Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.3, Jun. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.3.363

차분 계산 분석 대응을 위한 WBC-AES Dummy LUT 생성 방안 연구*

최 민 영,[1†] 석 병 진,[2] 서 승 희,[1] 이 창 훈[3‡]
[1,3]서울과학기술대학교 (대학원생, 교수)
[2]서울과학기술대학교 전기정보기술연구소 (연구원)

A Study on Creating WBC-AES Dummy LUT as a Countermeasure against DCA*

Minyeong Choi,[1†] Byoungjin Seok,[2] Seunghee Seo,[1] Changhoon Lee[3‡]
[1,3]Seoul National University of Science and Technology (Graduate student, Professor)
[2]Institute of Electric and Information Technology, Seoul National University of Science and Technology (Researcher)

요 약

화이트박스 환경이란 알고리즘의 내부 정보가 공개된 환경을 말한다. 2002년에 AES 화이트박스 암호가 최초로 발표되었으며, 2016년에는 화이트박스 암호에 대한 부채널 분석인 DCA(Differential Computation Analysis)가 제안되었다. DCA 분석은 화이트박스 암호의 메모리 정보를 부채널 정보로 활용하여 키를 찾아내는 강력한 부채널 공격기법이다. DCA에 대한 대응방안 연구가 국내외에 발표되었지만, DCA 분석에 Dummy 연산을 적용하는 하이딩 기법을 실험한 결과와 실제로 평가 또는 분석된 결과가 존재하지 않았다. 따라서, 본 논문에서는 2002년에 S. Chow가 발표한 WBC-AES 알고리즘에 LUT 형태의 Dummy 연산을 삽입하고, Dummy 크기에 따라서 DCA 분석의 대응의 변화 정도를 정량적으로 평가하였다. 2016년에 제안된 DCA 분석이 총 16바이트의 키를 복구하는 것에 비하여, 본 논문에서 제안하는 대응 기법은 Dummy의 크기가 작아질수록 최대 11바이트의 키를 복구하지 못하는 결과를 얻었으며, 이는 기존의 공격 성능보다 최대 약 68.8% 정도 낮아진 약 31.2%이다. 본 논문에서 제안한 대응방안은 작은 크기의 Dummy를 삽입함에 따라서 공격 성능이 크게 낮아지는 결과를 확인할 수 있었으며, 이러한 연구결과는 다양한 분야에서 활용될 수 있다.

ABSTRACT

A white-box environment refers to a situation where the internal information of an algorithm is disclosed. The AES white-box encryption was first announced in 2002, and in 2016, a side-channel analysis for white-box encryption called Differential Computation Analysis (DCA) was proposed. DCA analysis is a powerful side-channel attack technique that uses the memory information of white-box encryption as side-channel information to find the key. Although various countermeasure studies against DCA have been published domestically and internationally, there were no evaluated or analyzed results from experiments applying the hiding technique using dummy operations to DCA analysis. Therefore, in this

Received(02. 16. 2023), Modified(04. 03. 2023) Accepted(04. 03. 2023)
* 본 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2022-0-00627, 저사양 디바이스 지원을 위한 경량 사물 블록체인 네트워크 기술개발)
* 본 논문은 2022년도 한국정보보호학회 동계학술대회에 발표한 우수논문을 개선 및 확장한 것임.
† 주저자, mychoi0426@gmail.com
‡ 교신저자, chlee@seoultech.ac.kr(Corresponding author)

paper, we insert LU T-shaped dummy operations into the WBC-AES algorithm proposed by S. Chow in 2002 and quantitatively evaluate the degree of change in DCA analysis response depending on the size of the dummy. Compared to the DCA analysis proposed in 2016, which recovers a total of 16 bytes of the key, the countermeasure proposed in this paper was unable to recover up to 11 bytes of the key as the size of the dummy decreased, resulting in a maximum decrease in attack performance of about 68.8%, which is about 31.2% lower than the existing attack performance. The countermeasure proposed in this paper confirms that the attack performance significantly decreases as smaller dummy sizes are inserted and can be applied in various fields.

Keywords: White-box Cryptography, Side-Channel Analysis, Differential Computation Analysis, Hiding Scheme, Dummy

I. 서 론

최근 안전한 암호 알고리즘에 대한 수요의 증가로 화이트박스 암호 알고리즘에 관한 관심이 높아져 가고 있다. 블랙박스 암호 환경은 공격자가 알고리즘 내부의 동작 과정을 알 수 없고, 입력 평문과 출력 암호문만을 확인할 수 있는 구조이다. 블랙박스 암호 환경에서의 암호화키는 신뢰할 수 있는 장치 내부에서 관리 및 사용되어왔다. 하지만 2008년에는 'Cold Boot Attack'[3]과 같이 메모리 내부의 직접적인 분석을 통하여 암호화 키를 찾아내는 연구가 발표 됐으며, 또한, 개방형 플랫폼 사용의 증가는 누구에게나 역공학을 통한 애플리케이션의 내부 암호 알고리즘에 대한 분석이 가능하게 하였다.

화이트박스 환경이란 입력과 출력에 대한 정보만을 얻어내는 블랙박스 환경과는 다르게 내부 시스템에 대한 정보 대부분이 공개된 상태를 말한다. 화이트박스 환경에서의 공격자는 막강한 공격 자원을 갖게 된다. 이러한 이유로, 화이트박스 환경에서의 기존의 블랙박스 암호 알고리즘은 보안에 취약하게 되었다. 따라서, 화이트박스 환경에서 적합한 암호 알고리즘에 대한 공모사업이 활발히 진행되었으며, 화이트박스 환경에서 안전하게 암호 알고리즘을 운용하기 위해서는 알고리즘 내부의 정보가 노출되더라도 정보를 인식하지 못하게 하거나, 노출된 정보를 활용하지 못하게 해야 한다.

DES[1]와 AES[2]에 대한 최초의 화이트박스 암호 알고리즘은 2002년에 S. Chow 외 2인에 의해서 발표되었다[4][5]. 이 화이트박스 암호 알고리즘은 암호화 키와 내부 함수를 LUT(Look-Up Table) 형태로 구현하여, 공격자가 암호화 키를 유추하지 못하게 하며, 내외부에 인코딩 기법을 적용하여 입력값, 출력값 그리고 중간값을 공격자로부터 보호하게 된다. LUT 형태로 구현된 화이트박스 암호 알고리즘은 연산 과정에서 메모리에 읽고 쓰는 주소 참조가 많아지게 된다. 이 과정에서 발생하는 메모리 참조에 대한 정보를 부채널 정보로 활용하여 공격을 수행하는 DCA 분석이 J.W. Bos 외 3인에 의하여 2016년에 제안되었다[6].

