Workshop on Artificial Intelligence and Cryptography 2021

Deep Learning and Side-Channel Analysis

Guilherme Perin

TU Delft

Content

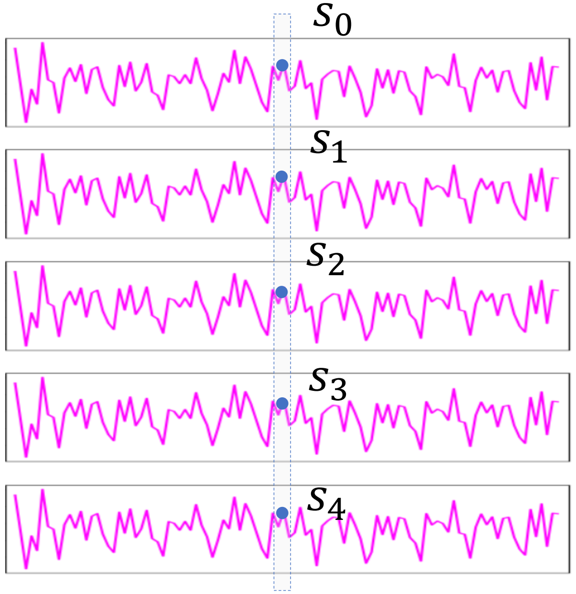
- Main threat models in DL-SCA
- Profiling attacks (from template attacks to deep learning)
- Basic steps
- DL-SCA in the last 6 Years (achievements and challenges)
- Overfitting, generalization, metrics

- AISY Framework
- Attack scenarios (demonstration)

Non-Profiling Attacks (DPA, CPA)

Measurements
(Observations)

Hypothetical leakage
for each key candidate



l_0
l_1
l_2
l_3
l_4

s vectors l

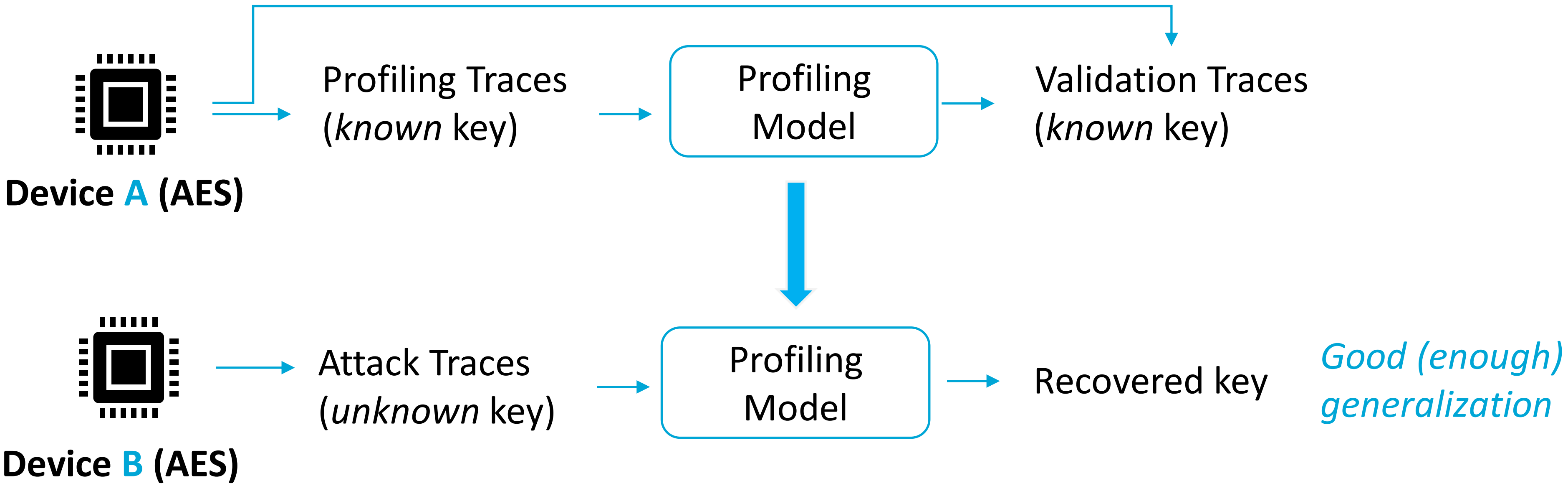
$l_i = S_{BOX}(p_i \oplus k_i)$

Distinguisher f

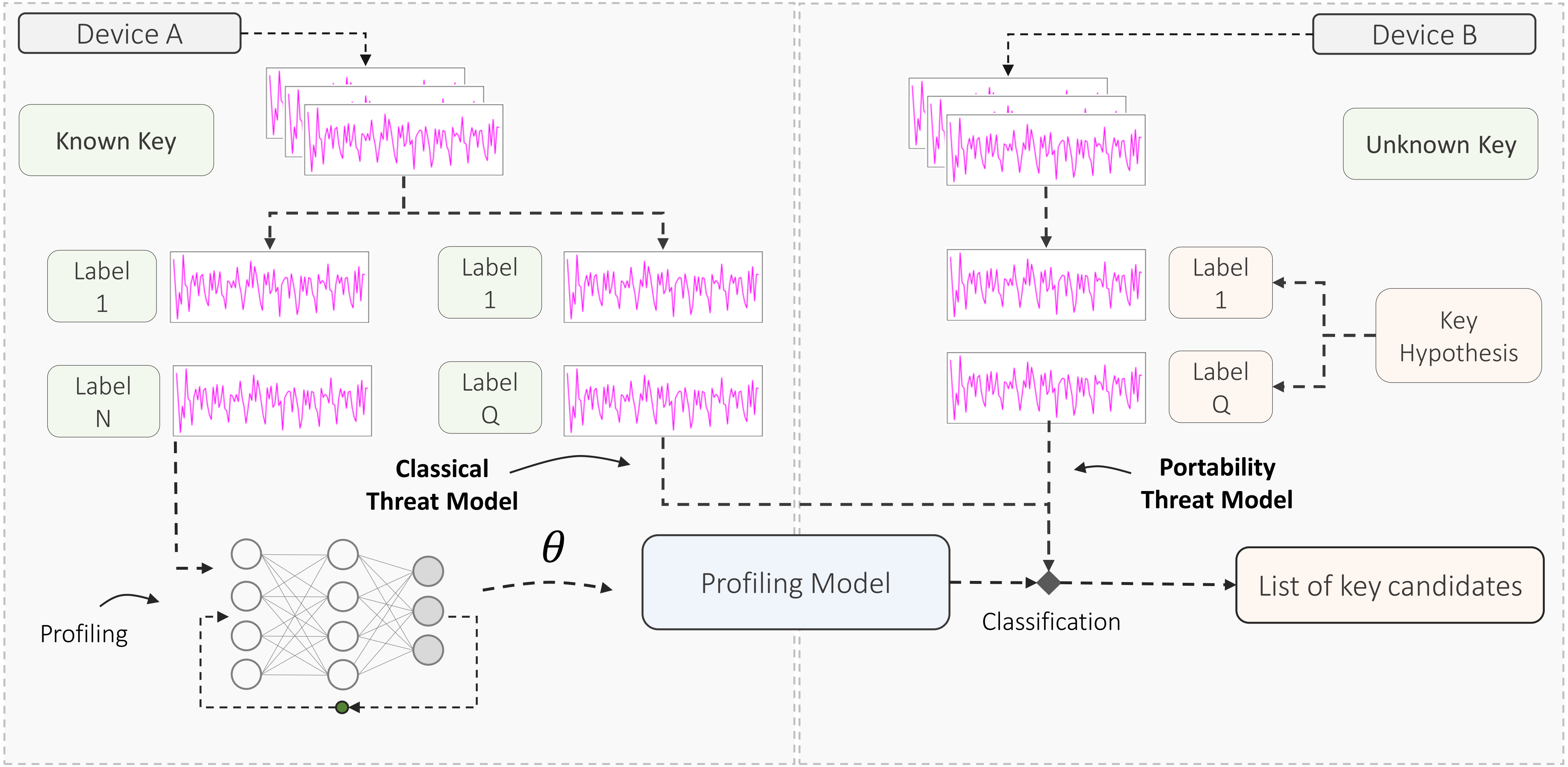
$$k^* = \max_k f(s, l)$$

- No assumptions about statistical distributions of measured leakages.
- Non-parametric approach
- High-order analysis

Profiling Attacks



Main threat models in DL-SCA (profiling)

