

Remove Some Noise: On Pre-processing of Side-channel Measurements with Autoencoders

<https://youtu.be/wH4hSJstOKw>

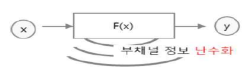

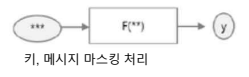
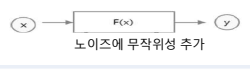
목적

- 노이즈 제거를 위해 오토인코더 사용 → Hiding 대응기법을 제거하는 새로운 접근법 제안
 - 기존 연구
 - 이미지 노이즈 제거를 위한 오토 인코더는 존재
 - 부채널 도메인 적용 x
- 다양한 유형의 노이즈 및 대응기법을 trace와 결합 → 동일한 딥러닝 모델로 노이즈 제거

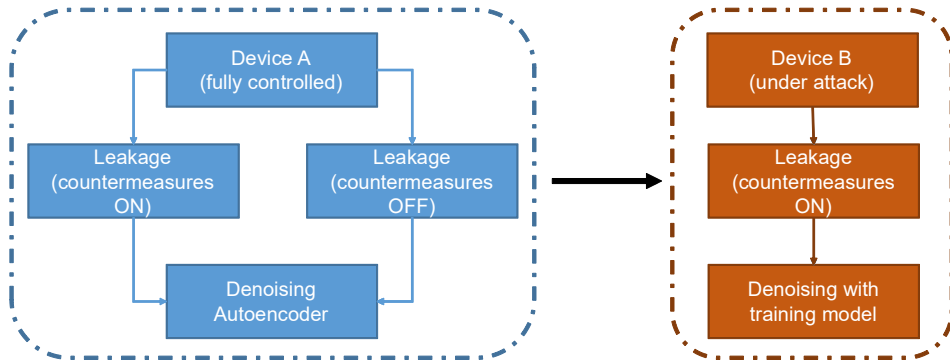
목적

- 하이딩 대응 기법을 잡음으로 간주
 - 입력 : 대응기법이 적용된 파형 사용
 - 라벨 : 대응기법이 없는 파형 사용
 - 실제 학습된 신경망 : 대응기법을 무력화하도록 파형 변환 가능

Side-Channel Attack(SCA) 대응책

대응기술	핵심요소	기법설명
Randomness 기법		유출 정보 난수 생성 계산 실제 값 유출방지
Blinding 기법		$y = f(x)$ x, y 블라인드 계산
Masking 기법	 <p>키, 메시지 마스킹 처리</p>	연산 시 중간값 숨김(랜덤화) 임의의 마스킹 값 교체
Hiding 기법	 <p>노이즈에 무작위성 추가</p>	연산 시 소모 전력량 랜덤화

Denoising strategy : white-box setting



Convolutional Auto-Encoder(CAE)

- Convolution 필터의 비지도학습을 위한 도구로 사용되는 CNN의 변형
- 최적의 필터를 학습하여 재구성 오류를 최소화하기 위해 이미지 노이즈 제거에 사용
- 노이즈가 있는 숫자 이미지를 깨끗한 숫자 이미지에 매핑하도록 오토 인코더 훈련



Convolutional Auto-Encoder(CAE)

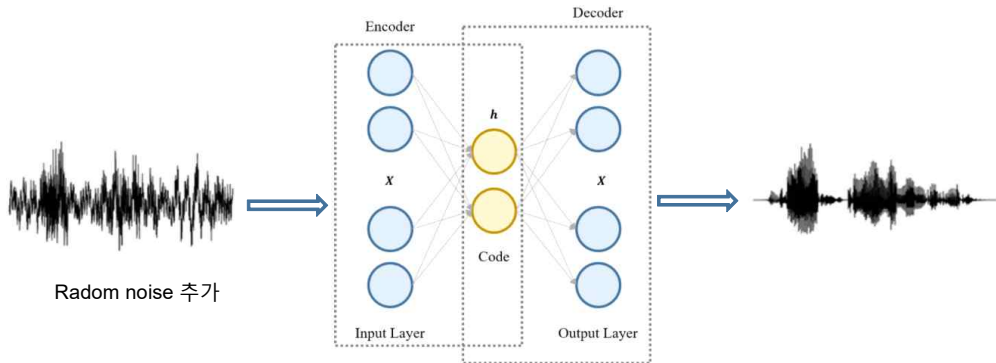
Table 1: CAE hyperparameter tuning.