DCA 분석은 하드웨어 부채널 공격에 적용하던 차분 전력분석(Differential Power Analysis; DPA)[7]와 상관전력분석(Correlation Power Analysis; CPA)[8]의 분석 방법을 소프트웨어의 환경에 적용한 공격 방법이다. 분석 방법은 화이트박스 알고리즘 내부에 존재하는 고정된 키값에 대한 반복적인 메모리 참조특성을 부채널 정보로써 활용하여 암호화키를 찾아내는 강력한 공격기법이다.

J.W. Bos 외 3인은 기존의 차분 전력분석, 상관전력분석에서 전력 파형 분석을 위해서 오실로스코프 기기를 사용하는 것 대신에, Valgrind[14] 라는 메모리 디버거를 사용하여 메모리 정보에 대한 파형을 분석하였다. DCA 분석은 전력 파형을 분석하는 기존의 방식과는 다르게 노이즈가 존재하지 않는 메모리 정보를 분석하여 짧은 시간 안에 암호화키를 찾게 된다.

본 연구에서 사용된 암호 알고리즘은 S. Chow가 제안한 화이트박스 암호 AES이며, 본 논문에서는 WBC-AES(White-Box Cryptography AES)[4]라고 표현한다. 실험은 WBC-AES 코드 내부에 Dummy 연산을 삽입하여 메모리 정보를 교란함으로써 노이즈와 같은 효과를 기대할 수 있으며, 이러한 Dummy 연산은 크기를 달리하여 DCA 분석에 대한 대응 정도를 정량적으로 제시한다.

J.W. Bos 외 3인은 추출된 메모리 정보를 이용하여 메모리 정보를 시각화할 수 있는 도구와 DCA 분석에 필요한 분석 도구를 Github를 통하여 배포하여 분석을 쉽게 하였다[13]. 본 논문의 실험은 메모리 정보를 시각화할 수 있는 도구와 DCA 분석에 필요한 분석 도구를 실험에 맞게 수정하여 사용하였다. 본 논문에서는 이러한 도구를 메모리 정보 시각

화 도구와 DCA 분석 도구라고 표현한다.

마지막으로, 본 논문의 구성은 다음과 같다. 2장에서는 화이트박스 암호 알고리즘에 대한 자세한 설명과 부채널 분석 방법에 대한 설명을 다룬다. 3장에서는 본 논문에서 제안하는 방법인 DCA 분석에 대응하기 위한 Dummy 연산의 삽입과 실험환경에 대한 설명을 다루며, 4장에서는 실험결과에 관한 내용을 다룬다. 마지막으로 5장에서는 결론과 향후 연구에 관한 내용을 다루며 논문을 마친다.

II. 관련연구

2.1 화이트박스 암호

공격자에게 막강한 권한을 제공하는 화이트박스 환경은 내부 시스템에 입력과 출력, 메모리 정보, 디버깅정보 등 정보 대부분이 공개된 상태를 말한다. 공개된 정보를 활용하여 공격자는 시스템에 다양한 공격을 시도하여 암호화키를 찾아낼 수 있게 된다. 화이트박스 환경에서 암호 알고리즘을 안전하게 운용하기 위해서는 암호 알고리즘의 입출력, 구현방식 등이 공개되더라도 공격자가 공개된 정보를 활용하여 공격할 수 없어야 한다. 이러한 이유로, 다양한 방법의 화이트박스 암호 알고리즘 구현 방법이 제시되었다. 2002년에 최초로 화이트박스 환경에서 적용 가능한 AES와 DES 암호가 발표되었다[4][5]. 이 화이트박스 암호 알고리즘은 내부연산과 키 정보를 감추기 위해서 내외부에 인코딩을 적용하여 내부 정보가 보호된 LUT 형태의 테이블을 생성한다. 이 암호 알고리즘은 테이블에 대한 참조연산만을 이용하여 AES 암호 알고리즘연산을 수행하도록 구현하였다.

2.1.1 WBC-AES

S. Chow 외 2인에 의하여 발표된 LUT 형태의 WBC-AES[4] 알고리즘에 대한 설명을 위해서 AES-128 알고리즘을 예로 든다. AES-128 알고리즘은 Fig. 1의 왼쪽과 같은 구조를 가지며, 10개의 라운드로 구성되며, 각각의 라운드의 내부연산 4X4 행렬의 연산으로 수행된다. 마지막 라운드에서의 MixColumns 과정이 생략되는 것을 제외하고는 모든 라운드가 SubBytes, ShiftRows, MixColumns, AddRoundKey 4가지 과정으로 구성되어있다.

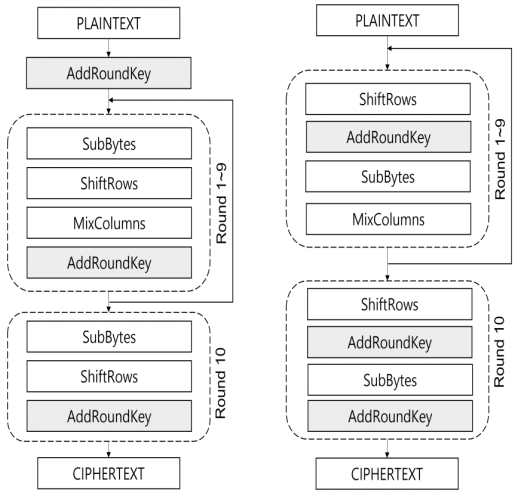

Fig. 1. AES(Left), WBC-AES(Right) Structure

WBC-AES[3]는 Fig. 1의 오른쪽과 같은 구조를 가진다. WBC-AES는 기존의 AES의 라운드 마지막에 수행되던 AddRoundKey 연산을 SubBytes 연산 이전에 수행하며, 고정된 키에 대한 AddRoundKey와 SubBytes의 연산을 결합하여, 고정된 키에 의존적인 LUT 테이블을 만든 후 T 테이블로 명명한다.

