\left(\frac{M}{2}\right)^2$	$R[1]=0$ $R[0]=M^2$	Adjacent Collision
1	1		$R[2]=\left(\frac{M^2}{2}\right)^2$	$R[1]=\left(\frac{M^2}{2}\right)^2$	$R[1]=0$ $R[0]=M^4$	Adjacent Collision
2	0	$R[0]=M^8$				
3	1		$R[2]=\left(\frac{M^8}{2}\right)^2$	$R[1]=\left(\frac{M^8}{2}\right)^2$	$R[1]=0$ $R[0]=M^{16}$	Adjacent Collision
4	0	$R[0]=M^{32}$				
5	0	$R[0]=M^{64}$				
6	1		$R[2]=\left(\frac{M^{64}}{2}\right)^2$	$R[1]=\left(\frac{M^{64}}{2}\right)^2$	$R[1]=0$ $R[0]=M^{128}$	Adjacent Collision
Return		$R[1]=0$				

Fig. 8. An example of Collision-based Combined attack

IV. 비교 분석 및 시뮬레이션

4.1 부채널 공격 방법 분석 및 대응 방법

본 논문에서는 지금까지 부채널 공격에 안전하다고 알려진 Square Always 멱승 알고리듬에 대한 3가지의 공격 방법을 제안하였다. 제안 방법 중에서 KPCA 공격과 변형 Doubling 공격은 전력 분석 공격 방법이며 다른 하나는 오류 주입 공격과 CPA 공격을 결합한 조합 공격 방법이다.

Table 1은 전력 분석 공격 방법 중 기존에 제시된 Doubling 공격, Relative Doubling 공격 그리고 논문에서 제안하는 전력 분석 공격 방법의 차이점을 비교한 것이다.

Doubling 공격이 Left-to-Right 형태의 Square-Multiply Always 알고리듬에 관한 공격이라면 제안하는 공격은 Right-to-Left 형태의 Square Always 멱승 알고리듬이다. 공격에 필요한 전력 파형은 모든 방식이 두 개만 있으면 공격이 가능하다. 특히, Doubling 공격, Relative Doubling 공격 그리고 변형 Doubling 공격은 입력이 M과 M^2인 멱승의 전력 파형만 있으면 충돌 쌍을 찾을 수 있다. 따라서 이 3가지 공격 방법은 입력에 제약을 받지 않는 서명이나 복호 시스템에서 공격이 가능하다. 하지만 RSASSA-PSS나 RSARSA-PKCS#1과 같은 표준 서명[21]에서는 멱승 연산에 대한 입력으로

Table 1. Comparison of power analysis attacks

	Doubling[9]	Relative Doubling[10]	Proposed	
			KPCA	Modified Doubling
Target algorithm	Square-Multiply Always	Montgomery Ladder	Square Always	Square Always
Exponent scan method	Left-to-Right	Right-to-Left	Right-to-Left	Right-to-Left
Exponentiation Power signal	M^d, $(M^2)^d$	M^d, $(M^2)^d$	M^d, M^{2^l}	M^d, $(M^2)^d$
Checking loops for collision test	① i-th for $(M^2)^d$ ② $(i+1)$-th for M^d	① i-th for $(M^2)^d$ ② $(i+1)$-th for M^d	① i-th for M^{2^l} ② $(i+1)$-th$(j=0)$ or $(i+1)$-th$(j=3)$ for M^d	① i-th for $(M^2)^d$ ② $(i+1)$-th$(j=0)$ or $(i+1)$-th$(j=3)$ for M^d
Secret Information	Absolutely d_i	Relationship of d_i and d_{i+1}	Absolutely d_i	Absolutely d_i

일정한 메시지 형식을 가지도록 하고 있다. 따라서 위의 공격 방법으로는 정해진 입력 메시지 형식을 갖는 서명 시스템을 공격할 수는 없다.

