情報保護學會論文誌
第 21 卷 第 2 號, 2011. 4

마스킹 테이블을 사용하지 않는 AES, ARIA, SEED S-box의 전력 분석 대응 기법*

한 동 국,[1†] 김 희 석,[2] 송 호 근,[3] 이 호 상,[3] 홍 석 희[2‡]
[1]국민대학교, [2]고려대학교, [3]한국조폐공사

A Power Analysis Attack Countermeasure Not Using Masked Table for S-box of AES, ARIA and SEED*

Dong-Guk Han,[1†] HeeSeok Kim,[2] Ho-geun Song,[3] Ho-sang Lee,[3] Seokhie Hong[2‡]
[1]Kookmin University, [2]Korea University, [3]Korea Minting and Security Printing Corporation

요 약

전력 분석 공격이 소개되면서 다양한 대응법들이 제안되었고 그러한 대응법들 중 블록 암호의 경우, 암/복호화 연산 도중 중간 값이 전력 측정에 의해 드러나지 않도록 하는 마스킹 기법이 잘 알려져 있다. 블록 암호의 마스킹 기법은 비선형 연산에 대한 비용이 가장 크며, 따라서 AES, ARIA, SEED의 경우 S-box에 대한 대응법을 효율적으로 설계해야만 한다. 하지만 기존의 AES, ARIA, SEED의 S-box에 대한 대응 방법은 마스킹 S-box 테이블을 사용하는 방법으로 하나의 S-box당 256 bytes의 RAM을 필수적으로 사용한다. 하지만 가용 RAM의 크기가 크지 않은 경량 보안 디바이스에 이러한 기존의 대응법은 사용이 부적합하다. 본 논문에서는 이러한 단점을 보완하기 위해 마스킹 S-box 테이블을 사용하지 않는 새로운 대응법을 제안한다. 본 논문에서 제안하는 새로운 대응 기법은 비용이 적은 ROM을 활용, RAM의 사용량을 줄일 뿐 아니라 마스킹 S-box 테이블 생성 시간을 소요하지 않으므로 축소 라운드 마스킹 기법 적용 시 고속화도 가능하다.

ABSTRACT

In the recent years, power analysis attacks were widely investigated, and so various countermeasures have been proposed. In the case of block ciphers, masking methods that blind the intermediate values in the en/decryption computations are well-known among these countermeasures. But the cost of non-linear part is extremely high in the masking method of block cipher, and so the countermeasure for S-box must be efficiently constructed in the case of AES, ARIA and SEED. Existing countermeasures for S-box use the masked S-box table to require 256 bytes RAM corresponding to one S-box. But, the usage of the these countermeasures is not adequate in the lightweight security devices having the small size of RAM. In this paper, we propose the new countermeasure not using the masked S-box table to make up for this weak point. Also, the new countermeasure reduces time-complexity as well as the usage of RAM because this does not consume the time for generating masked S-box table.

Keywords: Side-Channel Attack, Power Analysis, Masking method, AES S-box

접수일(2011년 1월 18일), 게재확정일(2011년 4월 12일)
* 이 논문의 일부는 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(20100024870)
* 이 논문의 일부는 한국조폐공사와의 공동연구 과제 사업의 지원비를 받아 수행되었음.
† 주저자, christa@kookmin.ac.kr
‡ 교신저자, shhong@korea.ac.kr

I. 서 론

수학적으로 안전한 것으로 알려진 알고리즘조차도 구현 단계에서 고려되지 못한 부가적인 정보의 누출이 있다는 것이 알려졌고, 이로부터 비밀 키의 값을 알아낼 수 있는 부채널 공격(Side Channel Attack)이 소개되었다[14]. 이러한 부채널 공격이 소개되면서 많은 암호시스템 설계자들은 효율적인 대응법들을 연구하기 시작했고, 부채널 공격 중 하나인 차분 전력 분석(Differential Power Analysis, DPA)[12, 13,15]에 대한 대응법으로는 마스킹 대응법(masking method)이 활발히 연구되어지고 있다[5,7, 8,9].