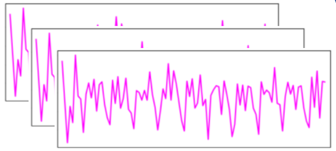


Profiling Attacks

- Profiling attacks are about class probabilities
  - Classification
- Worst-case security assessment
  - If profiling attacks fail, then is it secure?
- Realistic attacks?
  - Adversary needs an accessible device for profiling (change key, access to random values, source code)
  - JIL rating (smart cards): identification and exploitation phases
- If we relax adversary assumptions (knowledge about source code, randomness, etc.), are profiling attacks still real worlds threats? Why so much effort on this type of attack?
  - Feature selection becomes difficult
  - Deep learning/AI might change some strong assumptions from the community in future

Profiling Attacks (classical way)

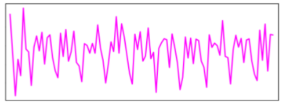
Measurements (Observations)



class a

class b

measurement x



parameters (classes a,b)

$pdf(x)$

$p(a|x)$

$p(b|x)$

(Approximate) Statistical Distribution Assumptions

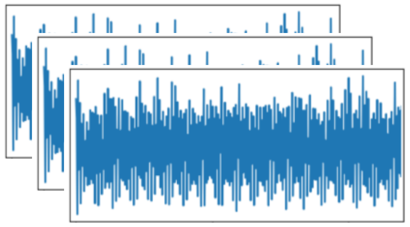
$$l_n = l(p,k) + \mathcal{N}(\mu,\sigma^2)$$

well-defined equation

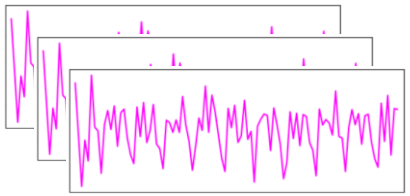
$$pdf(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$p(a|x) = 1 - p(b|x)$$

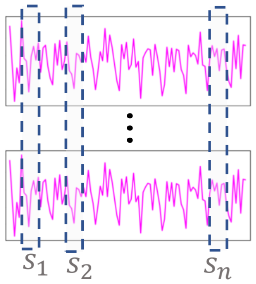
Template Attacks



Raw Traces

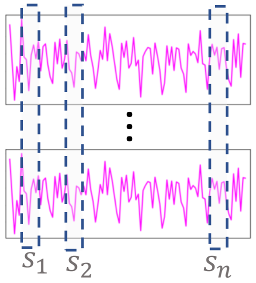


Pre-processed Traces
(trim, align, resample, filter)



Feature Selection
(Points-of-Interest)

Template Attacks (Gaussian Mixture Models)



$S = n \times N$

Hamming Weight

$l_i = HW(S_{BOX}(p_i \oplus k_i))$

$S_0 = n \times N_{HW=0} \rightarrow \mu_0, \Sigma_0$

$S_1 = n \times N_{HW=1} \rightarrow \mu_1, \Sigma_1$

$S_2 = n \times N_{HW=2} \rightarrow \mu_2, \Sigma_2$

$S_3 = n \times N_{HW=3} \rightarrow \mu_3, \Sigma_3$

$S_4 = n \times N_{HW=4} \rightarrow \mu_4, \Sigma_4$

$S_5 = n \times N_{HW=5} \rightarrow \mu_5, \Sigma_5$

$S_6 = n \times N_{HW=6} \rightarrow \mu_6, \Sigma_6$

$S_7 = n \times N_{HW=5} \rightarrow \mu_7, \Sigma_7$

$S_8 = n \times N_{HW=6} \rightarrow \mu_8, \Sigma_8$

n $\quad$ $n \times n$

$$p(l=0|x) = \frac{1}{\sqrt{(2\pi)^n|\Sigma_0|}} e^{\left(-\frac{1}{2}(x-\mu_0)^T {\Sigma_0}^{-1}(x-\mu_0)\right)}$$

$$p(l=1|x) = \frac{1}{\sqrt{(2\pi)^n|\Sigma_1|}} e^{\left(-\frac{1}{2}(x-\mu_1)^T {\Sigma_1}^{-1}(x-\mu_1)\right)}$$

$\vdots$

$$p(l=8|x) = \frac{1}{\sqrt{(2\pi)^n|\Sigma_8|}} e^{\left(-\frac{1}{2}(x-\mu_8)^T {\Sigma_8}^{-1}(x-\mu_8)\right)}$$

$$\sum_{i=0}^{8} p(l=i|x) = 1$$

$\max p(l|x)$

$$k^* = \max_k \left(\sum \log p(l|x) \right)$$

Machine Learning (incl. Deep Learning)

Measurements (Observations)

class a

class b

Hyperparameters

$pdf(x)$

Parameters are learned from data

No assumptions about data statistical distribution

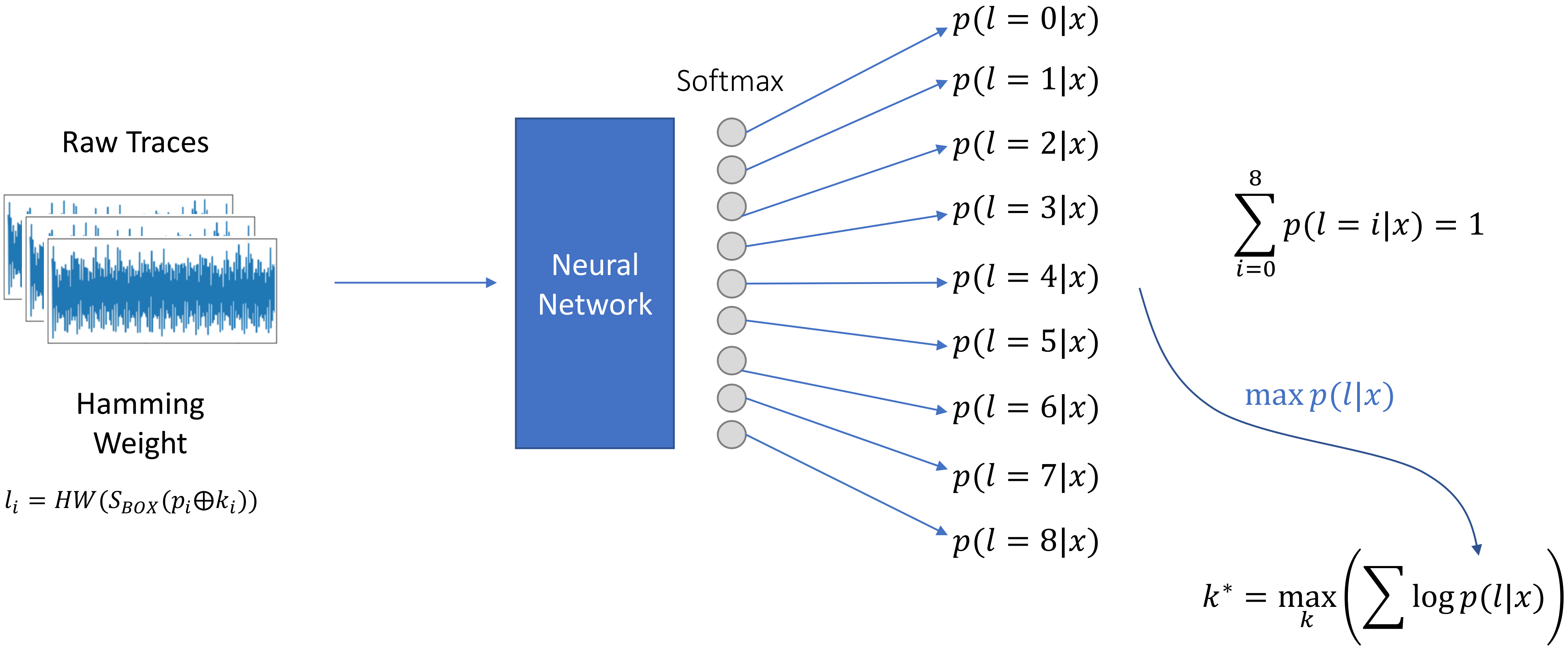
New measurement x

$p(a|x)$

$p(b|x)$

$$p(a|x) = 1 - p(b|x)$$

Deep Learning



Template vs Deep Learning: should we compare?

