A Comparison of Deep Learning Approaches for Power-based Side-channel Attacks

Roberto Capoferri, Alessandro Barenghi, Luca Breveglieri, Niccolò Izzo, Gerardo Pelosi

November 6th, 2024

POLITECNICO
MILANO 1863

Outline

POLITECNICO
MILANO 1863

Introduction

Side-channel attacks (SCAs) are a very relevant threat to the devices that offer cryptographic operations, even modern ones [9, 17].
SCAs exploit *information leakage* from the target to extract secret information.

Deep learning applied to SCA: overview

1. collection of *traces* from the target device
2. build the Neural Network (NN) which takes traces as input and produces a probability distribution over the keyspace
   2.1 tune the model hyperparameters
   2.2 training on the whole dataset: a classification task on target intermediate
3. testing: determine the *Guessing Entropy* (GE) on a different set of traces

Details of all the points in the methodology section.

POLITECNICO
MILANO 1863

Objectives

Provide a systematic comparison of the effects of:

- single/multiple devices to determine *portability*
- fixed/random key traces
- different target intermediates
- usage of plaintext information

Find the combination with the best attack performance.

Outline

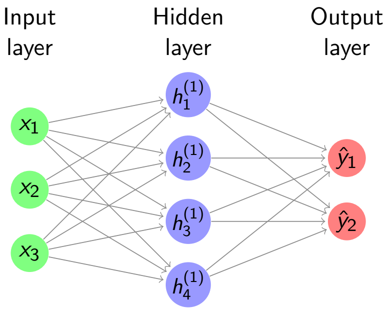
POLITECNICO
MILANO 1863

Background – Deep Learning (DL)

Branch of machine learning that focuses on neural networks.

We consider the Multi-Layer Perceptron (MLP), composed by multiple, fully connected layers of neurons:

- **input layer**: each neuron takes an input feature
- one or more **hidden layers**: transform the input features through some weight matrices and non-linear activations
- **output layer**: the number and the role of each neuron depend on the task



POLITECNICO
MILANO 1863

Background – Side-channel attacks (SCAs)

During cryptographic operations, collect *traces* of:

- execution time
- power consumption [7]
- electromagnetic emission [13]

Build a *leakage function* that correlates such measurements with the secret key.

- **non-profiled SCA**: measurements done on the target device, extract key by using statistical analysis (e.g., DPA [8], CPA [3])
- **profiled SCA**: collect traces from a clone device to build a model which is then used to attack the target (e.g., template attack [4])

Background – DL applied to SCAs

It's a type of profiled attack, can outperform traditional methods [10] and has some advantages:

- automatic feature extraction
- no hypotheses on noise distribution

The profiling phase (a.k.a. *training phase*) can be completely automated.
After the network is trained, only a few traces are needed to perform the attack.

POLITECNICO
MILANO 1863

Outline

POLITECNICO
MILANO 1863

Methodology

Evaluate different approaches to dataset collection and training:

- **target**: observe the effect of changing the target intermediate
  - `SBOX_OUT`: targets the output of the SBox
  - `HW_SO`: targets the hamming weight of the SBox output
- **multi-device**: check portability problem [2] and performance of single-device vs multi-device training in different scenarios
- **plaintext**: add plaintext information to raw trace data as input to MLP

All code available [16]

Methodology – Data collection

Traces captured on a Riscure Pinata, based on an STM32F4 (Cortex-M4) 32-bit microcontroller running at 168MHz.

Board instrumentation

- **current probe** to measure power consumption
- **serial connection** to handle communication
- **trigger probe** to start the capture on the scope

Oscilloscope: Tektronik MSO58, 625MS/s, 8 bit resolution

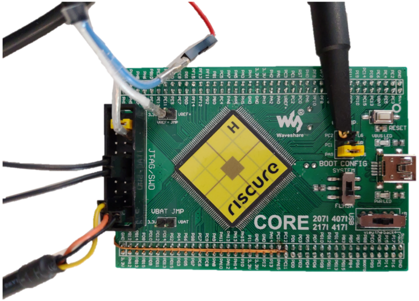


Figure: Capture setup

POLITECNICO
MILANO 1863

Methodology – Data collection

Traces are captured from **three different devices** running a software implementation of AES128.

