Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.4, Aug. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.4.527

CNN을 이용한 소비 전력 파형 기반 명령어 수준 역어셈블러 구현*

배 대 현,[1†] 하 재 철[2‡]
[1,2]호서대학교(학생, 교수)

Implementation of Instruction-Level Disassembler Based on Power Consumption Traces Using CNN*

Daehyeon Bae,[1†] Jaecheol Ha[2‡]
[1,2]Hoseo University(Student, Professor)

요 약

정보보호용 디바이스의 부채널 정보인 소비 전력 파형을 이용하면 내장된 비밀 키 뿐만 아니라 동작 명령어를 복구할 수 있음이 밝혀졌다. 최근에는 MLP 등과 같은 딥러닝 모델을 이용한 프로파일링 기반의 부채널 공격들이 연구되고 있다. 본 논문에서는 마이크로 컨트롤러 AVR XMEGA128-D4가 사용하는 명령어에 대한 역어셈블러를 구현하였다. 명령어에 대한 템플릿 파형을 수집하고 전처리하는 과정을 자동화하였으며 CNN 딥러닝 모델을 사용하여 명령-코드를 분류하였다. 실험 결과, 전체 명령어는 약 87.5%의 정확도로, 사용 빈도가 높은 주요 명령어는 99.6%의 정확도로 분류될 수 있음을 확인하였다.

ABSTRACT

It has been found that an attacker can extract the secret key embedded in a security device and recover the operation instruction using power consumption traces which are some kind of side channel information. Many profiling-based side channel attacks based on a deep learning model such as MLP(Multi-Layer Perceptron) method are recently researched. In this paper, we implemented a disassembler for operation instruction set used in the micro-controller AVR XMEGA128-D4. After measuring the template traces on each instruction, we automatically made the pre-processing process and classified the operation instruction set using a deep learning model CNN. As an experimental result, we showed that all instructions are classified with 87.5% accuracy and some core instructions used frequently in device operation are with 99.6% respectively.

Keywords: Side-Channel Attack, Power Analysis, Deep Learning, Convolutional Neural Network(CNN), Disassembler

I. 서 론

P. Kocher 등이 제안한 차분 전력 분석(Differential Power Analysis) 공격[1]을 비롯한 여러 부채널 분석 공격이 등장한 이래로 전력 분석 공격은 암호 알고리즘 구현 시 필수적으로 고려해야 할 사항 중 하나로 자리 잡았다. 최근에는 머신러닝 분야가 활성화됨에 따라 이를 전력 분석 공격 분야에 접목하여 프로파일링(profiling) 혹은 비프로파일링(non-profiling) 방식으로 비밀 키를 찾는

Received(07. 23. 2020), Modified(07. 28. 2020)
Accepted(07. 28. 2020)
* 이 논문은 2019년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임.(20190852)

* 본 논문은 2020년도 하계 학술대회에 발표한 우수논문을 개선 및 확장한 것임

† 주저자, noeyheadb@gmail.com
‡ 교신저자,jcha@hoseo.edu(Corresponding author)

연구도 활발히 진행되고 있다.

지금까지의 전력 분석 공격 연구의 대부분은 부채널 누수 정보인 소비 전력을 이용해 비밀 키를 복구하는 데 초점을 맞추고 있었으나, 최근에는 MCU(Micro Controller Unit)에서 동작하는 명령어들을 복구할 수 있다는 사실이 실험을 통해 증명되었다. 그러나 국내·외적으로 소비 전력 기반 명령어 역어셈블러(disassembler)에 관한 연구는 아직 미흡한 실정이며, 특히 국내 연구 사례는 전무하다.

한편, J. Park 등[2]은 ATMega328P가 탑재된 타겟 보드에서 측정된 소비 전력 파형을 CWT(Continuous Wavelet Transform), PCA(Principal Components Analysis)[3], KL-divergence[4]를 통해 변환 및 특징을 추출한 후 QDA(Quadratic Classifier)[5], LDA (Latent Dirichlet Allocation)[6], SVM (Support Vector Machine)[7] 등의 분석 모델을 이용해 명령-코드(opcode) 112개와 레지스터 64개를 99.0%의 정확도로 분류하는 데 성공하였다. 또한, 최근에는 V. Cristiani 등[8]이 PIC16F가 탑재된 타겟 보드에서 측정한 전자기파를 LDA와 QDA를 이용하여 명령어의 비트-레벨 기계어를 약 99.4%의 정확도로 복구하기도 하였다.

이와 같은 명령어 역어셈블러에 관한 연구는 IoT 장비의 SW에 대한 지적 재산권 침해와 같은 부정적 요소도 존재하지만 안티 바이러스(anti-virus) 솔루션을 설치할 수 없는 환경인 경량 디바이스에서 악성 행위나 프로그램의 수정 등과 같은 공격을 탐지하는데 활용될 수 있다. 또한, 인증 메커니즘, 암호 기법 등이 공개되지 않은 정보보호 디바이스에 대한 세부 동작 코드나 운영 과정을 파악하여 시스템의 취약점을 분석하는 데 유용하게 사용될 수도 있다.