- TA and DL have different purposes and applications

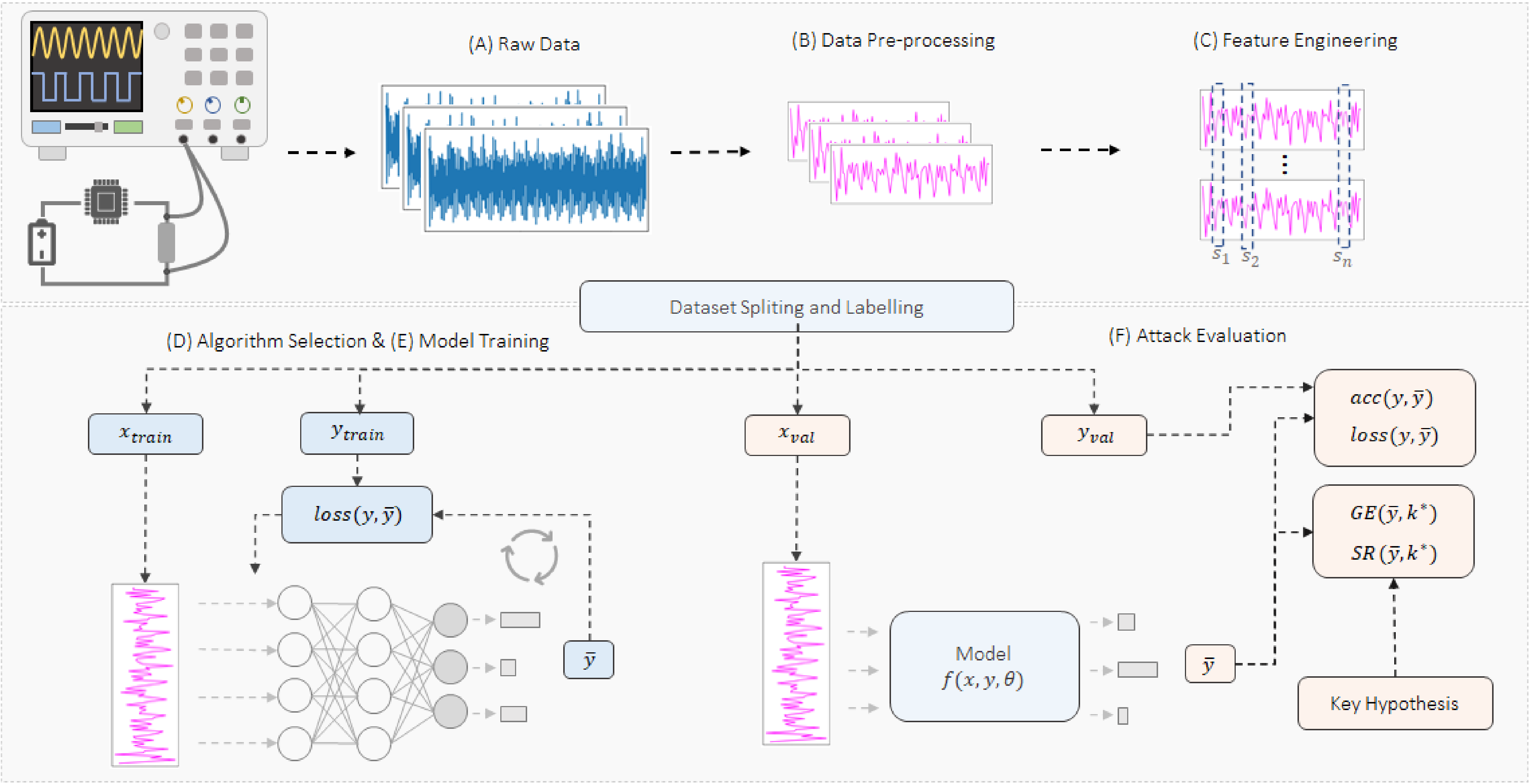
  - *Bronchain et al. "Breaking Masked Implementations with Many Shares on 32-bit Software Platforms or When the Security Order Does Not Matter", CHES 2021.*

- DL is not a replacement. It is an alternative. It is what comes next.
  - Highly exploratory.
  - But limitations are still unknown (this is a good direction for research).

Basic steps for DL-SCA

- Get measurements (profiling and attack traces)
- Leakage assessment ?
- Split profiling set into training and validation traces
- Label training, validation and attack traces
- Define the neural network (or neural network search process)
- Define the metric (Guessing Entropy, Number of Attack Traces)
- Train, validate, adjust, train, validate, adjust, ...
- Attack phase

Main DL-SCA steps



Workflow in DL-SCA

- DL-SCA attacks against (first-order protected) AES implementations became completely feasible (at least in 2021).
- What would happen if we would focus on more realistic and difficult targets?

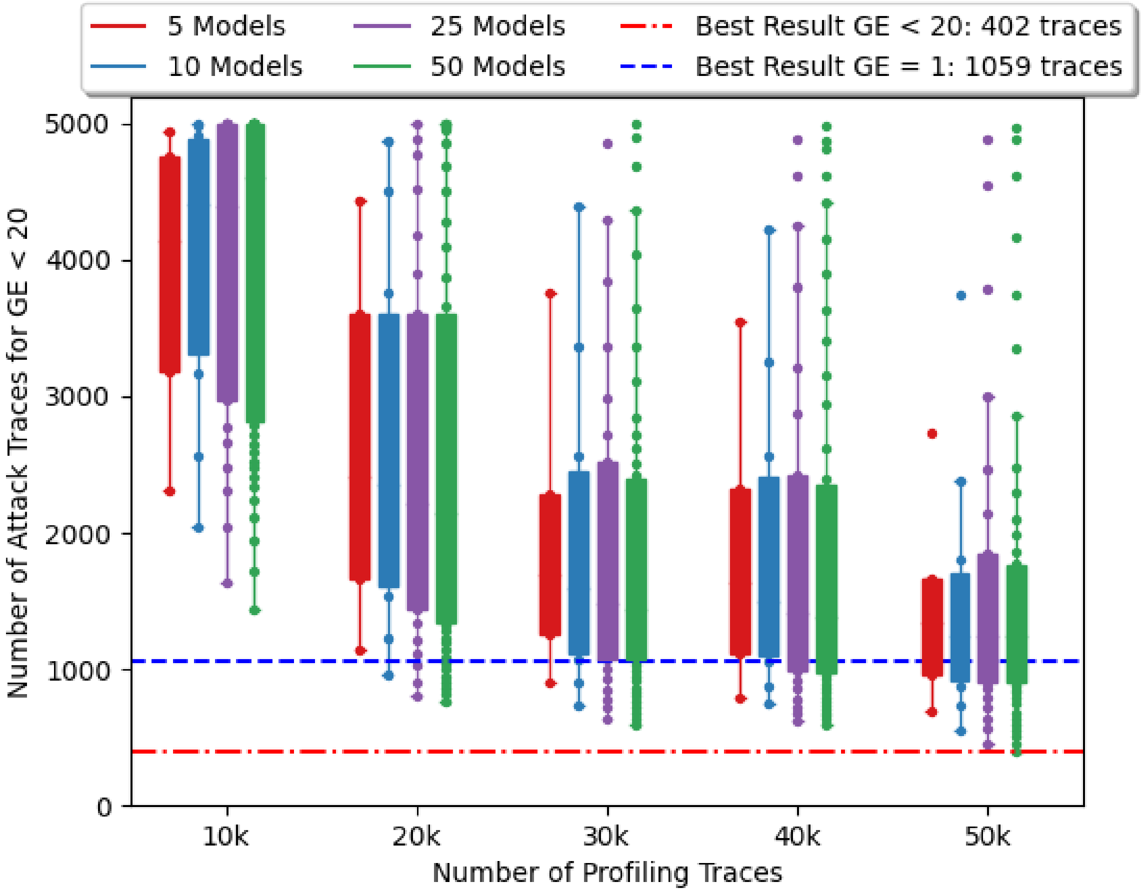
- We need to understand what impacts our attack efficiency

