IT융합 디바이스에 대한 물리적 2차 CPA 공격을 위한 새로운 전처리 기법

준회원 이 철 희*, 황 아 름*, 이 동 건*, 종신회원 김 형 남**, 김 호 원*°

New Pre-processing Method for Second-Order CPA on the IT Convergence Device

Chulhee Lee*, Ahreum Hwang*, Donggeon Lee* *Associate Members*,
Hyoungnam Kim**, Howon Kim*° *Lifelong Member*

요 약

본 논문에서는 스마트그리드나 AMI, Zigbee 기반 홈네트워크와 같은 대표적인 IT 융합 환경 상에서 그 환경을 구성하는 주요 디바이스들에 대하여 내부에 존재하는 비밀키 값과 같은 중요한 정보를 쉽게 찾아낼 수 있는 효율적인 2차 차분전력분석 기법을 제안한다. 이 기법은 1차 차분전력분석 기법에 대한 공격 방지 기법이 적용된 디바이스에서도 쉽게 그 키 값을 찾아내는 기법으로 먼저 기존의 사전처리함수를 이용한 2차 차분전력분석 공격 기법을 실제로 구현하여 성능을 분석하고 이후 마스킹이 적용된 알고리즘에 대하여 더 강화된 사전처리함수를 제안한다. 그리고 제안한 사전처리함수를 이용하여 2차 CPA 공격을 수행하고 그 결과를 분석함으로 마스킹 대응 기법에 대한 2차 차분전력분석 공격이 IT융합 보안 기술 분야에 있어 매우 위협적인 공격임을 실험적으로 검증한다.

Key Words : Second-order DPA, CPA, AES, Preprocessing function, side channel attack

ABSTRACT

In this paper, we propose the efficient Second-Order Differential Power Analysis attack, which has ability to find significant information such as secret key in the devices consisting IT convergence environment such as Smartgrid, Advanced Metering Infrastructure(AMI) and ZigBee-based home networking service. This method helps to find the secret key easily at a device, even though it uses a countermeasure like masking which makes First-Order DPA attack harder. First, we present the performance results of our implementation which implements practical Second-Order DPA attack using the existing preprocessing function, and analyze it. Then we propose a stronger preprocessing function which overcomes countermeasures like masking. Finally, we analyze the results of the Second-Order CPA attack using proposed preprocessing function and verify that proposed scheme is very threatening to the security fields of IT convergence technology through the experimental results.

I. 서 론

오늘날 IT 기술의 발달로 인해 언제 어디서나 원하는 정보를 각종 네트워크에 접속하여 검색, 저장, 전송할 수 있는 유비쿼터스 환경이 이루어짐으로써 모든 사물이 지능화, 정보화, 네트워크화 되어가고 있다.

※ 이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2008-0061842)
* 부산대학교 컴퓨터공학과 정보보호 및 임베디드 보안 연구실 ({2fehee, xlizhwang, guneez, howonkim}@gmail.com), (° : 교신저자)
** 부산대학교 전자전기공학과 통신 및 신호처리 연구실 (컴퓨터 및 정보통신 연구소) (hnkim@pusan.ac.kr)
논문번호 : KICS2010-03-144, 접수일자 : 2010년 3월 1일, 최종논문접수일자 : 2010년 8월 10일

이러한 유비쿼터스 환경은 에너지 분야로까지 확대되어 기존의 전기 에너지 분배 및 관리, 에너지 소비 및 관리를 위한 스마트 그리드 및 AMI(Advanced Metering Infrastructure) 기술에 대한 연구 개발로 진행 중에 있다. 특히 스마트 그리드 및 AMI와 같이 물리적인 환경이 IT 기술과 접목된 상황에선 새로운 보안 문제가 부각되거나 혹은 존재하지 않던 보안 문제가 새로 발생하게 되는데 대표적인 것이 물리적 공격이다.

물리적 공격의 의미는 암호화 과정에서 발생하는 각종 물리적인 신호들을 채득하여 이로부터 암호화에 사용되는 암호키를 추출하는 것을 말하며 현재 스마트카드와 , USB 보안 token, RFID, 센서노드, Zigbee, Bluetooth와 같은 통신용 칩이나 방송용 수신 칩, 등 많은 분야에 위협적인 공격으로 응용되고 있다. 스마트 그리드 및 AMI 환경에서의 핵심 무선 인프라 기술인 무선 센스 네트워크와 지능형 계량기, 원격검침 및 제어 시스템, AMI 스마트미터기 등은 이러한 물리적인 공격을 방어할만한 마땅한 대비책이 없는 실정이기에 만약 물리적인 공격으로 인한 보안성이 보장되지 않게 되면 제어 시스템의 오작동 및 화재/테러를 유발하여 인적/물적 피해를 가져올 것이며 AMI 단말기에 대한 검침 정보의 위변조와 각 가정의 가스 및 전기 사용량에 대한 도청을 통해 프라이버시 노출로 인한 인권침해 문제를 야기 시킬 것이다. 이러한 IT 융합 분야의 물리적인 보안에 있어 위협적인 물리적 공격은 여러 종류가 있으며 그 중에서도 부채널 공격(side-channel attack)[1-2]중의 하나인 차분 전력 분석(differential power analysis: DPA)[3] 공격은 가장 강력한 공격 방법이다.

현재 차분 전력 분석 공격을 막기 위한 대응방법으로 마스킹[4-9], 하이딩[10], 그리고 새로운 하드웨어 논리 회로[11-13] 등이 연구되었으며 그 중에서도 현재 대응 방법으로 가장 활발하게 연구 되고 있는 분야는 마스킹을 적용한 대응 기법이다. 반면 이러한 마스킹 대응 기법을 공격하는 연구 또한 많이 이루어 졌는데 대표적인 것으로 2차 DPA (second-order differential power analysis)[14-16]공격을 들 수 있다. 본 논문에서는 마스킹 기법에 대한 기존의 2차 DPA 공격에 사용된 사전처리함수(preprocessing function)들을 모두 구현하여 실질적인 공격을 통해 각각의 위협성을 알아보았고 실질적인 공격 환경에서의 노이즈 영향을 고려하여 공격 환경자체를 새롭게 모델링 하였으며 이러한 환경에 맞는 새로운 사전처리함수를 이용함으로 보다 강화된 2차 CPA(second-order correlation power analysis)공격 방법을 제시하였다. 다양한 사전처리함수를 실험하기 위해서 마스킹 기법이 적용된 AES를 마이크로 컨트롤러에 구현하였으며 제안하는 사전처리함수를 이용한 공격 방법이 기존 공격 방법에 비해 보다 효율적이고 강력한 공격임을 실험적으로 검증하였다.

본 논문의 구성은 다음과 같다. 2장에서 1차 DPA 공격 중에서도 많이 사용되는 CPA(correlation power analysis)[17]에 대해 간단히 소개하고 마스킹 기법과 2차 CPA공격에 대해 설명한다. 3장에서 사전처리함수의 특징과 기존의 사전처리함수들에 대한 각각의 성능에 대해 설명하고 본 논문에서 제안하는 새로운 사전처리함수를 이용한 2차 CPA공격 방법을 소개한다. 4장에서는 3장에서 언급한 기존의 사전처리 함수들과 제안한 사전처리함수를 직접 구현하여 2차 CPA 공격을 함으로써 실질적인 실험을 통해 제안된 방법이 가장 우수한 성능을 보임을 검증한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 마스킹 기반 2차 CPA 공격

2.1 상관도 분석 (correlation power analysis: CPA) 공격

DPA 공격에는 어떤 전력 모델(power model)을 선택하고 어떤 통계학적인 분석(statistical analysis)을 사용하는지에 따라서 다양한 공격 방법들이 존재한다. 그 중 공격 성능이 좋고 가장 대표적인 공격 방법으로 상관도 분석(correlation power analysis: CPA) 공격이 있으며 이것은 하나의 시점에 대한 공격으로 1차 (first-order) DPA공격에 속한다. 상관도 분석 공격을 그림 1과 같이 AES 알고리즘에 예를 들어 설명하면 다음과 같다.