마스킹 기법은 평문 x에 대하여 암호문 y를 얻기 위해 마스킹 난수 m을 이용 $x\oplus m(\oplus$: xor)의 암호문 $y'(=y\oplus m')$을 구한 후, 최종적으로 y를 얻기 위해 $y'\oplus m'$의 연산을 수행한다.(경우에 따라 마스킹 기법은 다르게 구성한다.) 따라서 암호화 중 중간 값을 알 수 없기 때문에 일반적인 전력 분석 공격은 성공할 수 없다. 이러한 마스킹 기법을 사용한 경우, $x\oplus m$의 암호문 $y'(=y\oplus m')$에서 m'을 알아야 실제 원하는 암호문 y를 얻을 수 있다. 하지만 블록암호 알고리즘은 비선형 연산을 수행하므로 수정되지 않은 블록 암호 시스템에서 m'값은 x에 따라 다르며 이 값을 중간 값의 누출 없이 아는 것도 상당한 연산을 필요로 한다. 따라서 마스킹 대응법은 이 비선형 연산에 대한 고려가 불가피하다. 표준 블록 암호 AES, ARIA, SEED의 비선형 연산은 S-box 연산으로 그 값이 $S(x\oplus m)=S(x)\oplus m_x$의 형태이며 m_x의 값이 x의 값마다 다르다. 따라서 일반적인 소프트웨어 마스킹 방법에서는 이 m_x의 값이 모든 x에 대해 같은 값이 되게끔 암호 알고리즘의 최초 수행시마다 새로운 마스킹 S-box(MS)를 만든다. 즉, 모든 x에 대해 $MS(x\oplus m)=S(x)\oplus m'$를 만족하는 MS를 생성한다. 하지만 이러한 MS 테이블의 생성은 AES, ARIA, SEED의 경우 하나의 256 bytes의 S-box마다 256 bytes의 RAM을 요구한다. 이는 가용 RAM이 크지 않은 경량 보안 디바이스에 적합하지 않을 수 있으며 마스킹 S-box 생성에 필요한 연산 시간으로 인해 많은 속도 저하를 가져온다.

본 논문에서는 AES, ARIA, SEED의 부채널 대응 방법 설계를 위해 가장 큰 연산량을 차지하는 S-box에 대해 효율적인 대응법을 제안하고자 한다. 이를 위해 가용 RAM이 크지 않은 경량 보안 디바이스에 쉽게 적용할 수 있도록 MS 테이블을 생성하지 않는 새로운 대응 기법을 설계한다. 제안 기법은 연산 중간 값을 저장하는데 필요한 메모리를 제외하고 RAM을 사용하지 않으며 뿐만 아니라 축소 라운드 마스킹 기법의 적용 시 마스킹 S-box 생성 시간 단축으로 인해 연산 시간 단축의 효과도 가진다. 실제로 제안 기법을 ARIA에 적용하여 성능을 비교하였을 때 속도 향상의 결과가 있음을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2절은 AES, ARIA, SEED의 S-box에 대해 설명하고, 3절에서는 기존의 마스킹 S-box 생성 기법을, 4절에서는 제안하는 마스킹 S-box 연산 방법에 대해 소개한다. 본 논문에서 제안하는 마스킹 방법의 효율성은 5절에서 소개하며 본 논문은 6절에서 결론짓는다.

II. AES, ARIA, SEED의 S-Box

블록 암호 알고리즘 AES, ARIA, SEED의 S-Box 연산은 x^{-1}의 아핀 변환(affine transform) 형태로 다음의 연산을 수행한다 (A:8×8 행렬, $a\in GF(2^8)$).

$$S\colon GF(2^8)\rightarrow GF(2^8)$$

$$S(x)=Ax^{-1}\oplus a$$

위의 연산에서 x^{-1}의 연산은 유한체 GF(28) (AES, ARIA의 기약 다항식 : $x^8+x^4+x^3+x+1$, SEED의 기약 다항식 : $x^8+x^6+x^5+x+1$) 상에서 이루어진다.

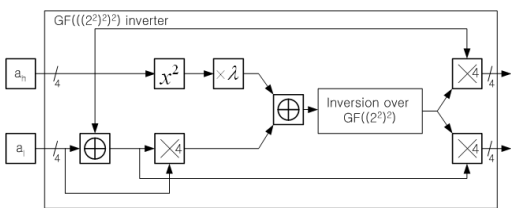
일반적으로 AES의 하드웨어 설계에 있어서 S-Box를 저장하고 호출하는 방식은 공간적인 제약이 많이 따르는 편이며, 이로 인해 S-Box를 연산하는 방법이 주로 사용된다. 하지만 x^{-1}의 연산과 아핀 변환으로 이루어지는 S-Box 연산은 x^{-1}, 즉 $GF(2^8)$에서의 역원계산에서 상당한 비용이 요구되며. 실제 역원 계산에 들어가는 비용은 블록 암호의 라운드 연산에서 상당한 부분을 차지한다. 따라서 역원 계산의 효율성은 전체 암호 알고리즘의 성능에 크게 영향을 미치며, 이러한 이유로 역원 계산의 비용을 감소시키기 위한 다양한 방법들이 연구되어졌으며 복합체의 개념도 소개되어졌다[3,6]. 즉, $GF(2^8)$ 위에서의 일반적인 역원계산이 아닌 역원 계산의 비용이 작은 복합체로의 변환을 수행한 후 역원 계산을 수행, 결과 값을 다시 $GF(2^8)$ 위의 원소로 역변환 하는 방법이 소개되어졌다.

