

# Deep Learning-based Side-channel Analysis: Challenges and Opportunities

Stjepan Picek

End-of-ESCADA Workshop, August 26, 2024

# Outline

- 1 Side-channel Analysis
- 2 Developments in the Last Eight Years (for AES)
- 3 What About ASCON?
- 4 What About Xoodyak?
- 5 Conclusions

# Outline

- 1 Side-channel Analysis
- 2 Developments in the Last Eight Years (for AES)
- 3 What About ASCON?
- 4 What About Xoodyak?
- 5 Conclusions

# Cryptographic Theory vs Physical Reality

- Cryptographic algorithms are (supposed to be) theoretically secure.
- Implementations leak in physical world.

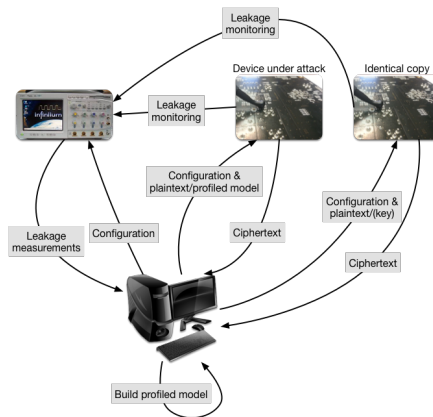
## Implementation attacks

Implementation attacks do not aim at the weaknesses of the algorithm but at its implementation.

# Profiling Attacks

- Profiling attacks have a prominent place as the most powerful among side-channel attacks.
- Within profiling phase the adversary estimates leakage models for targeted intermediate computations, which are then exploited to extract secret information in the actual attack phase.
- Template Attack (TA) is the most powerful attack from the information theoretic point of view.
- Some **machine learning** (ML) techniques also belong to the profiling attacks.

# Profiling Attacks



- Profiling attacks are more complicated than the direct attacks.
- The attacker must have a copy of the device to be attacked.

# Common “Traditional” Approaches in Profiling SCA

- Multilayer Perceptron.
- Naive Bayes.
- Support Vector Machines.
- Random Forest.

# Deep Learning

- Stacked neural networks, i.e., networks consisting of multiple layers.
- Layers are made of nodes.

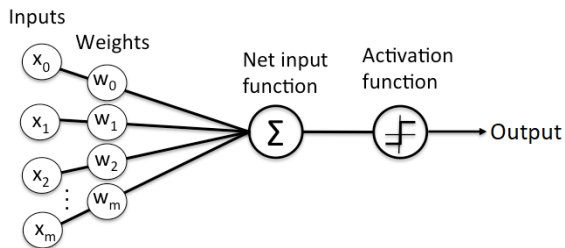


Figure: Perceptron.

# Multilayer Perceptron

- One input layer, one output layer, at least one hidden layer.

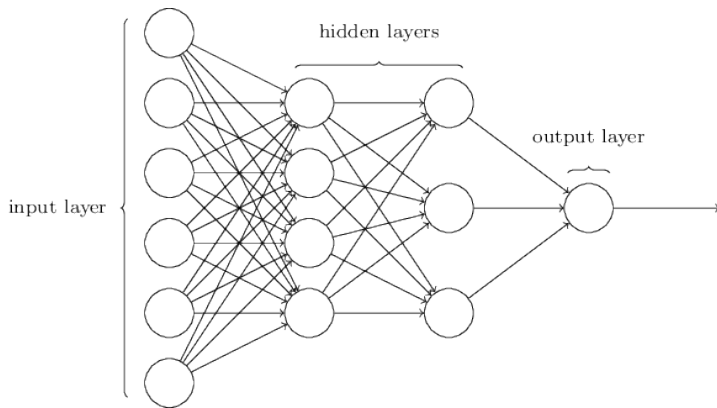


Figure: Multilayer perceptron.

# Convolutional Neural Networks

- CNNs represent a type of neural network first designed for 2-dimensional convolutions.
- They are primarily used for image classification, but lately, they have proven to be powerful classifiers in other domains.
- From the operational perspective, CNNs are similar to ordinary neural networks: they consist of a number of layers where each layer is made up of neurons.
- CNNs use three main types of layers: convolutional layers, pooling layers, and fully-connected layers.