본 논문에서는 현재 부채널 연구 및 실험에 많이 사용되고 있는 NewAE Technology사의 ChipWhisperer® 플랫폼[9]과 ATmel AVR XMEGA128-D4 MCU가 탑재된 타겟 보드를 대상으로 사용된 명령-코드를 복구하는 소비 전력 파형에 기반한 역어셈블러를 구현하고자 한다. 실험에서는 웨이블릿 변환을 이용해 명령어 동작 시 발생하는 소비 전력 파형을 이미지로 변환한 후 합성곱 신경망(Convolutional Neural Network, CNN)에 기반하여 명령-코드를 분류하는 전 과정을 자동화하였다. 또한, 구현된 역어셈블러가 실제 디바이스 구동 환경에 어느 정도 적합한지를 검증하기 위해 테스트 데이터를 3가지 형태로 정의하여 검증을 진행하였다.

II. 배경 지식

2.1 이산 웨이블릿 변환

이산 웨이블릿 변환(Discrete Wavelet Transform, DWT)은 n개의 샘플을 갖는 이산 신호에 적용하며 특정 웨이블릿 모-함수(mother-function)를 필터로 이용해 신호의 저주파 대역과 고주파 대역을 분리한다. 분리된 저주파 대역 신호와 고주파 대역 신호는 각각 $n/2$개의 샘플로 분해된다.

이렇게 분해된 신호 중 고주파 대역의 신호는 잡음으로 간주할 수 있으며 저주파 대역의 신호에 대해서는 반복적으로 이산 웨이블릿 변환을 적용하여 다운샘플링(downsampling)을 수행할 수 있다. 반복적으로 이산 웨이블릿 변환을 적용하는 것을 레벨(level)로 표기하며 레벨 n까지 적용할 경우 샘플 수는 $1/2^n$까지 줄어든다. 이러한 특성 때문에 이산 웨이블릿 변환 기법은 이미지나 오디오 압축 등에 많이 사용된다.

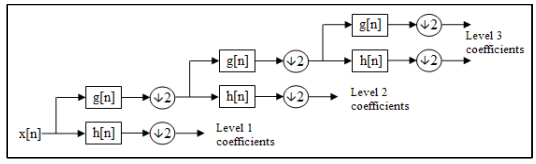

Fig. 1. An example of 3-level DWT.

2.2 연속 웨이블릿 변환

연속 웨이블릿 변환은 입력된 신호와 모-웨이블릿(mother-wavelet)과 합성곱 연산을 진행하는데 이때 모-웨이블릿의 스케일을 변화시키며 반복한다. 모-웨이블릿의 스케일 변경에 따른 여러 개의 합성곱 결과 벡터를 합쳐 시간-주파수를 축으로 하는 2차원 데이터로 변환하며 이때 웨이블릿의 스케일 변화에 따른 축이 주파수 축이 된다. 이렇게 생성된 시간-주파수 영역의 데이터는 Fig. 2와 같이 연속적인 형태로 표현할 수 있다.

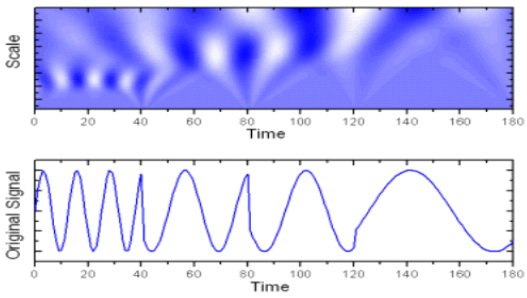

Fig. 2. An example of CWT.

2.3 합성곱 신경망

CNN은 Y. LeCun에 의해 1989년 처음 소개된 신경망을 이용한 딥러닝 구조 중 하나로서 1998년 제안한 LeNet 모델[10]을 단순화하여 현재의 일반 모델로 사용하고 있다. CNN은 완전 연결 계층(fully connected layer)으로 구성된 MLP(Multi Layer Perceptron)[11]의 앞단에 이미지의 특징을 추출할 수 있는 합성곱 계층이 추가된 구조이다.

기존 MLP를 이용해 이미지 분류 모델을 구성하는 경우, 입력된 이미지의 각 픽셀이 1차원으로 평탄화(flatten)되어 학습에 사용된다. 따라서 이미지의 형상이 반영되지 않아 인근 픽셀과의 관계까지는 학습하지 못하게 된다. 그러나 CNN은 2차원 이미지 데이터를 그대로 모델의 입력으로 사용하며 입력된 이미지 데이터는 합성곱 계층에서 커널(kernel) 또는 특징 맵(feature map)이라 불리는 2차원 형상의 필터와 합성곱 연산을 수행한다. 경우에 따라서는 풀링(pooling) 계층을 이용해 주요 특징만 추출함으로써 데이터 크기를 축소하여 사용하기도 한다. 즉, CNN은 합성곱 계층을 거친 2차원 이미지 데이터들을 1차원 데이터로 평탄화하여 MLP로 학습하고 분류하는 구조이며 Fig. 3과 같이 표현할 수 있다.

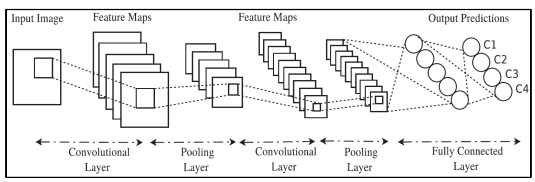

Fig. 3. The typical structure of CNN.