복합체에서의 역원연산은 부분체(subfield)에서의 연산을 통해 이루어진다. S-box 연산에서 사용되어지는 $GF(2^8)$ 위에서의 연산은 복합체인 $GF(((2^2)^2)^2)$ 위에서의 연산으로 변형되어질 수 있으며, 각 부분체에서 사용되어지는 기약다항식의 형태는 다음과 같다.

$$\begin{aligned} &GF(2^2)\ over\ GF(2) && : P_0(x)=x^2+x+1\\ &GF((2^2)^2)\ over\ GF(2^2) && : P_1(x)=x^2+x+\phi\\ &GF(((2^2)^2)^2)\ over\ GF((2^2)^2) && : P_2(x)=x^2+x+\lambda \end{aligned}$$

$P_0(x)$의 근을 α, $P_1(x)$의 근을 β, $P_2(x)$의 근을 γ라 한다면 ($\alpha\in GF(2^2)$, $\beta\in GF((2^2)^2)$, $\gamma\in GF(((2^2)^2)^2)$), $GF(2^2)$의 모든 원소는 $a_1\alpha+a_0$, $GF((2^2)^2)$의 모든 원소는$(a_3\alpha+a_2)\beta+(a_1\alpha+a_0)$, $GF(((2^2)^2)^2)$의 모든 원소는 $\{(a_7\alpha+a_6)\beta+(a_5\alpha+a_4)\}\gamma+\{(a_3\alpha+a_2)\beta+(a_1\alpha+a_0)\}$의 형태로 표현된다. 연산의 효율성을 위해 $P_1(x)$, $P_2(x)$가 기약인 성질을 만족하는 원소 중에 $GF(2^2)$의 원소 ϕ는 $\alpha((10)_2)$로, $GF((2^2)^2)$의 원소 λ는 $(\alpha+1)\beta((1100)_2)$로 선택하였다.

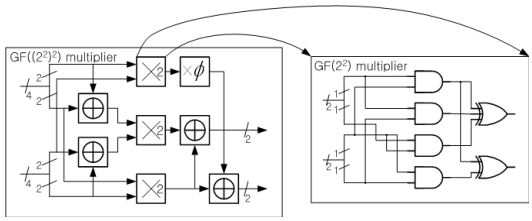
복합체 위에서의 역원 연산은 다음의 연산을 통해 이루어진다. $A\in GF(((2^2)^2)^2)$의 역원 A^{-1} 연산을 위해 $C^{-1}A^{16}(C=A^{17}\in GF((2^2)^2))$을 연산한다. 또한 $C\in GF((2^2)^2)$의 역원 C^{-1}는 $D^{-1}C^4(D=C^5\in GF(2^2))$ 연산을 통해 이루어진다. 즉 $GF(((2^2)^2)^2)$위에서의 역원 연산이 $GF(2^2)$ 위에서의 역원 연산을 통해 이루어진다. 물론, 역원 연산을 위해 추가적인 A^{16}, A^{17}과 같은 연산이 필요하지만 복합체 연산의 특성상 $A=a_h\gamma+a_l$의 16승 연산 결과는 $a_h\gamma+(a_h+a_l)$의 4비트 XOR 연산, $(a_h\gamma+a_l)^{17}$의 연산 결과는 $a_h^2\lambda+(a_h+a_l)a_l$로 $GF((2^2)^2)$ 위에서의 한 번의 제곱 연산과 곱셈 연산만을 요구한다. 다음 그림은 $GF(((2^2)^2)^2)$ 위에서의 역원 연산과 $GF((2^2)^2)$ 위에서의 곱셈, 제곱 연산의 그림이다[3].



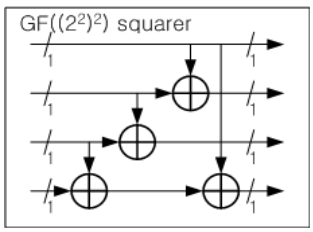
[그림 1] $GF(((2^2)^2)^2)$ 역원 연산[3]

[그림 1]에 따른 $A=a_h\gamma+a_l$의 역원 $A^{-1}=a_h'\gamma+a_l'$를 계산하는 수식은 식 (1)과 같다.

$$\begin{aligned} &d=\lambda a_h^2+a_l(a_h+a_l)\\ &d'=d^{-1}\\ &a_h'=d'a_h\\ &a_l'=d'(a_h+a_l) \end{aligned} \qquad (1)$$



[그림 2] $GF((2^2)^2)$ 곱셈 연산[3]



[그림 3] $GF((2^2)^2)$ 제곱 연산[3]

