부채널 분석 기술 개발 보고서

참 여 자

한성대학교 IT융합공학부 엄시우

한성대학교 IT융합공학부 심민주

한성대학교 IT융합공학부 송경주

문 제 1

AES-STM, Unprotected AES(fixed key)

최소의 파형수로 마스터키 16바이트를 찾을 수 있는 최적의 부채널 분석기술 개발

Type & number of traces provide

- Fixed key and 2,000 traces

Acquisition Board for Data set

- SCARF STM Board (32bit MCU STM32F411RET)

Data set information

- {identifier}-info.txt : information text file
- {identifier}.btr : trace binary file(see below for structure of the binary file)
- {identifier}-plain.txt : plaintext expressed in hexadecimal
- {identifier}-cipher.txt : cipher expressed in hexadecimal
- {identifier}-key.txt : key expressed in hexadecimal

Example code for processing data set

- pub_data_handler.py : data handlers for btr file format and hex text file
- handler_example.py : Reading btr file or showing using
- pub_data_handler.py

서론

부채널 공격(side channel attack)은 암호 체계의 물리적인 구현 과정의 정보를 기반으로 하는 공격 기법이다. 알고리즘의 약점을 찾거나 무차별 공격을 하는 것이 아닌 암호 알고리즘을 대상으로 하는 물리적 공격 기법을 말한다. 알려진 물리적 부채널 공격으로는 타이핑 공격(Timing Attack), 전력 분석 공격(Power Analysis Attack), 전자기 분석 공격(EM Attack) 등이 있다. 이번 보고서를 통해 주어진 Data Sets을 이용하여 최소의 파형수로 마스터키 16바이트를 찾을 수 있는 최적의 부채널 분석기술 개발 결과를 설명한다.

분석 방법

Hamming Distance

전력 소비에 대한 가장 간단한 모델 중 하나로, 두 이진수 사이의 서로 다른 비트 수를 의미한다. 예를 들면 1011과 0110이 있다고 한다면 두 이진수의 서로 다른 비트 수는 3개 이므로 Hamming Distance = 3 이 되게 된다. 쉽게 구하는 방법은 두 이진수를 XOR 연산을 하였을 때의 1의 개수가 Hamming Distance가 된다.

Pearson 상관 계수

Hamming Distance를 이용하여 전력 소비 추정치를 구했다면, 실제 측정된 트레이스와 전력 추정치를 비교할 방법이 필요하다. 이때 Pearson 상관 계수를 사용한다. Pearson 상관 계수는 두 변수 X와 Y간의 선형 상관 관계를 계량화한 수치이다. 추정치와 실제 측정된 트레이스와의 상관 관계를 수치화하여 키를 추측한다.