## Other Neural Network Architectures in SCA

- Autoencoder.
- Recurrent neural network.
- Residual neural network.
- Generative Adversarial Network.
- Transformers.

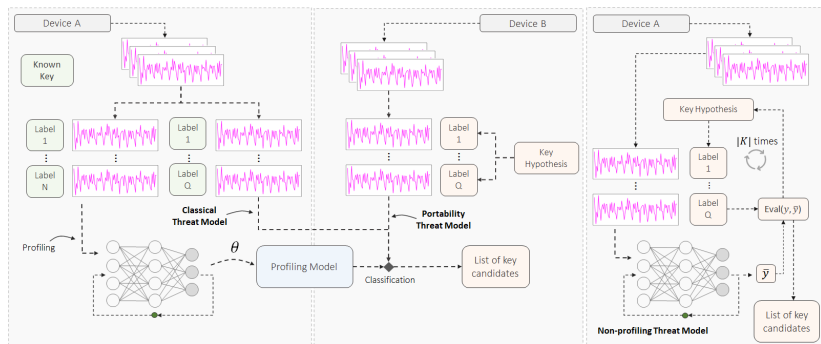
# Outline

- 1 Side-channel Analysis
- 2 Developments in the Last Eight Years (for AES)
- 3 What About ASCON?
- 4 What About Xoodyak?
- 5 Conclusions

# Datasets

| Dataset                              | Platform             | Traces (Features)   | Keys                       | Implementation | Countermeasures                      |
|--------------------------------------|----------------------|---------------------|----------------------------|----------------|--------------------------------------|
| DPAv2 [100] (2010)                   | SASEBO GII (FPGA)    | 100 000 (3 253)     | 1 fixed key                | AES128         | None                                 |
| DPAv4 [101] (2013)                   | Atmega (Software)    | 100 000 (435 000)   | 1 fixed key                | AES256         | First-order masking (RSM)            |
| DPAv4.2 [102] (2014)                 | Atmega (Software)    | 80 000 (1 704 400)  | 16 different keys          | AES128         | First-order masking (RSM)            |
| AES_HD [95] (2018)                   | SASEBO GII (FPGA)    | 50 000 (1 250)      | 1 fixed key                | AES128         | None                                 |
| AES_HD_MM [30] (2014)                | SASEBO GII (FPGA)    | 5 600 000 (3 125)   | 1 fixed key                | AES128         | BM + Affine Masking                  |
| AES_RD [25] (2009)                   | Atmel AVR (Software) | 50 000 (3 500)      | 1 fixed key                | AES128         | Hiding (Random Delay Interrupt)      |
| ASCADf [84] (2018)                   | Atmega (Software)    | 60 000 (100 000)    | 1 fixed key                | AES128         | First-order BM (XOR)                 |
| ASCADv1 [84] (2018)                  | Atmega (Software)    | 300 000 (250 000)   | Random keys + 1 fixed key  | AES128         | First-order BM (XOR)                 |
| ASCADv2 [2] (2021)                   | STM32 (Software)     | 810 000 (1 000 000) | Random + 1 fixed key       | AES128         | Second-order BM + Hiding (Shuffling) |
| CHES_CTF 2018 [92] (2018)            | STM32 (Software)     | 42 000 (650 000)    | Random Keys + 3 fixed keys | AES128         | First-order BM (XOR)                 |
| CHES_CTF 2020 [7] (2020)             | Software/Hardware    | †                   | Fixed/random               | Clyde128       | ISW masking ★                        |
| Portability [10] (2020)              | Atmega (Software)    | 50 000 (600)        | 4 fixed keys               | AES128         | None                                 |
| Ed25519 (WolFSSL) [113] (2019)       | STM32 (Software)     | 6 400 (1 000)       | Random Ephemeral keys      | EdDSA          | None                                 |
| Curve25519 ( $\mu$ NaCl) [20] (2020) | STM32 (Software)     | 5 997 (5,500)       | Random Ephemeral keys      | EdDSA          | CSWAP, Coord./Scalar Rand.           |
| Curve25519 [111]                     | STM32 (Software)     | 300 (8 000)         | Random keys                | EdDSA          | CSWAP, Coord./Scalar Rand.           |
| Curve25519 [111]                     | STM32 (Software)     | 300 (1 000)         | Random keys                | EdDSA          | CSPOINTER, Coord./Scalar Rand.       |