III. 전력 분석 공격에 안전한 기존 S-Box 대응 기법

AES, ARIA, SEED의 S-box 연산은 그 값이 $S(x\oplus m)=S(x)\oplus m_x$의 형태이며 m_x의 값이 x의 값마다 다르다. 따라서 기존의 대응법들은 이 m_x의 값이 모든 x에 대해 같은 값이 되게끔 암호 알고리즘의 최초 수행시마다 새로운 마스킹 S-box(MS)를 만든다. 즉, 알고리즘의 시작 전 두 난수 m, m'을 생성하고 이 두 난수로부터 모든 x에 대해 $MS(x\oplus m)=S(x)\oplus m'$를 만족하는 MS를 생성한다. 8 비트 입출력을 가지는 S-box를 사용하는 블록암호에서 이러한 MS 테이블을 생성하는 가장 일반적인 알고리즘은 다음과 같다[1,5,7].

알고리즘 1. MS **테이블 생성 알고리즘**
입력 S, m, m'
출력 마스킹 테이블 MS
1. For x from 0 to 255 :
  1.1. $MS(x)=S(x\oplus m)\oplus m'$
  End 1.
2. Return MS

알고리즘 1에서 보이는 것과 같이 MS 테이블 생성

알고리즘은 ARIA, SEED, AES의 경우, 하나의 S-box마다 기본적으로 256 bytes의 RAM을 사용한다. 하지만 이러한 RAM의 사용은 가용 RAM의 크기가 충분하지 않은 경량 보안 디바이스에 적합하지 못할 수 있다.

IV. 마스킹 S-box 테이블을 사용하지 않는 S-box 대응 기법

본 절에서는 RAM의 크기가 제약적인 환경을 고려하여 마스킹 S-box 테이블을 사용하지 않는 S-box 대응 기법을 설계하고자 한다.

4.1 제안 기법

제안 기법은 복합체 위에서의 역원 연산을 이용해 마스킹 S-box 함수를 구성하도록 한다. 제안하는 방법은 연산의 효율성을 위해 복합체 위에서의 연산을 사전 계산해 RAM에 비해 비용이 적은 ROM에 저장하는 방식을 사용한다.

우선 AES, ARIA, SEED의 S-box는 다음의 식을 만족한다.

$$S\colon GF(2^8) \rightarrow GF(2^8)$$

$$S(x) = Ax^{-1} \oplus a$$

위의 식으로부터 마스킹 S-box 함수는 다음의 식을 만족해야만 한다.

$$MS(x \oplus m) = Ax^{-1} \oplus a \oplus m'$$

제안하는 기법은 2절에서 설명한 복합체 위에서의 역원 연산의 개념을 이용하며 각 연산 단계에 해당하는 중간 값 a_i를 랜덤한 두 값 $(a_{i1},\ a_{i2})$으로 나누어 연산하도록 한다. 이 때, $a_i = a_{i1} \oplus a_{i2}$의 식이 항상 만족할 수 있도록 두 값을 연산한다. 이러한 방식으로 대응법을 설계했을 때, 공격자가 전력 분석에서 예측한 모든 중간 값은 랜덤한 난수 형태로 분해되어 연산되므로 제안 기법은 전력 분석 공격에 안전할 수 있다. 또한 랜덤하게 쪼개진 두 값의 XOR된 값은 실제 연산되어져야 할 중간 값의 형태가 되므로 이는 언제든 연산되어지는 값들을 통해 원래의 값으로 복원할 수 있음을 의미한다.

마스킹 S-box에 해당하는 연산을 안전하고 효율적으로 구성하기 위해 다음 5 개의 테이블을 사전 계산해 ROM에 저장한다.

- **스칼라 제곱 테이블 (256 bytes)**
  입력 : 8 비트 $X(=A_h \| A_l) \in GF((2^4)^2)$ $(A_h,\ A_l \in GF(2^4))$
  출력 : $T1[X] = \lambda A_h^2 \oplus (A_h \oplus A_l)A_l$
- **곱셈 테이블(256 bytes)**
  입력 : 4 비트 $X,\ Y \in GF(2^4)$
  출력 : $T2[X][Y] = XY$
- $GF((2^2)^2)$ **마스킹 역원 테이블 (256 bytes)**
  입력 : 4 비트 $\widetilde{X}(= X \oplus m),\ m \in GF(2^4)$
  출력 : $T3[m][\widetilde{X}] = X^{-1} \oplus m \in GF(2^4)$
- **isomorphism 테이블 (256 bytes)**
  입력 : 8 비트 $X \in GF(2^8)$
  출력 : $T4[X] = iso(X) \in GF(((2^2)^2)^2)$
- **inverse isomorphism + affine transform 테이블 (256 bytes)**
  입력 : 8 비트 $X \in GF(((2^2)^2)^2)$
  출력 : $T5[X] = A \cdot inviso(X) \oplus a \in GF(2^8)$

위의 다섯 개의 테이블로부터 마스킹되지 않은 S-box S는 입력 값 x로부터 출력 값 $S(x)$를 다음의 다섯 단계에 의해 연산한다.