먼저 128비트 키 중 첫 번째 8비트 키 K와 랜덤하게 생성한 M개의 평문 P를 만든다. 그리고 데이터와 키 값에 의존한 중간 값을 도출시킬 분류함수로 D(Pi, Kj)= $S(P_i \oplus K_j)$을 선택하고 분류함수를 거쳐 계산된 중간결과 값 행렬인 D를 얻는다. 그리고 얻은 중간 결과 값에서 소비전력을 추정하기 위해서 해밍 웨이트 모델을 적용하여 추정하는 소비전력 값인 H=HW ($S(P_i \oplus K_j)$)행렬을 구한다. 그리고 실제로 공격하려는 타겟으로부터 측정한 소비전력 트레이스인 T행렬과 앞서 구한 H행렬사이에 식 (1)을 이용한 상관계수를 계산함으로 R 행렬을 구하게 되고 R행렬 중 가장 높은 상관계수 값이 발생하는 지점과 해당하는 키를 구

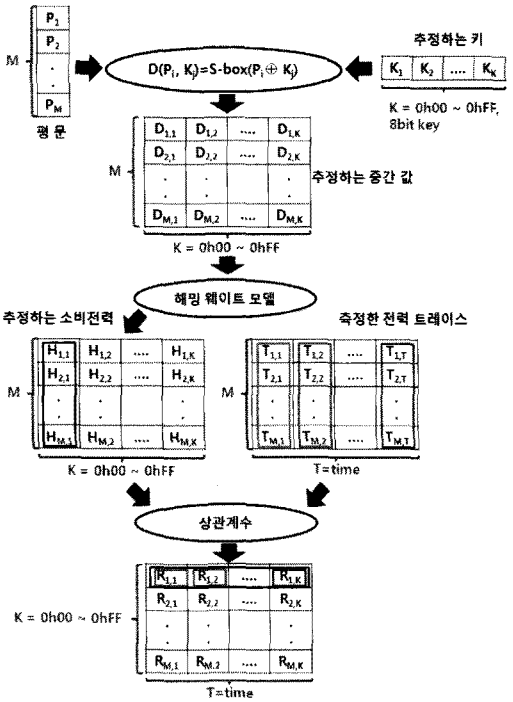


그림 1. AES S-box를 공격하기 위해 해밍 웨이트 모델을 이용한 CPA 공격 과정

함으로서 암호화 과정에 사용한 정확한 키 값과 $S(P_i \oplus K_j)$값이 출력되는 정확한 시점을 알 수 있게 된다. 이후 나머지 120비트 키들은 위의 절차를 반복함으로 모든 키를 구할 수 있게 된다.

$$R_{i,j} = \frac{\sum_{m=1}^{M}(H_{m,i} - \overline{H}_i)\cdot(T_{m,j} - \overline{T}_j)}{\sqrt{\sum_{m=1}^{M}(H_{m,i} - \overline{H}_i)^2 \cdot \sum_{m=1}^{M}(T_{m,j} - \overline{T}_j)^2}} \quad (1)$$

2.2 마스킹 기반 대응기법

마스킹 대응 기법은 랜덤한 마스크 값을 삽입하여 공격자로 하여금 분석하기 힘든 랜덤한 소비 전력 값이 발생하도록 함으로서 암호화 연산중에 발생하는 중간 값을 알 수 없게 하여 키와 데이터에 의존한 추정하는 모델 값과 실제 소비 전력 값 사이에 상관관계를 제거하는 방법이다. 이 방법은 구현 또한 쉽고 비용도 적게 들기 때문에, 차분전력분석 공격을 막기 위해 가장 많이 사용되고 있다.

마스킹에 대한 연구는 활발해서 여러 가지 방법으로 마스킹을 구현할 수 있다. 소프트웨어 적인 측면에서 마스킹을 구현하는 방법[4-9]도 있고, 하드웨어 적인 측면에서 마스킹을 구현하는 방법[18-19]도 있다. 또 어느 지점을 마스킹 하느냐에 따라 여러 방법이 존재한다. 본 논문에서는 3장에서 마스킹 된 암호 알고리즘에 2차 CPA 공격을 하기 위해 AES에 마스킹을 적용하였으며, Power Analysis Attacks[10]에 기술된 마스킹 기법에 근거하여 구현을 하였다. 구현한 마스킹에 대한 설명은 다음과 같다.

우선 AES의 암호화 연산 과정에서 중간 결과 값을 랜덤화 하기 위해서 S-box에 두 개의 랜덤한 마스크 m과 m'를 이용하여 다음과 같은 식 (2)의 성질을 만족시키는 마스킹이 적용된 S_m(masked S-box)을 생성한다.

$$S_m(p \oplus k \oplus m) = S(p \oplus k) \oplus m' \quad (2)$$

masked S-box의 입력 값 $p \oplus k \oplus m$은 키 값 k에 마스크 m을 XOR하고 AddRoundKey 단계에서 평문 p와의 XOR 연산을 거쳐 생성된 값이다. 이 입력 값이 masked S-box을 통과 했을 때의 출력 값 $S_m(p \oplus k \oplus m)$은 정상적인 입력이 원래의 S-box를 통과 한 후의 출력 값 $S(p \oplus k)$에 또 다른 마스크 m'을 XOR한 결과 값인 $S(p \oplus k) \oplus m'$와 같아지기 때문에 최종적인 출력 $S_m(p \oplus k \oplus m)$값에 마스크 m'를 XOR하게 되면 정상적인 $S(p \oplus k)$값이 도출 되게 된다. 결국 마스크 m을 통해 키 값을 랜덤하게 만들며 masked S-box을 통해서 SubByte단계를 지난 중간 결과 값을 공격자가 추정할 수 없게 만드는 효과를 주는 것이다. 이러한 마스킹 기법을 통해 CPA공격을 막음과 동시에 암호화 과정의 마지막 단계에서 m'를 XOR함으로 원래의 정상적인 암호 값을 구하기에 정상적인 암호화 과정 또한 수행할 수 있게 된다. 그림 2는 두 마스크 m과 m'을 이용한 마스킹 된 S-box인 masked S-box을 생성하는 알고리즘이다.

```
input : m, m'
output : m_sbox(p⊕k⊕m) = sbox(p⊕k)⊕m'
1. for i = 0 to 255 do
2.         m_sbox(i⊕m) = sbox(i)⊕m';
3. end for
4. return m_sbox
```

그림 2. 마스킹 된 S-box 생성을 위한 알고리즘

2.3 2차 CPA 공격

2.1장에서 설명한 CPA에 공격은 한 시점에 대한 공격으로 1차 DPA공격이라고 할 수 있다. 그래서 2.2에서 설명한 마스킹 대응 기법을 통해서 1차 DPA공