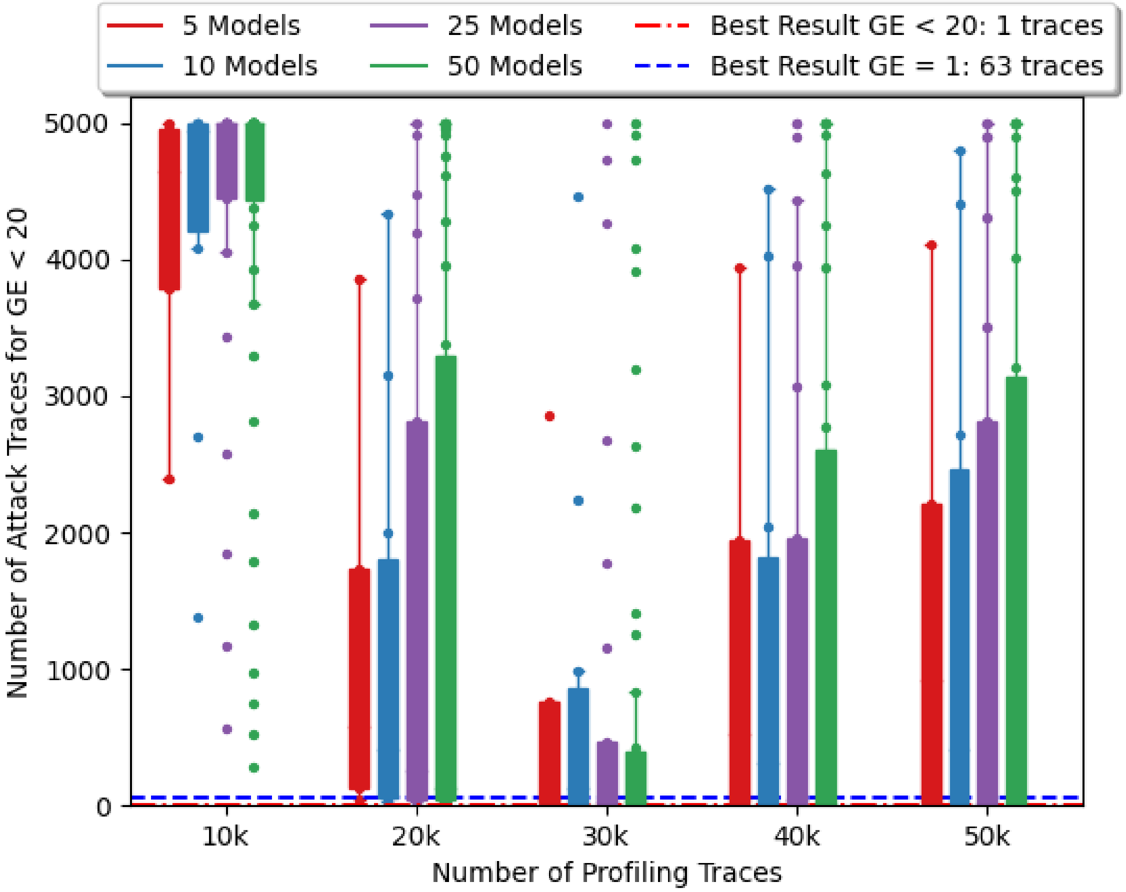
Attack Components

- Number of profiling traces
- Number of attack traces
- Learnability: number of neural networks configurations we can try

- What happen if we have limited traces (profiling, attack), but *infinite learnability capacity*?
  - We should be able to recover the key with a single attack trace.
  - "*Replace the human by the machine*" -> we are far from this point. And we need good frameworks and guidelines.

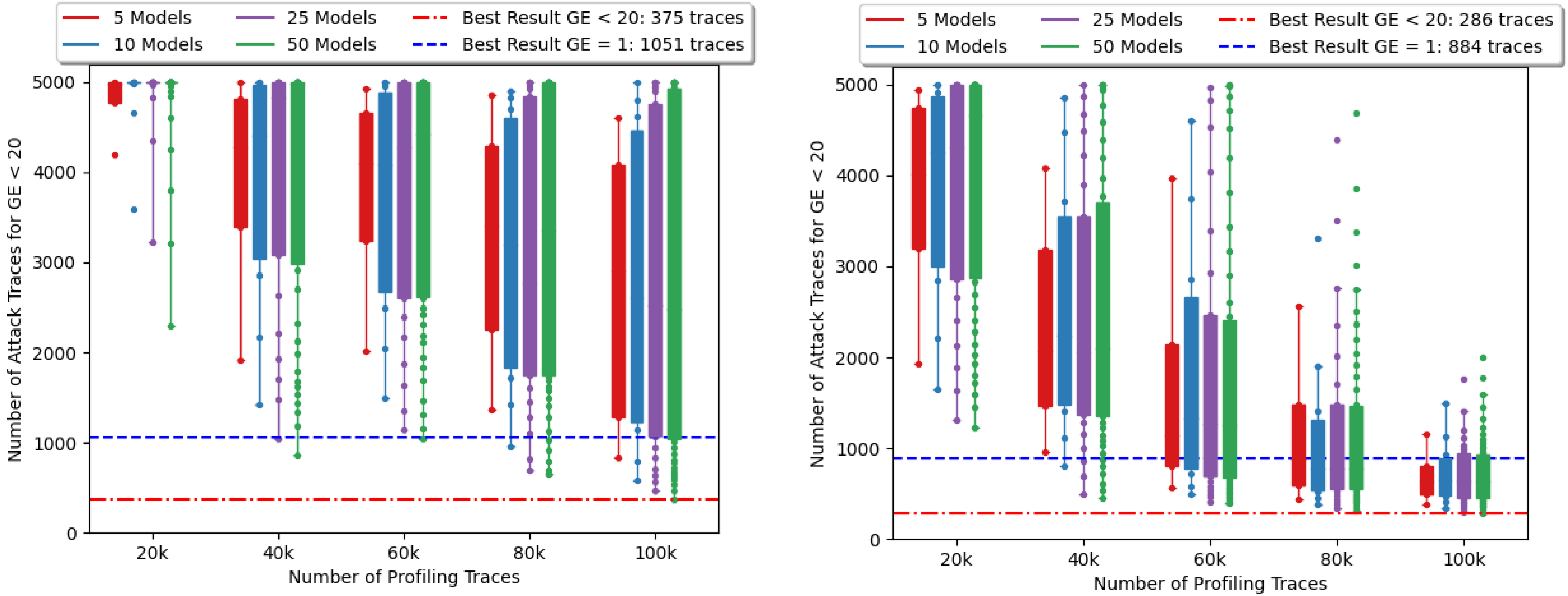
Efficient Attacker Framework





Number of profiilig traces impacts more than learnability

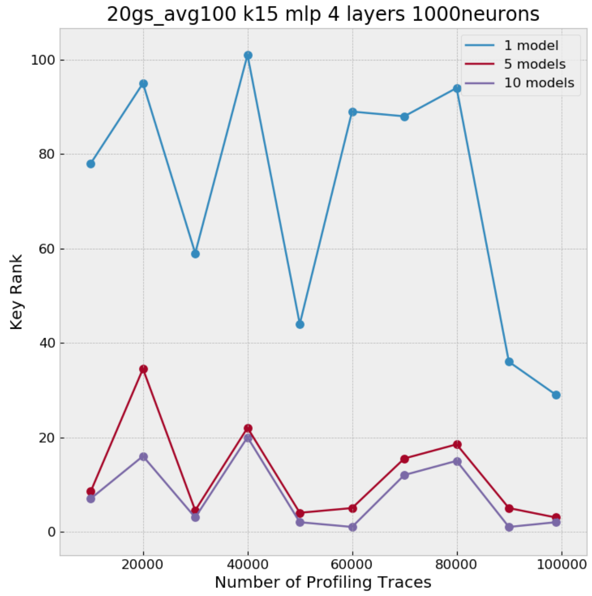
Efficient Attacker Framework

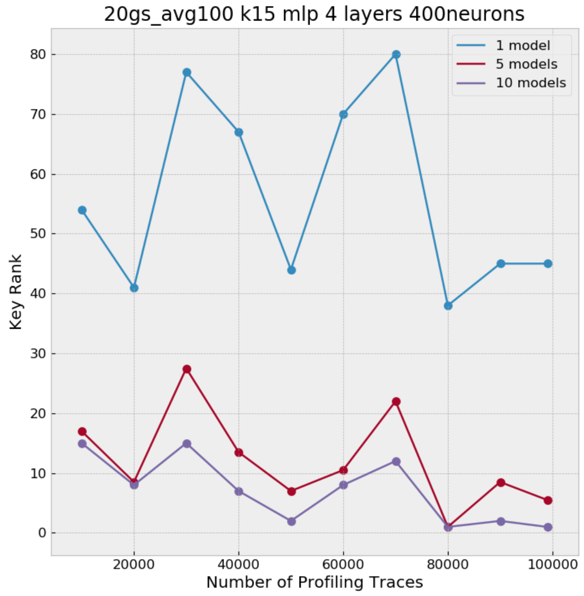


Number of profiling traces and learnability impacts the efficiency.

Efficient Attacker Framework

- Analyze how the number of profiling traces impacts the attack performance (Guessing Entropy, Success Rate)
- Analyze how number of models (neural network configurations) affects results (random hyperparameter search)





The number of profiling traces has a smaller influence in the attack performance with respect to number of models we try.

“Zooming out from the problem”.

