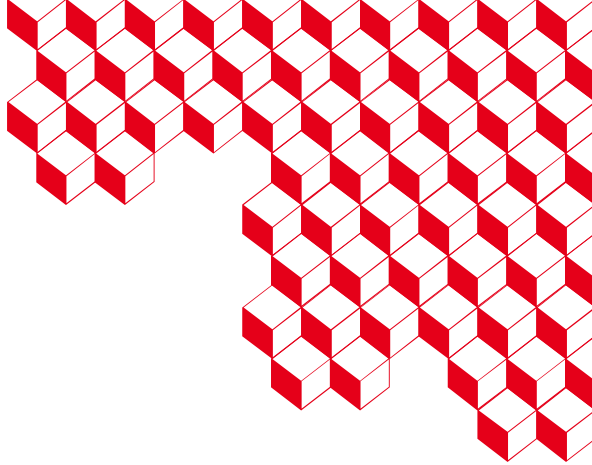
INNOVATION
DÉFENSE
LAB



Keep it Unsupervised: Horizontal Attacks Meet Simple Classifiers

Ninon Calleja Albillos – CEA LETI – France

Sana Boussam – THALES - France

With the guidance of Gabriel Zaid, Ange Martinelli, Eleonora Cagli, Simon Abelard, Antoine Loiseau, Julien Eynard and Guénaël Renault

THALES

Outline

IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2021, No. 1, pp. 343–372.
DOI:10.46586/tches.v2021.i1.343-372

THIS PAPER

Keep it Unsupervised: Horizontal Attacks Meet Simple Classifiers

Sana Boussam[2,3] and Ninon Calleja Albillos[1,4]

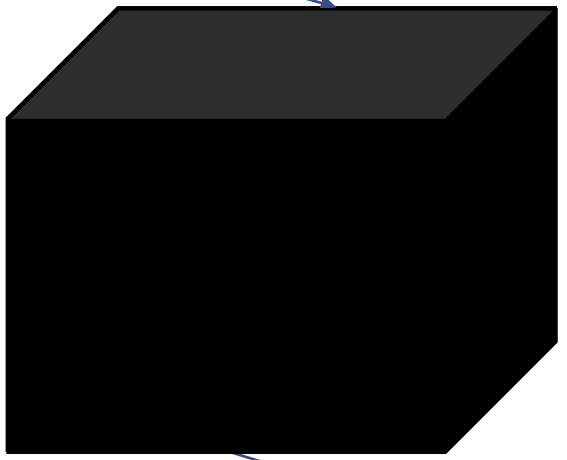
[1] CEA, LETI, MINATEC Campus, F-38054 Grenoble, France
[2] INRIA and LIX, Institut Polytechnique de Paris, Palaiseau, France
[3] Thales ITSEF, Toulouse, France
[4] Univ. Grenoble Alpes, F-38000, Grenoble, France
sana.boussam@inria.fr
ninon.callejaalbillos@cea.fr

When a simple horizontal attack can recover around 90% on average and to 100% for at least one of the attacked key bits, the proposed iterative framework improves the performance and remains fully unsupervised and excludes the need to know where the bits are located in each separate randomized private key.

Keywords: Side-channel Analysis · Public-key Algorithms · Horizontal Attacks · Deep Learning

52% accuracy key recovery

Benefits?
Limitations?
Reproducibility?