아래 식 (1)은 상기 설명을 수식으로 나타낸 것으로 $T_{(i,j)}^{(r)}(x)$는 1~9라운드까지의 T 테이블을 나타내며, $T_{(i,j)}^{(10)}(x)$는 10라운드의 T 테이블을 나타낸다. LUT 테이블의 i, j는 각각 행과 열을 의미하며, r은 라운드를 의미한다. S는 비선형 S-Box를 의미하며, $k_{(i,j)}^{(r)}$은 r라운드의 고정된 키에 대한 연산을 나타내며, $k_{(i,j)}^{(10)}$, $k_{(i,j)}^{(11)}$ 10라운드와 마지막 라운드에 키에 대한 연산을 표현한다.

$$\begin{aligned} &T_{(i,j)}^{(r)}(x) = S(x \oplus (k_{(i,j)}^{(r)})) \\ &T_{(i,j)}^{(10)}(x) = S(x \oplus (k_{(i,j)}^{(10)}) \oplus (k_{(i,j)}^{(11)})) \\ &(0 \le i,j \le 3, 1 \le r \le 9) \end{aligned} \tag{1}$$

다음으로 MixColumns의 연산은 아핀 변환(Affine Transformation)으로 구성된 연산을 테이블화하는 방식으로 구성되며, 식 (2)와 같이 A행렬을 열벡터로 분할하여 행렬 곱 연산을 수행한 다음에 XOR 연산을 수행한다.

$$GF(2)^m \rightarrow GF(2)^n \tag{2}$$

$$
\begin{aligned}
y &= Ax \oplus c \\
\begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix} &= \begin{bmatrix} A_{(0,0)} \cdots A_{(0,m)} \\ \vdots \quad \cdots \quad \vdots \\ A_{(n,0)} \cdots A_{(n,m)} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_m \end{bmatrix} \oplus \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \\
\begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix} &= \begin{bmatrix} A_{(0,0)} \\ \vdots \\ A_{(n,0)} \end{bmatrix} x_1 \oplus \begin{bmatrix} A_{(0,1)} \\ \vdots \\ A_{(n,1)} \end{bmatrix} x_2 \oplus \cdots \oplus \begin{bmatrix} A_{(0,m)} \\ \vdots \\ A_{(n,m)} \end{bmatrix} x_m \oplus \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix}
\end{aligned}
$$

식 (3)은 상기 설명된 방법을 적용하여 MixColumns 연산을 수행한 과정이다. 기존의 4X4 행렬은 4개의 열로 분할 하여 행렬 곱과 XOR 연산으로 정의할 수 있다. $MC_{(i,j)}$ 에서 i, j는 각각 MixColumns의 행과 열벡터를 의미한다.

$$
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3}
$$

$$
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} x_0 \oplus \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} x_1 \oplus \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} x_2 \oplus \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} x_3
$$

$$
[y_i] = \bigoplus_{j=0}^{3} (MC_i \bullet x_j) \qquad (i,j = 0,1,2,3)
$$

$$
MC_0 : y = \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} x,\ MC_1 : y = \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} x,
$$

$$
MC_2 : y = \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} x,\ MC_3 : y = \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} x
$$

WBC-AES에서의 ShiftRows는 state의 순서를 섞는 연산으로 라운드의 처음에 수행한다. 따라서, 한 라운드에 대한 수식은 식 (4)와 같으며 T 테이블과 MixColumns의 연산을 합쳐 TMC 테이블로 명명한다.

$$
\begin{aligned}
TMC_{(i,j)}^{(r)} &= MC_i \bullet T \\
&= mc_i \bullet T_{(i,j)}^{r} \\
&= mc_i \bullet S(x \oplus k_{(i,j)}^{(r)}) \\
&(0 \le i,j \le 3, 1 \le r \le 9)
\end{aligned} \tag{4}
$$

Fig. 2는 TMC 테이블에 대한 수행과정을 나타내며, 한 개의 라운드에서 TMC 테이블은 총 16개가 사용된다. Fig. 2와 같이 하나의 8bit의 입력은 32bit의 출력이 되고, 총 128bit의 입력은 512bit의 출력이 된다. 512bit 출력은 각각의 4bit 단위로

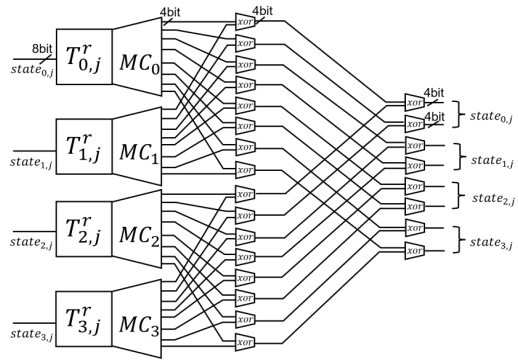

Fig. 2. Execution process of four TMC Tables

연산이 수행되며, 2개의 4bit 데이터는 XOR 연산을 거쳐 4bit를 출력한다. 한 개의 라운드에서 128bit의 입력은 총 128개의 XOR 연산을 통하여 다시 128bit를 출력한다.

2.2 부채널 분석

암호에 대한 분석은 입력 평문과 출력 암호문을 비교하여 내부 알고리즘에 대한 취약성을 찾아내는 방법으로 일반적으로 알고리즘 내부의 산술적 어려움에 기인한다. 부채널 분석은 암호 알고리즘에 대한 취약성을 분석하는 방법과는 다르게 물리적인 구현에 대한 취약성을 분석하는 방법이다. 암호학적으로 안전한 암호 알고리즘도 하드웨어로 구현하였을 때 취약점이 발생할 수 있다. 부채널 분석은 암호 알고리즘이 탑재된 장치에서 발생하는 열, 소리, 전자기파, 전력 소모와 같은 부가적인 정보들을 활용하여 암호화 키를 찾아내는 강력한 공격 방법으로 알려져 있다.

2.2.1 전력분석 공격(Power Analysis)

부채널 분석에 사용되는 대표적인 전력분석 공격으로는 SPA, DPA, CPA로 크게 3가지 방법이 존재한다[7][8]. 먼저, SPA는 한 개 혹은 소수의 전력 파형을 이용하여 암호화키를 분석해내는 방법이며, DPA는 수집된 다수의 전력 파형에 통계적인 방법을 적용하여 암호화 키를 찾아내는 분석 방법이다. 마지막으로 CPA는 다수의 전력 파형을 사용하여 측정된 전력 소모량과 계산 가능한 중간값 사이의 상관관계를 분석하여 암호화 키를 분석해내는 방법이다.

2.2.2 DCA(Differential Computation Analysis)

DCA 분석은 기존 부채널 분석 방법의 방법론을 소프트웨어적으로 수집한 부채널 정보에 응용한 분석 기법이다. 전력분석방법은 하드웨어나 소프트웨어에서 암호 알고리즘이 수행되는 동안 발생하는 전자파 또는 소비전력 등의 부채널 정보들의 파형 차이를 이용하는 분석 기법이지만, DCA 분석은 소프트웨어 환경에서 암호 알고리즘을 동작하며 발생하는 메모리 참조 정보를 부채널 정보로써 수집하게 된다. 부채널 정보로 사용하는 메모리 참조 정보는 메모리에 데이터가 쓰여지거나 읽어질 때 해당 메모리 주소의 최하위 바이트 정보를 수집하게 된다. DCA 분석은 수집된 메모리 정보에서 특정 메모리 주소에 대한 편향된 접근 정보를 통계적으로 해석하여 키를 복구한다.