격을 막을 수 있게 된다. 하지만 마스킹 대응 기법의 취약성 또한 존재하는데 그것은 다음과 같다. 두 마스크 m과 m'가 같게 되는 경우 masked S-box의 입력 값 p⊕k⊕m과 masked S-box의 출력 값 S_m(p⊕k⊕m)을 XOR하면 마스크가 제거 된 (p⊕k)⊕S(p⊕k) 값을 구할 수 있게 된다. 왜냐하면 식 (2)에 의해 S_m(p⊕k⊕m) = S(p⊕k)⊕m' 이기 때문에 (p⊕k⊕m)⊕S_m(p⊕k⊕m) = (p⊕k⊕m)⊕S(p⊕k)⊕m' = (p⊕k)⊕S(p⊕k) (단 m=m')가 성립하는 것이다. 이렇게 되면 결론적으로 (p⊕k)⊕S(p⊕k)은 마스크가 제거된 정상적인 S-box의 입력 값과 출력 값에 대한 XOR연산 결과가 되기 때문에 이 결과는 마찬가지로 키와 데이터에 의존한 값이 되고 결국 측정한 소비전력 트레이스와의 상관계수를 구하여 키 값을 도출 해 낼 수 있게 된다. 문제는 하나의 시점이 아닌 S-box의 입력 값과 출력 값이라는 두 시점의 값을 알아야 하기 때문에 측정한 소비전력 트레이스 상에서 이 두 시점을 어떻게 찾고 이 두 시점의 XOR 연산 관계를 어떻게 이끌어 낼 것인가가 2차 CPA공격의 핵심 포인트다. 2차 CPA 공격을 위해서는 다음의 두 단계를 거치게 되는데 첫 번째 단계로 추정하는 소비전력 모델 측면에서는 위에서 설명한 XOR 연산 관계를 이용한 마스크가 제거된 값을 얻기 위해 식 (3)과 같은 해밍 웨이트 모델식을 사용하여 추정하는 소비 전력 모델을 세운다.

$$HW((p\oplus k\oplus m)\oplus(S(p\oplus k)\oplus m)) = HW((p\oplus k)\oplus S(p\oplus k)) \quad (3)$$

두 번째 단계는 측정한 소비전력 트레이스로부터 식 (3)과 동등한 소비전력 패턴을 얻기 위해서 사전처리 단계를 거친다. 이때 식 (3)에서 사용된 S-box의 입력 값과 출력 값이 측정된 소비전력 트레이스의 각각 어느 시점에 존재 하는지 알아야 하기 때문에 S-box에 해당하는 연산이 다 포함되어 있는 적절한 구간을 선택하여 골라낸 후 그림 3과 같이 그 구간에 대해서 서로 다른 두 시점마다 모든 경우의 수를 반영하여 사전처리 함수를 수행함으로 구간을 확장한다. 결국 원래의 구간 길이가 l일 경우 총 확장된 구간의 길이는 $(l-1)+(l-2)+...+2+1=l\cdot(l-1)/2$가 된다. 그림 3에서 t는 측정한 트레이스이며 pre()는 사전처리함수를 뜻하고 r은 해당하는 번째의 트레이스 수를 의미하며 +숫자는 해당하는 시간 포인트를 뜻한다. 사전처리함수를 사용하는 이유는 측정한 소비전력에서 두 시점에 대한 값의 관계가 식 (3)의 XOR 연산 후의

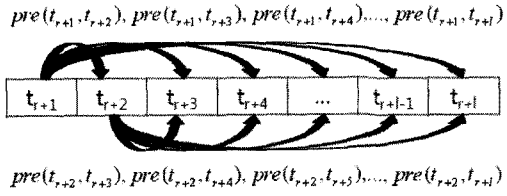


그림 3. 두 시점을 찾기 위한 구간 확장 과정과 사전처리 함수

해밍 웨이트를 취한 값과 동등한 관계를 가지도록 하기 위해서이다. 즉 사전처리 함수를 어떠한 것을 사용하느냐에 따라 얼마나 동등한 관계가 이루어질지가 결정되어 지고 이것은 공격자가 추정하는 소비전력 모델과 측정한 소비전력 트레이스 사이에 높은 상관관계를 이끌기 위한 아주 중요한 요소가 되게 된다.

Ⅲ. 기존의 사전처리함수와 제안하는 사전처리함수

이 장에서는 기존의 사전처리 함수들의 특징들을 현재 분석되어 있는 결과를 토대로 설명하며 제안하는 함수에 대한 특징과 우수성에 대해 설명한다.

3.1 기존의 사전처리함수

사전처리함수는 2.3장에서 설명했듯이 측정한 소비전력 트레이스에서 추정하는 소비전력 모델과 동등한 패턴의 결과를 이끌어 내기 위해 DPA공격 이전에 적용시켜 최대한 높은 상관계수를 이끌어 내는 함수를 말한다. 이러한 사전처리함수에는 현재 여러 형태의 함수가 있으며 이전에 연구 되어 진 대표적인 사전처리함수들로 두 포인터에 대한 곱인 $pre(t_a,t_b) = t_a \cdot t_b$[20], 두 포인터에 대한 차이 값의 절대치인 $pre(t_a,t_b) = |t_a-t_b|$[21,22], 두 포인터에 대한 최적화된 곱인 $pre(t_a,t_b) = (t_a-E[t_a]\cdot t_b-E[t_b])$[22], 두 포인터에 대한 합의 제곱인 $pre(t_a,t_b) = (t_a+t_b)^2$[15] 등이 있다. 이 함수들 중에서 추정하는 소비 전력 모델인 $HW(t_a\oplus t_b)$와 가장 높은 상관관계를 가지는 함수는 $pre(t_a,t_b) = (t_a-E[t_a]\cdot t_b-E[t_a])$이지만 이 함수는 실제 공격 시 연산의 오버헤드가 크기 때문에 제외하고 나머지 함수들 중에서 우리는 연산의 오버헤드가 많지 않으면서도 $HW(t_a\oplus t_b)$와의 상관계수 값이 높게 나타나는 함수를 찾기 위해 각각의 함수들과 $HW(t_a\oplus t_b)$와의 상관계수 값을 구하여 비교해 보았다. 이때 연산 결과 $HW(t_a\oplus t_b)$와 가장 높은 상관계수 값을 나타내는 함수는 $pre(t_a,t_b) = |t_a-t_b|$이며 그 이유는 표 1을 통해 알 수 있다.

표 1은 1비트 상에서 t_{am}과 t_{bm}의 값에 따른 $HW(t_a\oplus t_b)$가 가질 수 있는 모든 경우의 값들과 각각의 사

표 1. 사전처리함수에 따른 상관계수

	값				상관 계수
t_{am}(masked a시점)	0	0	1	1	
t_{bm}(masked b시점)	0	1	0	1	
$HW(t_a \oplus t_b)$	0	1	1	0	
$HW(t_{am}) \cdot HW(t_{bm})$	0	0	0	1	-0.57
$\lvert HW(t_{am})-HW(t_{bm})\rvert$	0	1	1	0	1
$(HW(t_{am})+HW(t_{bm}))^2$	0	1	1	4	-0.33

전처리함수들을 통해 얻은 결과 값과의 상관관계를 보여 주는 것이다. 표 1에서 확인 할 수 있듯이 $|HW(t_{am}) - HW(t_{bm})|$ 함수에서 상관계수가 1로 가장 높게 뜨는 것을 확인 할 수 있다. 하지만 1비트에서 8비트로 비트 수가 늘어나게 되면 표 2와 같이 상관계수가 떨어지는 것을 볼 수 있다[10]. 이것은 각각의 사전처리 함수들이 $HW(t_a \oplus t_b)$ 연산 결과 값을 정확하게 표현하지 못하기 때문이다. 그리고 실제 공격 환경에서는 노이즈 영향을 고려해야한다. 그래서 우리는 노이즈가 있는 환경을 새롭게 모델링 하고 노이즈가 존재하는 환경에서 $HW(t_a \oplus t_b)$ 연산 결과 값과의 상관계수를 구했을 때 기존의 사전처리함수들 보다 더 높은 상관계수 값을 이끌어내는 새로운 사전처리함수를 제안한다.