III. 명령어 구성 및 파형 수집

3.1 역어셈블러 대상 장비

ChipWhisperer-Lite®(이하 CW-lite)는 NewAE사에서 제작한 전력 분석 공격 등의 부채널 공격 실험을 위한 하드웨어 및 소프트웨어 플랫폼이며 구체적인 모델명은 CW-1173이다. CW-lite는 최대 24,400개 까지의 샘플을 7.3846MS/s 또는 29.538MS/s의 표본화율로 측정할 수 있으며 이는 각각 클럭당 1개 혹은 4개의 샘플을 측정할 수 있음을 의미한다. 논문에 사용된 모든 소비 전력 파형은 29.538MS/s 표본화율로 측정하였으며 CW-lite와 타겟 보드인 CW-303의 구성은 Fig. 4와 같다.

본 실험 대상 타겟 보드인 ChipWhisperer 시리즈의 CW-303에는 ATmel AVR XMEGA128-D4가 탑재되어 있으며 7.37MHz 외부 클럭으로 동작한다. 이 MCU에서는 137개의 명령어 집합(instruction set)을 제공하고 있으며 하버드 구조(Harvard architecture)를 채택함으로써 프로그램 메모리와 실행 메모리가 분리되어 있다. 또한, 32개의 범용 레지스터(R0~R31)를 제공하며, 이 레지스터들은 모두 ALU(Arithmetic Logic Unit)와 직접 연결되어 있다. 이 외에도 AVR XMEGA128-D4 MCU는 모든 클럭 사이클마다 명령어가 수행될 수 있도록 단일-레벨 명령어 파이프라이닝(single-level pipelining)을 적용하고 있다.

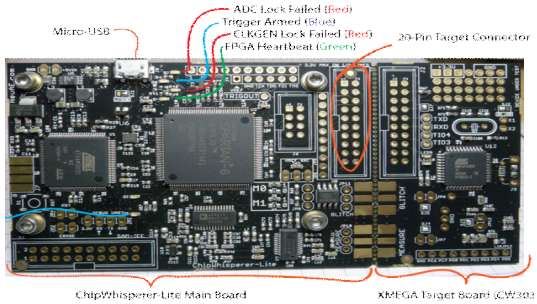

Fig. 4. ChipWhisperer-Lite and CW303.

3.2 역어셈블러 구현 및 자동화 절차

역어셈블러 구현을 위해서는 먼저 자체 제작한 명령어 데이터베이스를 바탕으로 C언어 기반 인라인

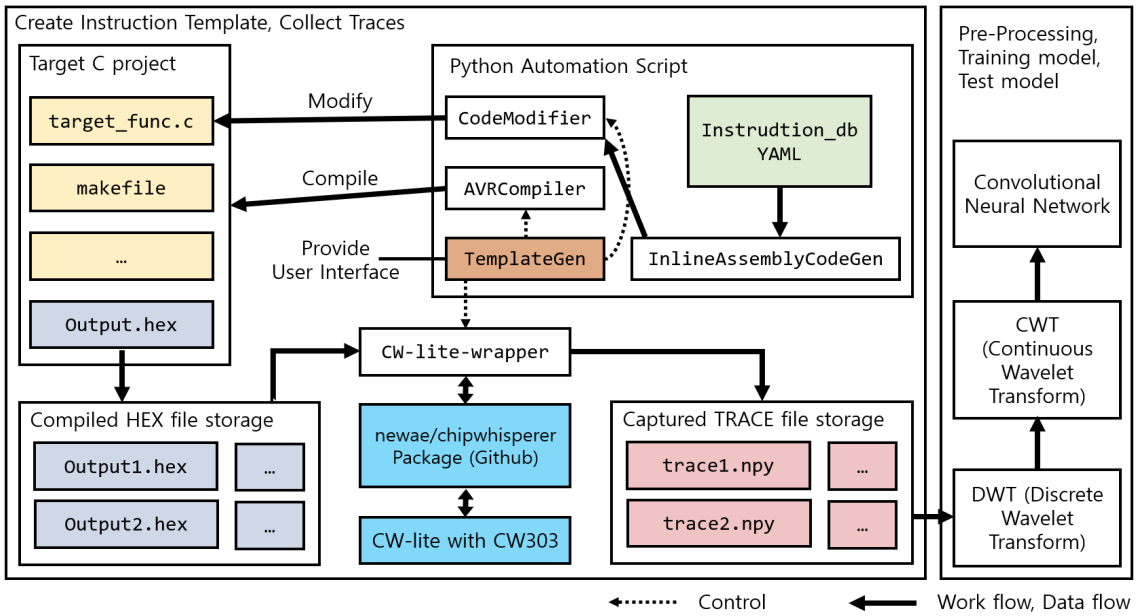

Fig. 5. Automation process of CNN-based disassembler

(inline) 어셈블리 코드를 생성한 후 컴파일 과정을 거쳐 실행 파일(.hex)을 생성한다. 그리고 컴파일된 파일을 타겟 보드에 적재한 후 CW-lite를 통해 PoI(Point Of Interest)에 해당하는 소비 전력 파형만 부분적으로 수집한다. 그리고 수집한 파형을 이산 웨이블릿 변환, 연속 웨이블릿 변환을 이용해 전처리 후 CNN 모델을 학습하고 테스트한다. 논문에서는 명령어 구성에서부터 분류까지 역어셈블러 구현 과정을 모두 파이썬 스크립트로 자동화하였으며 세부 실험 절차 및 자동화 프로세스를 나타낸 것이 Fig. 5이다.