기존의 전력분석 방법은 측정환경에 따라서 수많은 노이즈가 발생하여 실제 전력 값과는 다른 전력 값들이 수집될 가능성이 있다. 이러한 이유로, 전력 소모에 대한 데이터를 수집하기 위해 큰 노력이 요구되지만, DCA 분석의 경우, Valgrind 라는 소프트웨어 디버거를 이용하여 암호 알고리즘에 대한 메모리 정보만을 노이즈 없이 명확한 형태로 수집할 수 있다. DCA 분석은 이처럼 이상적인 부채널 정보를 활용하여 화이트박스 암호에 대한 공격을 수행하며, 그 성능이 빠르고 정확하여 강력한 공격기법으로 분류되고 있다.

J.W. Bos 외 3인은 WBC-AES에 대하여 DCA 분석 결과 16바이트의 암호화 키를 모두 복구할 수 있는 것을 성능지표로 제시하였다.

2.2.3 부채널 분석 대응 방안

부채널 분석은 측정된 값과 계산 가능한 중간값 사이에 연관성을 분석하여 암호화 키를 찾아내는 분석 방법이다. 이러한 부채널 분석에 대응하는 방안은 측정된 데이터와 추측값 사이에 연관성을 제거하여 분석에 어려움을 주거나 암호화 키를 찾지 못하게 한다. 대표적인 방법으로는 암호 알고리즘에 하이딩 기법이나 마스킹 기법을 적용하는 방법이 존재한다.

먼저, 하이딩 기법이란, 알고리즘에 Dummy 연산을 적용하여, 예상하는 중간값 지점의 계산 값과 실제 측정값의 차이를 발생시키는 방법이다[9].

하이딩 기법에는 이처럼 Dummy 연산을 삽입하는 방법 이외에도 셔플링과 같이 알고리즘의 수행순서를 변경하여 차이를 발생시키는 등 다양한 방법이 존재한다.

마스킹 기법이란 예상하는 중간값 지점의 계산 값을 랜덤하게 만들어 실제 측정된 데이터의 값과 차이를 발생시키는 방법이다. 마스킹 기법에는 Boolean 마스킹, Arithmetic 마스킹 등 다양한 종류의 마스킹 기법이 존재한다[10].

DCA 분석에 관한 대응연구는, S. Lee 외 2인이 S. Chow의 테이블 중간에 마스킹 정보를 생성하는 테이블을 설계하여, 기존의 부채널 대응 기술인 마스킹 기법과 같이 라운드의 연산 결과와 무관한 난수가 발생하는 연구가 진행되었다[11]. 또한, Y. Lee 외 2인은 기존의 S. Lee 외 2인 제안한 DCA 분석에 대한 대응연구에 올바른 시점 분석을 통하여 새롭게 조합한 2차 trace를 이용하여 마스킹 화이트박스를 무력화시키는 연구가 2020년에 제안되었다[12]. 기존의 부채널 분석에 대응하기 위해서 다양한 연구가 진행되었으나, DCA 분석에 대하여 하이딩기법을 적용한 연구사례는 전무하였다.

III. Dummy LUT 연산을 이용한 대응 방안

DCA 분석 도구는 기존의 CPA 분석의 방법으로 비밀 키를 찾게 된다. DCA 분석은 비밀 키를 이용하여 측정된 값과 16개의 바이트에 대하여 256가지의 추측 키($00 \sim FF$)를 이용한 중간값과의 상관관계 정도를 피어슨 상관계수를 이용하여 계산하고, 가장 높은 상관관계의 값을 갖는 16바이트의 키 후보군을 제시한다.

Fig. 3의 그래프는 16개의 바이트에 대하여 각 바이트의 256가지의 추측 키($00 \sim FF$)를 이용한 중간값과 측정값 사이의 상관관계의 값을 비교하여 그래프에 나타냈으며, 한 바이트당 256가지의 값 중 가장 큰 첨두값에 점을 찍은 모습이다. 이 과정을 16번 반복하여 그래프에 상위 16개의 값을 표시하였다. 이 그래프는 노이즈가 없는 이상적인 데이터를 이용하여 상관관계 분석을 진행한 그래프로, 틀린 키(wrong key)에 비하여 옳은 키(right key)의 상관계수가 현저하게 높은 것을 확인할 수 있다.

본 연구에서 제안하는 DCA 분석의 대응방안은 WBC- AES의 LUT로 구현된 코드와 유사한 형태의 Dummy LUT 연산을 삽입하여 메모리 정보에 혼선을 일으키는 것을 목표로 한다.

Table. 1은 제안한 WBC-AES의 내부 알고리즘

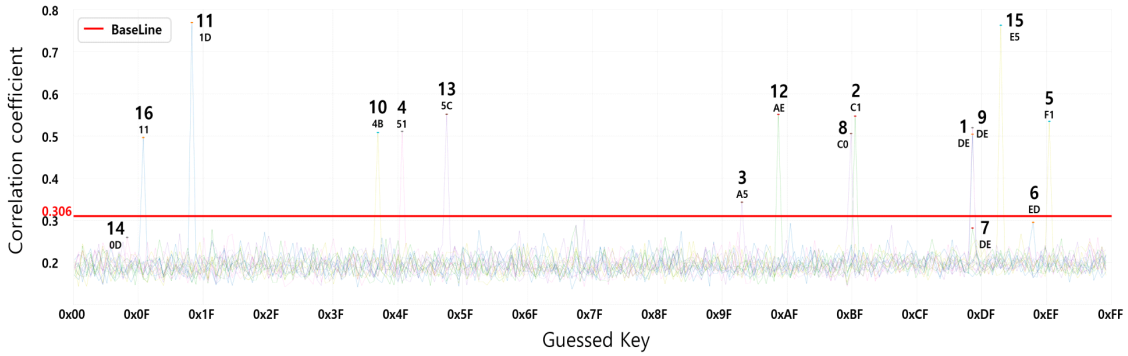

Fig. 3. Correlation Analysis Graph (WBC-AES)

의 내부 수행과정을 나타낸다. 먼저 평문이 입력되게 되면 평문은 배열로 나뉘어 미리 계산된 LUT 테이블을 참조하며 다른 값들을 연쇄적으로 나타내게 된다. 결과적으로 입력된 평문은 전체 4,048개의 LUT 테이블을 참조하여 암호문을 생성하게 된다.

Table. 2는 본 논문에서 제안하고자 하는 Dummy LUT의 생성원리를 나타낸다. 본 논문에서는 상기 설명한 바와 같이 기존의 WBC-AES의 LUT 생성원리와 유사한 방식의 LUT를 생성하여, 메모리 참조 과정에서 혼란을 일으키는 것을 목표로 한다.