데이터셋 분석

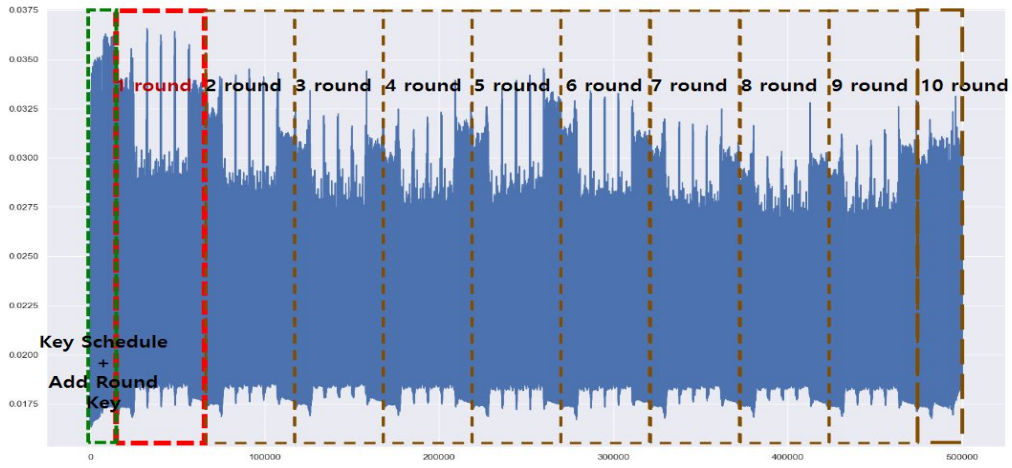
데이터셋을 그래프로 표현하여 확인하면 데이터의 구조를 쉽게 파악할 수 있다. 주어진 데이터셋을 그래프로 확인하기 위하여 예제 코드중 handler_example.py 파일의 show_mean_trace_btr 함수를 이용하였다. show_mean_trace_btr 함수는 Binary traces(.btr) 파일로 이루어진 파형의 평균을 시각화 하는 함수이기 때문에 주어진 btr 파일의 데이터셋을 그래프로 확인할 수 있었다.

```
from pub_data_handler import show_mean_trace_btr

#파일을 읽기 전에 전체 구조를 파악하여 부분적으로 읽어들일 수 있도록 평균 파형을 보여주는 함수
show_mean_trace_btr("../data/STM-AES.btr",  # 읽어들일 btr 파일
                    None) # 읽어올 파형의 샘플 인덱스. 전체를 읽을경우 None, 혹은 생략 가능.
```

[그림 1] handler_example.py 의 show_mean_trace_btr 함수

show_mean_trace_btr 함수를 통해 그려진 그래프를 AES-128 과정에 대입 하였다. 이후 아래와 같이 AES-128 각각의 단계에 해당하는 부분을 예측하여 그래프를 나눠 분석 하였다.



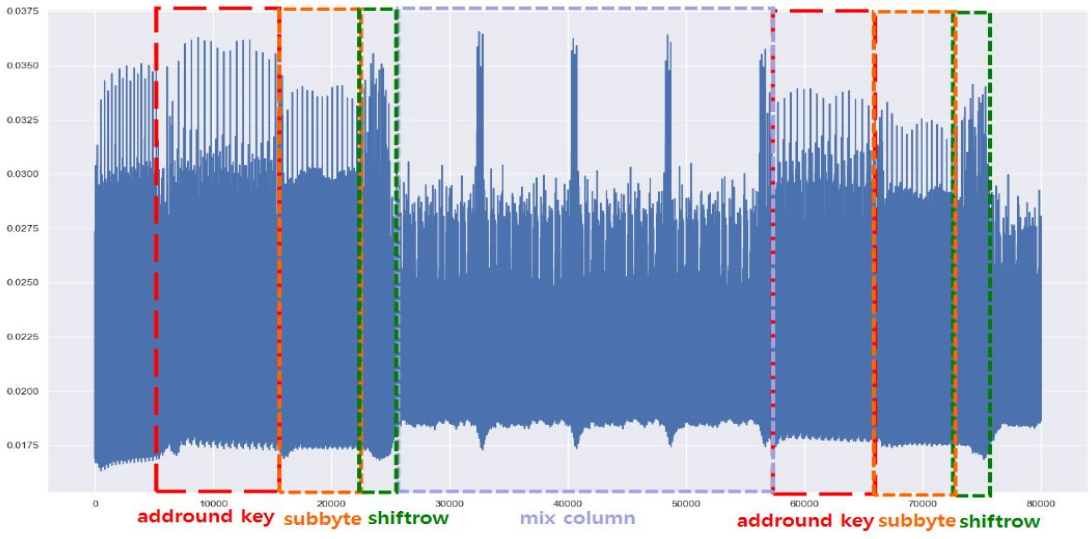
[그림 2] 전체데이터 그래프

[전체데이터 그래프 분석 결과]