# Threat Models

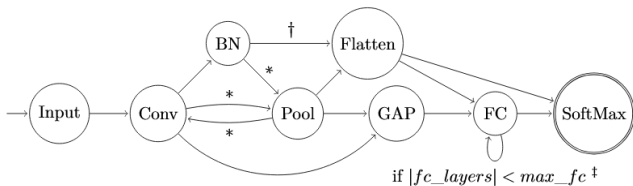


# Hyperparameter Tuning

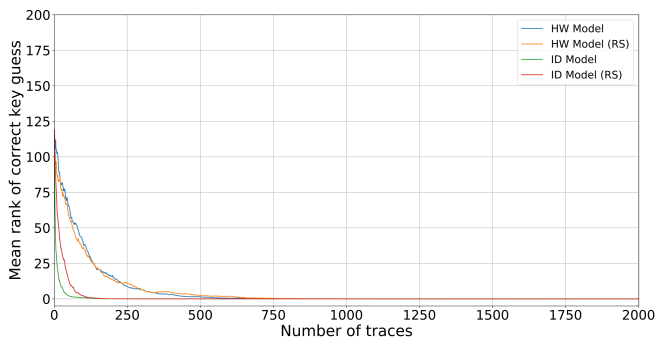
- Hyperparameter tuning is extremely important.
- Different algorithms have different hyperparameters.
- General design principles.
- Methodologies.
- Random search.
- Grid search.
- Advanced techniques.

# Reinforcement Learning

- Reinforcement learning attempts to teach an agent how to perform a task by letting the agent experiment and experience the environment, maximizing some reward signal.
- <https://github.com/AISyLab/Reinforcement-Learning-for-SCA>



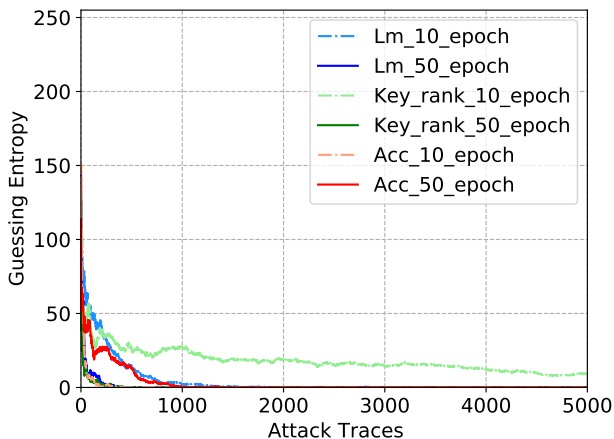
# Reinforcement Learning



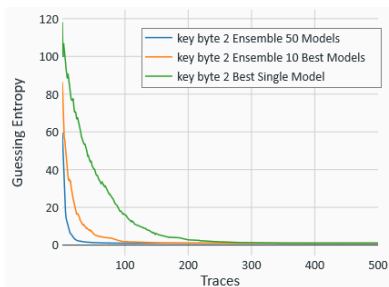
# Bayesian Optimization

- In Bayesian optimization, the aim is to build a probabilistic model of the underlying function.
- We first require a probabilistic model of a function (often referred to as the surrogate model), where there are several ways to model it.
- Second, we require an acquisition function for Bayesian optimization to generate the next neural network architecture to observe, i.e., to select what point to sample next.

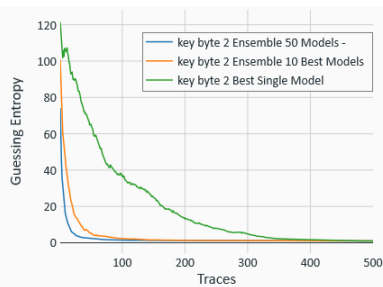
# Bayesian Optimization



# Deep Learning Ensembles



(a) MLP results.



(b) CNN results.

Figure: Guessing entropy for ASCAD for the Hamming weight leakage model.