그러나 제안하는 공격 방법 중 KPCA 방법은 입력 형식에 대한 연속된 자승 파형만 미리 측정할 수 있다면 공격이 적용 가능하다. 즉, 메시지 형식에 의해 서명을 수행하더라도 이 형식화된 메시지에 대한 연속된 자승 파형을 사전에 미리 측정할 수 있는 장치가 있다면 공격이 가능하다는 면에서 다른 3가지 방법보다 강력한 공격 방법이라고 할 수 있다.

논문에서 제안하는 조합 공격은 논문 [12]의 조합 공격과 오류 주입 모델은 동일하지만 전력 분석 방법에서 차이가 있다. 논문 [12]의 조합 공격은 오류 주입 후 0과 곱해지는 부분을 SPA로 관측하여 비밀 키를 구하는 방법인 반면, 제안하는 조합 공격에서는 오류 주입 후 측정된 파형에서 충돌 쌍이 발생하는지 여부를 전력 분석을 통해 확인하여 비밀 키를 찾아낸다.

논문에서 제시한 전력 분석 공격은 소비 전력 측정을 어렵게 하거나 기존의 DPA 대응 기법을 적용하여 방어할 수 있다. 특히, 멱승에 사용되는 지수를 랜덤화 하거나 입력 메시지를 랜덤화 하는 블라인딩(blinding) 기법을 사용하면 실제 멱승 연산에 사용되는 입력과 중간 계산 값의 변화로 인해 충돌 쌍을 찾을 수 없어 KPCA와 변형 Doubling 공격을 방어할 수 있다. 그러나 충돌 쌍에 기반한 조합 공격은 오류 주입과 단 하나의 파형만으로도 비밀 키를 찾아내게 되므로 블라인딩을 적용한 DPA 대응 기법을 무력화할 수 있다.

4.2 충돌 쌍 추출 시뮬레이션

현재까지 부채널 공격에 대해 안전한 것으로 알려진 Square Always 알고리듬은 곱셈 연산 없이 자승만으로 멱승을 처리한다는 점에서 획기적인 멱승 연산 방법이다. 그러나 Atomicity에 기반한 속성을 그대로 유지하고 있다는 점이 다른 부채널 공격이 가능한 빌미가 된다. Atomic 알고리듬들은 모든 연산을 자승 연산으로 처리한다 하더라도 그 결과 값은 비밀 키 비트가 1일 때나 0일 때 가지는 중간 과정 값을 그대로 유지하고 있다. 따라서 일정한 메시지에 대한 연속된 자승 값은 멱승을 수행하는 동안 항상 존재하게 되며 이 값에 대한 충돌 여부를 전력 분석을 통해 확인할 수 있다면 비밀 키가 노출될 수 있다.

논문에서 제시한 공격 방법이 실제 환경에서 적용 가능한 지는 스마트 카드나 임베디드 시스템에 Square Always 알고리듬을 구현한 후 시스템의 동작 과정에서 소비 전력을 측정하거나 오류를 주입한 후 공격을 시도함으로써 확인해 볼 수 있다. 실제로 Fiex와 Venelli는 RSA 시스템에서 충돌 쌍을 찾아 비밀 키를 찾아내는 공격을 실험한 바 있으며[8], Amiel과 Villegas는 오류 주입과 단순 전력 분석을 조합한 공격 실험을 통해 비밀 키가 노출될 수 있음을 검증하였다. 논문에서는 실제 하드웨어적인 실험 환경을 쉽게 구성할 수 없는 점을 고려하여 Square Always 알고리듬에 대한 전력 분석 공격과 조합 공격이 가능함을 컴퓨터 시뮬레이션을 통해 검증하고자 한다.