표 2. 비트 수에 따른 사전처리함수의 상관 계수

	t_{am}과 t_{bm}의 비트 수			
	1	2	4	8
$HW(t_{am}) \cdot HW(t_{bm})$	-0.58	0.32	-0.17	-0.09
$\lvert HW(t_{am})-HW(t_{bm})\rvert$	1.00	0.53	0.34	0.24
$(HW(t_{am})+HW(t_{bm}))^2$	-0.33	-0.16	0.08	-0.04

3.2 제안하는 사전처리 함수

실제로 전력을 측정하는 상황에서는 내부 신호나 cell의 출력 신호가 변할 때 발생하는 동적인 전력 소모량 외에도 고정적인 전력 소모량과 노이즈 값이 존재하기 때문에 8비트 상에서 t_a와 t_b가 이론상으로는 $00000000_{(2)}$인 값이 존재하지만 실제 측정된 전력 값은 0의 값으로 존재하지 않는다. 그래서 우리는 이를 고려하여 모델링을 하였다. 먼저 노이즈 값이 반영된 실제 모델 환경을 모델링하기 위해서 2차 CPA 공격 상에 두 시점에 대한 0과 1일 때 소비되는 전력을 각각 다음과 같이 정의한다.

X_0 : 첫 번째 시점의 0의 소비전력
X_1 : 첫 번째 시점의 1의 소비전력
Y_0 : 두 번째 시점의 0의 소비전력
Y_1 : 두 번째 시점의 1의 소비전력

그리고 우리는 다음과 같은 가정을 두고 모델링을 하였다.

- 가정 1 : 첫 번째 시점의 X_0와 X_1의 차이 값 비율은 두 번째 시점의 Y_0와 Y_1의 차이 값 비율과 같다.
- 가정 2 : 첫 번째 시점의 0의 소비전력 값과 두 번째 시점의 0의 소비전력 값은 노이즈에 의해 서로 다르다.

따라서 위의 가정을 기반으로 다음과 같은 모델식이 유도된다.

$$X_0 = P_{const};$$
$$X_1 = \alpha * X_0 + \beta;$$
$$Y_0 = P_{const} + P_{noise};$$
$$Y_1 = \alpha * Y_0 + \beta;$$

P_{const}는 0일 때 발생하는 소비전력 값을 의미한다. P_{noise}는 노이즈로 인해 발생하는 서로 다른 시점에서 0일 때 발생하는 소비전력 값 차이를 의미한다. α, β는 0의 소비전력에 비해 1의 소비전력이 얼마나 큰 소비전력 값을 발생시키는지를 나타낸다.

예를 들어 첫 번째 시점을 X, 두 번째 시점을 Y, $P_{const} = 1$, $P_{noise} = 0.2$, $\alpha = 2$, $\beta = 0$으로 하고 이때의 8비트 값이 $X = 00000000_{(2)}$인 경우와 $Y = 10010100_{(2)}$인 경우의 소비 전력 값을 계산 해본다면 처음 초기 설정 값은 다음과 같을 것이다.

$$X_0 = P_{const} = 1,$$
$$X_1 = \alpha * X_0 + \beta = 2 * 1 + 0 = 2,$$
$$Y_0 = P_{const} + P_{noise} = 1 + 0.2 = 1.2,$$
$$Y_1 = \alpha * Y_0 + \beta = 2 * 1.2 + 0 = 2.4$$

그리고 이것을 이용해서 먼저 $X = 00000000_{(2)}$을 계산해보면 0이 8bit고 1이 0bit 이므로 첫 번째 시점 X의 소비전력은 $(X_0 * 8) + (X_1 * 0) = (1 * 8) + (2 * 0) = 8$ 이 된다. 이때 두 번째 시점 $Y = 10010100_{(2)}$을 계산해보면 0이 5bit 1이 3bit 이므로 두 번째 시점 Y의 소비전력은 $(Y_0 * 5) + (Y_1 * 3) = (1.2 * 5) + (2.4 * 3) = 13.2$ 가 된다.

우리는 앞에서 제시한 모델 환경에서 두 시점에 대한 노이즈 차이 값을 잡아내면서 동시에 추정하는 소비 전력 모델인 $HW(t_a \oplus t_b)$와도 높은 상관계수 값을 이끌어 내기위해 서로 각각의 상대 시점을 고려한 t_{am}

값과 t_{bm}의 비율을 적용하고 그 비율사이에 차이 값을 구하는 새로운 사전처리함수를 제안한다. 식 (4)는 두 시점의 비율 값과 그 비율 값 사이에 차이 값이 적용된 제안하는 사전처리함수이다.

$$\left|\frac{HW(t_{bm})}{HW(t_{am})}-\frac{HW(t_{am})}{HW(t_{bm})}\right| \quad (4)$$

여기서 우리는 일단 노이즈가 없는 환경에서 1일 때 소비전력 값이 0일 때 소비전력 보다 2배정도 크다고 보고 P_{noise}=0, P_{const}=1, α=2, β=0로 두어 1비트부터 8비트까지 $HW(t_a \oplus t_b)$ 연산 결과 값과 두 사전처리 함수 $|HW(t_{am}) - HW(t_{bm})|$, $|HW(t_{am})/HW(t_{bm}) - HW(t_{bm})/HW(t_{am})|$의 상관계수 값을 각각 구하여 비교해 보았다. 표 3은 그 결과를 보여준다. ($HW(t_{am}) \rightarrow t_a$, $HW(t_{bm}) \rightarrow t_b$)

우리가 모델링한 환경에서 기존의 사전처리함수 $|t_a-t_b|$의 상관계수 결과 값이 Prouff, E[22]등이 제시한 이상적인 모델에서의 $|t_a-t_b|$ 함수 상관계수 결과 값과 같음을 표 3과 표 4를 통해 확인할 수 있다.

표 3을 통해 알 수 있는 정보는 노이즈가 전혀 없는 환경에서는 비트가 증가하면 할수록 사전처리함수 $|t_a-t_b|$가 제안하는 사전처리함수 $|t_a/t_b-t_b/t_a|$보다 상관계수가 높으며 그 차이 정도도 증가 한다는 것이다. 하지만 실제 공격 환경에서 노이즈가 전혀 없을 수는 없기 때문에 노이즈가 존재하는 8비트 환경에서 노이즈가 얼마나 존재하느냐에 따라 두 사전처리함수가 나타내는 상관계수 값이 어떻게 달라지는지를 비교할 필요가 있다. 표 5는 노이즈가 존재하는 환경에서 1일 때 소비전력 값이 0일 때 소비전력 보다 2배정도 크다고 보고 P_{noise}=(0.1~0.8), P_{const}=1, α=2, β=0로 두어 8비트 상에서 노이즈 변화를 0.1~0.8로 주었을 때 $HW(t_a \oplus t_b)$ 연산 결과 값과 두 사전처리 함수 $|HW(t_{am}) - HW(t_{bm})|$, $|HW(t_{am})/HW(t_{bm}) - HW(t_{bm})/HW(t_{am})|$의 상관계수 값을 각각 구하여 비교해 본 결과이다. ($HW(t_{am}) \rightarrow t_a$, $HW(t_{bm}) \rightarrow t_b$)

표 5에서 확인할 수 있듯이 노이즈가 증가할수록 제안하는 사전처리함수 $|t_a/t_b-t_b/t_a|$가 더 큰 차이로 기존의 사전처리함수 $|t_a-t_b|$보다 높은 상관계수를 나타냄을 알 수 있다. 이러한 결과는 앞에서 언급했듯이 t_a/t_b 또는 t_b/t_a 연산을 통해 제안한 함수 자체가 한쪽의 시점이 아닌 다른 쪽 시점까지 고려함으로 두 시점 사이에 노이즈로 인해 발생하는 차이 값을 잡아내기 때문이다. 따라서 공격자가 공격하는 환경이 노이즈가 많이 발생하는 환경이라면 제안하는 전처리 함수를 사용하는 것이 더 유용함을 알 수 있다.