1. Mix column은 10라운드를 제외한 나머지 9개의 라운드에만 존재
2. 10 라운드는 해당 그래프 마지막 부분에 푹 꺼진 이외의 부분이 Subbyte + Shiftrow 예측
3. 그래프의 맨 앞을 Key Schedule과 Add Round key로 예측
4. 한 라운드에 4종류의 연산이 수행되므로 그래프를 정확히 4가지 구조로 나누기 위해 1라운드에 해당되는 0~8만 구간의 그래프를 추출해 분석 진행

[AES 동작원리]

1. AES의 Round는 총 10 Round로 구성되어 있으며, 한 Round 당 Subbyte - Shiftrow - Mix column - Add Round Key로 구성
2. 1 Round 전에 Key schedule과 Add Round Key 수행
3. 마지막 10 Round에서는Mix column 연산을 수행하지 않음



[그림 3] 80000 포인트까지의 그래프

한 라운드의 구성을 자세히 보기 위해서 1라운드까지의 구간으로 예측되는 80000 포인트까지를 [그림 3] 과 같이 그래프로 추출하여 분석을 진행하였다.

부채널 분석은 선형연산인 부분을 통해 중간값 유추가 가능하여 비선형 연산인 Mix column을 제외한 Add round Key - Subbyte - Shiftrow에 해당되는 구간을 15000~25000으로 예측하였다.

최종적으로 해당 구간에서 측정된 데이터와 계산된 추정치를 통해 키 예측이 가능할 것이라 생각하여 이에 대한 분석을 하였다.

분석 과정

```
# 16진수 문자열로이루어진 텍스트파일을 읽는 함수
plain = convert_hex_data_2_npy("D:₩₩AVR-SEED₩₩AVR-SEED-plain.txt",  # 16진수 문자열로 이루어진 txt 파일
                               None)  # 변환된 객체를 파일로 저장하려면 경로 및 파일명 입력. 생략 가능.

cipher = convert_hex_data_2_npy("D:₩₩AVR-SEED₩₩AVR-SEED-cipher.txt",  # 16진수 문자열로 이루어진 txt 파일
                                None)  # 변환된 객체를 파일로 저장하려면 경로 및 파일명 입력. 생략 가능.

keys = convert_hex_data_2_npy("D:₩₩AVR-SEED₩₩AVR-SEED-key.txt",  # 16진수 문자열로 이루어진 txt 파일
                              "D:₩₩key_backup.npy")  # 변환된 객체를 파일로 저장하려면 경로 및 파일명 입력. 생략 가능.
```

[그림 4] convert_hex_data_2_npy

예제 코드로 주어진 handler_example.py 파일의 convert_hex_data_2_npy 파일을 활용하여 데이터로 주어진 STM-AES.btr, STM-AES-plain.txt, STM-AES-cipher.txt, STM-AES-key.txt 를 npy 파일로 변환하여 저장하였다.

키 추측에 사용된 코드는 ChipWhisperer Wiki 사이트에 올라온 코드를 사용하여 분석을 진행하였다.

(코드 출처 : https://wiki.newae.com/V4:Tutorial_B6_Breaking_AES_(Manual_CPA_Attack)

```
Subkey 15, hyp = a4:
0.5959163812089185
Subkey 15, hyp = a5:
0.5960392051776819
Subkey 15, hyp = a6:
0.5309443087521735
Subkey 15, hyp = a7:
0.6448565186393206
Subkey 15, hyp = a8:
0.44883111378153984
Subkey 15, hyp = a9:
0.6649087474724927
Subkey 15, hyp = aa:
0.5588249050024928
Subkey 15, hyp = ab:
0.5124082094949217
Subkey 15, hyp = ac:
0.5974799834050265
Subkey 15, hyp = ad:
0.6032937283398736
Subkey 15, hyp = ae:
0.49004381077153064
Subkey 15, hyp = af:
0.6855177012712006
Subkey 15, hyp = b0:
0.5344439514174113
```

[그림 5] chipwhisperer cpa 예제 코드 실행

```
Best Key Guess:
39
4d
8f
af
9d
90
71
e0
59
f7
19
e3
e0
bc
45
63
```

[그림 6] chipwhisperer cpa 예제 코드 키 추측 결과

ChipWhisperer CPA 예제 코드의 구간을 조금씩 수정하여 추측키를 확인 하였고 추측키들을 정리하여 규칙을 찾는 과정을 수행하였다.