- **마스킹 되지 않은 S-box 연산**
  1. $(A_h \| A_l) = iso(x) = T4[x]$
  2. $d = \lambda A_h^2 \oplus A_l(A_h \oplus A_l) = T1[A_h \| A_l]$
  3. $d' = d^{-1} = T3[0][d]$
  4. $(A_h' \| A_l') = (d'A_h \| d'(A_h \oplus A_l)) = (T2[d'][A_h] \| T2[d'][A_h \oplus A_l])$
  5. $Ax^{-1} \oplus a = A \cdot inviso(A_h' \| A_l') \oplus a$

제안하고자 하는 MS 함수는 두 난수 m, m'과 마스킹된 S-box 입력 값 $\tilde{x}(= x \oplus m)$으로부터 마스킹된 S-box 출력 값 $Ax^{-1} \oplus a \oplus m'$을 연산한다. 이 연산은 다음의 여섯 단계에 의해 수행되어진다. 각 단계에서 수행되어지는 연산은 위의 다섯 단계의 연산에 해당하는 중간 값을 랜덤한 두 값으로 나누어 연산한다.

- **제안하는 마스킹 S-box 연산**
  1. $(\widetilde{A}(\widetilde{A_h} \| \widetilde{A_l}),\ \widetilde{m}(m_h \| m_l)) = (T4[\tilde{x}],\ T4[m])$

$(\widetilde{A_h}=A_h\oplus m_h,\ \widetilde{A_l}=A_l\oplus m_l)$

2. $(inv_{in},\ m_{out})=(T1[\widetilde{A}],\ T1[\widetilde{m}]\oplus T2[m_h][\widetilde{A_l}]\oplus T2[m_l][\widetilde{A_h}])$
3. $(inv_{out},\ m_{out})=(T3[m_{out}][inv_{in}],\ m_{out})$
4-1. $\widetilde{A_h}'=(T2[inv_{out}][\widetilde{A_h}]\oplus T2[m_{out}][\widetilde{A_h}]\oplus T2[inv_{out}][m_h])$
$m_h'=T2[m_{out}][m_h]$
4-2. $\widetilde{A_l}'=(T2[inv_{out}][\widetilde{A_l}]\oplus T2[m_{out}][\widetilde{A_l}]\oplus T2[inv_{out}][m_l])\oplus\widetilde{A_h}'\ m_l'=T2[m_{out}][m_h\oplus m_l]$
5. Return $T5[\widetilde{A_h}'\|\widetilde{A_l}']\oplus m'\oplus T5[m_h'\|m_l']\oplus a$

다음은 제안하는 마스킹 S-box 연산이 정상적으로 동작함을 증명하기 위해 각 단계에서 연산된 한 쌍의 중간 값을 XOR한 값이 마스킹되지 않은 S-box 연산의 중간 값과 일치함을 보인다.

제안하는 마스킹 S-box의 soundness 증명

1. $\widetilde{A}\oplus\widetilde{m}=T4[\tilde{x}]\oplus T4[m]=T4[\tilde{x}\oplus m]=iso(x)$
2. $inv_{in}\oplus m_{out}$
$=T1[\widetilde{A}]\oplus T1[\widetilde{m}]\oplus T2[m_h][\widetilde{A_l}]\oplus T2[m_l][\widetilde{A_h}]$
$=\{\lambda\widetilde{A_h^2}\oplus(\widetilde{A_h}\oplus\widetilde{A_l})\widetilde{A_l}\}\oplus\{\lambda m_h^2\oplus(m_h\oplus m_l)m_l\}\oplus m_h\widetilde{A_l}\oplus m_l\widetilde{A_h}$
$=\{\lambda A_h^2\oplus\lambda m_h^2\oplus(A_h\oplus A_l\oplus m_h\oplus m_l)(A_l\oplus m_l)\}\oplus\{\lambda m_h^2\oplus(m_h\oplus m_l)m_l\}\oplus m_h(A_l\oplus m_l)\oplus m_l(A_h\oplus m_h)$
$=\lambda A_h^2\oplus(A_h\oplus A_l)A_l$
$=d$
3. $inv_{out}\oplus m_{out}=(d^{-1}\oplus m_{out})\oplus m_{out}=d^{-1}=d'$
$\Rightarrow B=d'\oplus m_{out}$
4-1. $\widetilde{A_h}'\oplus m_h'=(T2[inv_{out}][\widetilde{A_h}]\oplus T2[m_{out}][\widetilde{A_h}]\oplus T2[inv_{out}][m_h])\oplus T2[m_{out}][m_h]$
$=\{(d'\oplus m_{out})(A_h\oplus m_h)\oplus m_{out}(A_h\oplus m_h)\oplus(d'\oplus m_{out})m_h\}\oplus m_{out}m_h$
$=d'A_h$
4-2. $\widetilde{A_l}'\oplus m_l'=(T2[inv_{out}][\widetilde{A_l}]\oplus T2[m_{out}][\widetilde{A_l}]\oplus T2[inv_{out}][m_l])\oplus\widetilde{A_h}'\oplus T2[m_{out}][m_h\oplus m_l]$
$=\{(d'\oplus m_{out})(A_l\oplus m_l)\oplus m_{out}(A_l\oplus m_l)\oplus(d'\oplus m_{out})m_l\oplus(d'A_h\oplus m_{out}m_h)\}\oplus m_{out}(m_h\oplus m_l)$
$=d'(A_h\oplus A_l)$
5. $T5[\widetilde{A_h}'\|\widetilde{A_l}']\oplus m'\oplus T5[m_h'\|m_l']\oplus a$
$=A\bullet inviso(\widetilde{A_h}'\|\widetilde{A_l}')\oplus A\bullet inviso(m_h'\|m_l')\oplus a\oplus m'$
$=A\bullet(inviso((\widetilde{A_h}'\|\widetilde{A_l}')\oplus(m_h'\|m_l')))\oplus a\oplus m'$
$=A\bullet x^{-1}\oplus a\oplus m'=S(x)\oplus m'$