## Feature Selection for Deep Learning SCA

| Scenario | Knowledge of $r$ mask share | POI selection and pre-processing                                                          | Noisy/non-leaking samples |
|----------|-----------------------------|-------------------------------------------------------------------------------------------|---------------------------|
| RPOI     | Yes                         | Main SNR peaks of $r$ and $s_r$ .<br>No pre-processing required.                          | No                        |
| OPOI     | Yes                         | Minimum trace interval including SNR peaks of $r$ and $s_r$ . No pre-processing required. | Reduced                   |
| NOPOI    | No                          | No POI selection and pre-processing is required.                                          | All available             |

**Table:** Possible feature selection scenarios for deep learning-based SCA with the synchronized measurements.

# Feature Selection for Deep Learning SCA

**Table:** Points of interest, minimum number of attack traces to get guessing entropy equal to 1, model search success (when GE=1), and number of trainable parameters for all datasets and feature selection scenarios.

| Dataset | Neural Network Model | Feature Selection Scenario | Amount of POIs (HW/ID) | Attack Traces (HW/ID) | Search Success (%) (HW/ID) | Trainable Parameters (HW/ID) |
|---------|----------------------|----------------------------|------------------------|-----------------------|----------------------------|------------------------------|
| ASCADf  | MLP                  | RPOI                       | 200/100                | 5/ <b>1</b>           | 99.22%/96.86%              | 82 209/429 256               |
| ASCADf  | CNN                  | RPOI                       | 400/200                | 5/ <b>1</b>           | 99.23%/99.08%              | 499 533/158 108              |
| ASCADf  | MLP                  | OPOI                       | 700/700                | 480/104               | 82.80%/68.80%              | 16 309/10 266                |
| ASCADf  | CNN                  | OPOI                       | 700/700                | 744/87                | 55.53%/35.33%              | 594 305/62 396               |
| ASCADf  | MLP                  | NOPOI                      | 2 500/2 500            | 7/ <b>1</b>           | 74.50%/39.00%              | 2 203 009/5 379 256          |
| ASCADf  | CNN                  | NOPOI                      | 10 000/10 000          | 7/ <b>1</b>           | 15.40%/2.45%               | 545 693/439 348              |
| ASCADf  | CNN                  | NOPOI desync               | 10 000/10 000          | 532/36                | 2.44%/2.64%                | 268 433/64 002               |
| ASCADr  | MLP                  | RPOI                       | 200/20                 | 3/ <b>1</b>           | 99.23%/100%                | 565 209/639 756              |
| ASCADr  | CNN                  | RPOI                       | 400/30                 | 5/ <b>1</b>           | 100%/100%                  | 575 369/636 224              |
| ASCADr  | MLP                  | OPOI                       | 1 400/1 400            | 328/129               | 71.40%/37.25%              | 31 149/34 236                |
| ASCADr  | CNN                  | OPOI                       | 1 400/1 400            | 538/78                | 47.92%/23.95%              | 270 953/87 632               |
| ASCADr  | MLP                  | NOPOI                      | 25 000/25 000          | 6/ <b>1</b>           | 44.39%/7.02%               | 5 243 209/12 628 756         |
| ASCADr  | CNN                  | NOPOI                      | 25 000/25 000          | 7/ <b>1</b>           | 19.17%/4.35%               | 369 109/721 012              |
| ASCADr  | CNN                  | NOPOI desync               | 25 000/25 000          | 305/73                | 0.71%/1.04%                | 22 889/90 368                |

## Plaintext/Ciphertext-based Non-profiling SCA

- Supervised deep learning-based SCA learns a mapping based on known plaintexts and keys.
- Then, the adversary estimates the conditional probability given a leakage trace with the unknown key.
- In unsupervised setting, we do not know the key.
- But, the key is commonly fixed for all traces.

# Plaintext/Ciphertext-based Non-profiling SCA

- Supervised deep learning-based SCA learns a mapping based on known plaintexts and keys.
- Then, the adversary estimates the conditional probability given a leakage trace with the unknown key.
- In unsupervised setting, we do not know the key.
- But, the key is commonly fixed for all traces.
- The label  $l(k, d_i)$  and  $d_i$  would satisfy:

$$d_i \mapsto l(k, d_i). \quad (1)$$

$$l(k, d_i) = \text{map}_k(d_i), \quad (2)$$

# Plaintext/Ciphertext-based Non-profiling SCA

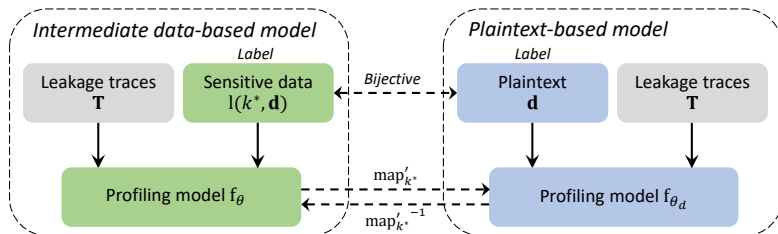


**Figure:** The relationship between intermediate data-based model and plaintext-based model.

# Plaintext/Ciphertext-based Non-profiling SCA

- For supervised DLSCA, if a profiling model is generalized well on the leakage traces, the probability of the incorrect value is closely correlated with the correct label.
- In unsupervised setting, we can still estimate the label distance, providing us with plaintext/ciphertext distribution.

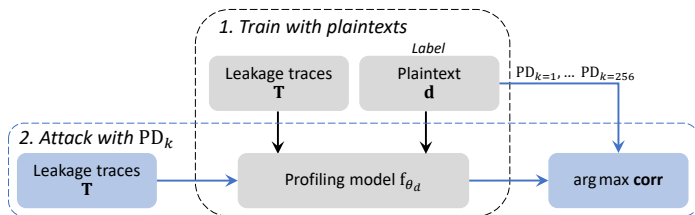


Figure: Attack scheme of the Plaintext Labeling Deep Learning (PLDL).

$$k^* = \arg \max_k \text{corr}(PD_k^M(\mathbf{d}), f_{\theta_d}(\mathbf{T})), \quad k \in \mathcal{K}. \quad (3)$$

# Plaintext/Ciphertext-based Non-profiling SCA

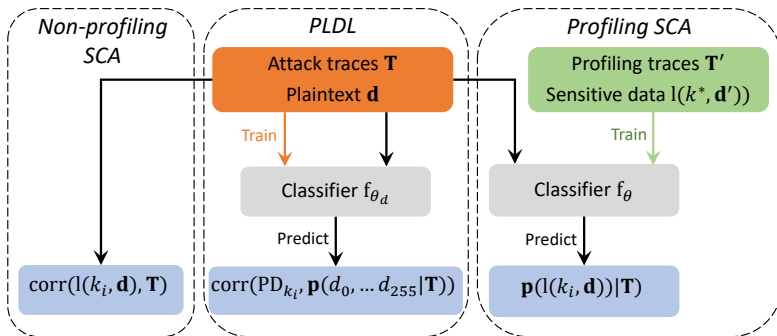


**Figure:** A demonstration of non-profiling SCA, PLDL, and profiling SCA.

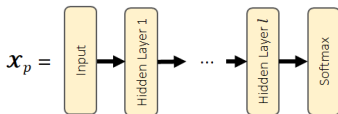
# Plaintext/Ciphertext-based Non-profiling SCA

**Table:** Performance benchmark with non-profiling attacks.

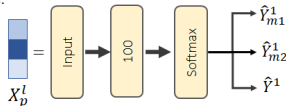
| Dataset  | CPA         | MOR        | DDLA        | PLDL             |
|----------|-------------|------------|-------------|------------------|
| ASCAD_F  | KR161/KR47  | 1 957/638  | KR7/309     | <b>8/111</b>     |
| ASCAD_R  | KR64/KR8    | KR28/KR9   | 27 266/KR48 | <b>20/19</b>     |
| CHES_CTF | KR139/KR220 | KR6/KR31   | KR54/KR85   | <b>6 121/KR2</b> |
| AES_RD   | KR2/KR31    | KR33/3 112 | 2 541/KR2   | <b>1/57</b>      |
| AES_HD   | KR19/KR145  | 5 593/KR10 | KR26/KR20   | <b>60/KR6</b>    |

# Explainability Approach

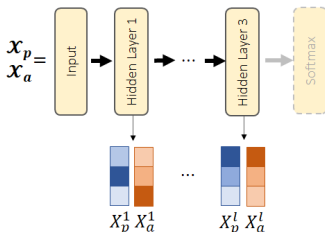
1. Train baseline model for  $E$  epochs:



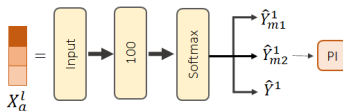
3. Add a Softmax layer to each encoded profiling dataset  $X_p^l$  (for each layer  $l$ ) and train new softmax weights for 5 epochs:



2. At training epoch  $e$ , Get layer activations by predicting profiling ( $x_p$ ) and attack sets ( $x_a$ ):



4. Predict each encoded attack set  $X_a^l$  (for each layer  $l$ ) and compute Perceived Information (PI):



# Explainability Approach

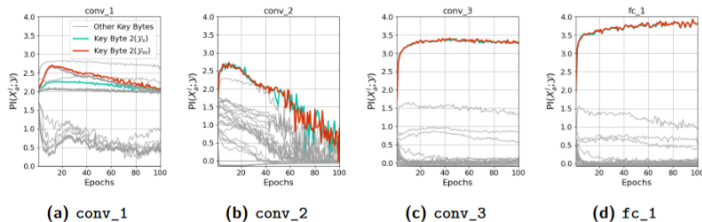


**Figure :** The compression of irrelevant features with CNN. Red and green lines indicate the perceived information from relevant features, which are the mask  $\mathcal{Y}_m[2]$  and the masked S-Box output  $\mathcal{Y}_s[2]$  labels related to key byte 2. Irrelevant features associated with the rest of the key bytes are illustrated as gray lines.

# AI Explainability and SCA

- **Where.** Our empirical results indicate that compression of  $\mathcal{X}$  mostly happens in the first hidden layer in any MLP configuration. Generalization to  $\mathcal{Y}$  is stronger in hidden layers closer to the output layer, and this conclusion comes from higher  $\widehat{PI}(X_a^l; \mathcal{Y})$  values obtained for the outer layer in comparison to hidden layers closer to the input layer.
- **What.** We verified that to generalize to  $\mathcal{Y}$ , the first hidden layer compresses noise and irrelevant features and transmits information from relevant secret shares to the subsequent hidden layers. This also suggests that hidden layers perform unmasking by combining the two secret shares.

## AI Explainability and SCA

- **Why.** Our analyses indicate that MLP implements IB theory as the bottleneck is usually implemented already by the first hidden layer. In essence, all intermediate network representations follow the IB theory, but the bottleneck (compression of irrelevant features) is usually more evident in the first layer.

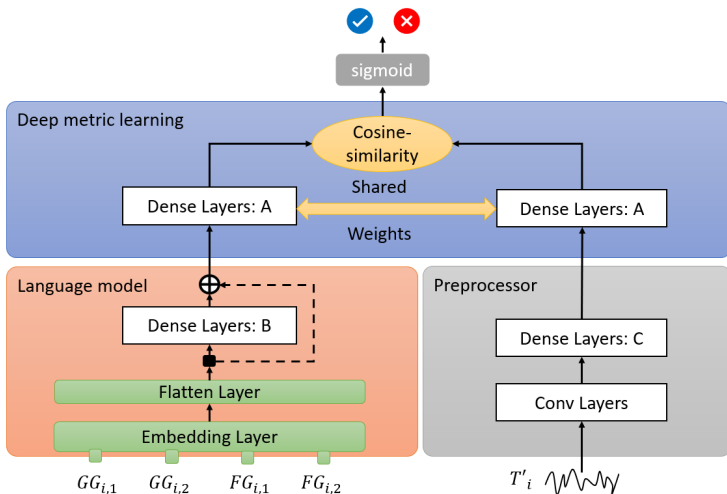
# LLM Concepts

- In NLP, a language model offers a way to assign probabilities to sequences of words or characters.
- An  $n$ -gram is a contiguous sequence of  $n$  items from a given text, where an “item” can be a word, character, or any other unit of text.
- Integer tokenization, or indexing, is a critical NLP step that assigns unique numerical identifiers to each token obtained during tokenization.
- A language model uses the frequency of  $n$ -grams in a corpus of text to predict the next word.

# LLM for DLSCA

- For SCA, the measurement can be labeled in many ways, like plaintext, ciphertext, key, masks, or output from an S-Box.
- These labels can be categorized into two groups: data tokens or intermediate values.
- Unlike NLP sentences with arbitrary lengths and words occurring in any position within an n-gram, these SCA sentences have two unique properties.
- First, they have a fixed length.
- Second, a word (i.e., a value for some data token or intermediate value) occurs in a fixed position based on the data token or intermediate value it represents.