Ⅳ. 실험 및 결과 분석

4.1 실험 환경

주요 실험 장비로 공격 대상 칩은 PIC18F452 마이크로 컨트롤러를 사용하였고 AES-128비트를 공격할 암호 알고리즘으로 칩에 구현하였다. 칩의 동작 과정을 측정하기 위해 Agilent사의 MS06102A 오실로스

표 3. 노이즈가 존재하지 않는 환경에서의 제안한 사전처리 함수 $|t_a/t_b-t_b/t_a|$와 기존의 사전처리 함수 $|t_a-t_b|$의 상관계수 값 비교

함수 \ bit	1	2	3	4	5	6	7	8
$\|t_a/t_b-t_b/t_a\|$	1	0.5345	0.4109	0.3468	0.3057	0.2764	0.2542	0.2366
$\|t_a-t_b\|$	1	0.5345	0.4121	0.3486	0.3079	0.2789	0.2568	0.2394
차이정도	0	0	-0.0012	-0.0018	-0.0022	-0.0025	-0.0026	-0.0028

표 4. 이상적인 환경 내에서 $|t_a-t_b|$ 함수의 상관계수[22]

n	1	2	3	4	5	6	7	8
H	1.00	0.53	0.41	0.35	0.31	0.28	0.26	0.24

표 5. 노이즈가 존재하는 환경에서의 제안한 사전처리 함수 $|t_a/t_b-t_b/t_a|$와 기존의 사전처리 함수 $|t_a-t_b|$의 상관계수 값 비교

함수 \ 노이즈	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$\|t_a/t_b-t_b/t_a\|$	0.1790	0.0987	0.0500	0.0270	0.0180	0.0152	0.0151	0.0158
$\|t_a-t_b\|$	0.1782	0.0938	0.0424	0.0172	0.0065	0.0022	0.0006	0.0002
차이정도	+0.0008	+0.0049	+0.0076	+0.0098	+0.0115	+0.0130	+0.145	+0.0156

코프를 사용하였다. 또한 전원 공급을 위해 DAP-3005t모델의 power supply를 사용하여 전압은 5V, 전류는 0.5A를 주었고 전력 소비를 측정하기 위해 1Ω 시멘트 저항을 삽입하였다. 그림 4는 앞서 설명한 부채널 공격을 위한 실험 환경으로 오실로스코프의 ㉮의 프로브로 시멘트 저항 양단을 연결하여 소비 전력을 측정하게 하고 또한 여러 전력 트레이스간의 시작 지점을 일치시키기 위하여 General Purpose I/O 핀을 이용하여 암호화가 시작될 때 별도의 시작 신호를 트리거 하도록 하였다. 그리고 이 신호를 ㉯의 프로브를 이용하여 측정하며, 이를 트리거 신호로 사용하여 시작점을 일치시킨 상태에서 원하는 구간의 전력 파형을 측정 할 수 있게 하였다.

그리고 계측기를 통한 소비 전력 측정을 Vee-Pro 언어를 이용하여 자동화함으로 그림 5와 같이 오실로스코프에서 측정한 전력 파형을 필요로 하는 개수만큼 지정한 파일이름으로 쉽게 저장할 수 있게 하였다. 마지막으로 공격자가 추정하는 소비전력 모델을 Matlab 상에서 구현한 뒤 측정한 파형들과의 상관계수를 구하여 최종적인 결과 값을 이끌어 내었다.

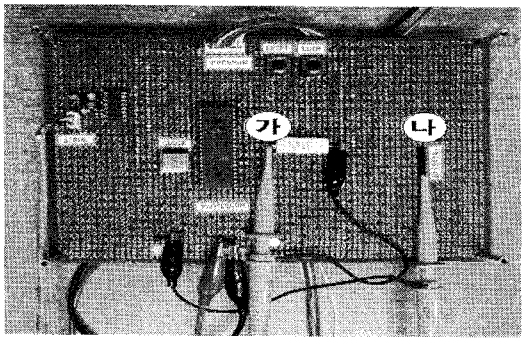


그림 4. 부채널 공격을 위한 실험 환경

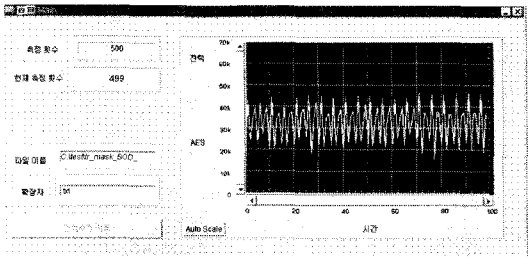


그림 5. 오실로스코프를 이용한 파형 측정

4.2 1차 CPA 및 마스킹 기반의 2차 CPA

먼저 1차 CPA 공격이 성공적으로 이루어지는 것을 확인하기 위해 시간 축으로 10000point 그리고 총 1000개의 소비전력 트레이스를 측정하였고 AES 128 비트 키, 즉 1byte키 16개를 실험하여 모든 키가 정확하게 추출되는 것을 확인하였다. 그림 6은 첫 번째 키인 16진수 값 '1A'에 대한 키 추출 성공 및 각 키에 대한 상관계수 수치를 보여준다. 다음으로 CPA 공격을 막기 위한 마스킹 기법을 공격 타겟 보드에 구현한 뒤 2차 CPA 공격이 아닌 일반적인 CPA공격을 막을 수 있는지를 실험 하였고 그림 7과 그림 8에서 볼 수

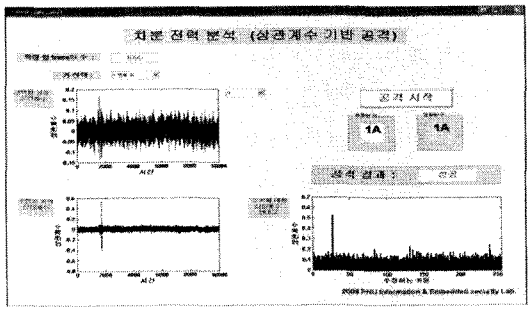


그림 6. CPA 공격 시뮬레이션 결과

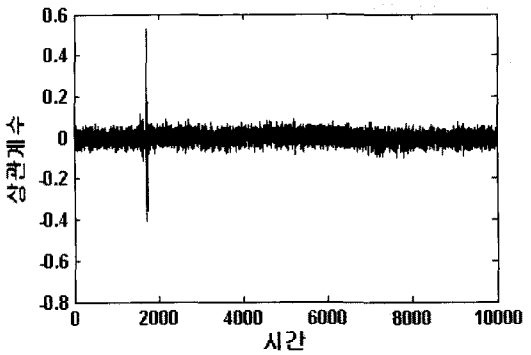


그림 7. 마스킹 기법이 사용되지 않은 타겟의 첫 번째 키에 CPA 공격 결과

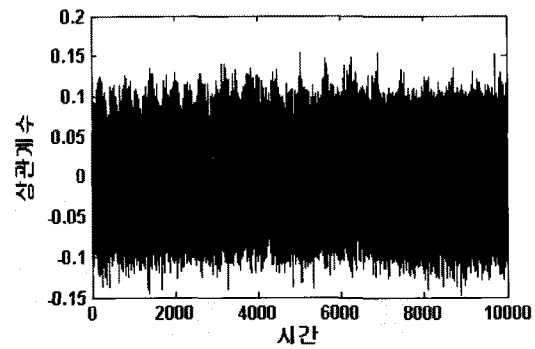


그림 8. 마스킹 기법이 사용 된 타겟의 첫 번째 키에 CPA 공격 결과

있듯이 마스킹 기법이 적용되지 않은 경우 0.5이상의 높은 상관계수로 키가 추출되지만 마스킹이 적용된 경우 낮은 상관계수 값으로 정확한 키 추출에 실패 하는 것을 확인할 수 있었다. 이로써 마스킹이 적용 된 경우 한 시점에 대한 공격인 1차 CPA공격으로는 키를 추출 할 수 없음이 검증 되었다.