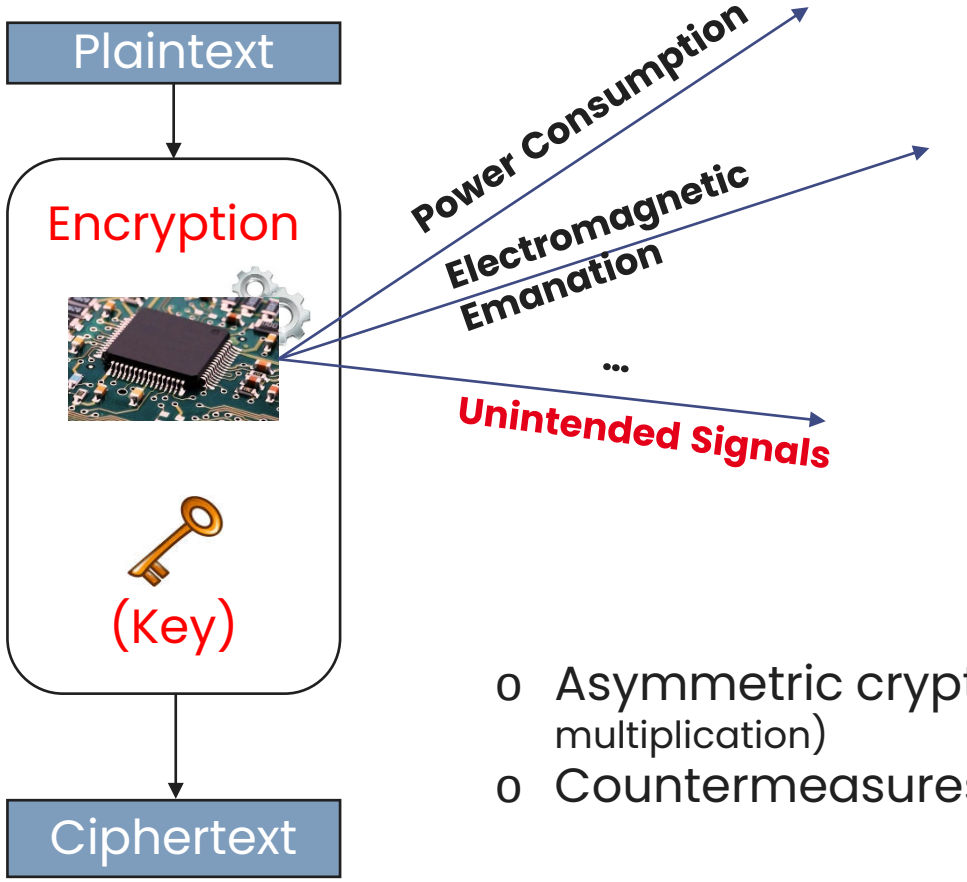
100% accuracy key recovery

Agenda

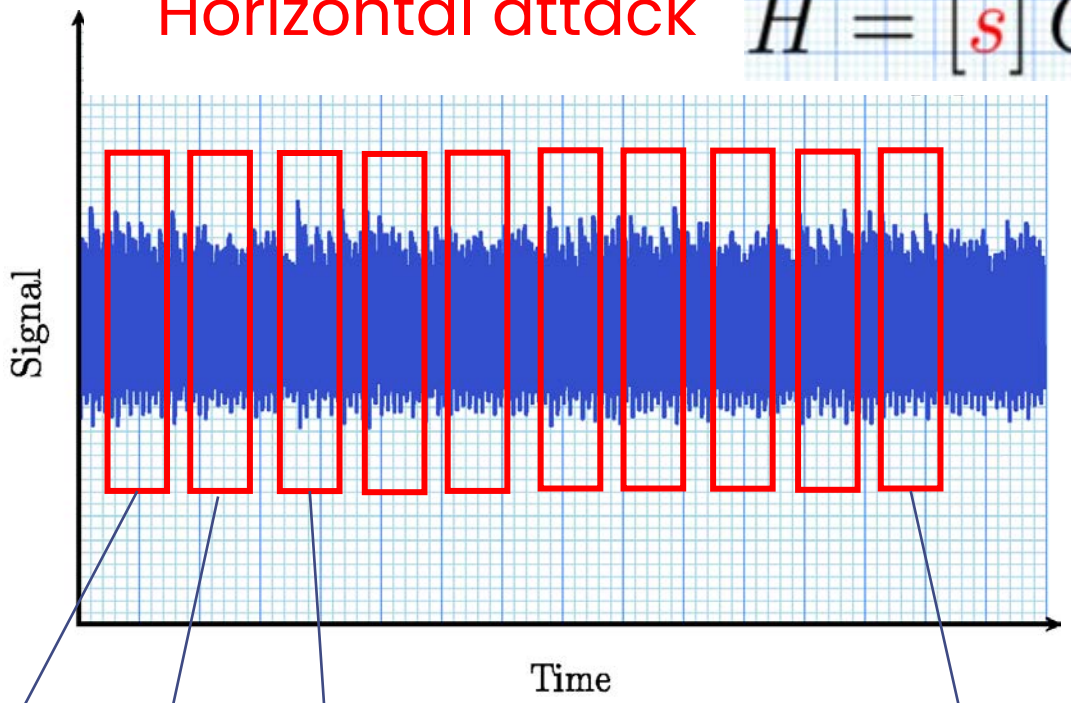
1. **Introduction**
2. **Overview of Perin et al. CHES 2021 [PCBP20]**
3. **In-depth analysis of the iterative framework**
   Simplification of the neural networks
   Correction capability of the iterative framework
4. **Deep Learning Free Approach**
5. **Conclusion**