Table 1. WBC-AES Computing Process

```
Input  : Plaintext
Output: Ciphertext
p = Plaintext
LUT = S. Chow's Pre-Computed LUT
    for Total Number of LUT do
        r = LUT(p)
    end for
Ciphertext = r
return Ciphertext
```

Table 2. Dummy LUT Making Algorithm

```
Input  : Plaintext for Dummy LUT
Output: Dummy LUT
pr_p = Pre-Fixed Plaintext
LUT = S. Chow's Pre-Computed LUT
    for D_LUT = (128B, 256B, 2KB, 4KB,
                 6KB, 8KB, 10KB, 20KB,
                 30KB, 40KB, 50KB) do
        d = LUT(pr_p)
    end for
Dummy LUT = d
return Dummy LUT
*B = Byte, KB = KByte
```

Dummy LUT는 기존에 LUT에서 평문이 입력되는 것과는 다르게, 사전에 정의해놓은 데이터를 평문으로 할당하게 된다. 미리 정의된 평문 데이터를 기 알고리즘과 같은 LUT에 적용하여 128B, 256B, 2KB, 4KB, 6KB, 8KB, 10KB, 20KB, 30KB, 40KB, 50KB 크기의 임의의 데이터인 Dummy LUT를 생성한다. B는 Byte를 의미하며 KB는 KBytes를 의미한다.

Fig. 4는 본 논문에서 제안하고자 하는 Dummy LUT가 삽입된 WBC-AES 알고리즘 순서를 나타

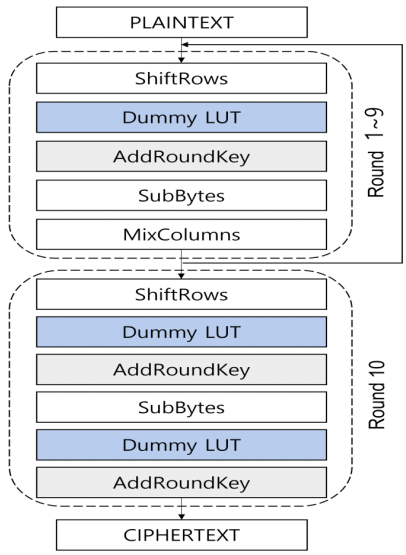

Fig. 4. Overview of the Proposed method

낸다. Dummy LUT 연산은 Fig. 4에서 보는 바와 같이 기존의 WBC-AES의 AddRoundKey 연산 앞쪽에 파란색으로 Dummy LUT 형태로 삽입하여 표현하였다.

본 연구는 WBC-AES의 메모리 참조특성을 이용한 실험으로, 고정된 키에 대한 데이터를 메모리를 통하여 읽어드리는 과정에서 Dummy LUT의 삽입이 반복적인 메모리 참조에 혼선을 발생시킬 수 있으며, 이로 인하여 옳은 키를 추측하지 못하는 것을 기대한다. 또한, 총 11번의 Dummy LUT가 삽입하는 과정에서 매번 난수에 의하여 무작위로 결정된 1~10 사이의 수만큼 Dummy LUT를 반복하여 삽입한다.

이처럼 Dummy LUT의 크기를 난수에 의해서 선택된 임의의 횟수만큼 반복하여 구성한 이유는 다음과 같다. DCA 분석은 DPA나 CPA 분석처럼 암호 알고리즘에 대한 반복수행을 통하여 얻은 부채널 정보를 통계적으로 분석하게 된다. 총 11회에 걸쳐서 Dummy LUT를 고정된 특정 위치에 삽입하게 되면, 반복되는 알고리즘의 수행과정에서 고정된 같은 위치에서 매번 다른 데이터가 존재하게 되는 효과를 얻을 것으로 예상된다. 이러한 과정은 DCA 분석에 노이즈 역할을 충분히 수행할 것으로 판단되었다. 본 논문에서는 이렇게 고정된 위치에 삽입된 Dummy LUT의 크기가 기존에 다양한 부채널 분석 중 효율적이고 강력한 공격기법으로 평가된 DCA 분석에 얼마나 효과적으로 대응할 수 있는지 파악하고자, Dummy LUT의 크기를 변경하여 실험을 진행하였다. Table. 3은 본 논문에서 제안하는 방안에 대한 알고리즘을 나타낸다.

4장에서는 Dummy LUT의 크기를 다르게 삽입함에 따라서, DCA 분석의 상관관계값의 변화량을 조사하여 자세하게 설명한다.

Table 3. Dummy LUT Insertion Algorithm

Input : Plaintext
Output: Ciphertext

```
rnd_value = random(1,10)
D_LUT = Dummy LUT
  for D_LUT = (128B, 256B, 2KB, 4KB,
               6KB, 8KB, 10KB, 20KB,
               30KB, 40KB, 50KB) do
  state = Plaintext

    // 1~9 Round
    for r=1 to 9
      state = Shiftrows(state)
      for rnd_value do
        Insertion D_LUT
      end for
      state = AddRoundKey(state, k^r)
      state = SubBytes(state)
      state = Mixcoulmns(state)
    end for

    // 10 Round Process
    state = Shiftrows(state)
    for rnd_value do
      Insertion D_LUT
    end for
    state = AddRoundKey(state, k^r10)
    state = SubBytes(state)
    for rnd_value do
      Insertion D_LUT
    end for
    state = AddRoundKey(state, k^r11)
  end for
Ciphertext= state
return Ciphertext
```

IV. 실험 및 결과

4.1 실험환경

DCA 분석을 위한 실험환경은 Table. 4와 같으며, WBC-AES에 구성된 LUT의 종류는 Table. 5와 같이 구성되며, 종류는 Type 1, Type 2, Type 3, Type 4로 구성되어있다. Type 1 은 8bit 입력으로 128bit 출력하게 되고, 입력과 출력에 16개씩 존재한다. Type 2와 Type 3은 상기 설명한 TMC 테이블을 구현하는 데 사용된다. 마지막으로 Type 4는 4bit의 입력 2개를 이용하여 XOR 연산을 수행한 후 하나의 4bit를 출력하는 기능을 한다.