Capture SBox computation

1. start AES computation
2. raise GPIO pin before *SBox* operation
3. lower GPIO pin before *MixColumns* operation

Dataset publicly available for further studies [14, 15]

For each device

- 200,000 traces collected with a fixed key (`key 0`)
- 200,000 traces collected with a random key
- 30,000 traces collected with a different fixed key (`key 1`), used for testing

POLITECNICO
MILANO 1863

Methodology – Hyperparameter tuning

Explore different MLP structures to get the best model.

Genetic algorithm [5]

Start by selecting random parameters:

1. train model
2. select top performing
3. produce offspring
4. iterate

Table: Hyperparameter space

parameter	possible values
hidden layers	[1, 2, 3, 4, 5]
hidden neurons	[100, 200, 300, 400, 500]
dropout rate	[0.0, 0.1, 0.2, 0.3]
l2	$[0.0, 5\cdot10^{-2}, 1\cdot10^{-2}, 5\cdot10^{-3}, 1\cdot10^{-3}, 5\cdot10^{-4}, 1\cdot10^{-4}]$
optimizer	['adam', 'rmsprop', 'sgd']
learning rate	$[5\cdot10^{-3}, 1\cdot10^{-3}, 5\cdot10^{-4}, 1\cdot10^{-4}, 5\cdot10^{-5}, 1\cdot10^{-5}]$
batch size	[128, 256, 512, 1024]

Table: Parameters for the genetic algorithm

nGen	popSize	selPerc	scProb	mProb
20	15	30%	20%	20%

POLITECNICO MILANO 1863

Methodology – Training and performance evaluation

Use a 90/10 training-validation split and rescale input:

$$X_{\texttt{scaled}} = (X - \mathbf{min}(X)) \ / \ (\mathbf{max}(X) - \mathbf{min}(X))$$

Use of regularization options to improve model performance

- **Dropout:** which percentage of neurons to turn off in a layer during training.
- **L2 regularization**: limits the value of the weights by adding a penalty to larger values.
- **Batch normalization** [6]: normalizes the input to each layer, stabilizing the training process.
- **Early stopping** [12]: monitors training and validation accuracy, and stops the training process if overfitting is detected

Evaluation metric: Guessing Entropy [18]

- rank of the correct key byte over an increasing number of test traces
- average over multiple runs

POLITECNICO
MILANO 1863

Outline

POLITECNICO
MILANO 1863

Results

Performance

- measured using the GE
- graphs show evolution of the position of the correct key when incrementing the number of attack traces
- tested from 1 to 300 attack traces
- average over 100 runs on different traces (use up to all $30,000$ test traces)