다음은 2차 CPA공격으로 마스킹 기법이 적용 된 타겟에 공격이 성공하는지를 실험하기 위해 여러 가지 기존의 사전처리함수 중 가장 높은 상관계수를 보여준 $pre(t_a,t_b) = |t_a-t_b|$를 이용하여 1차 CPA공격 때와 동일하게 S-box 16개를 계산하는 구간 전체를 시간축으로 10000point로 설정하고 총 2000개의 소비 전력 트레이스를 측정한 뒤 2차 CPA 공격을 수행 하였다. 이때 키 값을 변경한 경우도 결과가 예상대로 잘 나오는지를 확인하기 위해 16byte의 키를 새로 설정하여 공격하였다. 2차 CPA 공격은 사전처리 단계에서 트레이스 구간이 확장되기 때문에 공격에 필요로 하는 두 시점이 들어 있는 최소한의 영역만 가지고 공격하는 것이 전체 공격 연산 수행에 있어 효율적이다. 그래서 먼저 visual inspections을 통해서 전체 10000point의 구간 중에 16개 S-box연산과 관련이 없는 앞쪽 830point 이하 부분과 뒤쪽 9200point 이상부분을 잘라내고 남은 8370point 구간을 5분의 1인 1674point로 compression을 하였다. 이때 사용한 compression 기법은 Raw Integration[10]을 사용하였고 1674point를 다시 1개의 99point와 15개의 105point로 16개의 S-box구간을 하나씩 나누었다. 그리고 하나의 S-box의 입력과 출력 값이 모두 포함되어 있는 시간 영역 105point를 선택하여 사전처리 단계를 거친 후 총 $l \cdot (l-1)/2 = 105 \cdot (105-1)/2 = 5460$point로 구간을 확장하였다. 그림 9와 그림 10은 세 번째 키에 대한 2차 CPA 공격 결과이다. 추출하려

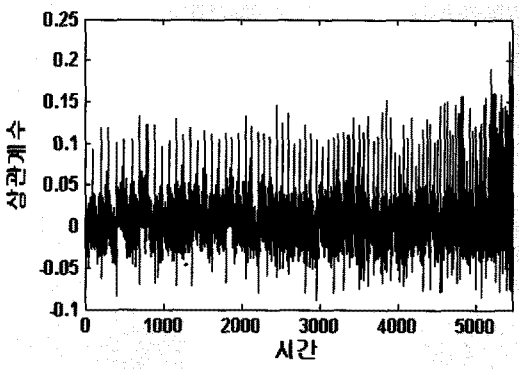


그림 9. 마스킹 기법이 사용 된 타겟의 세 번째 키에 2차 DPA 공격 결과

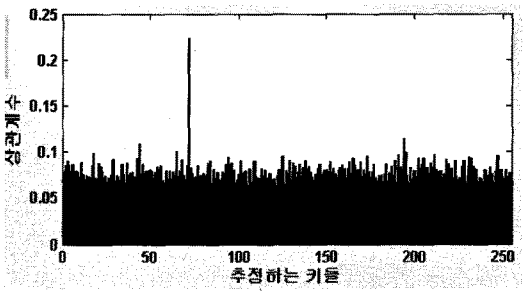


그림 10. 각각의 추정하는 키들이 가지는 최대 상관계수

는 정확한 키 값은 10진수 71이었고 마스킹의 영향으로 상관계수 값은 전체적으로 1차 CPA 공격 보다는 낮게 나왔지만 키 값은 71을 정확히 추출 하였다. 그림 9는 추정하는 키 값이 71일 때 5460 포인트에 발생한 상관계수 값들을 다 표현한 것이고 그림 10은 추정하는 모든 키 0 ~ 255 사이의 값들마다 가지고 있는 가장 높은 상관계수 값들을 다 표현한 것이다. 추정하는 키 값 71 지점에서 가장 높은 상관계수가 발생하는 것을 확인 할 수 있다.

4.3 사전처리함수 결과 분석

2차 CPA공격에 적용한 제안된 새로운 사전처리함수의 성능을 확인하기 위해, AES 암호화 알고리즘의 첫 번째 라운드에 사용된 16byte의 키를 모두 적용하여 파형을 측정하였다. 성능을 평가할 사전처리 함수는 연산의 오버헤드가 $pre(t_a,t_b) = (t_a-E[t_a] \cdot t_b-E[t_b])$보다 적으면서도 기존의 가장 우수한 성능을 보이는 $pre(t_a,t_b) = |t_a-t_b|$와 제안한 $pre(t_a,t_b) = |t_a/t_b-t_b/t_a|$ 함수를 선택했다. 그리고 각각의 키에 대해서 총 2000개의 소비전력 트레이스를 측정하였고 1000개, 1300개, 1500개, 1700개, 2000개의 트레이스 수로 각각의 사전처리함수를 이용하여 2차 CPA 공격을 하였으며 그 결과는 표 6에 나타내었다.

표 6에서 확인 할 수 있듯이 두 가지 사전처리함수는 AES 첫 번째 라운드의 16byte의 키 모두를 정확하게 추출 하는데 성공하였고 두 사전처리함수의 가장 높은 피크가 뜨는 correct peak 상관계수를 비교해 보면 16byte의 키 모두에서 제안한 $|t_a/t_b-t_b/t_a|$ 함수가 $|t_a-t_b|$ 함수에 비해 더 높은 상관계수 값을 나타내는 것을 볼 수 있다. 그리고 상관계수 peak를 보면 각 S-box별로 키를 추출하는데 필요한 correct peak가 있는 반면 두 번째로 높은 peak인 ghost peak가 존재한다. 이 때 correct peak/ghost peak을 계산해보면 이 수치가 높으면 높을수록 correct peak와 ghost peak의 차이 비율이 크다는 것을 의미하기 때문에 더 정확하

표 6. 사전처리함수에 따른 2차 CPA 공격 결과 (1000~2000개 소비 전력 트레이스)