위의 수식은 각 단계에서 연산되어지는 중간 값 a_i가 $a_i=a_{i1}\oplus a_{i2}$의 식이 항상 만족하도록 랜덤한 두 값 $(a_{i1},\ a_{i2})$으로 나누어 연산됨을 의미한다. 또한 최종 결과는 MS의 출력 값에 해당하는 $S(x)\oplus m'$임을 확인할 수 있다. 위의 수식으로부터 사전 계산된 테이블을 이용해 세 입력 값 $(\tilde{x}(=x\oplus m), m, m')$으로부터 MS의 출력 값을 안전하게 연산하는 알고리즘은 다음과 같다.

〔알고리즘 2〕 제안하는 Masking S-box $MS(\tilde{x}, m, m')$ 연산 기법

입력 $\tilde{x}(=x\oplus m),\ m,\ m'$
출력 $S(x)\oplus m'$

1. $B=T4[\tilde{x}],\ temp=T4[m]$
2. $Ah=B\gg 4,\ Al=B\&0xf,\ mh=temp\gg 4,\ ml=temp\&0xf$
3. $m_{out}=T1[temp]\oplus T2[mh][Al]\oplus T2[ml][Ah]$
4. $B=T3[m_{out}][T1[B]]$
5. $temp=(T2[B][Ah]\oplus T2[m_{out}][Ah]\oplus T2[B][mh])$
6. $temp=(temp\ll 4)\oplus(T2[B][Al]\oplus T2[m_{out}][Al]\oplus T2[B][ml])\oplus temp$
7. $m_{out}=$
$(T2[m_{out}][mh]\ll 4)\oplus(T2[m_{out}][mh\oplus ml])$
8. Return $T5[temp]\oplus m'\oplus T5[m_{out}]\oplus a$

4.2 제안 기법의 안전성 증명

제안 기법의 안전성은 본 소절의 다섯 개의 Lemma로부터 증명 가능하다.

Lemma 1. a가 $GF(2^n)$의 임의의 원소이고 m이 a에 독립인 $GF(2^n)$에서 균일하게 분포된 난수라고 한다면, $a\oplus m$은 a에 독립이다.

Lemma 2. a가 $GF(2^n)$의 임의의 원소이고 m_1, m_2가 a에 독립인 $GF(2^n)$에서 균일하게 분포된 난수들이라고 한다면, $a\oplus m_1\oplus m_2$는 a에 독립이다.

위의 두 Lemma에 대한 증명은 직관적이므로 생략하도록 한다. 위의 두 Lemma는 덧셈 마스킹이 일차 전력 분석에 안전한 근본이 되는 이유이다. 이 Lemma들에 의해 독립인 난수로 마스킹된 값을 출력하는 $T3$, $T4$, $T5$ 테이블은 일차 전력 분석에 안전하다.

제안 기법의 안전성을 증명하기 위해 다음의 추가적인 Lemma들이 필요하다. 다음의 Lemma들은 하드웨어 환경에서 AES S-box의 마스킹 기법 설계 시 이미 증명되어졌다[8].

Lemma 3. a, b가 $GF(2^n)$의 임의의 원소이고 m_a, m_b는 $GF(2^n)$에서 균일하게 분포된 서로 독립인 난수라고 한다면, $(a\oplus m_a)(b\oplus m_b)$는 a, b에 모두

독립이다.