Scenarios

- attack the same device(s) as the training one(s)
- attack a different device

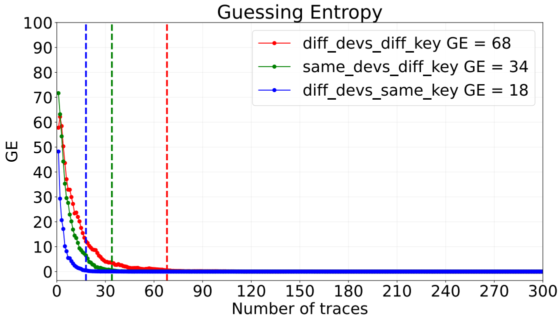


Figure: Fixed key, 1 device w/ ptx, `HW_SO`. Vertical lines where $GE < 0.5$

POLITECNICO
MILANO 1863

Results – Plaintext

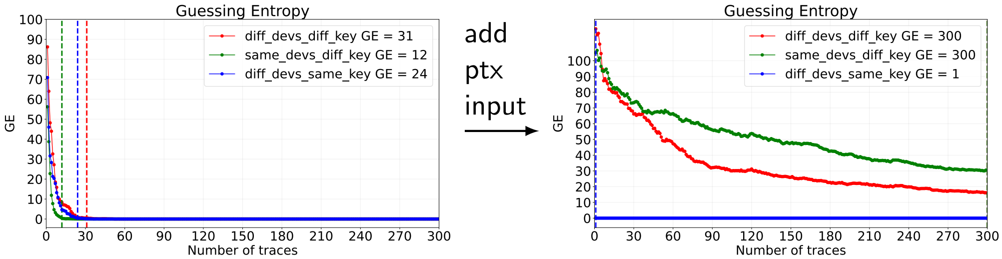


Figure: Effect of ptx on model performance using `SBOX_OUT`

Fixed key scenario

- adding plaintext information breaks the model when considering `SBOX_OUT`
- for `HW_SO` the model still works but performance is works
- probably due to overfitting on the key

POLITECNICO
MILANO 1863

Results – Plaintext

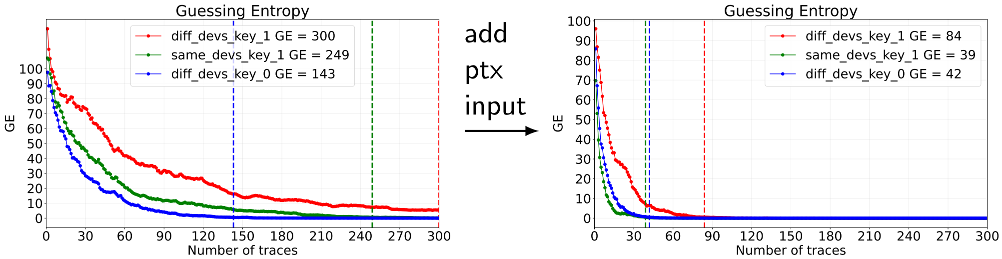


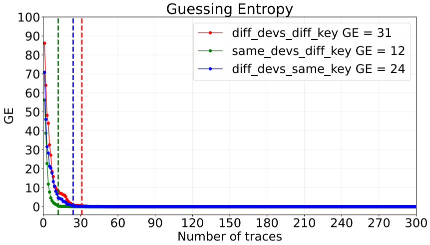
Figure: Effect of ptx on model performance using `SBOX_OUT`

Random key scenario

- in this case performance is improved
- similar behaviour with both `SBOX_OUT` and `HW_SO` targets
- also applies when considering multiple training devices

POLITECNICO
MILANO 1863

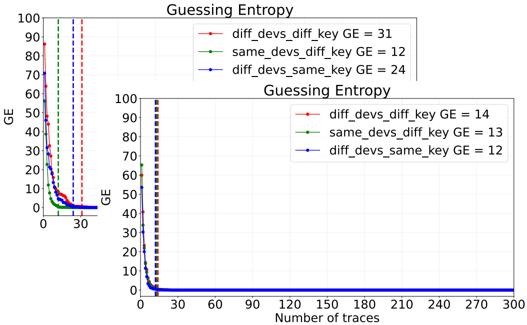
Results – Multi-device



Fixed key GE with 1 training device.

POLITECNICO
MILANO 1863

Results – Multi-device

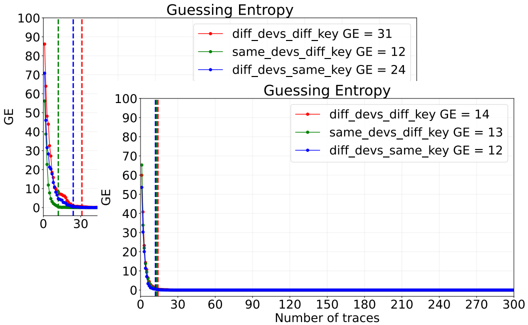


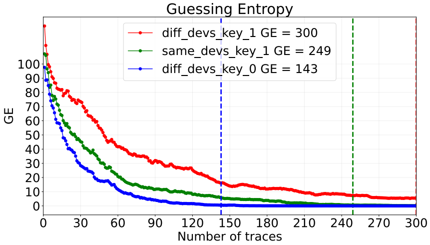
Multi-device

- improves performance in general
- reduces performance loss when considering portability