Table 4. Experimental Environment Configuration

OS	Ubuntu 18.04.6 LTS
Memory	32GB
CPU	Intel Core i7 - 12700
Source Code	WBC-AES
DCA Tool	J.W.Bos's Github

Table 5. WBC-AES LUT Configuration
(Info. : Information, Conf. :Configuration)

Conf. \ Info.	Quantity	Memory Size (B)	Total (KB)
Type 1 (8bit to 128bit)	32	4096	128
Type 2 (8bit to 32bit)	144	1024	144
Type 3 (8bit to 32bit)	144	1024	144
Type 4 (8bit to 4bit)	2,688	128	336

4.2 실험방법

실험은 3장에서의 설명과 같이 Dummy LUT를 WBC-AES 매 라운드의 AddRoundKey의 앞부분에 삽입하여 실험을 진행하였다. 삽입되는 Dummy LUT의 크기가 DCA 분석 성능 저하에 미치는 영향을 파악하기 위해서 Dummy LUT의 크기를 128B, 256B, 2KB, 4KB, 6KB, 8KB, 10KB, 20KB, 30KB, 40KB, 50KB로 변경하며 결과를 분석하였다. 3장에서 설명한 바와 같이 Dummy LUT가 DCA 분석에 대한 대응이 된다면, Fig. 3의 그래프의 첨두값은 값이 낮아질 것이며, 다른 데이터들과 비슷한 수치가 될 것이다. 또한, 첨두값과 다른 바이트의 상관관계값과의 차이가 없어지게 되면, 다른 추측 키값들이 첨두값이 될 가능성이 커지게 된다. 결과적으로, DCA 분석은 틀린 키를 추측하게 될 가능성이 커진다.

Fig. 3의 BaseLine(L)은 실험의 결과를 해석하기 위한 도구로써 아래의 식 (5)와 같다. 식 (5)에서 GB(Guessed Byte)는 총 16바이트 키 값의 자리 수를 의미하며, GK(Guessed Key)는 총 256자리의 추측 키를 의미한다. D는 256가지의 추측키에 대한 데이터를 GB만큼 반복하여 계산한 데이터의 상관관계 값을 의미한다. 그래프의 모든 상관관계 데이터들의 평균($M, mean(D)$)와 분산($V, variance(D)$)를 구한 후, 그 합을 L로 잡았다. L은 데이터의 분포 정도를 고려하여, 분산 값이 클수록 L은 크게 높아지고, 분산 값이 작을수록 L은 작게 높아지도록 설정하였다.

$$\begin{aligned} D &= (GB, GK) \\ M &= mean(D) \\ V &= variance(D) \\ L &= M + V \end{aligned} \tag{5}$$

실험에서는 데이터의 분포 정도를 확인하기 위해서 L을 활용하여 L보다 큰 값을 갖는 데이터의 개수와 이러한 데이터의 값들의 표준편차를 계산한다. 계산된 표준편차의 값에 따라서, L 이상의 데이터들 분포 정도를 확인하게 된다. 앞서 설명한 바와 같이, Dummy LUT가 DCA 분석에 대응이 된다면 첨두값은 낮아지게 되며, L 이상의 데이터들의 수가 증가할 것이다. 또한, L 이상의 데이터들의 표준편차는 낮아지게 되어, 데이터들이 밀집된 형태를 갖추게 될 것이다. 그 결과 다른 상관관계 값들과의 혼선이 발생하여 옳은 키를 추측하지 못하게 될 것이다.

4.3 실험 결과

DCA 분석은 알고리즘을 반복 수행하며 얻어진 데이터를 통계적으로 분석하는 공격으로, 알고리즘을 반복적으로 수행할수록 분석에 약간의 오차가 발생한다. 또한, Dummy LUT는 알고리즘의 라운드마다 1~10 사이의 난수에 의해서 랜덤한 반복횟수로 삽입되는데, 이때의 Dummy LUT는 통계적으로 반복수행을 거듭할수록 약 66번 등장하게 된다.

Fig. 5는 메모리 정보 시각화 도구[13]를 이용하여 WBC-AES와 8KB의 Dummy LUT를 삽입한 WBC-AES에 대한 암호화 과정의 메모리 정보를 시각화한 그래프로 Y축은 알고리즘의 시작부터 종료

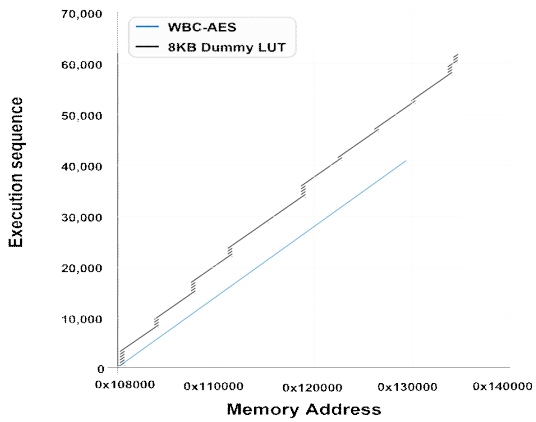

Fig. 5. Comparison Graph (WBC-AES vs Proposed WBC-AES)

시점까지의 순서를 0부터 순서대로 나타냈으며, X축에는 알고리즘의 수행과정에서 참조하는 메모리 주소의 주소값이 기록된다.

Fig. 5의 8KB의 Dummy LUT를 삽입한 그래프를 보면, 고정된 11개의 지점에 1~10 사이에 임의의 개수만큼 Dummy LUT가 삽입된 모습을 확인할 수 있다. 그래프를 통하여 반복되는 수행과정에서 라운드마다 메모리 정보에 대한 데이터가 매번 상이하게 시작됨을 확인할 수 있다.

Fig. 6은 256B Dummy LUT를 적용한 DCA 분석의 측정값과 추측한 키의 중간값 사이의 상관관계를 분석하여 그래프로 나타낸 그래프이다. 총 16개의 그래프를 겹쳐서 표시하였으며, 각 그래프의 첨두값을 데이터와 함께 표시하였다. Dummy LUT를 사용한 경우, Fig. 3과 Fig. 6과 비교하면 가장 높은 상관관계값을 갖는 데이터가 Fig. 3에서 약 0.74였으나, Fig. 6에서는 0.17로 현저하게 낮아짐을 확인할 수 있다. 또한, Fig. 6의 첨두값 16개 바이트의 값들이 약 0.17 ~ 0.13 사이의 값으로 낮아짐을 확인할 수 있다. 이러한 첨두값들은 Fig. 3과 비교하였을 때 전체적으로 현저하게 낮아진 값들이다. 결과적으로, WBC-AES의 경우보다, 본 논문에서 제안하는 Dummy LUT를 적용하였을 때 그래프의 상관관계값들이 전반적으로 낮은 값이 됨을 확인하였다.

Table. 6은 Dummy LUT의 크기에 따른 데이터의 분포에 대한 실험을 표로 정리한 내용이다. Table. 6에서 표기된 L은 BaseLine의 값을 의미하고, NL은 L 이상의 데이터 개수를 의미한다. 마지막 STD는 NL의 표준편차 값을 의미한다. 실험에 따르면, Dummy LUT의 크기가 작으면 작을수록

Table 6. Results of the Experiment
(Info. : Information, Conf. :Configuration)

Conf. \ Info.	L	NL	STD
WBC-AES	0.307	13	0.010136
50KB	0.136	19	0.001181
40KB	0.132	21	0.000604
30KB	0.129	34	0.000501
20KB	0.129	36	0.000414
10KB	0.128	34	0.000387
8KB	0.125	54	0.000140
6KB	0.126	52	0.000189
4KB	0.123	66	0.000072
2KB	0.130	64	0.000147
256B	0.124	61	0.000074
128B	0.131	69	0.000102

많은 데이터가 L 보다 큰 값을 갖게 되는 것을 확인할 수 있다.