Achievements
Challenges

DL-SCA in security evaluations

- Hyperparameter search that delivers *pass* verdict is enough?
  - 100 Models (4 activation functions): 2 activation functions are inefficient -> 50% of models are useless.
  - We need more realistic open-source datasets
  - How to judge/evaluate attacker’s decisions?
- AI, deep learning, reinforcement learning fields are evolving fast. Are security evaluators adopting state-of-the-art?
  - How feasible is to do that?
  - It is becoming a standard to publish source code with paper. That is a good thing!
  - We need good frameworks.

DL-SCA progress over the years

2016	2017	2018	2019	2020	2021
MLP CNN	CNN Against Trace Misalignment	ASCAD Database Non-Profiled Attack	Methodologies for CNNs Visualization Data Augmentation DL vs SCA Metrics	Ensembles DL Preprocessing (AE) Portability DL for Horizontal Attacks	Reinforcement Learning SCA-based Loss Functions Advanced DL models End-to-End Attack Explainability Pruning Frameworks
1	2	7	28	52	> 52

A lot of enthusiasm

Moment to find out lots of unknowns

It is changing...
(Full session on CHES)

Where are we today?

- (MILESTONE) Successful and efficient profiling attacks against *first-order* (Boolean) *masking* schemes (symmetric crypto) and *protected* public-key algorithms (RSA, ECC)
    - Software (8, 16 and 32-bit platforms) and Hardware (FPGA)
    - Noisy measurements
    - Misaligned measurements
    - Unified deep learning structures for multiple targets
    - Efficient methods for hyperparameters search

Challenges

- Attacks on high-order masking schemes (e.g., ASCADv2)
- We need more realistic open-source datasets (countermeasures, platforms)
- Definition of best profiled attack setup (several methods, which one should we use?)
- What is the path to define an efficient neural network?
  - For profiling SCA, do we always have to run hundreds of trainings?
  - Can we have a universal DL model for multiple targets?
- Metrics that can evaluate how much the model is bypassing countermeasures
  - Could we measure how much my neural network is bypassing misalignment during training? If yes, we could adapt hyperparameters earlier during hyperparameter search.
  - Hyperparameter search/optimization based on the SCA context
- Efficient model interpretation (to avoid wrong security assessments conclusions)
  - Explainability and interpretability
- Going from <u>local</u> to broad generalization (F. Chollet, "*On the measure of intelligence*", 2019)

- Unsupervised DL-SCA attack

Generalization
Overfitting
Metrics

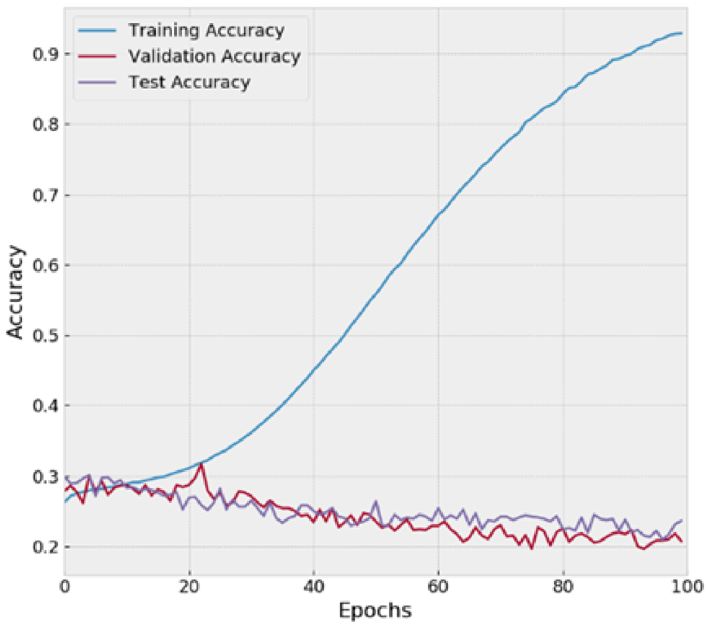
Minimizing loss

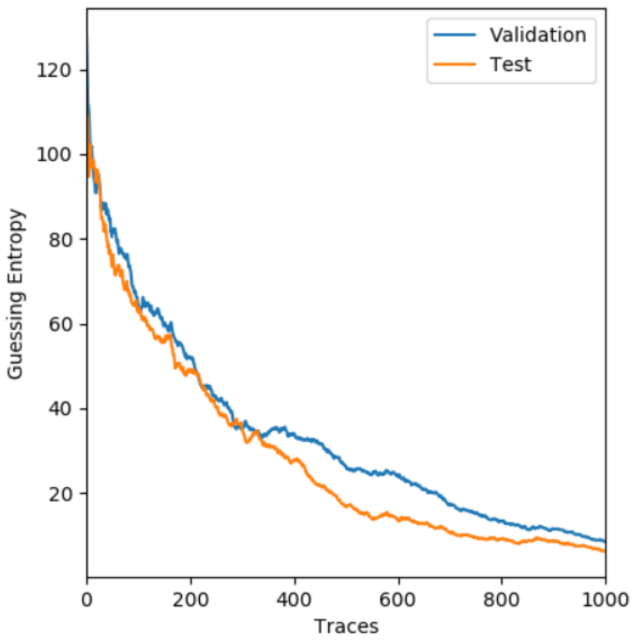
- Minimizing **categorical cross-entropy** is equivalent to maximize **Perceived Information (PI)**
  - *Masure et al. "A Comprehensive Study of Deep Learning for Side-Channel Analysis", TCHES2020.*

- Other directions using SCA-based loss functions:
  - Cross-Entropy Ratio (CER) Loss Function (TCHES 2020)
  - Ranking Loss (RKL) (TCHES 2021)
  - Ensemble Loss (TCHES 2021)

SCA vs DL Metrics

- Validation loss and validation accuracy



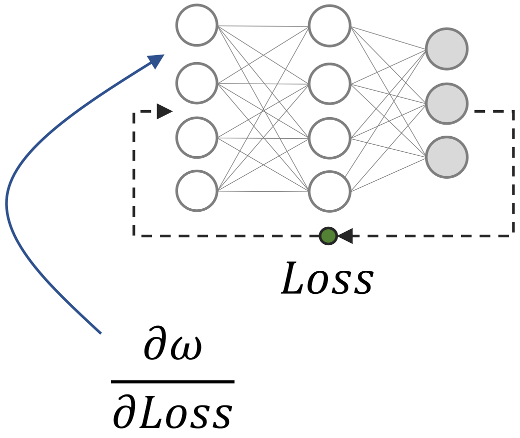


Overfitting and Generalization – Can we measure?

- Not clear method for that. Only estimation.
- What we can do is to check generalization capability.
- Can Guessing Entropy indicate generalization?

  - Guessing entropy vector = average of 100 key rank executions
  - From a large set of U attack traces, randomly select Q attack traces for each key rank execution, where U >>>> Q
  - Ex.: U = 100000, Q = 1000

Visualization



Other methods:

- Occlusion
- Saliency maps
- Layer-wise relevance propagation
- LIME

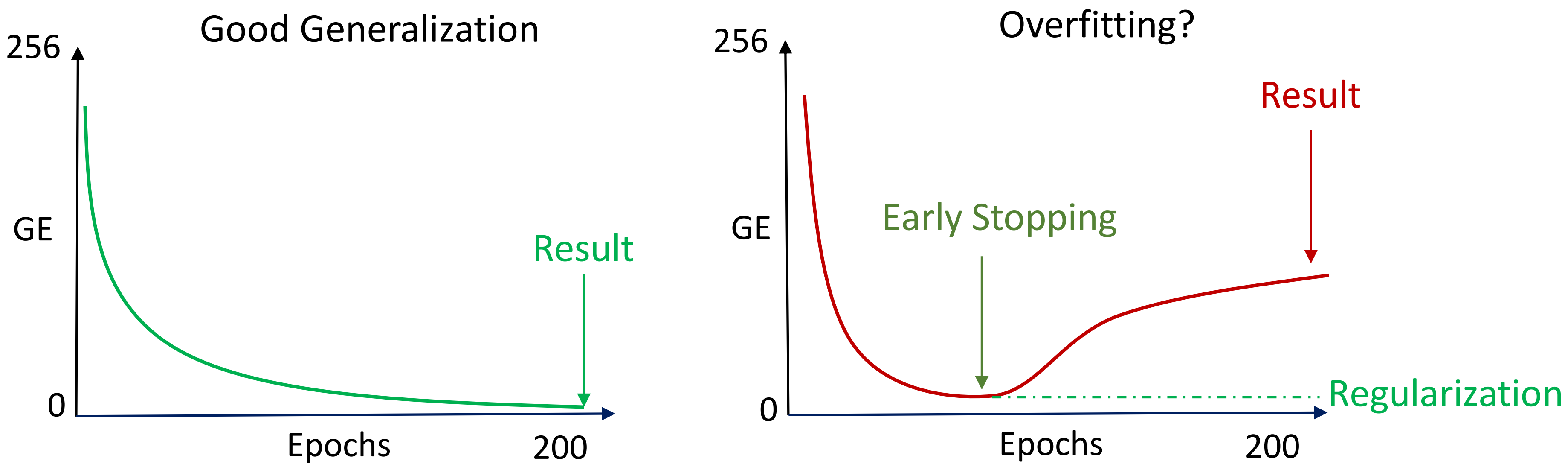
Input Gradient: measures how much changing ω affects $Loss$

- Most important features for the trained model
- Good tool to understand/visualize if a neural network is able to automatically reject what is not leakage.