1. Introduction

Side-Channel Attacks



Horizontal attack $H = [s]G$



- Asymmetric cryptographic primitives (e.g. RSA exponentiation, ECC scalar multiplication)
- Countermeasures (e.g. key randomization, message randomization, ...)

$s_1 = 1$ $s_2 = 1$ $s_3 = 0$ $s_{255} = 1$

CLASSIFICATION (supervised) / CLUSTERING (unsupervised)

2. Overview of Perin et al. CHES 2021 [PCBP20]

Keep it Unsupervised: Horizontal Attacks Meet Deep Learning

Target

Algorithm 1 Montgomery ladder with cswap and coordinate re-randomization

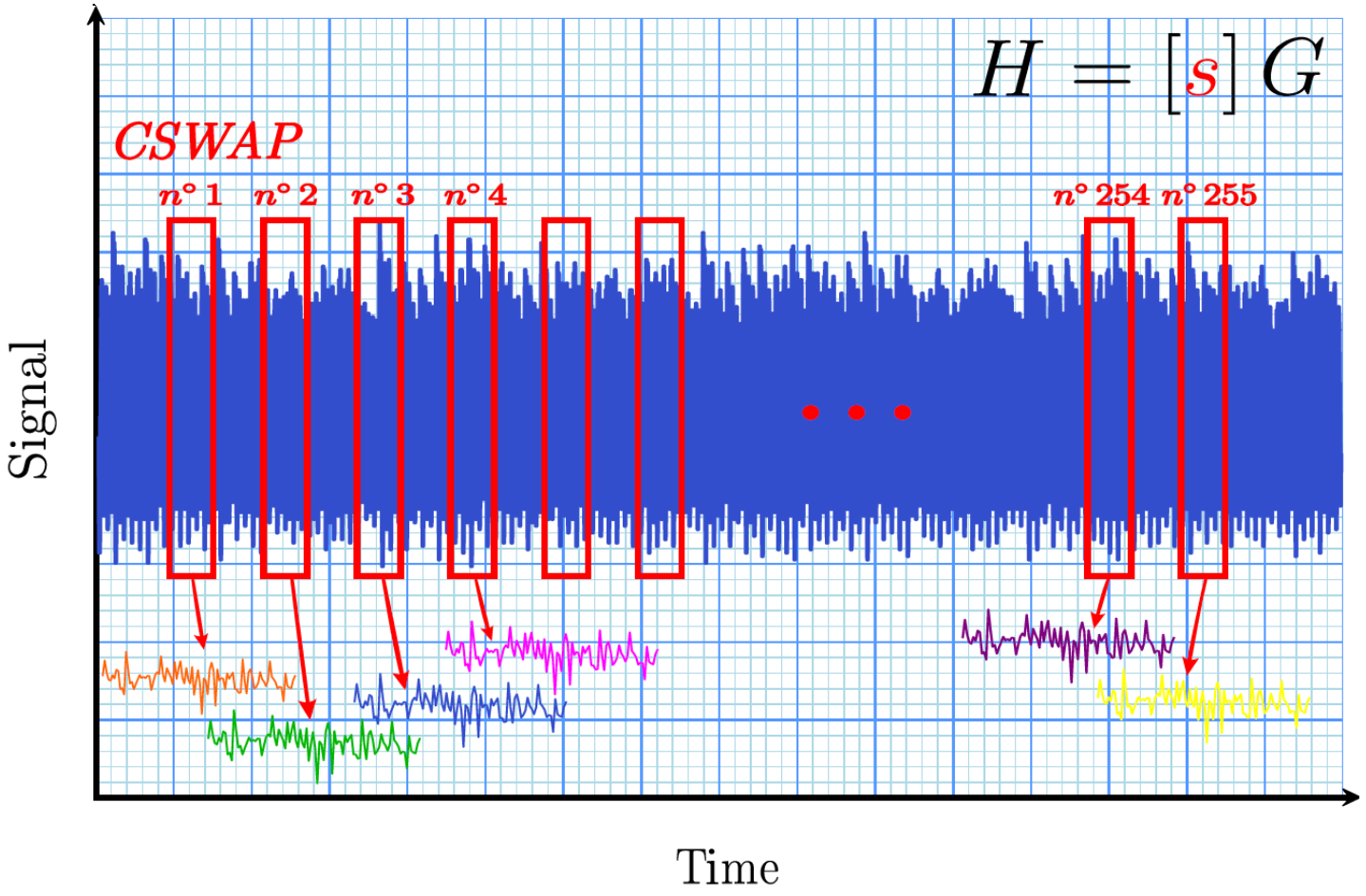
...initialization omitted...
$bprev \leftarrow 0$
for $i = 254...0$ **do**
  $RE\_RANDOMIZE\_COORDS(work\_state)$
  $b \leftarrow$ bit i of the secret scalar
  $s \leftarrow b \oplus bprev$
  $bprev \leftarrow b$
  $CSWAP(work\_state, s)$ ▷ Leakage (see [NCOS16,NC17])
  $LADDERSTEP(work\_state)$
end
...return omitted...