이러한 결과는 Dummy LUT의 크기가 작아짐에 따라서, 데이터들의 특징 없이 서로 유사한 데이터가 됨을 의미한다. Table. 6에서 STD를 비교하면, WBC-AES는 0.0101의 값을 가지며, Dummy LUT의 크기가 작은 128B의 경우 STD의 값이 약 0.0001의 값으로 작아지게 된다. 이것은 WBC-AES의 L 이상의 13개의 데이터는 전반적으로 흩어져 있으며, 128B의 Dummy LUT를 삽입했을 때의 L 이상의 69개의 데이터는 서로 비슷한 값들이며 모여있음을 의미한다. Fig. 7은 Table. 6에서의 NL의 수를 그래프에 표현하였으며, Fig. 8은 Table. 6 에서의 STD의 값을 그래프에 표현하였다.

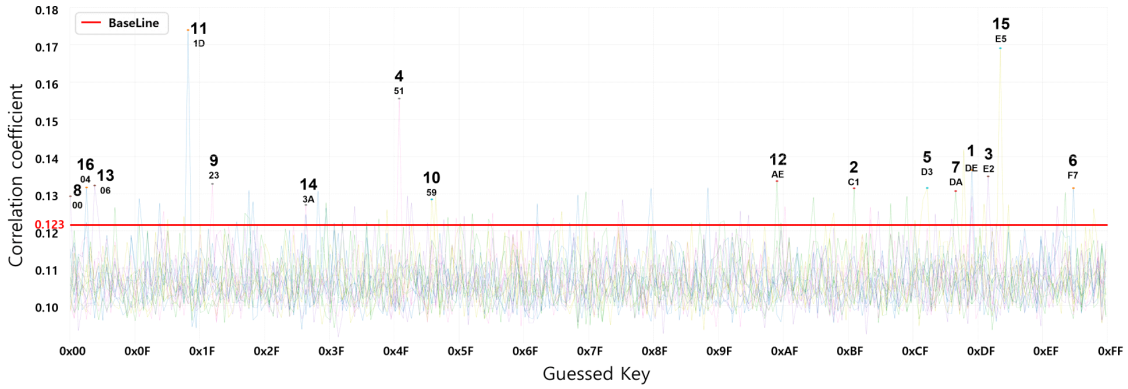

Fig. 6. Correlation Analysis Graph (Dummy LUT - 256B)

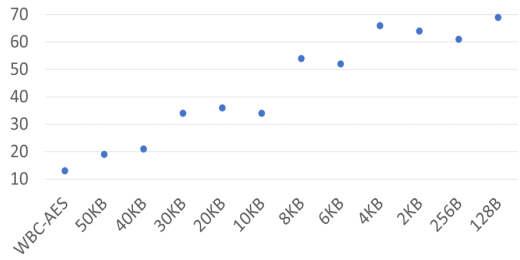

Fig. 7. Number of Data above the BaseLine

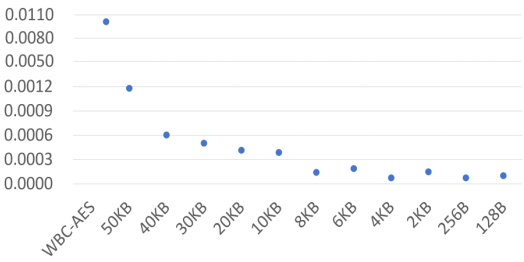

Fig. 8. Standard deviation values of data above the BaseLine

Table 7. Performance comparison table (*Perf. : Performance, Conf,:Configuration) (Excution Time : 10,000 iteration)

Perf. / Conf.	Memory Size (KB)	Execution Time (sec)	Number of Correct Key Guessed
WBC-AES	508	0.0241	16
50KB	3300	0.1064	11
40KB	1980	0.0898	11
30KB	1650	0.0761	10
20KB	1320	0.0632	8
10KB	660	0.0478	8
8KB	528	0.0448	7
6KB	396	0.0412	7
4KB	264	0.0371	6
2KB	132	0.0349	5
256B	16.5	0.0343	6
128B	8.25	0.0340	6

Table. 7은 Dummy LUT의 크기를 변경하며 실험한 결과를 나타낸다. 전체적으로 메모리의 크기가 커짐에 따라서 암호 알고리즘을 10000회 반복한 수행 시간이 늘어난 것을 확인할 수 있었다. 또한, Dummy LUT의 크기가 50KB일 경우 기존 DCA 분석 대비 약 31.25% 감소하여 공격 성능은 약 68.75%로 확인되었으며, 또한, Dummy의 크기가 4KB 이하로 작아질 경우 기존 공격 대비 약 68.8% 정도 낮아진 약 31.2%로 공격 성능이 크게 낮아짐을 확인할 수 있었다.

V. 결 론

본 논문에서는 화이트박스 환경에서의 부채널공격인 DCA 분석에 효과적인 대응을 하기위해서 WBC-AES에 Dummy LUT를 삽입하여, 공격성능을 낮추는 연구를 수행하였으며, 또한 이러한 Dummy LUT의 크기가 공격성능에 끼치는 영향을 정량적 나타내고자 연구를 수행하였다. 먼저, Fig. 8에서 WBC-AES와 50K Dummy LUT를 삽입한 경우의 STD 를 비교해본 결과 50K Dummy LUT가 약 10배 작았으며, 또한 128B Dummy LUT를 삽입한 경우 WBC-AES의 STD에 비하여 약 100배 작은 값을 갖는 것을 확인할 수 있었다.

실험결과 DCA 분석이 정상적으로 수행되는 경우 옳은 키(right key)에서만 첨두값이 높이 치솟는 Fig. 3과 같은 구조를 갖게 되고, Fig. 7의 결과 NL의 갯수가 13개로 현저하게 낮았다. 그 값들은 Fig. 8의 결과 밀집되어있지 않고, 흩어져 있어서, 키를 정확하고 빠르게 찾아낼 수 있게 된다. 하지만, Dummy LUT를 삽입하고, 그 크기가 작아지면, Fig. 7과 같이 NL의 수가 많아지며, Fig. 8의 결과, WBC-AES 보다 4KB 이하의 데이터에서 NL의 STD가 0에 가까운 값들을 갖는 것을 확인할 수 있었다. 이것은 4KB 이하의 데이터의 NL이 밀집되어있는 것을 의미한다. 결과적으로 작은 Dummy LUT를 삽입할수록 상관관계의 값들의 특징이 점점 사라지고 서로 유사한 데이터가 되어서 옳은 키를 추측하지 못하는 결과를 발생하게 된다. 강력한 공격으로 평가되는 DCA 분석에 아주 간단한 Dummy LUT를 추가함으로써, DCA 분석에 효과적으로 대응됨을 확인하였다.