[그림 5] 는 ChipWhisperer CPA 예제 코드를 실행했을 때 모습이며 [그림 6] 는 ChipWhisperer CPA 예제 코드 키 추측 결과이다.

아래의 [표 1] 과 [표 2] 는 구간을 수정하며 규칙을 찾는 과정을 표로 시작은 2000으로 고정하고 100씩 증가시켜가며 추측 결과를 표로 정리한 것이다.

[표 1] 의 맨 왼쪽의 파란색 라인은 STM-AES-key.txt 에 나와있는 키 이며 예측 된 키를 이와 비교하여 규칙을 찾아보았다.

START	2000	2000	2000	2000	2000	2000	2000	2000	2000
END	2100	2200	2300	2400	2500	2600	2700	2800	2900
f2	c4	c3	c3	c3	e3	e3	e3	e3	e3
6d	cd	c0	c0	c0	c0	c0	c0	c0	c0
2b	61	61	e0	e0	e0	14	14	14	14
46	64	59	59	59	59	59	59	59	59
7d	3c	b2	b2	b2	b2	b2	b2	b2	b2
2d	c6	c6	c6	c7	c7	c7	c7	ba	ba
b2	a1	59	35	b9	4a	4a	4a	4a	4a
0e	76	76	44	44	0	0	0	0	0
a9	ae	ae	ae	ae	b8	b8	b8	b8	b8
d7	4f	59	59	59	59	59	59	59	59
2a	db	db	db	db	db	db	db	db	db
3c	c6	c6	c6	c6	c6	c6	c6	c6	c6
35	b1	f3	f3	c2	c2	c2	c2	c2	c2
e5	9c	9c	8	e2	e2	e2	e2	e2	e2
19	f6	3a	9b	c2	c2	c2	c6	c6	c6
65	ed	ed	ed	ed	ed	ed	db	db	db

[표 1] 임의 구간 추측 키1

구간을 조금씩 수정해가며 추측해 본 결과 3~400 단위로 키 값이 비슷하게 추측되는 것을 확인하였다.

START	3000	3000	3000	START	2000	3000	4000	5000	6000	7000	8000
END	3100	3200	3300	END	2300	3300	4300	5300	6300	7300	8300
f2	b9	cc	cc	f2	c3	cc	c3	48	c3	9	2
6d	0b	0	13	6d	c0	13	0	f6	43	d7	6f
2b	61	4b	4b	2b	e0	4b	b7	f1	f3	5d	42
46	8e	8e	8e	46	59	8e	55	6d	8e	3d	6e
7d	c4	c4	c4	7d	b2	c4	f9	0a	4	e8	6e
2d	3c	3c	3c	2d	c6	3c	2a	78	0d	6e	b5
b2	2	7	7	b2	35	7	2	2	bf	11	14
0e	f3	5f	f0	0e	44	f0	fc	3c	47	7b	cd
a9	7b	c5	c5	a9	ae	c5	bb	d0	77	ac	45
d7	30	ef	ef	d7	59	ef	23	ee	d5	d5	1b
2a	c2	60	60	2a	db	60	c2	6e	16	8	c2
3c	dd	b3	b3	3c	c6	b3	b3	69	36	c8	f7
35	b1	e2	e2	35	f3	e2	ce	9a	b1	f5	ff
e5	f4	f4	f4	e5	8	f4	2a	b7	2f	8	8
19	3f	c2	bf	19	9b	bf	d4	bd	2b	3	ec
65	45	a1	a1	65	ed	a1	5a	2e	d6	ef	ed

[표 2] 임의 구간 추측 키2

시작 부분(START)을 1000단위로 증가시켜가며 비교해본 결과, 다른 키 값이 추측하는 것을 알 수 있다. 100단위로 증가시켜가며 찾아보면 키 값을 찾을 수 있을 것으로 예상하였다. 이에 키의 첫번째 byte인 'F2'를 추측할 수 있는 구간을 알 수 있다면, 해당 구간에서 구간을 조금씩 수정해가면 키 전체를 찾을 수 있을 거라 예상하고 분석을 진행하였다.