이를 위해 먼저 자승 연산만을 이용하여 멱승 연산

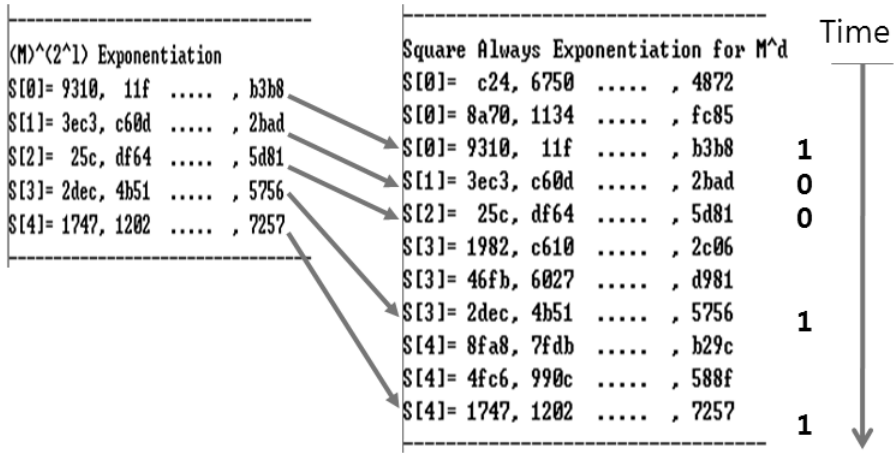


Fig. 9. Simulation of KPCA attack

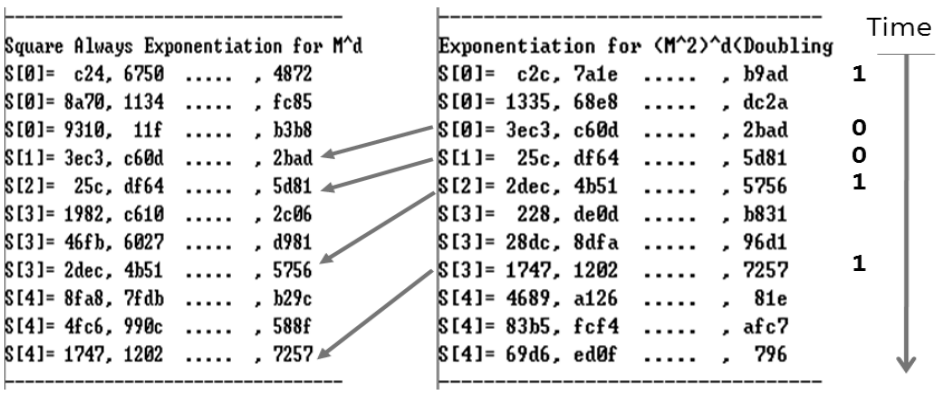


Fig. 10. Simulation of modified Doubling attack

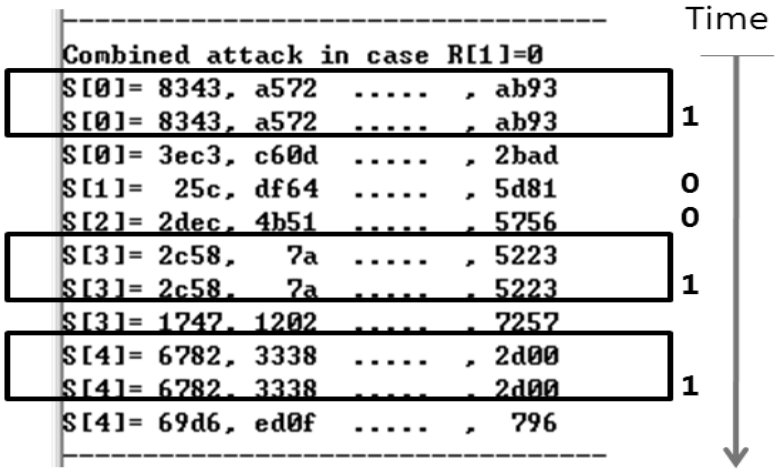


Fig. 11. Simulation of Collision-based Combined Attack

을 수행하는 Square Always 알고리듬을 C언어로 개발하고 KPCA와 변형 Doubling 공격 그리고 조합 공격에서 충돌 쌍이 발생하는 것을 확인하였다. 시뮬레이션에서는 각각 1024비트 합성수 N과 비밀 키 d에 대해 멱승을 수행하였다. 단, 시뮬레이션 화면에서는 충돌 쌍 발생 여부만 확인할 수 있도록 각 결과값의 최상위 자리와 최하위 자리 값만 표시하였다. 또한, 1024비트 비밀 키 d의 중 최하위 자리는 0xDA19를 사용하였다.