Two datasets
- CSWAP_ARITH
- CSWAP_POINTER

Targets: protected ECC software implementations on Curve25519 from μNaCl

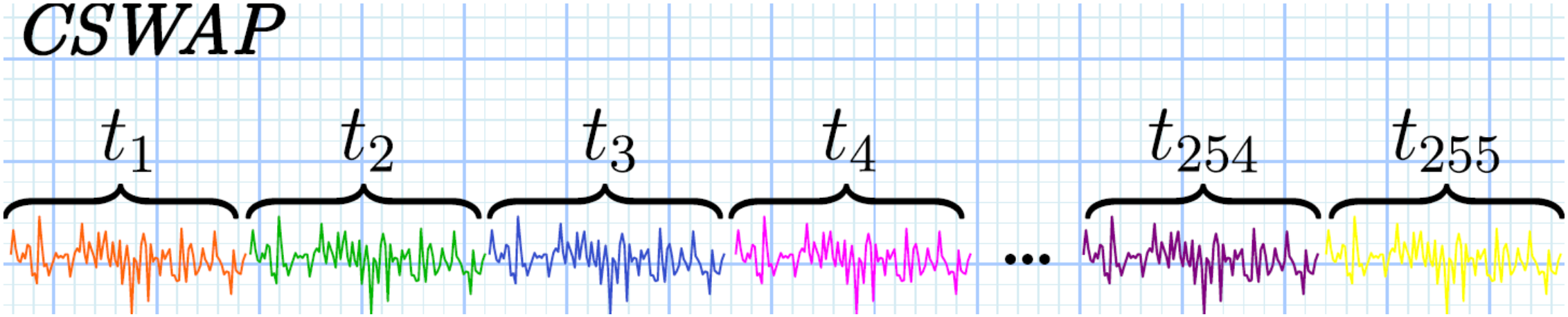
https://munacl.cryptojedi.org/curve25519-cortexm0.shtml

Target – EM acquisition



Selection of subtraces corresponding to CSWAP executions during an ECSM.