Lemma 4. a가 $GF(2^n)$의 임의의 원소이고 m_a, m_b는 $GF(2^n)$에서 균일하게 분포된 서로 독립인 난수라고 한다면, $(a \oplus m_a)m_b$는 a에 독립이다.

Lemma 5. a가 $GF(2^n)$의 임의의 원소이고, λ가 고정된 상수일 때, m_a가 $GF(2^n)$에서 균일하게 분포된 a, λ에 독립인 난수라고 한다면, $(a \oplus m_a)^2\lambda$는 a에 독립이다.

Lemma 3과 Lemma 4는 곱셈 연산, 즉 $T2$ 테이블 사용 시, 두 입력 값이 어떤 조건을 만족해야하는지를 나타낸다. 이 두 Lemma에 따르면 $T2$ 테이블의 두 입력 값은 서로 독립인 난수로 마스킹되어 있는 값이어야만 한다. 또한 Lemma 5는 $T1$ 테이블의 출력 값에 해당하는 $GF(2^n)$에서 제곱 연산 후 스칼라 곱셈 연산의 출력 값이 일차 전력 분석에 안전함을 의미한다.

제안하는 마스킹 S-box는 위의 다섯 개의 Lemma들을 이용해 설계되었다. 따라서, 제안 기법은 일차 전력 분석으로부터 안전성을 제공한다.

V. 사용 메모리 및 효율성 비교

제안하는 마스킹 기법과 기존 기법의 성능을 비교하기 위해 일반적인 마스킹 테이블 생성기법인 알고리즘 1과 제안 기법을 ARIA에 적용하여 성능을 측정하였다. 실험 환경은 다음과 같다.

- ATmega128
- AVR Studio 4
- ARIA 128 bit
- I) 축소 라운드 마스킹 (1, 2, 11. 12 라운드)
  II) 전체 라운드 마스킹

제안 기법은 암호 연산에서 비교적 많은 시간을 소요하지만 마스킹 S-box 테이블을 생성하지 않으므로 이에 해당하는 Masked S-box 테이블 생성 시간을 단축할 수 있었다. 따라서, 일반적인 ARIA 암호 대응기법[1]은 Masked S-box 테이블 생성 → Encrypt의 순서로 동작한다. 하지만 마스킹 S-box를 사용하지 않는 제안 기법은 Masked S-box 테이블 생성 단계 없이 동작한다. [표 1]은 축소 라운드 마스킹 적용 시, 암호화 시간을 비교한 것이다.

다음 표는 전체 라운드 마스킹 적용시 속도를 비교한 것이다.

[표 1] ARIA 축소 라운드 마스킹 기법의 암호화 속도 비교 (단위:cc(clock cycle))

	Typical ARIA	Algorithm 1[1,5,7]	Proposed method
Masked S-box 테이블 생성	-	41865cc	-
Encrypt	73177cc	73745cc	84169cc
Total	73177cc	115610cc	84169cc

[표 2] ARIA 전체 라운드 마스킹 기법의 암호화 속도 비교 (단위:cc(clock cycle))

	Typical ARIA	Algorithm 1[1,5,7]	Proposed method
Masked S-box 테이블 생성	-	41865cc	-
Encrypt	73177cc	74995cc	106093cc
Total	73177cc	116860cc	106093cc

[표 3] ARIA 암호화 메모리 사용량 비교

	Algorithm 1[1,5,7]	Proposed method
RAM 사이즈 (마스킹 S-box 테이블 사이즈)	1024 Bytes	-
ROM 사이즈	1024 Bytes	2304 Bytes

[표 3]는 제안하는 마스킹 S-box 연산 기법을 적용한 ARIA 구현의 메모리 사용량을 알고리즘 1을 적용한 ARIA의 마스킹 기법과 비교하고 있다. 제안 기법은 기존 기법과 비교하여, 1024 bytes의 RAM을 절약할 수 있다. 이는 가용 RAM의 크기가 크지 않은 경량 보안 디바이스에 새로운 알고리즘이 적합함을 의미한다.

VI. 결 론

본 논문에서는 경량 보안 디바이스에 기존 전력 분석 대응 기법 적용 시 문제가 되었던 RAM의 사용량을 줄이기 위해 마스킹 S-box 테이블을 사용하지 않는 새로운 대응 기법을 제안하였다. 제안하는 방법은 연산을 위한 중간 값 저장에 필요한 메모리 이외의 별도의 RAM을 사용하지 않으며 축소 라운드 마스킹 기법을 적용 시 마스킹 S-box 테이블 생성 시간 단축으로 인해 연산 속도도 향상 시킬 수 있었다.

참고문헌

〔1〕 유형소, 하재철, 김창균, 박일환, 문상재, "랜덤 마스킹 기법을 이용한 DPA 공격에 안전한 ARIA 구현," 한국정보보호학회논문지 16(2), pp. 129-139, April 2006

〔2〕 Advanced Encryption Standard (AES), FIPS PUB 197, November 26, 2001, available at http://csrc.nist.gov/encryption/aes.