3.3 소비 전력 파형 측정 범위

XMEGA128-D4는 단일-레벨 명령어 파이프라이닝이 적용되어 하나의 명령어가 실행될 때, 다음 실행될 명령어가 선인출(pre-fetch)된다. 즉, 하나의 명령어에 해당하는 소비 전력 파형을 측정할 경우, 해당 명령어가 실행될 때의 소비 전력 정보뿐만 아니라 다음 명령어가 선인출될 때 소모하는 전력 정보도 포함된다. 따라서 파형을 측정하는 구간은 측정 대상 이전 명령어의 실행 구간(대상 명령어의 선인출 구간)과 대상 명령어의 실행 구간(다음 명령어의 선인출 구간)이 된다.

다음 Fig. 6에서 보는 바와 같이 Target 명령어는 이전 명령어인 Inst. 1이 실행될 때 선인출되며, Target 명령어가 실행될 때는 Inst. 3이 선인출된다. 이때, Target 명령어에 접해 있는 Inst. 1과 Inst. 3은 모든 경우의 명령어가 위치할 수 있다. 그러나 모든 명령어 구성을 학습하는 것은 과중한 학습 과정이 있어야 하기 때문에 측정 대상 명령어 Target에 대하여 Inst. 1, Inst. 3을 명령어 집합 중 임의로 선택하여 구성하되 적정량의 명령어를 학습하는 것이 유리하다. 이때 Inst. 1, Inst. 3은 물론 Target의 피연산자(operand)도 임의로 선택하여 사용한다. 그리고 Target 명령어가 선인출되는 시점부터 실제 동작할 때까지의 C1~C2 구간 파형을 측정하게 된다.

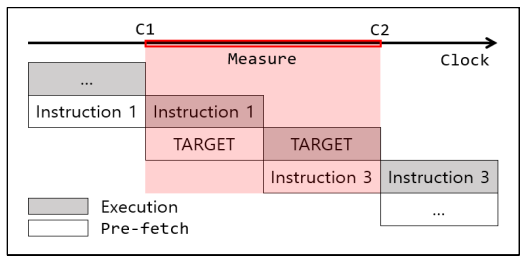

Fig. 6. Instruction sequence and measurement range for instruction template.

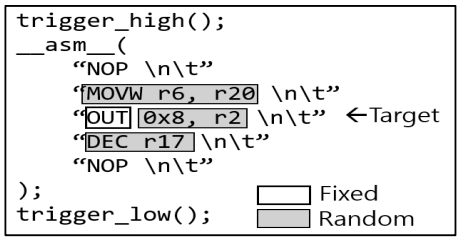

Fig. 7. Inline assembly code of C language for template construction.

결과적으로 명령-코드에 대한 템플릿을 제작하기 위해 Fig. 7과 같이 타겟 보드의 명령-코드를 제외한 Inst. 1~Inst. 3 구간의 모든 명령-코드와 피연산자를 임의로 선택해 템플릿을 구성한다. 또한, 트리거 설정 및 해제와 같은 다른 명령어의 간섭을 최소화하기 위해 Inst. 1의 이전, Inst. 3의 이후 명령어는 NOP으로 설정하였다.

3.4 대상 명령어 선택 및 임의 코드 생성

C언어 인라인 어셈블리 코드 문법에 부합하는 임의의 명령-코드와 피연산자를 생성하기 위해 본 실험에서 명령어 데이터베이스를 자체 구성하고 제작하였다. XMEGA128-D4에는 137개의 명령어 집합이 있지만, 파형 측정상의 어려움으로 인해 분기 명령어 등을 제외한 77개의 명령-코드에 대한 템플릿을 구성하였으며 이는 Table 1에 나타낸 바와 같다.

MCU 명령어 매뉴얼을 기준으로 77개의 명령-코드를 선정하였지만, 이 중 피연산자는 다르고 동일한

Table 1. Instructions used to build YAML databases and training CNN model.