앞서 그래프를 통해 15000 ~ 25000 구간을 분석하면 키를 유추할 수 있을 것으로 예측하여, 15000부터 25000까지 100씩 증가시키며 START 포인트와 END 포인트의 차이를 100으로 고정하고 진행하였다.

분석 결과

15000	15100	118	148	116	97	111	120	61	59	180	218	216	96	226	8	168	216
15100	15200	180	89	13	142	111	119	2	57	26	218	106	163	245	8	236	177
15200	15300	55	89	185	142	70	181	17	160	39	213	191	179	226	99	177	219
15300	15400	187	0	189	162	17	119	17	160	180	213	194	163	193	8	106	213
15400	15500	198	89	5	110	2	231	89	125	144	179	194	5	245	8	168	213
15500	15600	85	32	241	123	4	80	2	160	236	15	110	200	226	0	217	32
15600	15700	232	235	251	162	4	117	25	248	118	79	57	179	226	212	43	228
15700	15800	237	192	171	193	17	199	42	95	186	66	125	163	206	8	58	228
15800	15900	237	94	178	89	249	228	242	194	57	8	249	163	226	175	212	237
15900	16000	147	89	75	79	133	102	2	59	254	214	160	216	206	175	54	219
16000	16100	158	89	27	254	4	120	196	71	71	244	196	104	226	175	212	182
16100	16200	161	217	173	94	4	163	2	25	80	213	197	93	61	183	238	237
16200	16300	227	9	183	123	188	102	82	163	84	16	83	0	14	164	145	97
16300	16400	128	225	182	184	43	186	61	160	26	34	208	104	61	150	121	219
16400	16500	227	217	24	142	29	120	42	213	68	218	106	165	40	99	236	237
16500	16600	223	54	207	97	52	181	226	124	71	218	160	156	193	244	226	47
16600	16700	232	246	234	10	114	27	13	163	158	90	249	190	47	8	156	111
16700	16800	232	192	184	152	52	90	115	44	93	213	194	179	177	8	6	85
16800	16900	46	192	234	84	174	181	226	180	196	25	106	163	206	175	212	68
16900	17000	214	207	114	89	205	60	2	167	13	212	71	199	47	8	161	213
17000	17100	232	67	225	142	205	12	61	196	106	219	2	165	29	8	225	95
17100	17200	188	67	217	89	205	60	53	196	68	244	230	199	154	175	12	219
17200	17300	223	14	225	100	128	90	211	103	180	250	216	163	243	8	161	51
17300	17400	219	20	96	204	107	27	2	192	181	213	62	161	239	8	100	213
17400	17500	161	39	217	146	21	81	138	0	235	239	211	65	206	47	50	213
17500	17600	3	231	141	152	226	105	203	168	173	186	124	184	112	240	13	216
17600	17700	87	82	164	142	103	203	232	93	173	228	177	249	239	8	159	56
17700	17800	158	192	185	152	250	193	2	59	172	157	176	71	245	8	36	56
17800	17900	232	225	216	89	107	206	2	95	160	206	106	217	29	115	7	76
17900	18000	158	207	234	85	1	186	68	101	235	10	106	5	88	8	35	104
18000	18100	174	1	117	89	50	181	244	160	235	46	8	179	177	8	115	213
18100	18200	254	0	56	162	11	193	121	240	235	48	106	86	34	8	31	237
18200	18300	80	228	174	142	9	193	185	167	71	213	238	4	196	175	63	219
18300	18400	167	82	74	51	210	186	235	190	181	131	140	9	29	40	187	119
18400	18500	158	231	82	61	235	181	89	206	121	213	127	199	29	181	17	219
18500	18600	85	192	34	89	207	206	142	208	121	87	249	179	61	27	218	219
18600	18700	158	203	123	89	235	60	25	197	27	213	176	175	205	203	79	250
18700	18800	128	85	36	183	217	88	54	120	62	59	194	179	48	7	58	250
18800	18900	80	61	201	89	249	250	54	190	229	213	38	179	14	113	212	46
18900	19000	196	97	173	86	252	199	88	57	93	213	216	179	231	231	212	161
19000	19100	198	0	232	162	107	223	89	144	182	218	104	179	112	203	91	250
19100	19200	217	207	52	27	252	111	5	160	115	245	45	5	112	130	191	8
19200	19300	181	97	182	162	250	89	128	146	163	213	176	163	177	32	191	219
19300	19400	195	97	0	79	4	27	128	196	26	119	99	93	232	175	191	90
19400	19500	161	97	41	89	235	62	42	160	185	216	106	217	193	240	94	250
19500	19600	195	119	161	142	235	27	42	196	185	216	47	1	206	103	89	237
19600	19700	158	97	129	56	4	3	242	44	26	210	62	46	14	255	89	63
19700	19800	237	232	173	204	224	231	94	180	217	213	226	199	136	240	212	250
19800	19900	161	210	5	110	224	111	226	113	180	213	194	163	237	8	49	213
19900	20000	195	207	122	110	241	163	205	144	121	213	159	190	245	210	63	124
20000	20100	223	164	122	162	217	163	86	201	153	215	244	158	218	103	168	250
20100	20200	196	97	187	162	5	88	2	223	118	239	196	200	189	19	109	230
20200	20300	158	54	59	89	132	5	53	226	181	216	58	163	226	245	12	109
20300	20400	183	225	201	186	35	120	176	145	198	213	36	179	246	47	108	206
20400	20500	217	97	203	107	231	162	24	62	68	220	178	163	61	163	34	161
20500	20600	180	207	183	97	132	110	219	95	71	245	150	163	40	40	163	32
20600	20700	86	219	201	162	249	110	196	89	180	19	39	60	225	40	63	219
20700	20800	6	89	251	27	132	88	193	145	181	247	106	199	245	8	171	7
20800	20900	196	97	50	146	140	103	78	183	71	207	38	5	206	29	124	60