POLITECNICO
MILANO 1863

Results – Multi-device





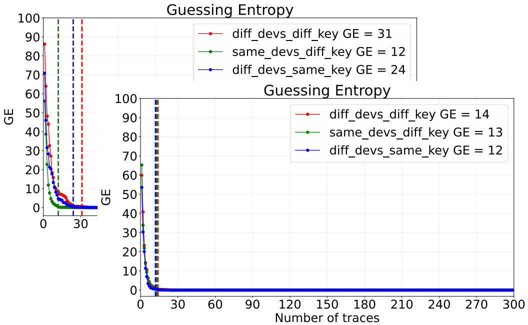
Multi-device

- improves performance in general
- reduces performance loss when considering portability

Random key GE with 1 training device.

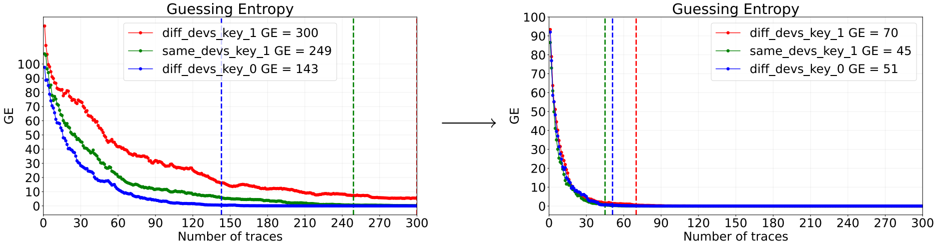
POLITECNICO
MILANO 1863

Results – Multi-device



Multi-device

- improves performance in general
- reduces performance loss when considering portability
- has a greater impact on random key



Random key GE with 1 and 2 training devices.

POLITECNICO
MILANO 1863

Results – Target intermediate

- different behaviour depending on target
- `SBOX_OUT` performs better overall in different situations
- there are cases when `HW_SO` is better (e.g., fixed key dataset w/ ptx)

Reason

Using the `HW_SO` model can induce a class imbalance [11] in the dataset, impacting performance.

Results – Recap

Effect of different parameters:

Plaintext

- for fixed key datasets it breaks the model
- performance increase for random key, but only in `SBOX_OUT` scenario

Data collection

- fixed key works better than random key
- random key can get closer when adding plaintext information

Multi-device and portability

- confirm that model performs worse when tested on new device
- using traces from multiple devices yields a more robust model
- some exception can happen, random key in the `HW_SO` scenario perform worse

POLITECNICO
MILANO 1863

Outline

Concluding remarks

Achieved results

- studied the effect of different parameters for training
- new study on plaintext information
- validate multi-device model for portability
- provide a new dataset to further study portability

Future work

- different implementations
- different points of interest
- different CPU architectures [1]

POLITECNICO
MILANO 1863

Questions?

POLITECNICO
MILANO 1863

References I

[1] Alessandro Barenghi and Gerardo Pelosi. "Side-channel security of superscalar CPUs: evaluating the impact of micro-architectural features". In: *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*. ACM, 2018, 120:1–120:6. DOI: 10.1145/3195970.3196112. URL: https://doi.org/10.1145/3195970.3196112.

[2] Shivam Bhasin et al. "Mind the Portability: A Warriors Guide through Realistic Profiled Side-channel Analysis". In: *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. URL: https://www.ndss-symposium.org/ndss-paper/mind-the-portability-a-warriors-guide-through-realistic-profiled-side-channel-analysis/.

[3] Eric Brier, Christophe Clavier, and Francis Olivier. "Optimal Statistical Power Analysis". In: *IACR Cryptol. ePrint Arch.* (2003), p. 152. URL: http://eprint.iacr.org/2003/152.

[4] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 13–28. DOI: 10.1007/3-540-36400-5_3. URL: https://doi.org/10.1007/3-540-36400-5%5C_3.