Hyperparameter	Range	Selected
Optimizer	Adam, RMSProp, SGD	Adam
Activation function	Tanh, ReLU, SeLU	SeLU
Batch size	32, 64, 128, 256	128
Epochs	30, 50, 70, 100, 200	100
Training sets	1 000, 5 000, 10 000, 20 000	10 000
Validation sets	2 000, 5 000	5 000

```
# 1st convolutional block
x = Conv1D(4, 1, kernel_initializer='he_uniform', activation='selu', padding='same', name='block1_conv1')(img_input)
x = BatchNormalization()(x)
x = AveragePooling1D(2, strides=2, name='block1_pool')(x)
```

SeLU 사용 이유? SeLU의 활성화 기능을 사용하여 gradient 문제해결

Denoising autoencoder



manifold 상에서는 동일
원본 데이터와는 다른 데이터로 encoder와 decoder 학습하는 네트워크

Denoising “clean” traces

```
def addGaussianNoise(traces, noise_level):
    print('Add Gaussian noise...')
    if noise_level == 0:
        return traces
    else:
        output_traces = np.zeros(np.shape(traces))
        print(np.shape(output_traces))
        for trace in range(len(traces)):
            if (trace % 5000 == 0):
                print(str(trace) + '/' + str(len(traces)))
                profile_trace = traces[trace]
                noise = np.random.normal(
                    0, noise_level, size=np.shape(profile_trace))
                output_traces[trace] = profile_trace + noise
        return output_traces
```

- 해당 코드 실행전, 파형에 노이즈 추가 (add noise)
- 인코더에서 노이즈를 제거하고 다시 디코더 실행
- 원본 데이터를 다시 거치면 => 노이즈가 제거된 파형
- 공격 코드 실행

```
def old_cnn(lr, input_length):
    img_input = Input(shape=(input_length, 1))
    #encoder
    x = conv(img_input, 256, 2, 'selu', 0)
    x = conv(x, 256, 2, 'selu', 0)
    x = conv(x, 256, 2, 'selu', 0)
    x = conv(x, 256, 2, 'selu', 0)
    x = conv(x, 256, 2, 'selu', 0)
    x = conv(x, 256, 2, 'selu', 5)
    x = conv(x, 128, 2, 'selu', 0)
    x = conv(x, 128, 2, 'selu', 0)
    x = conv(x, 128, 2, 'selu', 0)
    x = conv(x, 128, 2, 'selu', 2)
    x = conv(x, 64, 2, 'selu', 0)
    x = conv(x, 64, 2, 'selu', 0)
    x = conv(x, 64, 2, 'selu', 0)
    x = conv(x, 64, 2, 'selu', 2)
    x = Flatten(name='flatten')(x)

    x = Dense(512, activation='selu')(x)

    x = Dense(2240, activation='selu')(x)
    x = Reshape((35, 64))(x)
    x = deconv(x, 64, 2, 'selu', 2)
    x = deconv(x, 64, 2, 'selu', 0)
    x = deconv(x, 64, 2, 'selu', 0)
    x = deconv(x, 64, 2, 'selu', 0)
    x = deconv(x, 128, 2, 'selu', 2)
    x = deconv(x, 128, 2, 'selu', 0)
    x = deconv(x, 128, 2, 'selu', 0)
    x = deconv(x, 128, 2, 'selu', 0)
    x = deconv(x, 256, 2, 'selu', 5)
    x = deconv(x, 256, 2, 'selu', 0)
    x = deconv(x, 256, 2, 'selu', 0)
    x = deconv(x, 256, 2, 'selu', 0)
    x = deconv(x, 256, 2, 'selu', 0)
    x = deconv(x, 256, 2, 'selu', 0)

    x = deconv(x, 1, 2, 'sigmoid', 0)

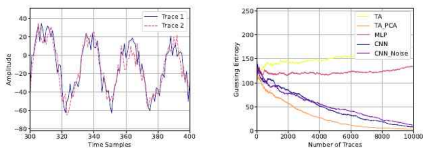
    model = Model(img_input, x)
    @model = multi_gpu_model(model)
    opt = RMSProp(lr=lr)
    model.compile(loss='mse', optimizer='adan', metrics=['accuracy'])
    model.summary()
    return model
```

Attack

- cnn_best 와 mlp_best를 사용하여 공격 시도

이미지 분류 예측을 위한 모델 사용

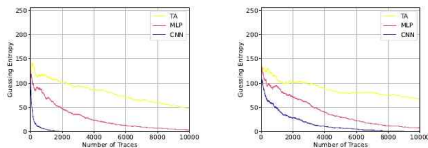
- 기존 CAE에 비해 CAE에 trace 재구성 후, trace 감소 (831 → 751)
- 노이즈 입력 계층 추가 시 trace 감소(742 → 647)
- 노이즈 추가 CNN의 성능 > 노이즈 없는 CNN의 성능



(a) Gaussian noise: zoom-in view.

(b) Gaussian noise: guessing entropy.

Figure 1: Gaussian noise: demonstration and its influence on guessing entropy.



(a) GE: denoise with averaging.

(b) GE: denoise with CAE.

Figure 2: Guessing entropy: denoising Gaussian noise with averaging (a) and CAE (b).

결론

- 누출된 trace로부터 noise와 countermeasure를 제거하는 CAE 기법 제안
- 제안된 CAE는 노이즈를 제거/축소하고 ground truth를 결정하여 공격 성능을 크게 향상
- 오토인코더가 훈련 프로세스에서 사용된 모든 노이즈를 포함 x
 - 노이즈/대응책 안정적으로 제거
- 프로파일링된 공격 수행 $x \rightarrow$ 측정을 사전 처리하여 공격 전략을 적용하는 것

Q & A