키 값	사전처리함수	상관계수		C/G ratio	트레이스 수	전체 공격 수행시간(s)	전처리 함수 수행시간(s)	수행시간 비율(%)
		Correct peak	Ghost peak					
1번째 키	$\lvert t_a-t_b\rvert$	0.2459	0.1519	1.6188	1000	21.633128	16.482734	76.1921
'0x49'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2473	0.1522	1.6248	1000	21.746754	16.505073	75.8967
2번째 키	$\lvert t_a-t_b\rvert$	0.2447	0.1201	2.0375	1700	65.915489	51.810816	78.6019
'0x38'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2449	0.1196	2.0477	1700	66.313842	52.369307	78.9719
3번째 키	$\lvert t_a-t_b\rvert$	0.2233	0.1138	1.9622	2000	88.463865	70.834477	80.0717
'0x47'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2238	0.1133	1.9753	2000	89.408189	71.912635	80.4318
4번째 키	$\lvert t_a-t_b\rvert$	0.2287	0.1084	2.1098	2000	87.945327	71.374159	81.1574
'0x56'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2310	0.1088	2.1232	2000	88.870760	71.668603	80.6436
5번째 키	$\lvert t_a-t_b\rvert$	0.2298	0.1113	2.0647	2000	88.658661	70.789058	79.8445
'0x65'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2312	0.1103	2.0961	2000	88.792895	71.192556	80.1782
6번째 키	$\lvert t_a-t_b\rvert$	0.1943	0.1114	1.7442	2000	88.930969	71.490519	80.3888
'0x74'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.1951	0.1115	1.7498	2000	89.597630	72.015147	80.3762
7번째 키	$\lvert t_a-t_b\rvert$	0.2453	0.1136	2.1593	2000	89.492118	70.775727	79.0860
'0x83'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2456	0.1129	2.1754	2000	88.910360	72.444773	81.4807
8번째 키	$\lvert t_a-t_b\rvert$	0.2450	0.1340	1.8284	1500	51.281031	41.049427	80.0480
'0x92'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2457	0.1338	1.8363	1500	51.431888	41.411444	80.5171
9번째 키	$\lvert t_a-t_b\rvert$	0.2580	0.1199	2.1518	1500	51.637239	40.430907	78.2980
'0x11'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2586	0.1197	2.1604	1500	51.817387	41.047224	79.2152
10번째 키	$\lvert t_a-t_b\rvert$	0.2170	0.1189	1.8251	2000	88.092172	72.279584	82.0500
'0x20'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2181	0.1195	1.8251	2000	88.844894	74.442511	83.7893
11번째 키	$\lvert t_a-t_b\rvert$	0.2213	0.1172	1.8882	2000	88.886923	71.521078	80.4630
'0x39'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2226	0.1172	1.8993	2000	89.022827	71.726423	80.5708
12번째 키	$\lvert t_a-t_b\rvert$	0.2308	0.1531	1.5075	1000	23.249300	18.415832	79.2103
'0x48'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2328	0.1532	1.5196	1000	23.651033	18.451638	78.0162
13번째 키	$\lvert t_a-t_b\rvert$	0.2462	0.1313	1.8751	1300	38.911424	30.350860	77.9999
'0x57'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2469	0.1304	1.8934	1300	38.971068	30.540710	78.3676
14번째 키	$\lvert t_a-t_b\rvert$	0.2456	0.1093	2.2470	2000	89.483905	72.008156	80.4705
'0x66'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2459	0.1080	2.2769	2000	90.222466	72.153193	79.9725
15번째 키	$\lvert t_a-t_b\rvert$	0.1774	0.1106	1.6040	2000	88.820628	71.471669	80.4674
'0x75'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.1788	0.1112	1.6079	2000	89.265402	71.590366	80.1995
16번째 키	$\lvert t_a-t_b\rvert$	0.2515	0.1517	1.6579	1000	23.656586	17.963199	75.9332
'0x84'	$\lvert t_a/t_b-t_b/t_a\rvert$	0.2521	0.1569	1.6068	1000	23.718539	17.997170	75.8781

게 키 값을 추출할 수 있음을 알 수 있다. 이러한 두 peak의 차이 비율 값을 나타내는 C/G(Correct peak / Ghost peak) ratio를 표 6에 나타내었고 16byte의 키 모두에서 제안한 $|t_a/t_b-t_b/t_a|$ 함수가 $|t_a-t_b|$ 함수에 비해 C/G ratio 값이 크기 때문에 정확한 키를 추출하는데 있어 더 성능이 좋음을 확인할 수 있다. 그리고 수행시간 측면에서는 Matlab 상에서 전체 소스 코드의 수행 시간과 전처리 함수 부분만 수행 했을 때의 수행시간을 측정하였고 각각의 전처리 함수가 전체 공격 수행시간에 대해 몇 퍼센트의 시간을 소비하는지를 비율로 알아보았다. 표 6에 수행시간 비율을 통해 알 수 있는 것은 $|t_a-t_b|$ 함수에 비해 제안한 $|t_a/t_b-t_b/t_a|$ 함수의 수행 시간 overhead가 -1% ~ +1%로 거의 없다는 것이다. 따라서 공격 수행시간에 있어서도 전혀 문제가 되지 않는 것을 확인 할 수 있다. 마지막으로 표 7은 16byte의 키를 추출하는데 필요한 최소한의 트레이스 수 즉 C/G ratio가 1 보다 커지기 시작할 때의 트레이스 수를 나타낸 것이고 이때 $|t_a-t_b|$ 함수와 제안한 $|t_a/t_b-t_b/t_a|$ 함수 사이에 트레이스 수 차이가 얼마나 발생하는지를 나타낸 것이다. 제안한 $|t_a/t_b-t_b/t_a|$ 함수가 $|t_a-t_b|$ 함수에 비해 더 많은 키에서 필요한 트레이스 수를 줄임을 확인할 수 있다.

결론적으로 제안한 함수가 비슷한 전체 공격 수행시간 동안 같은 트레이스수를 가지고 더 높은 상관계수를 이끌어 내었으며 키 추출에 필요한 최소한의 트레이스 또한 제안한 함수가 더 많은 키에서 기존의 함수보다 필요한 트레이스 수를 더 줄였음을 확인할 수 있었다.

표 7. 사전처리 함수에 따른 16byte의 키를 추출하는데 필요한 최소한의 트레이스 수

키 값	사전처리함수	C/G ratio	키 추출에 필요한 최소 트레이스 수	줄어든 트레이스 수
1번째 키	$\|t_a-t_b\|$	1.0186	350	0
'0x49'	$\|t_a/t_b-t_b/t_a\|$	1.0124	350	
2번째 키	$\|t_a-t_b\|$	1.0247	225	0
'0x38'	$\|t_a/t_b-t_b/t_a\|$	1.0092	225	
3번째 키	$\|t_a-t_b\|$	1.0114	275	0
'0x47'	$\|t_a/t_b-t_b/t_a\|$	1.0142	275	
4번째 키	$\|t_a-t_b\|$	1.0196	292	0
'0x56'	$\|t_a/t_b-t_b/t_a\|$	1.0094	292	
5번째 키	$\|t_a-t_b\|$	1.0119	396	+2
'0x65'	$\|t_a/t_b-t_b/t_a\|$	1.0461	394	
6번째 키	$\|t_a-t_b\|$	1.0196	766	0
'0x74'	$\|t_a/t_b-t_b/t_a\|$	1.0100	766	
7번째 키	$\|t_a-t_b\|$	1.0040	240	+1
'0x83'	$\|t_a/t_b-t_b/t_a\|$	1.0005	239	
8번째 키	$\|t_a-t_b\|$	1.0162	254	0
'0x92'	$\|t_a/t_b-t_b/t_a\|$	1.0192	254	
9번째 키	$\|t_a-t_b\|$	1.0024	339	-1
'0x11'	$\|t_a/t_b-t_b/t_a\|$	1.0106	340	
10번째	$\|t_a-t_b\|$	1.0029	236	-1
키 '0x20'	$\|t_a/t_b-t_b/t_a\|$	1.0196	237	
11번째	$\|t_a-t_b\|$	1.0073	140	+1
키 '0x39'	$\|t_a/t_b-t_b/t_a\|$	1.0018	139	
12번째	$\|t_a-t_b\|$	1.0113	330	0
키 '0x48'	$\|t_a/t_b-t_b/t_a\|$	1.0121	330	
13번째	$\|t_a-t_b\|$	1.0248	423	+1
키 '0x57'	$\|t_a/t_b-t_b/t_a\|$	1.0083	422	
14번째	$\|t_a-t_b\|$	1.0003	330	-1
키 '0x66'	$\|t_a/t_b-t_b/t_a\|$	1.0014	331	
15번째	$\|t_a-t_b\|$	1.0122	602	+5
키 '0x75'	$\|t_a/t_b-t_b/t_a\|$	1.0080	597	
16번째	$\|t_a-t_b\|$	1.0345	269	0
키 '0x84'	$\|t_a/t_b-t_b/t_a\|$	1.0394	269	