[5] Matt Harvey and Norman Heckscher. *Evolve a neural network with a genetic algorithm*. https://github.com/harvitronix/neural-network-genetic-algorithm. 2017.

[6] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[7] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25. URL: https://doi.org/10.1007/3-540-48405-1%5C_25.

[8] Paul C. Kocher et al. "Introduction to differential power analysis". In: *J. Cryptogr. Eng.* 1.1 (2011), pp. 5–27. DOI: 10.1007/S13389-011-0006-Y. URL: https://doi.org/10.1007/s13389-011-0006-y.

[9] Oleksiy Lisovets et al. "Let's Take it Offline: Boosting Brute-Force Attacks on iPhone's User Authentication through SCA". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.3 (2021), pp. 496–519. DOI: 10.46586/TCHES.V2021.I3.496-519. URL: https://doi.org/10.46586/tches.v2021.i3.496-519.

References II

[10] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*. Ed. by Claude Carlet, M. Anwar Hasan, and Vishal Saraswat. Vol. 10076. Lecture Notes in Computer Science. Springer, 2016, pp. 3–26. DOI: 10.1007/978-3-319-49445-6_1. URL: https://doi.org/10.1007/978-3-319-49445-6%5C_1.

[11] Stjepan Picek et al. "The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.1 (2019), pp. 209–237. DOI: 10.13154/TCHES.V2019.I1.209-237. URL: https://doi.org/10.13154/tches.v2019.i1.209-237.

[12] Lutz Prechelt. "Early Stopping - But When?" In: *Neural Networks: Tricks of the Trade - Second Edition*. Ed. by Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller. Vol. 7700. Lecture Notes in Computer Science. Springer, 2012, pp. 53–67. DOI: 10.1007/978-3-642-35289-8_5. URL: https://doi.org/10.1007/978-3-642-35289-8%5C_5.

[13] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards". In: *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*. Ed. by Isabelle Attali and Thomas P. Jensen. Vol. 2140. Lecture Notes in Computer Science. Springer, 2001, pp. 200–210. DOI: 10.1007/3-540-45418-7_17. URL: https://doi.org/10.1007/3-540-45418-7%5C_17.

[14] Roberto Capoferri. *Fixed-key dataset for three riscure Pinata devices*. https://zenodo.org/records/11443025. 2024.

[15] Roberto Capoferri. *Random-key dataset for three riscure Pinata devices*. https://zenodo.org/records/11199202. 2024.

[16] Roberto Capoferri. *Source code and models*. https://github.com/RobertoCapoferri/DLSCA-article. 2024.

[17] Thomas Roche. *EUCLEAK*. Cryptology ePrint Archive, Paper 2024/1380. 2024. URL: https://eprint.iacr.org/2024/1380.

[18] Lichao Wu et al. *On the Attack Evaluation and the Generalization Ability in Profiling Side-channel Analysis*. Cryptology ePrint Archive, Paper 2020/899. 2020. URL: https://eprint.iacr.org/2020/899.

Appendix A - full comparison tables I

Table: Results for the **fixed key** dataset. No. of traces to get $GE < 0.5$

target	variant	n. devices	same_devs_diff_key	diff_devs_same_key	diff_devs_diff_key
SBOX OUT	no ptx	1	12	24	31
		2	13	12	14
	ptx	1	> 300	1	> 300
		2	> 300	1	> 300
HW SO	no ptx	1	20	29	35
		2	18	13	28
	ptx	1	34	18	68
		2	> 300	8	> 300

Appendix A - full comparison tables II

Table: Results for the **random key** dataset. No. of traces to get $GE < 0.5$

target	variant	n. devices	same_devs_key_1	diff_devs_key_0	diff_devs_key_1
SBOX OUT	no ptx	1	249	143	> 300
		2	45	51	70
	ptx	1	39	42	84
		2	20	25	25
HW SO	no ptx	1	165	> 300	> 300
		2	117	70	117
	ptx	1	75	192	188
		2	> 300	> 300	> 300