Opcode	Operands	Clocks	Opcode	Operands	Clocks	Opcode	Operands	Clocks
ADD	Rd, Rr	1	MOVW	Rd, Rr	1	LSL	Rd	1
ADC	Rd, Rr	1	LDI	Rd, K	1	LSR	Rd	1
ADIW	Rd, K	2	LDS	Rd, k	2	ROL	Rd	1
SUB	Rd, Rr	1	LD	Rd, X	1	ROR	Rd	1
SUBI	Rd, K	1	LD	Rd, X+	1	ASR	Rd	1
SBC	Rd, Rr	1	LD	Rd, -X	2	SWAP	Rd	1
SBCI	Rd, K	1	LD	Rd, Y	1	SEC	None	1
SBIW	Rd, K	2	LD	Rd, Y+	1	CLC	None	1
AND	Rd, Rr	1	LD	Rd, -Y	2	SEN	None	1
ANDI	Rd, K	1	LDD	Rd, Y+q	2	CLN	None	1
OR	Rd, Rr	1	LD	Rd, Z	1	SEZ	None	1
ORI	Rd, K	1	LD	Rd, Z+	1	CLZ	None	1
EOR	Rd, Rr	1	LD	Rd, -Z	2	SEI	None	1
COM	Rd	1	LDD	Rd, Z+q	2	SES	None	1
NEG	Rd	1	STS	k, Rr	2	CLS	None	1
SBR	Rd, K	1	ST	X, Rr	1	SEV	None	1
CBR	Rd, K	1	ST	X+, Rr	1	CLV	None	1
INC	Rd	1	ST	-X, Rr	2	SET	None	1
DEC	Rd	1	ST	Y, Rr	1	CLT	None	1
TST	Rd	1	ST	Y+, Rr	1	SEH	None	1
CLR	Rd	1	ST	-Y, Rr	2	CLH	None	1
SER	Rd	1	STD	Y+q, Rr	2	IN	Rd, A	1
CP	Rd, Rr	1	ST	Z, Rr	1	OUT	A, Rr	1
CPC	Rd, Rr	1	ST	Z+, Rr	1	PUSH	Rr	1
CPI	Rd, K	1	ST	-Z, Rr	2	POP	Rd	2
MOV	Rd, Rr	1	STD	Z+q, Rr	2			

명령-코드를 사용하는 경우도 있어 명령-코드를 통일하는 등의 작업을 거쳐 템플릿으로 통합 및 조정하여 자체 제작 데이터베이스는 61개의 항목으로 구성하였다. 다음 Fig. 8은 ADIW와 STD 명령-코드에 대한 데이터베이스 항목을 예시한 것이다.

논문에서는 실험 대상으로 선택한 명령-코드들의 실제 사용 빈도를 확인하기 위해 블록 암호 알고리즘인 AES, LEA, SEED 등의 동작 코드를 컴파일하여 명령어 빈도를 분석해 본 결과 평균 5% 이하의 명령어만이 템플릿 데이터베이스에 포함되지 않았음을 확인했다.

```
-
  opcode          : ADIW
  opcode_id       : 11
  clock           : 2
  n_oprnd         : 2
  oprnd_1_type    : r
  oprnd_1_option  : {range: 24-30, even_num: true}
  oprnd_2_type    : h
  oprnd_2_option  : {range: 0-63}
  extra_info      : {decription: Add imimediate to Word}
-
  opcode          : STD
  opcode_id       : 41
  clock           : 2
  n_oprnd         : 2
  oprnd_1_type    : p
  oprnd_1_option  : {pointer: [Y, Z], displacement_range: 0-63}
  oprnd_2_type    : r
  oprnd_2_option  : {range: 0-31}
  extra_info      : {decription: Store indirect with displacement}
```

Fig. 8. Examples of ADIW and STS instruction in YAML format.

3.5 테스트 데이터의 구분

본 논문에서는 실제 역어셈블러 사용 환경을 고려하여 Fig. 9와 같이 테스트 데이터를 3종류로 구분하였다. 먼저 타입 1은 학습 데이터와 함께 측정된 것으로서 측정 파형을 학습 데이터와 일정 비율로 분리한 모델 검증용 테스트 데이터이다.

한편, 디바이스에서는 같은 명령어 구성으로 이루어진 동일한 실행 파일(.hex)에서 측정한 파형일지라도 측정할 때마다 잡음 요인, 공변량(covariate)

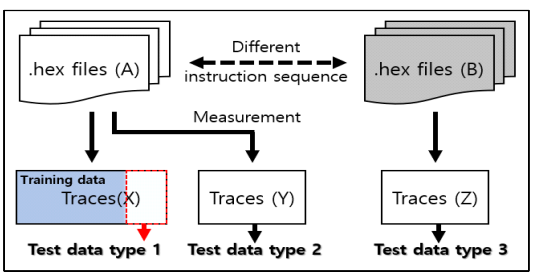

Fig. 9. Category of test data by executable file(.hex) and measurement group.

변화 등의 문제로 인해 파형의 분포가 미세하게 변화한다. 여기서 공변량이란 동일한 명령어를 다른 디바이스나 동일 디바이스의 다른 시간대에 측정할 경우 환경 변이에 따른 잡음 인자를 의미한다. 따라서 공변량 변화에 따른 분류 성능을 확인하기 위해 학습 데이터와 동일한 실행 파일을 독립된 환경에서 측정한 파형을 타입 2로 정의하였다.

마지막으로 CNN 분류 모델이 학습되지 않은 새로운 구성의 테스트 명령어에 대한 분류 성능을 확인할 필요가 있다. 따라서 임의로 구성된 실행 파일을 수행한 후 발생한 소비 전력 파형을 타입 3으로 정의하였다. 본 실험에서 학습용 소비 전력 파형과 3종류의 테스트용 파형을 Table 2와 같이 구성하고 측정하였다.

Table 2. The number of instruction sequence and power traces according to type of test data.