## DLSCA BLM



## DLSCA BLM

| Dataset       | $Acc$               |                       |                     | $T_{GE0}$           |                       |                     |
|---------------|---------------------|-----------------------|---------------------|---------------------|-----------------------|---------------------|
|               | $\mathcal{M}_{Stn}$ | $\mathcal{M}_{MTask}$ | $\mathcal{M}_{OvC}$ | $\mathcal{M}_{Stn}$ | $\mathcal{M}_{MTask}$ | $\mathcal{M}_{OvC}$ |
| ASCAD-V1-FK   | 0.009               | NA                    | 1.000               | 87 [PWP22]          | NA                    | 1                   |
| ASCAD-V1-VK   | 0.009               | NA                    | 0.721               | 78 [WPP22]          | NA                    | 61                  |
| ASCAD-V1-FK-F | 0.972               | NA                    | 1.000               | 1 [PWP22]           | NA                    | 1                   |
| ASCAD-V1-VK-F | 1.000               | NA                    | 1.000               | 1 [PWP22]           | NA                    | 1                   |
| ASCAD-V2      | 0.004               | 0.016/*               | 0.742               | ✗ [MS23]            | 60 [MS23]             | 47                  |

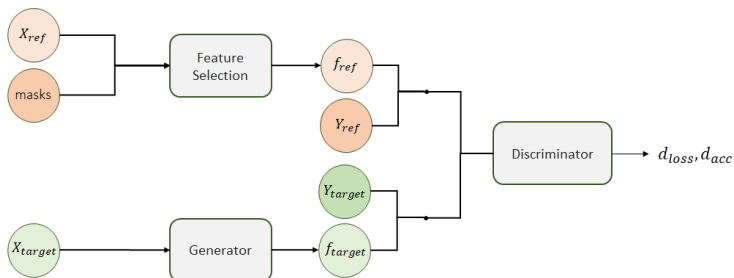
\* = (0.992, 0.211, 0.889): accuracies for ( $r_m, \beta, p[i]$ ) respectively as reported in [MS23].

NA: results are not available

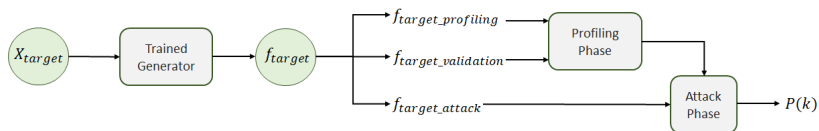
## From Black Box to White Box

- The adversary possesses a similar implementation that can be used as a white-box reference design.
- We create an adversarial dataset by extracting features or points of interest from this reference design.
- These features are then utilized for training a novel conditional generative adversarial network (CGAN) framework, enabling a generative model to extract features from high-order leakages in protected implementation without any assumptions about the masking scheme or secret masks.

# CGAN Framework



# CGAN Profiling and Attack



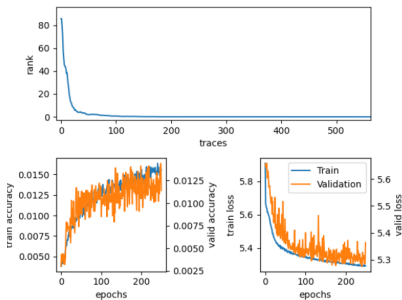
# Outline

- 1 Side-channel Analysis
- 2 Developments in the Last Eight Years (for AES)
- 3 What About ASCON?**
- 4 What About Xoodyak?
- 5 Conclusions

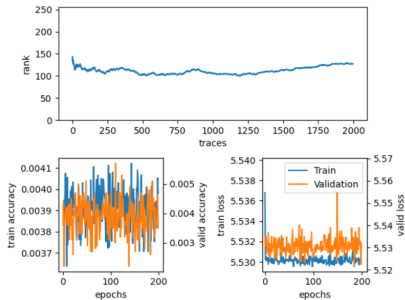
# ASCON

- The target of side-channel attacks against Ascon is the value of the key used in the initialization phase.
- For masked implementation, we consider masked Toffoli gates for the Chi function inside the S-box with the addition of re-used randomness that is cyclically refreshed at each call.
- An additional SCA countermeasure is set to rotate the offsets between the shares.
- While the Ascon S-box is bit-sliced, the output leakage can be stronger for bytes of the output state when considering a software implementation because of the architecture of the microprocessor and leakage during the storing operation.

