

딥러닝을 통한 S-PRESENT 알려진 평문 공격

임세진

<https://youtu.be/OEXAc2qVtxw>

Contents

01. 시행착오..

02. PRESENT / S-PRESENT

03. 딥러닝을 사용하여 알려진 평문 공격 수행

04. 모델별 암호 분석 (CNN, MLP)

05. 결론



01. 시행착오..

1) CHAM 암호 분석 시도

→ plaintext 64bit, key 128bit로 너무 커서 **1비트 키공간, 1비트 블록공간에 대해서 1라운드 조차 공격이 불가능**

2) 논문으로 존재하는 S-PRESENT 분석 시도 —————→

→ 블록 크기만 8bit로 줄임, key는 그대로 80bit 였음.. 공격 실패

Small Scale Variants Of The Block Cipher PRESENT

Gregor Leander

DTU Mathematics
Technical University of Denmark

3) 지금의 S-PRESENT로 정착

→ 깃허브에서 찾음. 블록 크기 8bit, key 16bit

Table 1. $n = 2$ (plaintext= 0 key=0)

round	state	key	state xor key	S(state xor key)
0	00	00	00	cc
1	f0	00	f0	2c
2	58	01	59	0e
3	54	01	55	00
4	00	62	62	a6
5	9c	2a	b6	8a
6	c4	33	f7	2d
7	59	5b	02	c6
8	b4	4c	f8	23
9	0d	84	89	3e
10	5e	55	0b	



02. PRESENT / S-PRESENT

PRESENT

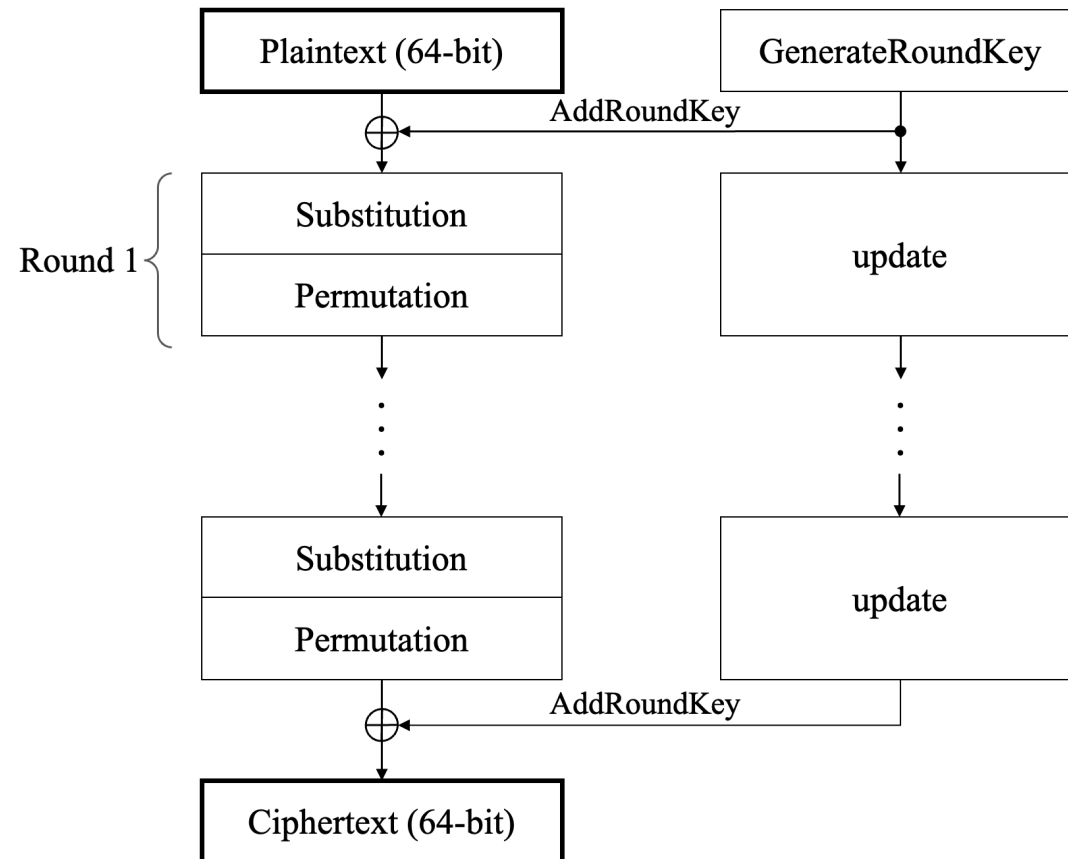
- 2007년 CHES에서 제안된 AES 기반의 SPN 구조 경량 블록 암호