	Instruction sequence per opcode	Number of traces
Train data	30~40	163,660
Test data type 1		70,140
Test data type 2		21,880
Test data type 3	8~10	2,950

IV. 파형 전처리 및 명령어 역어셈블러 구현

4.1 이산 웨이블릿 변환을 이용한 차원 축소

연속 웨이블릿 변환을 이용해 시간-주파수 영역의 2차원 데이터로 변환하기 위해서는 파형의 크기가 일정해야 하는데 템플릿을 구성하고자 하는 명령어 대부분은 1클럭으로 동작하지만, 일부 명령어는 2클럭으로 동작한다. 즉, CW-lite를 이용해 29.538MS/s 표본화율로 측정할 경우, 1클럭당 4개의 샘플이 측정되므로 1클럭 명령어는 4개의 샘플, 2클럭 명령어는 8개의 샘플이 측정된다. 따라서 이를 일정하게 맞추기 위해 본 논문에서는 Fig. 10과 같이 이산 웨이블릿 변환을 사용하여 8샘플로 구성된 2클럭 명령어를 4샘플로 변환하도록 하였다.

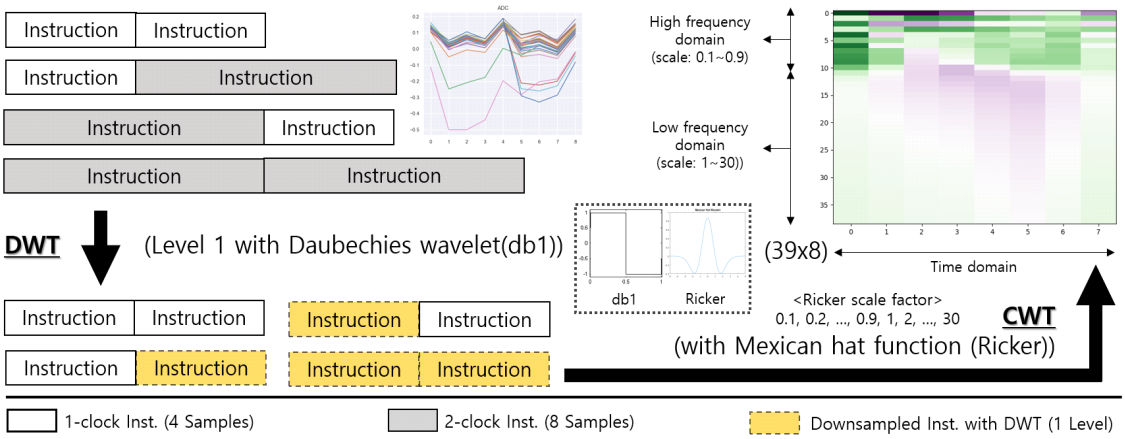

Fig. 10. Two-dimensional image conversion process of traces using DWT and CWT.

4.2 연속 웨이블릿을 이용한 이미지 변환

이산 웨이블릿 변환을 거쳐 한 명령어당 4샘플로 일원화된 소비 전력 파형들은 연속 웨이블릿 변환을 거쳐 시간-주파수 영역의 2차원 데이터로 변환된다. 이때 입력 파형과 합성곱 연산이 진행되는 웨이블릿은 Mexican hat function으로 알려진 Ricker 웨이블릿을 사용하였다. 다음 Fig. 10에서 보는 바와 같이 Ricker 웨이블릿의 길이 관련 파라미터인 스케일(scale)로는 0.1, 0.2, …, 0.9, 1, 2, …, 29, 30까지 총 39개를 사용하였으므로 주파수 축의 길이도 39가 된다.

4.3 CNN 모델 구성

본 실험에서 사용한 CNN의 구조는 Table 3과 같이 풀링 계층 없이 합성곱 4계층을 거쳐 특징을 추출하며, 추출된 특징들은 500개의 노드로 구성된 1개의 은닉 계층을 가진 완전 연결 계층에서 학습된다. 모델에서는 과적합(overfitting)을 방지하기 위해 드롭아웃(dropout)계층을 사용하는데 이를 통해 학습 과정에서 특정 비율의 노드를 비활성화시켜 일부 가중치만 갱신하게 된다. 세 개의 명령어를 조합할 수 있는 경우의 수는 매우 많아 모두 학습할 수 없고, 파형을 측정할 때마다 신호 분포가 미세하게 변화하기 때문에 드롭아웃과 같은 과적합 방지 대책은 테스트 파형에 대한 분류 정확도를 높이게 된다.

Table 3. Layer component of CNN model.

Layer	Shape/Kernel/ Node/Dropout ratio	Activation
(Input)	39x8x1	N/A
Conv2D	32 (7x4)	ReLU
Conv2D	32 (5x3)	ReLU
Conv2D	32 (4x2)	ReLU
Conv2D	32 (2x2)	ReLU
Dropout	0.25	N/A
(Input)	9,984	N/A
Dense	500	ReLU
Dropout	0.25	N/A
Dense	61	Softmax

4.4 명령-코드 분류 모델 학습 결과

본 논문에서 구현 및 학습시킨 CNN 모델을 타입 1의 테스트 데이터로 검증한 결과 87.5%, 타입 2는 42.1%, 타입 3은 18.31%의 정확도로 명령-코드를 분류할 수 있었다. Fig. 11의 (A)~(C)는 테스트 데이터 타입 1~3을 이용한 분류 결과의 혼동 행렬(confusion matrix)이다. 그림에서 혼동 행렬은 실제 데이터 클래스와 모델의 예측 클래스의 일치성을 보여주는데 X축은 모델이 예측한 클래스이며 Y축은 실제 데이터의 클래스를 의미한다. 그림에서 (D)~(F)는 실험한 CNN 모델에서 각 명령어 클래스별 분류 정확도를 나타낸 것이다.