Fig. 9는 KPCA 공격에서 충돌 쌍이 발생하는 것을 보인 것인데 왼쪽은 사전에 M^{2^l} 값을 계산하는 과정에서 출력되는 자승 값들을 순차적으로 표시하였다. 그리고 오른쪽은 실제로 Square Always 알고리듬을 수행하는 과정의 각 루프의 값을 표시한 것이다. 화면에 보는 바와 같이 왼쪽의 값들이 오른쪽 값들과 충돌이 일어나고 있으며 이를 통해 비밀 키를 하위 비트부터 추측해 보면 "1, 0, 0, 1, 1..."이 됨을 알 수 있다. 즉, 사용된 비밀 키의 최하위 디지트는 0x19임을 알 수 있다.

다음 Fig. 10은 변형 Doubling 공격에서 충돌 쌍이 발생하는 것을 보인 것이다. 그림의 왼쪽은 M을 입력으로 한 멱승이고 오른쪽은 M^2을 입력으로 한 멱승 결과이다. 그림에서도 충돌 쌍이 발생하는 위치가 다음 루프인지 그렇지 않은지에 따라 비밀 키를 유추할 수 있음을 알 수 있다.

또한 Fig. 11에서 보는 바와 같이 레지스터 $R[1]$이 0으로 세팅된 오류 주입 공격을 가정하면 충돌 쌍이 발생하는 것을 알 수 있다. 레지스터 $R[1]$에 대해 오류를 주입한 후 바로 인접한 다음 루프에서 충돌 쌍이 발견되면 비밀 키 비트는 1이 되고 그렇지 않은 경

우에는 0이 된다는 사실에 기반하여 전체 비밀 키를 찾아낼 수 있다. 이 공격에서는 오류 주입 기능이 추가로 필요한 반면 하나의 전력 파형만 분석함으로써 비밀 키를 찾아낼 수 있다.

V. 결 론

임베디드 암호 시스템에 필요한 멱승 알고리듬을 구현할 경우 전력 분석과 오류 주입 공격과 같은 부채널 공격을 방어하기 위한 대응책이 필요하다. 최근 제안된 Square Always 알고리듬은 자승 연산만으로 멱승을 실현할 수 있는 방법으로서 고속 구현이 가능하며 여러 부채널 공격에도 안전한 것으로 알려져 왔다.

본 논문에서는 이러한 Square Always 멱승 알고리듬을 사용할 경우 발생할 수 있는 구현상의 취약점을 통해 비밀 키가 노출될 수 있음을 지적하였다. 연산 과정에서 발생하는 소비 전력을 분석하면 자승 값에 대한 충돌 쌍이 발생하는 위치를 추측할 수 있고 이 정보만으로 비밀 키를 추출할 수 있다. 또한, 오류 주입과 충돌 쌍에 기반한 전력 분석 공격을 결합한 조합 공격에 의해서도 비밀 키가 노출될 수 있다. 따라서 개발자들은 멱승 알고리듬을 임베디드 장치에 구현할 경우 부채널 공격에 대한 세심한 주의와 더불어 강인한 특성을 가진 멱승 알고리듬을 선택하여 사용하여야 한다.

References

[1] R. Rivest, A Shamir, and L, Adelman, "A method for obtaining digital signature and public-key cryptosystems," Comm. of the ACM vol. 21, no. 2, pp. 120-126, 1978.

[2] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol. IT-22, no. 6, pp. 644-654, Nov. 1976.

[3] D. Gordon, "A survey of fast exponentiation methods," Journal of Algorithms," vol. 27, pp. 129-146, 1998.

[4] P. Kocher, "Timing Attacks on Implementation of Diffie-Hellman, RSA, DSS, and Other Systems," CRYPTO'96, LNCS 1109, pp. 104-113, 1996.