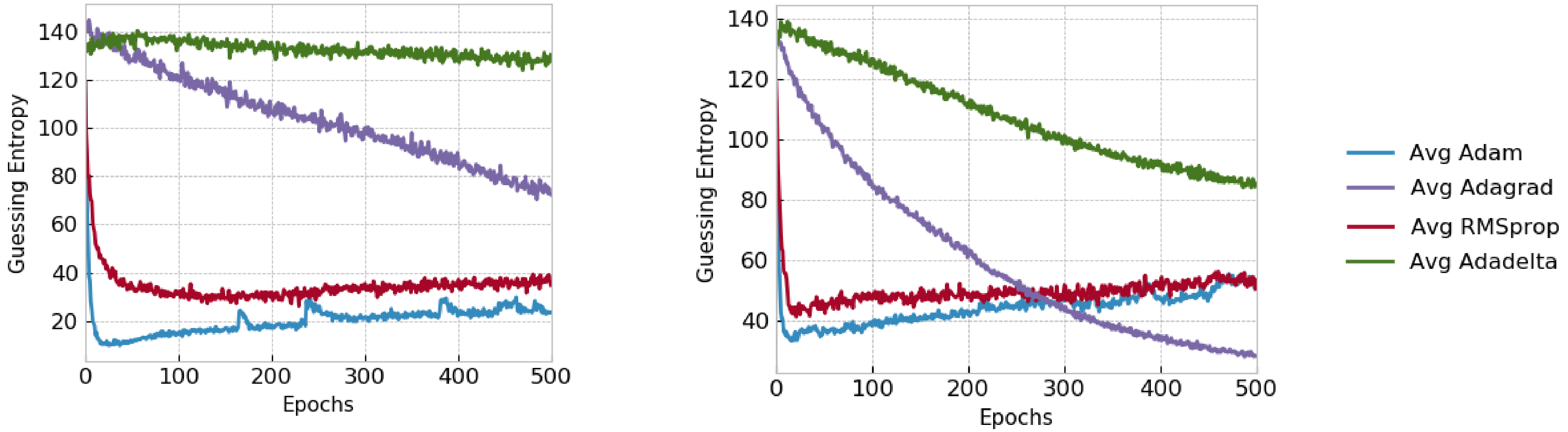
Regularization

- Implicit regularization (small models, large training sets): less overfitting
  - Small model capacity adds regularization
  - Small learning rates
- Explicit regularization (large models, small training sets): easy overfitting
  - Gaussian or noise layers
  - Early stopping
  - Batch normalization
  - Data augmentation (traces shifts, noise)

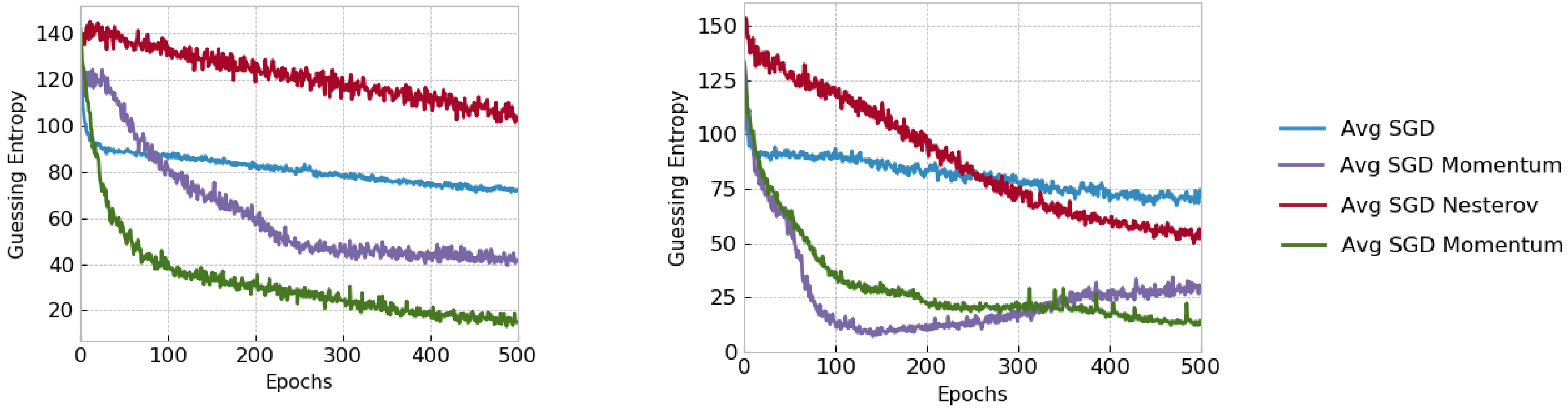
Regularization



Optimizers



Guessing Entropy
Epochs
Avg Adam
Avg Adagrad
Avg RMSprop
Avg Adadelta



Guessing Entropy
Epochs
Avg SGD
Avg SGD Momentum
Avg SGD Nesterov
Avg SGD Momentum

Lottery Ticket Hypothesis (LTH)

- J. Frankle et al, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks", ICLR 2019.

- Technique to find efficient deep learning models without tuning hyperparameters
- Alternative regularization method

- Train a baseline model (large one)
- Pruning
- **Reinitialize** the pruned model with **initial weights** from baseline
- Train pruned model
- Accuracy Pruned $\cong$ Accuracy Baseline

AISY Framework

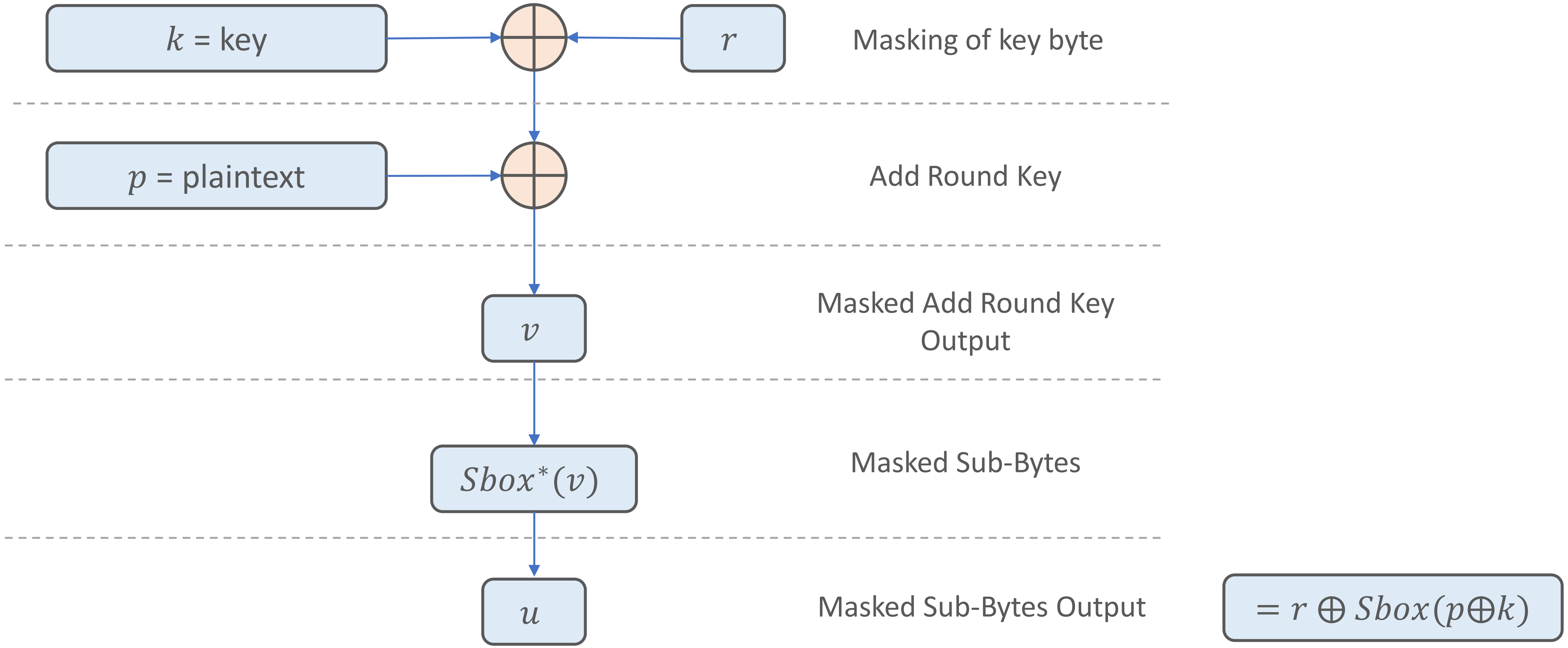
ASCAD Database (AES128, 8-bit Software)