〔3〕 A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," ASIACRYPT'01, LNCS 2248, pp. 239-254, 2001.

〔4〕 B. Zakeri, M. Salmasizadeh, A. Moradi, M. Tabandeh, and M. Shalmani, "Compact and Secure Design of Masked AES S-Box," ICICS'07, LNCS 4861, pp. 216-229, 2007.

〔5〕 C. Herbst, E. Oswald, and S. Mangard, "An AES Smart Card Implementation Resistant to Power Analysis Attacks," ACNS'06, LNCS 3989, pp. 239-252, 2006.

〔6〕 D. Canright, "A Very Compact Rijndael S-box.Technical Report," NPS-MA-04-001, Naval Postgraduate School (September 2004), http://web.nps.navy.mil/~dcanrig/pub/NPS-M A-05-001.pdf

〔7〕 E. Oswald and K. Schramm. "An Efficient Masking Scheme for AES Software Implementations," WISA'05, LNCS 3786, pp. 292-305, 2006.

〔8〕 E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen., "A Side-Channel Analysis Resistant Description of the AES S-box," FSE'05, LNCS 3557, pp. 413-423, 2005.

〔9〕 J. Bl¨omer, J. Guajardo, and V. Krummel. "Provably Secure Masking of AES," SAC'04, LNCS 3357, pp. 69-83, 2005.

〔10〕 Jovan D. Golic, Christophe Tymen. "Multiplicative Masking and Power Analysis of AES," CHES'02, LNCS 2523, pp. 198-212, 2003.

〔11〕 Mehdi-Laurent Akkar and Christophe Giraud. "An Implementation of DES and AES, Secure against Some Attacks," CHES'01, LNCS 2162, pp. 309-318, 2001.

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

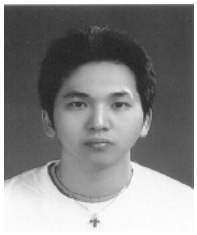
〔13〕 P. Kocher, J. Jaffe, and B. Jun, "Introduction to differential power analysis and related attacks," http://www.cryptography.com/dpa/te chnical, June 1998.

〔14〕 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, 1996.

〔15〕 T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks on modular exponentiation in Smart cards," CHES'99, LNCS 1717, pp. 144-157, 1999.

〈著者紹介〉

한 동 국 (Dong-Guk Han) 정회원
1999년 고려대학교 수학과 졸업(학사)
2002년 고려대학교 수학과 석사 (이학석사)
2005년 고려대학교 정보보호대학원 박사 (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 수학과 조교수
〈관심분야〉 암호시스템 안전성 분석 및 고속 구현, 부채널 분석, RFID/USN 정보보호 기술



김 희 석 (HeeSeok Kim) 학생회원
2006년 2월: 연세대학교 수학과 졸업(학사)
2008년 2월: 고려대학교 정보경영공학전문대학원 공학석사
2008년 3월~현재: 고려대학교 정보경영공학전문대학원 박사과정
〈관심분야〉 부채널 공격, 암호시스템 안전성 분석 및 고속구현, 암호칩 설계 기술

송 호 근 (Ho-geun Song) 정회원
1998년 경북대학교 전자공학과 졸업(학사)
2000년 경북대학교 전자공학과 석사(석사)
2000년 8월~2007년 12월: 삼성SDS SOC개발실 선임연구원
2007년 12월~현재: 한국조폐공사 정보기술연구실 선임연구원
〈관심분야〉 스마트카드 정보보호 기술, 부채널 분석

이 호 상 (Ho-sang Lee) 정회원
2000년 한양대학교 기계공학과 졸업(학사)
2002년 한국과학기술원 기계공학과 졸업(석사)
2002년 2월~2007년 12월: 삼성SDS SOC개발실 선임연구원
2007년 12월~현재: 한국조폐공사 정보기술연구실 선임연구원
〈관심분야〉 스마트카드 정보보호 기술, 부채널 분석

홍 석 희 (SeokHie Hong) 종신회원
1995년: 고려대학교 수학과 학사
1997년: 고려대학교 수학과 석사
2001년: 고려대학교 수학과 박사
1999년 8월~2004년 2월: (주)시큐리티 테크놀로지스 선임연구원
2003년 3월~2004년 2월: 고려대학교 시간강사
2004년 4월~2005년 2월: K.U. Leuven 박사후연구원
2005년 3월~2008년 8월: 고려대학교 정보경영공학전문대학원 조교수
2008년 9월~현재: 고려대학교 정보경영공학전문대학원 부교수
〈관심분야〉 대칭키 암호 알고리즘, 공개키 암호 알고리즘, 포렌식