S-PRESENT

- PRESENT의 Toy-Example

	n-bit	k-bit	r
PRESENT-64/80	64	80	31
PRESENT-64/128	64	128	31
S-PRESENT8/16	8	16	3

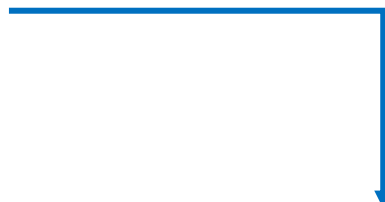
두 암호의 블록 크기, 키 크기, 라운드 수 비교



03. 딥러닝을 사용하여 알려진 평문 공격 수행

과정

- 모든 키의 경우의 수에 대해 랜덤 평문을 암호화하여 암호문을 생성
- 평문과 암호문을 연결한 쌍을 입력 데이터로, 비밀키를 라벨로 사용
- S-PRESENT의 경우 입력 데이터(16bit), 출력 데이터(16bit)
- 입력 데이터에 대응하는 키 예측
- 본 실험에서 데이터셋의 수는 총 102만 4000개



```
import secrets

SPRE = {}
count = 4000
counter = 1
for i in range(count):
    for ky in range(256):
        key = ky
        cipher = MiniPresent(key)
        plain = secrets.randbelow(256)
        cipher = cipher.encrypt(plain)
```

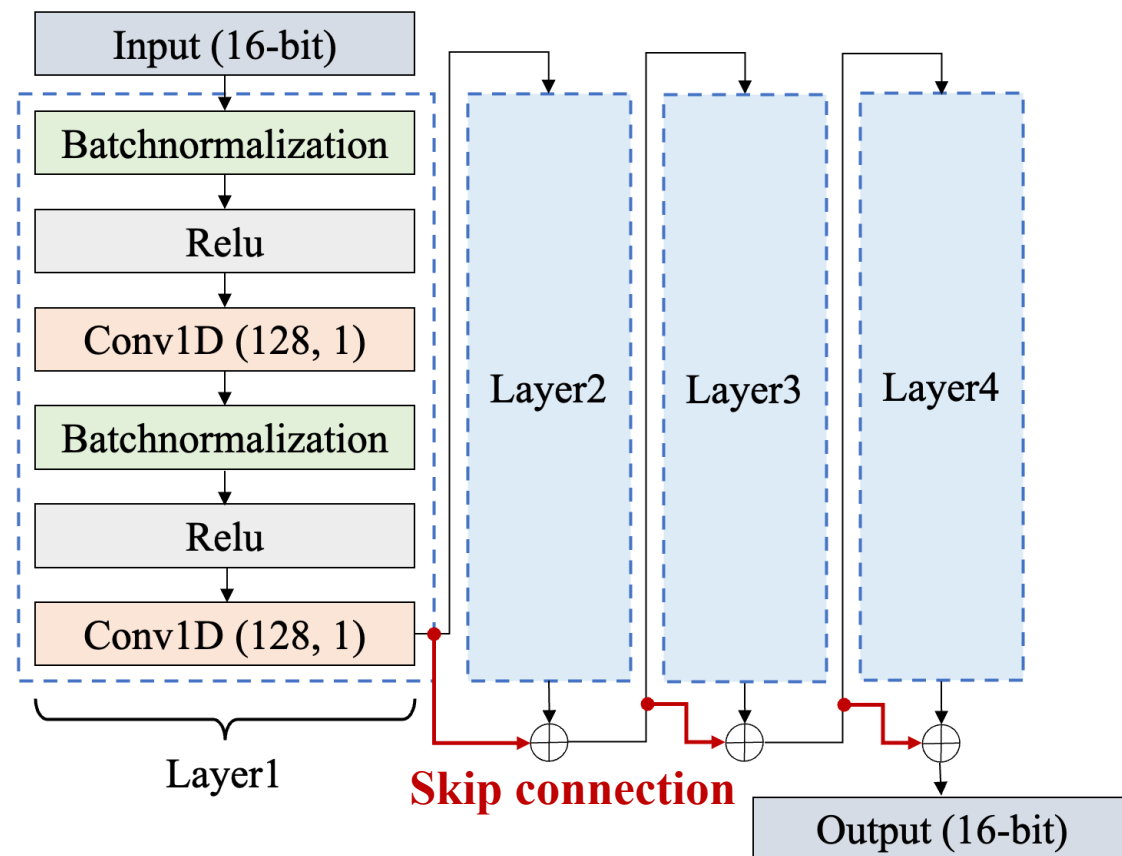
Python에서 random 대신 사용
(암호학적으로 강력한 난수 생성)