# First Results

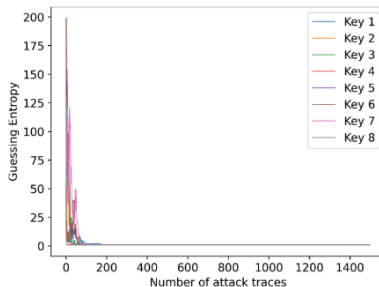


Reference implementation.

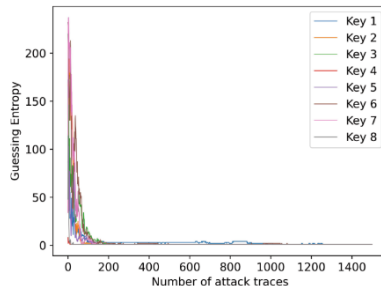


Protected implementation.

# Results with Ensembles

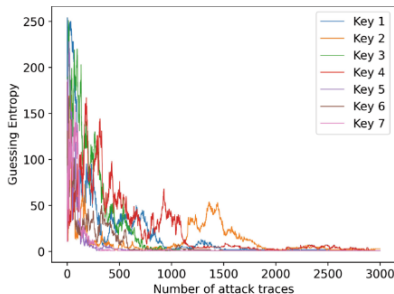


Ensemble of five MLPs.

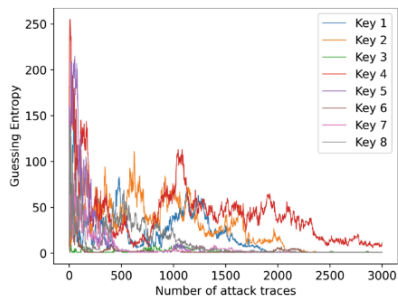


Ensemble of five CNNs.

# Results with Ensembles



Ensemble of five MLPs.



Ensemble of five CNNs.

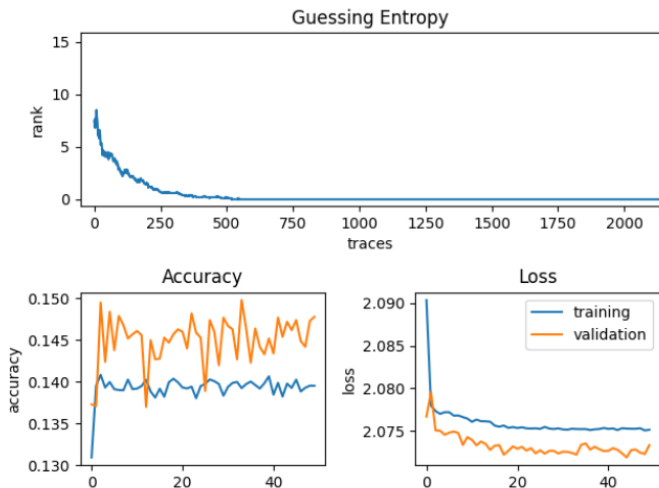
# Outline

- 1 Side-channel Analysis
- 2 Developments in the Last Eight Years (for AES)
- 3 What About ASCON?
- 4 What About Xoodoo?**
- 5 Conclusions

# Xoodoo

- Focus on the initialization phase of Xoodoo in the authenticated encryption with associated data (AEAD) mode.
- Sakura-G FPGA board equipped with a Xilinx Spartan6 XC6SLX25 FPGA.
- Unprotected implementation.

# First Results



# Outline

- 1 Side-channel Analysis
- 2 Developments in the Last Eight Years (for AES)
- 3 What About ASCON?
- 4 What About Xoodyak?
- 5 Conclusions**

# Conclusions

- Deep learning is efficient and powerful option for SCA.
- Current results are very promising as we can break protected targets even with very small architectures.
- What is less clear is how would the approach scale for more realistic targets (or different ciphers).
- It is not easy to select the most promising approaches from all that is available.
- The big challenges are unsupervised deep learning-based SCA and explainable AI for SCA.
- Not so much results besides AES.