[표 3] 15000~25000 구간 추측 키 앞 부분

15000 ~ 25000 구간에서 100씩 증가시켜가며 키를 추측한 결과를 .csv 파일에 기록하였다. [표 3] 에서 색칠된 부분은 실제 키와 자리와 값이 같은 부분이다.

더 많은 구간을 저장하여 분석해 보았지만, [표 3] 에서 확인할 수 있듯이 실제 키와 같은 값을 갖는 구간을 찾기 어려웠다. 증가 폭이나 15000~25000구간이 아닌 다른 구간도 비슷한 방법을 통해 수집하였지만, [표 3] 과 비슷한 결과를 얻었다.

실패 원인 분석

1. 그래프 분석 실패로 인한 잘못된 구간 설정
2. 잘못된 AES 분석
3. 부적합한 코드 사용으로 인한 잘못된 키 예측
4. 문제에 대한 잘못된 접근

부채널 공격이 직접 알고리즘을 분석하는 것이 아닌 물리적 정보를 통해 분석해야 하기 때문에 직접 알고리즘 분석을 하는 것 이상의 지식이 필요하다고 느꼈다. 예측을 통한 분석이라 정확한 문제해결을 위한 솔루션을 찾는데 어려움이 있었다. 하지만 시스템 내부를 알지 않고도 물리적 정보를 가지고 공격할 수 있다는 점에서 굉장히 강력한 공격기술이라 생각되었다.