- ASCAD fixed key (1 device):
  - 60,000 measurements
  - 100,000 points per measurement

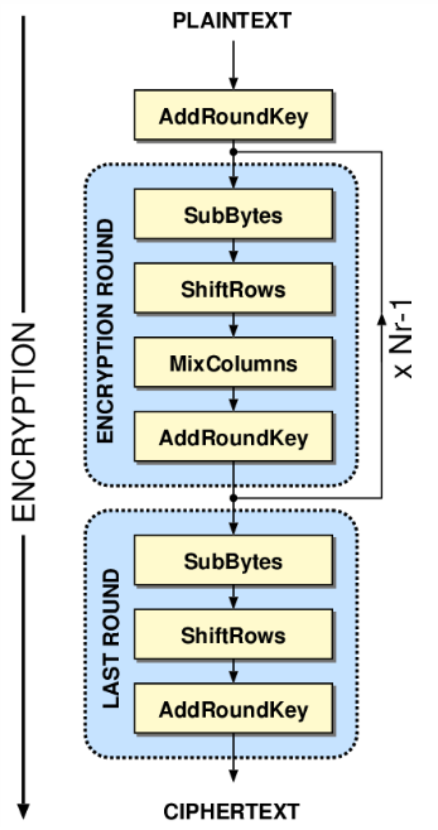
- ASCAD random keys (1 device):
  - 200,000 measurements (random key)
  - 100,000 measurements (fixed key)
  - 250,000 points per measurement

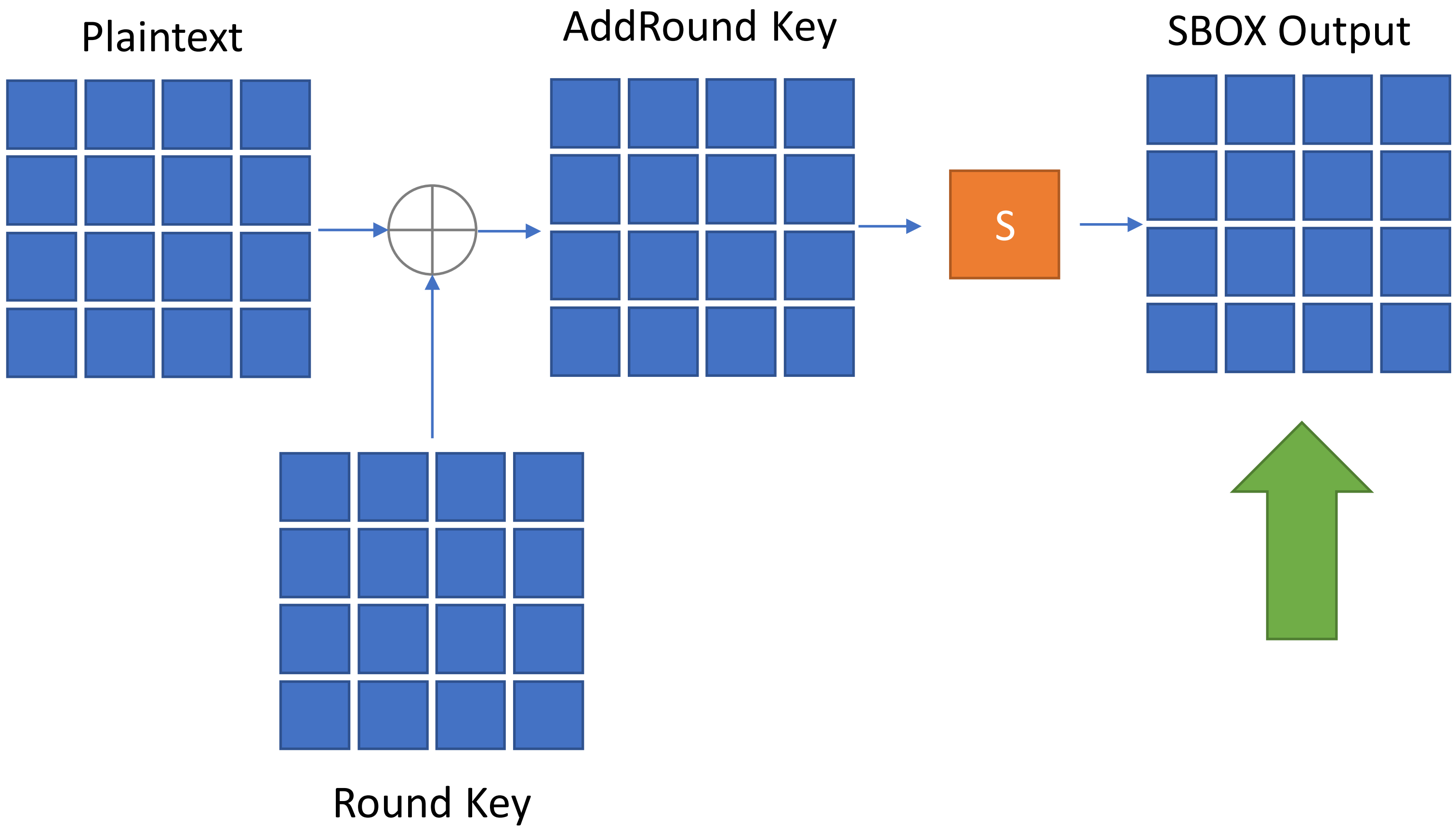
- Key bytes 0 and 1: unprotected (masks are equal to ZERO)
- Key bytes 2 to 15: 1^{st} order Boolean masking countermeasure

ASCAD Database (1st order Boolean Masking)



AES-128 Encryption





AISY Framework

AISY Framework

- custom
- scripts
- webapp
- app.py
- requirements.txt

- script_aes.py
- script_aes_cpa.py
- script_aes_custom_callback.py
- script_aes_custom_metrics.py
- script_aes_custom_table.py
- script_aes_data_augmentation.py
- script_aes_ensembles.py
- script_aes_grid_search.py
- script_aes_lth.py
- script_aes_neural_network_parameters.py
- script_aes_plot_probability_ranks.py
- script_aes_random_search.py
- script_aes_save_to_npz.py
- script_aes_visualization.py
- script_open_npz_file.py

AISY Framework

AISY Framework

- custom
- scripts
- webapp
- app.py
- requirements.txt

Path settings (datasets, databases, models, figures)
Localhost application (*Flask*)

AISY Framework

- custom
- scripts
- webapp
- app.py
- requirements.txt

```
numpy==1.19.4
tensorflow-gpu==2.0.0
matplotlib
joblib
keras==2.1.6
plotly==4.5.2
sqlalchemy
flask
python-dotenv
flaskcode
pandas
hiplot
dash
h5py==2.10.0
termcolor
pytz
sklearn
scipy
dash_bootstrap_components
aisy-sca==0.1.7
aisy-database==0.1.0
```