8bit key

04. 모델별 암호 분석 (CNN, MLP)

CNN / MLP 모델

- Conv1D (128, 1) → Linear (128)로 바꾸면 MLP 모델 구조임 (동일한 구조)
- MLP 모델의 구조는 현지언니 논문에서 사용된 구조 중 하나
 - Skip connection 사용
 - 128차원으로 고정
 - lr의 경우 MLP는 0.1 / CNN 0.01
- CNN의 경우 **커널 크기를 1로 설정**
 - GoogLeNet이라는 논문에서 소개된 기법
 - 장점 : 채널 수 조절, 연산량 감소, 비선형성 증가
 - but 해당 모델 구조에서는 △



05. 결론

- CNN 모델을 사용했을 때 전체 16bit key 중 12bit까지 공격 성공 (전체 정확도 평균은 두 모델이 유사)
- 9, 10, 11bit는 취약, 13bit는 안전
- 알려진 평문 공격을 통해 분석이 가능한 키 비트에 한계가 있음 (놔 줄 생각..)

Model	Key	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th	16th	AVG
CNN	8-bit	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.78	0.77	0.79	0.68	0.64	0.71	0.71	0.7	0.86
	9-bit	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.62	0.64	0.65	0.64	0.56	0.57	0.57	0.57	0.56	0.77
	10-bit	1.0	1.0	1.0	1.0	1.0	1.0	0.56	0.54	0.56	0.57	0.57	0.51	0.52	0.52	0.52	0.51	0.71
	11-bit	1.0	1.0	1.0	1.0	1.0	0.53	0.56	0.51	0.53	0.53	0.53	0.5	0.5	0.5	0.51	0.5	0.67
	12-bit	1.0	1.0	1.0	1.0	0.5	0.53	0.55	0.5	0.53	0.52	0.53	0.51	0.5	0.5	0.5	0.5	0.64
MLP	8-bit	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.79	0.78	0.79	0.74	0.58	0.76	0.74	0.73	0.87
	9-bit	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.59	0.6	0.62	0.61	0.53	0.54	0.59	0.55	0.54	0.76
	10-bit	1.0	1.0	1.0	1.0	1.0	1.0	0.57	0.54	0.56	0.57	0.56	0.51	0.52	0.52	0.52	0.52	0.71
	11-bit	1.0	1.0	1.0	1.0	1.0	0.52	0.56	0.51	0.53	0.52	0.53	0.51	0.5	0.5	0.5	0.5	0.67

감사합니다.