[5] P. Kocher, J. Jae, and B. Jun, "Differential power analysis," CRYPTO' 99, LNCS 1666, pp. 388-397, 1999.

[6] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," EUROCRYPTO'97, LNCS 1233, pp. 37-51, 1997.

[7] M. Joye, J. Quisquater, F. Bao and R. H. Deng, "RSA-type signatures in the presence of transient faults," Cryptography and Coding, LNCS 1355, pp. 109-121, 1997.

[8] M. Witteman, J. Woudenberg, and F. Menarini, "Defeating RSA Multiply-Always and Message Blinding Countermeasures," CT-RSA'11, LNCS 6558, pp. 77-88, 2011.

[9] P. Fouque and F. Valette, "The doubling attack- why upwards is better than downwards," CHES'03, LNCS 2779, pp. 269-280, 2003.

[10] S. Yen, L. Ko, S. Moon, and J. Ha, "Relative doubling attack against Montgomery ladder," ICISC'05, LNCS 3935, pp. 117-128, 2005.

[11] S. Yen, S. Kim, S. Lim, and S. Moon, "A countermeasure against one physical cryptanalysis may benefit another attack," ICISC'01, LNCS 2288, pp. 414-427, 2002.

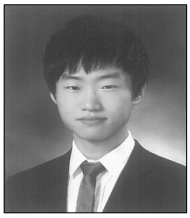
[12] F. Amiel, K. Villegas, B. Feix, and L. Mercel, "Passive and Active Combined Attacks: Combining fault attacks and side channel analysis," FDTC'07, IEEE-CS, pp. 92-102, 2007.

[13] H. Kim and J. Ha, "A physical combined attack and its countermeasure on BNP exponentiation algorithm," Journal of The Korea Institute of Information Security & Cryptology(JKIISC), 23(4), pp. 585-591, 2013.

[14] C. Clavier, B. Feix, G. Gagnerot, and M. Roussellet, "Square Always ex-

ponentiation," INDOCRYPT'11, LNCS 7107, pp. 40-57, 2011.
〔15〕 J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," CHES'99, LNCS 1717, pp. 292-302, 1999.
〔16〕 M. Joye and S. M. Yen, "The Montgomery Powering Ladder," CHES'02, LNCS 2523, pp. 291-302, 2002.
〔17〕 B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity," IEEE Trans. on Computers vol. 53, no. 6, pp. 760-768, 2004.
〔18〕 F. Amiel, B. Feix, M. Tunstall, C. Whelen, and W. Marnane, "Distinguishing Multiplications from Squaring Operations," SAC'08, LNCS 5381, pp. 346-360, 2009.
〔19〕 C. Couvreur and J. J. Quisquater, "Fast decipherment algorithm for RSA public-key cryptosystem," Electronics Letters, vol. 18, no. 21, pp. 905-907, 1982.
〔20〕 A. Boscher, R. Naciri, and E. Prouff, "CRT-RSA Algorithm Protected Against Fault Attacks," WISTP'07, LNCS 4462, pp. 237-252, 2007.
〔21〕 RSA Inc., "PKCS#1 v2.1, RSA Cryptography Standard," Jan. 2001. Available at ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf

〈저자소개〉



정 승 교 (Seung-Gyo Jung) 학생회원
2013년 2월 : 호서대학교 정보보호학과 (공학사)
2013년 2월 ~ 현재: 호서대학교 대학원 정보보호학과 (석사과정)
〈관심분야〉 정보보호, 스마트폰 보안, 무선 네트워크 보안

하 재 철 (Jae-Cheol Ha) 종신회원
1989년 2월: 경북대학교 전자공학과 졸업
1993년 8월: 경북대학교 전자공학과 석사
1998년 2월: 경북대학교 전자공학과 박사
1998년 3월~2007년 2월: 나사렛대학교 정보통신학과 부교수
2007년 3월~현재: 호서대학교 정보보호학과 교수
〈관심분야〉 정보보호, 네트워크 보안, 부채널 공격