V. 결 론

본 논문에서는 물리적 공격 중 대표적인 부채널 공격인 마스킹 기법에 대한 2차 CPA 공격에 대해 알아보고 실제 공격 환경에 적합한 노이즈가 존재하는 환경을 모델링함으로서 이러한 환경에서 더 좋은 성능을 내는 새로운 사전처리함수를 제안하였다. 이렇게 제안한 사전처리함수에 대해 8비트 상에 발생할 수 있는 모든 경우의 수를 고려하여 차이 값의 절대치를 취한 사전처리함수와 상관계수를 비교하였으며 노이즈가 많은 환경일수록 제안한 사전처리함수가 더 좋은 성능을 보임을 이론적으로 증명하였다. 그리고 실험 환경을 구축하여 실제 2차 CPA 공격을 구현 후 사전처리함수에 따른 공격의 성능을 분석하였으며 실험 결과, 같은 수의 트레이스 상에서 제안한 사전처리함수가 기존의 우수한 성능을 보인 두 포인터에 대한 차이 값의 절대치 사전처리함수에 비해 더 높은 상관계수를 나타냄을 확인하였다. 또한 제안한 사전처리함수가 기존의 사전처리함수보다 더 많은 키에서 키 추출에 필요한 최소한의 트레이스 수를 줄였음을 확인 하였다. 결국 제안한 사전처리함수를 이용한 2차 CPA 공격은 기존의 2차 CPA 공격의 강도와 효율성을 높임으로서 실제 노이즈가 존재하는 환경인 AMI용 스마트미터기나 Zigbee 디바이스에 구현되어 있는 비밀키 값을 보다 빠르고 정확하게 낼 수 있었으며 이 실험을 통해 2차 CPA 공격이 결국 IT 융합 보안 분야에 있어 상당히 위협적인 물리적 공격임을 검증하였다.

참 고 문 헌

[1] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall, "Side Channel Cryptanlysis of Product Cipher," Proceedings of ESORICS'98, pp.97-112, Springer-Verlag, Sep. 1998. 115-126, 1997

[2] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall, "Side Channel Cryptanlysis of Product Cipher (final version)," in the site, 2000.

[3] P. Kocher, J. Jaffe and B.Jun, "Differential Power Analysis," CRYPTO'99, LNCS 1666, pp.388-397, Springer-Verlag, 1999.

[4] M. L. Akkar and C. Giraud. "An Implementation of DES and AES, Secure against Some Attacks," In CHES2001, LNCS, Vol.2162, pp.309-318, Springer-Verlag, 2001.

[5] J. D. Golic and C. Tymen. "Multiplicative masking and power analysis of AES," In CHES2002, LNCS, Vol.2523, pp.198-212, Springererlag, 2002.

[6] T. S. Messerges, "Securing the AES finalists against power analysis attacks, In FSE'00, LNCS 1978, pp.150-164, Springer-Verlag, 2000.

[7] E. Trichina, D. D. Seta, and L. Germani. "Simplified adaptive multiplicative masking for AES," In CHES'02, LNCS 2535, pp.187-197, Springer-Verlag, 2003.

[8] J. Blomer, J. Guajardo, and V. Krummel, "Provably secure masking of AES", in Proc. SAC'04, LNCS 3357, pp.69-83, Springer-Verlag, 2004.

[9] E. Oswald, S. Mangard, and N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the AES S-box," In FSE'05, LNCS 3557, pp.413-423, Springererlag, 2005.

[10] S. Mangard, E. Oswald, and T. Popp, Power Analysis Attacks: Revealing the Secrets of Smart Cards, 2007 Springer Science+Business Media, LLC. pp 167-272.

[11] C. Clavier, J-S. Coron, and N. Dabbous. "Differential power analysis in the presence of hardware countermeasures", in Proc. CHES2000, LNCS, Vol.1965, pp.252-263, Springer-Verlag, 2000.

[12] K. Tiri, M. Akmal and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," In ESSCIRC'02, 2002.

[13] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," In DATE'04, pp.246-251, 2004.

[14] T. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software,"' In CHES'00, LNCS 1965, pp.238-251, Springer-Verlag, 2004.

[15] J. Waddle and D. Wagner, "Towards Efficient Second-Order Power Analysis,"' In CHES'04, LNCS 3156, pp.1-15, Springer-Verlag, 2004.

[16] M. Joye, P. Paillier, and B. Schoenmakers, "On Second-Order Differential Power Analysis,"' In CHES'05, LNCS 3659, pp.293-308, Springer-Verlag, 2005.

[17] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in Proceedings of CHES 2004, LNCS 3156, pp.16-29, 2004.

[18] T. Messerges, E. A. Dabbish and L. Puhl, "Method and apparatus for preventing information leakage attacks on a micro-electronic assembly," U.S. Patent 6,295,606 B1, Sep. 2001.

[19] E. Trichina, "Combinational logic design for AES subbyte transformation on masked data," Cryptology ePrint Archive, Report 2003/236, 2003.

[20] S. Chari, C. Jutla, J. Rao, and P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks", "In CRYPTO'99, LNCS 1666, pp.398-412. Springer-Verlag, 1999.

[21] T. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software,"' In CHES'00, LNCS 1965, pp.238-251, Springer-Verlag, 2004.

[22] Prouff, E., Rivain, M., B´evan, R.: Statistical Analysis of Second Order Differential Power Analysis. IEEE Transactions on Computers (58-6), 799-811 (2009)

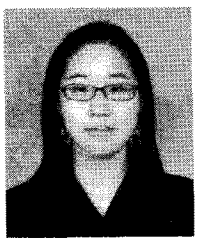
이 철 희 (Chulhee Lee) 준회원

2009년 2월 동명대학교 멀티미디어공학과 학사

2009년 3월~현재 부산대학교 컴퓨터공학과 석사과정

<관심분야> 부채널 공격 기법 및 대응, 암호 이론, VLSI 설계, embedded System

황 아 름 (Ahreum Hwang) 준회원



2010년 2월 부산대학교 정보컴퓨터공학부 학사

2010년 3월~현재 부산대학교 물류IT학과 석사과정

<관심분야> 부채널 공격 기법 및 대응, RFID/USN 정보보호 기술

이 동 건 (Donggeon Lee) 준회원

2009년 2월 부산대학교 정보컴퓨터공학부 학사

2009년 3월~현재 부산대학교 컴퓨터공학과 석사과정

<관심분야> RFID/USN 정보보호 기술, 페어링 기반 암호 이론, VLSI 설계, embedded system

김 형 남 (Hyoungnam Kim) 종신회원

1993년 2월 포항공과대학교 전자전기공학과 공학사

1995년 2월 포항공과대학교 전자전기공학과 공학석사

2000년 2월 포항공과대학교 전자전기공학과 공학박사

2000년 4월 포항공과대학교 전자컴퓨터공학부 박사후 연구원

2003년 3월 한국전자통신연구원 무선방송연구소 선임 연구원

2007년 2월 부산대학교 전자공학과 조교수

2007년 3월~현재 부산대학교 전자공학과 부교수

<관심분야> 적응신호처리, 레이더 및 소나시스템, 생체신호처리, 디지털 TV, 디지털 통신, OFDM 시스템, RFID, 멀티미디어 시스템

김 호 원 (Howon Kim) 종신회원

1993년 2월 경북대학교 전자공학과 학사

1995년 2월 포항공과대학교 전자전기공학과 공학석사

1999년 2월 포항공과대학교 전자전기공학과 공학박사

1998년 12월~2008년 2월 한국전자통신연구원(ETRI) 정보보호연구단 선임연구원/ 팀장

2008년 3월~현재 부산대학교 정보컴퓨터공학부 조교수

<관심분야> 스마트그리드 보안, RFID/USN 정보보호 기술, PKC 암호, VLSI 설계, embedded system 보안