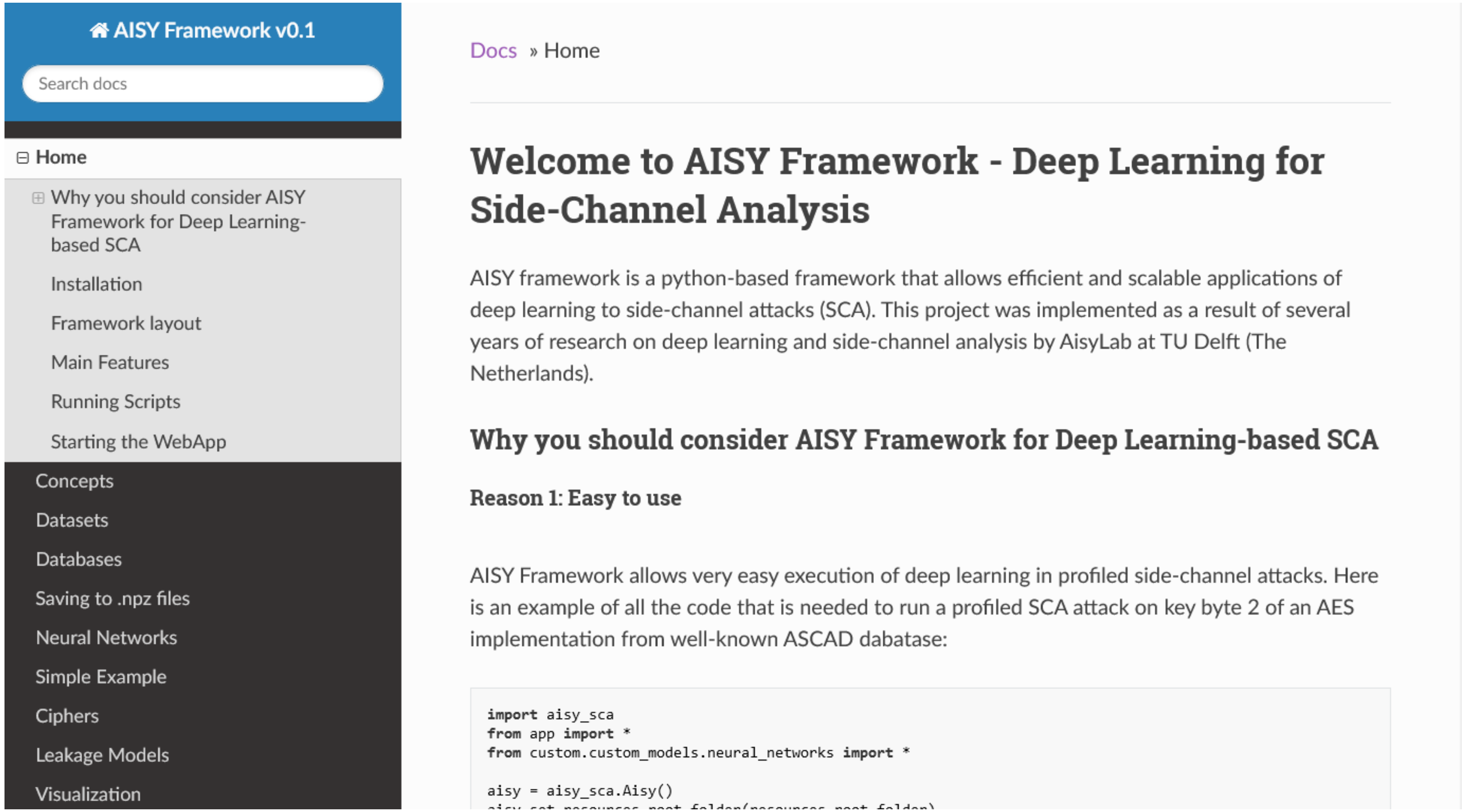
Installation

```
git clone https://github.com/AISyLab/AISY_Framework.git
cd AISY_Framework

virtualenv env
source env/Scripts/activate (Windows)
source env/bin/activate (Linux/MacOS)

pip install –r requirements.txt
```

https://aisylab.github.io/AISY_docs/



AISY Framework v0.1

Search docs

- Home
  - Why you should consider AISY Framework for Deep Learning-based SCA
  - Installation
  - Framework layout
  - Main Features
  - Running Scripts
  - Starting the WebApp
- Concepts
- Datasets
- Databases
- Saving to .npz files
- Neural Networks
- Simple Example
- Ciphers
- Leakage Models
- Visualization

Docs » Home

Welcome to AISY Framework - Deep Learning for Side-Channel Analysis

AISY framework is a python-based framework that allows efficient and scalable applications of deep learning to side-channel attacks (SCA). This project was implemented as a result of several years of research on deep learning and side-channel analysis by AisyLab at TU Delft (The Netherlands).

Why you should consider AISY Framework for Deep Learning-based SCA

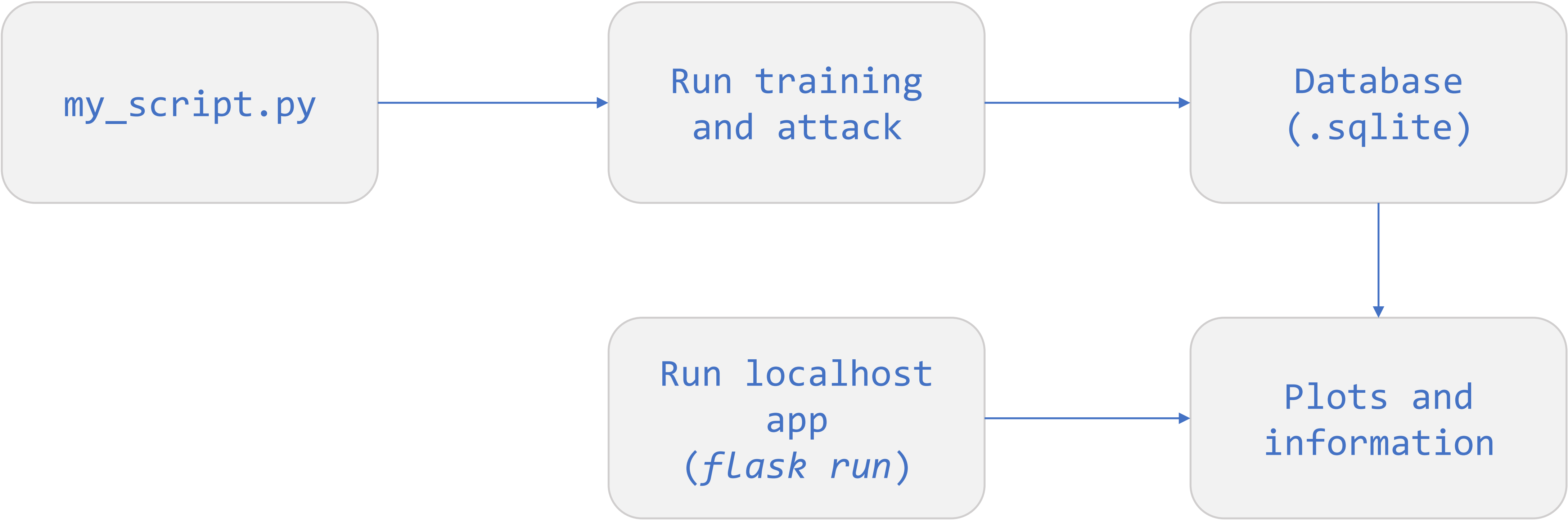
Reason 1: Easy to use

AISY Framework allows very easy execution of deep learning in profiled side-channel attacks. Here is an example of all the code that is needed to run a profiled SCA attack on key byte 2 of an AES implementation from well-known ASCAD dabatase:

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
```

Framework Structure



Open-source framework limitations

- AES128 (demonstration only)
- MLPs and CNNs (users may add other topologies)
- Limited functionalities (e.g., no custom loss function, to be added in future)
- You may find bugs and errors.

Keras

```
model.compile(...)
```

```
model.fit(
      ...,
      callbacks=my_callbacks)
```

```
model.fitgenerator(...)
```

AISY Framework

Custom metrics

Custom callbacks

Custom data augmentation

1) Simple script to attack one AES key byte

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)
aisy.run()
```

1) Simple script to attack one AES key byte

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)
aisy.run()
```

Path definitions (Unchanged)

1) Simple script to attack one AES key byte

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)
aisy.run()
```

Database filename

1) Simple script to attack one AES key byte

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)
aisy.run()
```

Dataset filename

1) Simple script to attack one AES key byte

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)
aisy.run()
```

Leakage Function (Labels)

1) Simple script to attack one AES key byte

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)
aisy.run()
```

Training settings

1) Simple script to attack one AES key byte

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)
aisy.run()
```

Neural Network

2) Visualization

```
import aisy_sca
from app import *
from custom.custom_models.neural_networks import *

aisy = aisy_sca.Aisy()
aisy.set_resources_root_folder(resources_root_folder)
aisy.set_database_root_folder(databases_root_folder)
aisy.set_datasets_root_folder(datasets_root_folder)
aisy.set_database_name("database_ascad.sqlite")
aisy.set_dataset(datasets_dict["ascad-variable.h5"])
aisy.set_aes_leakage_model(leakage_model="HW", byte=2)
aisy.set_batch_size(400)
aisy.set_epochs(20)
aisy.set_neural_network(mlp)

aisy.run(visualization=[4000])
```

Thank you!

g.perin@tudelft.nl