먼저, 테스트 데이터 타입 1의 경우, 학습 파형과 함께 측정되어 가장 유사한 분포를 가지기에 87.5% 정도의 높은 정확도로 명령-코드를 분류할 수 있었다. 특히, 명령어 구성 비율이 높은 데이터 메모리 읽기/쓰기 명령어인 ST, STD, STS, LD, LDD,

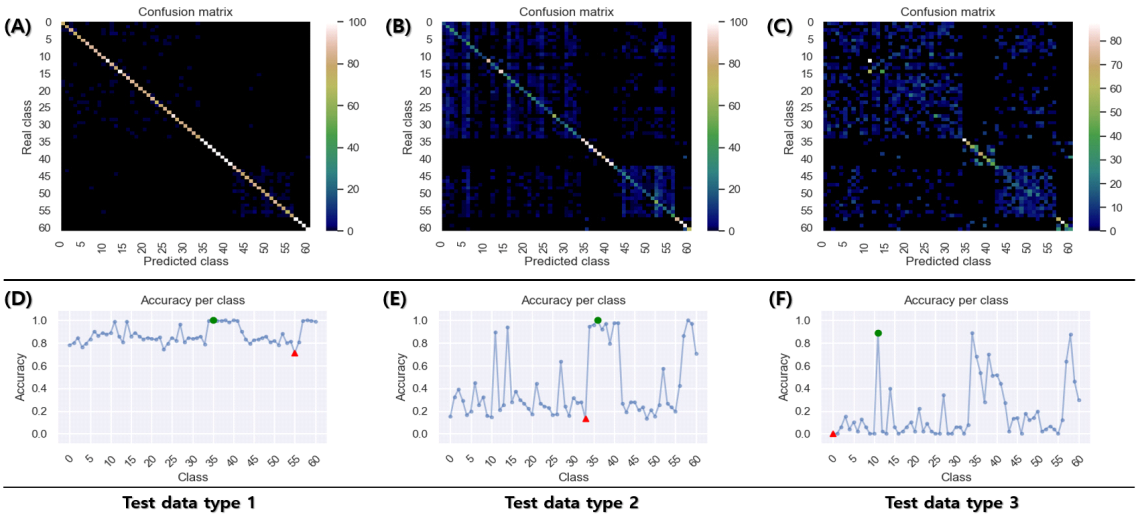

Fig. 11. Confusion matrix and accuracy for opcode classifier according to test data type.

LDS와 스택 및 I/O 관련 명령어인 PUSH, POP, IN, OUT 그리고 2클럭 명령어인 ADIW, SBIW에 대해서는 99.6% 정도의 정확도로 분류할 수 있었다. 해당 명령-코드들에 대한 분류 정확도가 비교적 높은 이유는, 이들의 소비 전력 파형이 다른 명령-코드의 소비 전력 파형과 뚜렷하게 구분될 수 있을 정도로 다른 분포를 갖기 때문이라 분석된다.

반면 테스트 데이터 타입 2의 경우, 파형의 분포가 미세하게 변화해 42.1%의 다소 낮은 정확도로 명령-코드를 분류할 수 있었다. 그림 Fig. 11의 (B)에서 보면, 클래스 35~40을 기준 축으로 제 2사분면, 제4사분면이 제 1사분면, 제 3사분면보다 높은 구성 비율을 차지하고 있는 것으로 보아, 비슷한 피연산자를 가진 명령어 내에서는 모델이 원활한 분류를 하지 못하는 것을 알 수 있었다.

마지막으로 테스트 데이터 타입 3의 경우 18.3%의 낮은 분류 정확도를 보이지만 (F)와 같이 데이터 메모리, 스택, I/O 관련 명령어의 정확도는 다소 높은 것을 확인할 수 있다. 또한, (C)에서 보는 바와 같이 올바르게 예측하지 못한 부분들이 (B)에 비해 전체적으로 높게 구성된 것으로 보아 타입 2보다 전반적인 명령-코드를 정확히 분류하지 못함을 알 수 있다. 타입 3의 정확도는 모델 학습시 각 명령-코드별 더 많은 명령어 구성을 학습함으로써 일부 향상시킬 수 있을 것이다. 테스트 데이터 타입에 따른 CNN 모델의 전체 명령어 및 주요 명령어 분류 정확도를 정리하면 Table 4와 같다.

Table 4. Accuracy of opcode classification model according to test data type and instruction group.