하지만 이러한 하이딩 기법은 단순한 공격기법이며, 부채널 분야에서도 공격에 대응 연구가 많이 진행되었다. 본 연구 또한, 간단한 Dummy LUT를 삽입하여서 공격 성능을 효과적으로 낮추었지만, 이는 기존의 Dummy를 이용한 공격에 같은 대응 방식으로 시행횟수를 크게 늘리면, 다시 상관관계의 값이

상승하게 된다. 하지만 본 연구는 임의의 횟수만큼 Dummy LUT를 반복 삽입하였고, 이때의 임의의 횟수는 설계자가 임의로 조정이 가능하다. 이러한 이유로 임의의 횟수가 늘어나면, 공격자가 통계적인 분석을 하기 위해서는 더 많은 부채널 정보를 수집해야 하는 불편함을 제공하게 된다.

본 논문에서 제안하는 방안은 작은 Dummy LUT를 이용한 대응이 효과적으로 분석 성능을 낮추었기 때문에, 메모리가 제한적인 환경에서도 DCA 분석에 대한 대응 가능성을 제시하였다. 하지만, Table. 7의 128B Dummy LUT를 이용한 실험결과는 기존의 WBC-AES에 1.6%에 해당하는 작은 메모리 상승이 있었으나, 알고리즘의 수행 시간은 약 1.5배 느려지는 것을 확인할 수 있었다. 이에 앞으로는 Dummy LUT가 차지하는 메모리 크기와 알고리즘의 수행 시간 사이의 상관관계를 분석하여 최소의 메모리사용으로 최적의 수행 시간을 갖는 효율적인 Dummy LUT에 대한 연구를 진행할 예정이다.

References

[1] National Institute of Standards and Technology (NIST), "Data Encryption Standard (DES)", FIPS Publication 46-3, U.S. Department of Commerce/NIST, Washington, D.C., 1999.

[2] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", FIPS-197, U.S. Department of Commerce/NIST, Washington, D.C., 2001.

[3] J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum, E.W. Felten, "Lest we remember: cold boot attacks on encryption keys", USENIX Security Symposium, pp. 45-60, Aug. 2008.

[4] S. Chow, P. Eisen, H. Johnson, P.C van Oorschot, "White-box cryptography and an AES implementation," Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002, Springer, Heidelberg, pp. 250-270, Jan. 2003.

[5] S. Chow, P. Eisen, H. Johnson, P.C van Oorschot, "A White-box DES Implementation for DRM Applications," Security and Privacy in Digital Rights Management: ACM CCS-9 Workshop, DRM 2002, Springer, pp. 1-15, Nov. 2003.

[6] J.W. Bos, C. Hubain, W. Michiels, P. Teuwen, "Differential computation analysis: hiding your white-box designs is not enough", Cryptographic Hardware and Embedded Systems: CHES 2016, Springer, Heidelberg, pp. 215-236, Aug. 2016.

[7] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis", Advances in Cryptology, CRYPTO'99, LNCS 1666, pp. 388-397, Aug. 1999.

[8] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model", Cryptographic Hardware and Embedded Systems, CHES'04, LNCS 3156, pp. 1-15, Aug. 2004.

[9] Herbst, C., Oswald, E., & Mangard, S. "An AES Smart Card Implementation Resistant to Power Analysis Attacks". In ACNS 2006, LNCS, Vol. 3989, pp. 239-252, Aug. 2006.

[10] L. Goubin, "A sound method for switching between Boolean and arithmetic mask-ing," Cryptographic Hardware and Embedded Systems, CHES'01, LNCS 2162, pp. 3-15, Jan. 2001

[11] S. Lee, T. Kim, and Y. Kang, "A Masked White-Box Cryptographic Implementation for Protecting Against Differential Computation Analysis", IEEE Transactions on Information Forensics and Security, vol. 13, no. 10, pp. 2602-2615, Apr. 2018.

[12] Y. Lee, S. Jin, H. Kim, H. Kim, & S.

Hong, "A new higher-order differential computation analysis technique for masked white-box AES", Journal of the Korean Institute of Information Security and Cryptology, vol. 30, no. 1, pp. 1-15. Feb. 2020

〔13〕 "SideChannelMarvels," GitHub, https://github.com/SideChannelMarvels (2022.12.21).

〔14〕 "Valgrind: a programming tool for memory debugging, memory leak detection, and profiling," Valgrind.org, https://valgrind.org (2022.12.23.).

〈저자소개〉

최 민 영 (Minyeong Choi) 학생회원
2019년 2월: 전북대학교 전자공학과 졸업
2021년 3월~현재: 서울과학기술대학교 일반대학원 컴퓨터공학과 석사과정
〈관심분야〉 정보보호, 암호학, 부채널 분석 등

석 병 진 (Byoungjin Seok) 종신회원
2017년 8월: 서울과학기술대학교 컴퓨터공학과 졸업
2019년 2월: 서울과학기술대학교 컴퓨터공학과 석사
2022년 2월: 서울과학기술대학교 컴퓨터공학과 박사
2022년 3월~현재 서울과학기술대학교 전기정보기술연구소 연구원
〈관심분야〉 정보보호, 암호학, 암호분석, 디지털포렌식 등

서 승 희 (Seunghee Seo) 학생회원
2017년 2월: 서울과학기술대학교 컴퓨터공학 졸업
2019년 2월: 서울과학기술대학교 컴퓨터공학 석사
2020년 3월~현재: 서울과학기술대학교 대학원 컴퓨터공학과 박사과정
〈관심분야〉 메모리 포렌식, 패스워드 복구, 디지털포렌식 등

이 창 훈 (Changhoon Lee) 종신회원
2001년 3월: 한양대학교 자연과학부 수학전공 졸업
2003년 3월: 고려대학교 정보보호대학원 석사
2008년 3월: 고려대학교 정보경영전문대학원 정보보호전공 박사
2008년 4월~2008년 12월: 고려대학교 정보보호연구원 연구교수
2009년 3월~2012년 2월: 한신대학교 컴퓨터공학부 조교수
2012년 3월~2015년 3월: 서울과학기술대학교 컴퓨터공학과 조교수
2015년 4월~2019년 3월: 서울과학기술대학교 컴퓨터공학과 부교수
2019년 4월~현재 서울과학기술대학교 컴퓨터공학과 교수
〈관심분야〉 정보보호, 암호학, 디지털포렌식, 사이버보안 등