	All instruction	Data Memory, Stack, IO, 2-clock instruction
Test data type 1	87.5%	99.6%
Test data type 2	42.1%	92.6%
Test data type 3	18.3%	57.2%

4.5 선행 연구와의 비교 분석

유사한 선행 연구에 해당하는 배 등의 논문[12]에서는 파형의 차원 축소를 위해 이산 웨이블릿 변환이 아닌 주성분 분석을 사용하고 전체 파형에 대해서도 다시 주성분 분석을 통해 좌표를 재표현하여 1차원 파형 데이터를 MLP로 학습 및 분류하였다. 이 MLP 모델에서 타입 2와 3에 해당하는 테스트 데이터로 검증한 결과, 2% 이하의 낮은 정확도를 보여 대부분의 명령-코드 및 레지스터를 분류하지는 못하였다.

그러나 본 논문에서는 문헌 [12]와 달리 파형에 대한 전처리를 방법을 주성분 분석에서 이산 웨이블릿 변환 및 연속 웨이블릿 변환으로 변경하였고 분류 모델을 MLP에서 CNN으로 대체함으로써 타입 2,

3에 대한 모델의 정확도를 향상시켰다. 이는 전처리 과정을 통해 파형의 공변량 변화 문제를 개선하였으며 딥러닝에서의 드롭아웃 기능을 통해 모델의 과적합 문제를 해결한 결과로 분석된다.

V. 결 론

본 논문에서는 특정 마이크로 컨트롤러가 사용하는 명령어도 전력 분석을 이용한 부채널 공격에 의해 충분히 복구될 수 있음을 보이고, 이를 검증할 수 있는 자동화된 역어셈블러를 구현하였다. 구현된 역어셈블러는 명령-코드 분류를 위해 CNN 기반의 딥러닝 기법을 사용하였다. 특히, 각 명령어별로 측정된 소비 전력 파형들은 잡음 제거 및 공변량 감소를 위해 두 종류의 웨이블릿 변환을 이용해 파형을 전처리하였다.

구현한 역어셈블러는 모든 명령어에 대해서 약 87.5%의 정확도로 명령-코드를 분류할 수 있었으며, 사용 빈도가 높은 주요 명령어에 대해서는 99.6%의 정확도로 분류할 수 있음을 확인하였다. 결론적으로 본 논문에서 구현된 역어셈블러는 IoT 디바이스내의 악성 코드 분석, 프로그램 수정 탐지 그리고 비공개 암호 메커니즘 분석 등에 효과적으로 사용할 수 있다.

References

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” CRYPTO’99, LNCS 1666, pp. 388-397, 1999.

[2] J. Park, X. Xu, Y. Jin and D. Forte, “Power-based side-channel instruction-level disassembler,” Proceedings of the 55th Annual Design Automation Conference(DAC), pp. 1-6, 2018.

[3] S. Wold, K. Esbensen and P. Geladi, “Principal component analysis,” Chemometrics and intelligent laboratory systems, Vol. 2, No. 1-3, pp. 37-52, 1987.

[4] S. Kullback and R. Leibler, “On Information and Sufficiency,” The Annals of Mathematical Statistics, Vol. 22, No. 1, pp. 79-86, 1951.

[5] T. Alaa, "Linear vs. quadratic discriminant analysis classifier: a tutorial," International Journal of Applied Pattern Recognition. Vol. 3 No. 2, pp. 145-180, 2016.

[6] J. Pritchard, M. Stephens and P. Donnelly, "Inference of population structure using multilocus genotype data," Genetics. Vol. 155, No. 2, pp. 945 - 959, 2000.

[7] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, Vol. 20, No. 3, pp. 273 - 297, 1995.

[8] V. Cristiani, M. Lecomte and T. Hiscock, "A Bit-Level Approach to Side Channel Based Disassembling," Smart Card Research and Advanced Applications(CARDIS'19), pp. 143-158, 2019.

[9] ChipWhisperer® - NewAE Technology Inc., "chipwhisperer," Available at http://newae.com/tools/chipwhisperer/, 2017.

[10] Y. LeCun, B. Léon, B. Yoshua and H. Patrick, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, Vol. 86, No. 11, pp. 2278 - 2324, 1998.

[11] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," Psychological Review, Vol. 65, No. 6, 1958.

[12] D. Bae and J. Ha, "Implementation of Instruction-Level Disassembler Based on Power of the Microcontroller," Conference on Information Security and Cryptography - Summer 2020 (CISC-S20), Vol. 30, No. 1, pp. 649-653, 2020.

〈저자소개〉

배 대 현 (Daehyeon Bae) 학생회원
2017년 3월: 호서대학교 컴퓨터정보공학부 입학
2017년 3월~현재: 호서대학교 컴퓨터정보공학부 학부과정
〈관심분야〉 암호학, 부채널 공격, 인공지능 보안

하 재 철 (Jaecheol Ha) 종신회원
1989년 2월: 경북대학교 전자공학과 학사
1993년 8월: 경북대학교 전자공학과 석사
1998년 2월: 경북대학교 전자공학과 박사
1998년 3월~2007년 2월: 나사렛대학교 정보통신학과 교수
2007년 3월~현재: 호서대학교 컴퓨터정보공학부 교수
2013년 1월~현재: 한국정보보호학회 상임부회장
2009년 1월~현재: 한국산학기술학회 이사
〈관심분야〉 정보보호, 네트워크 보안, 부채널 공격