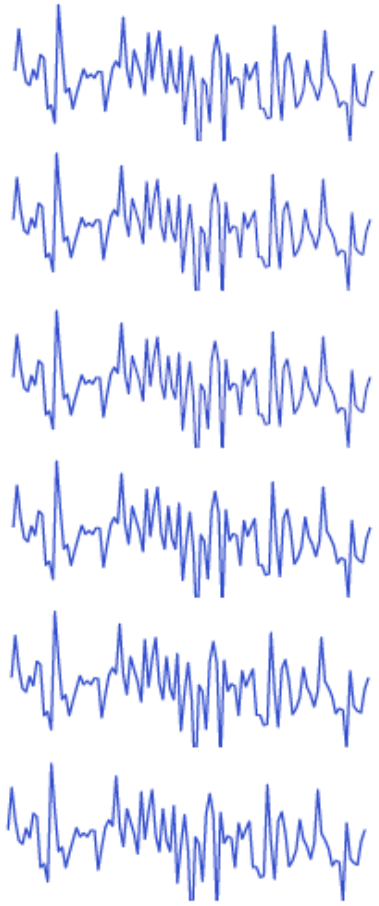
Target – Dataset construction



- A trace corresponds to the 255 CSWAPs concatenation. Each subtrace t_i corresponds to the processing of the i-th bit of the secret scalar **s**.
- Attack bit by bit proposed in [PCBP20] to recover the 255-bit secret scalar **s**.

Overall attack methodology presented in [PCBP20]

CSWAP



Applying Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations
Nascimento, Chmielewski [NC17]

Keep It Unsupervised : Horizontal Attacks Meet Deep Learning
Perin, Chmielewski, Batina, Picek [PCBP20]

Step 1 : Prelabeling phase

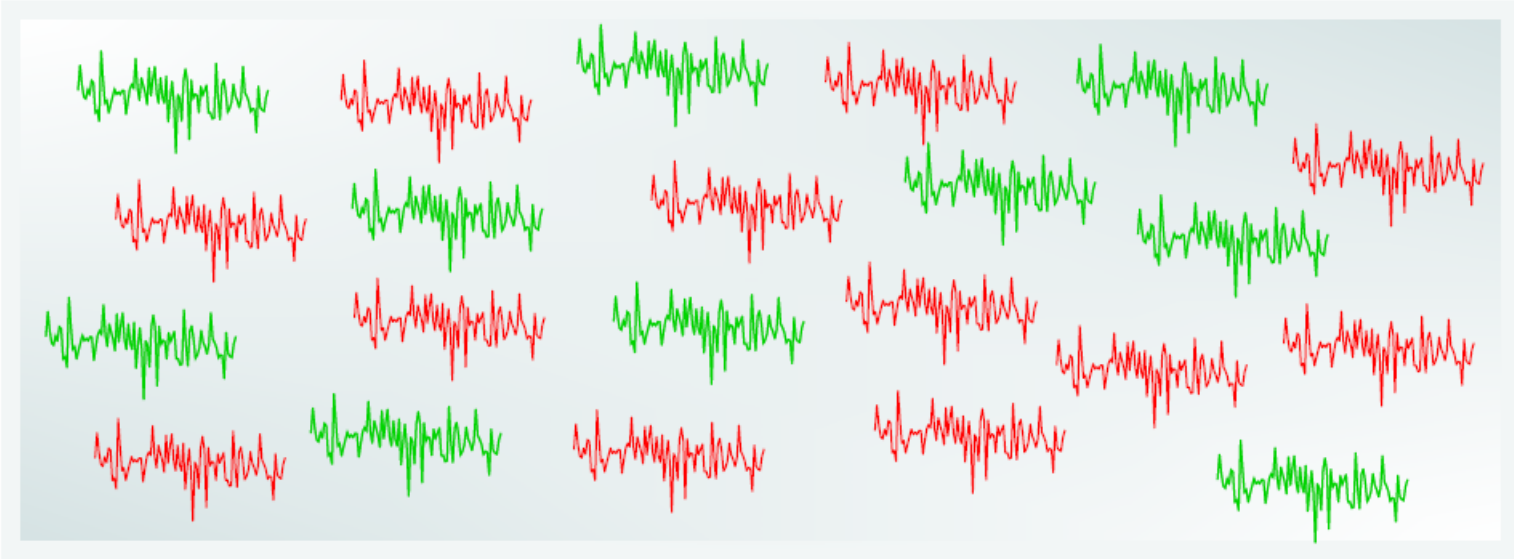
Step 2 : Corrective phase

Traces processing bit 0

Traces processing bit 1

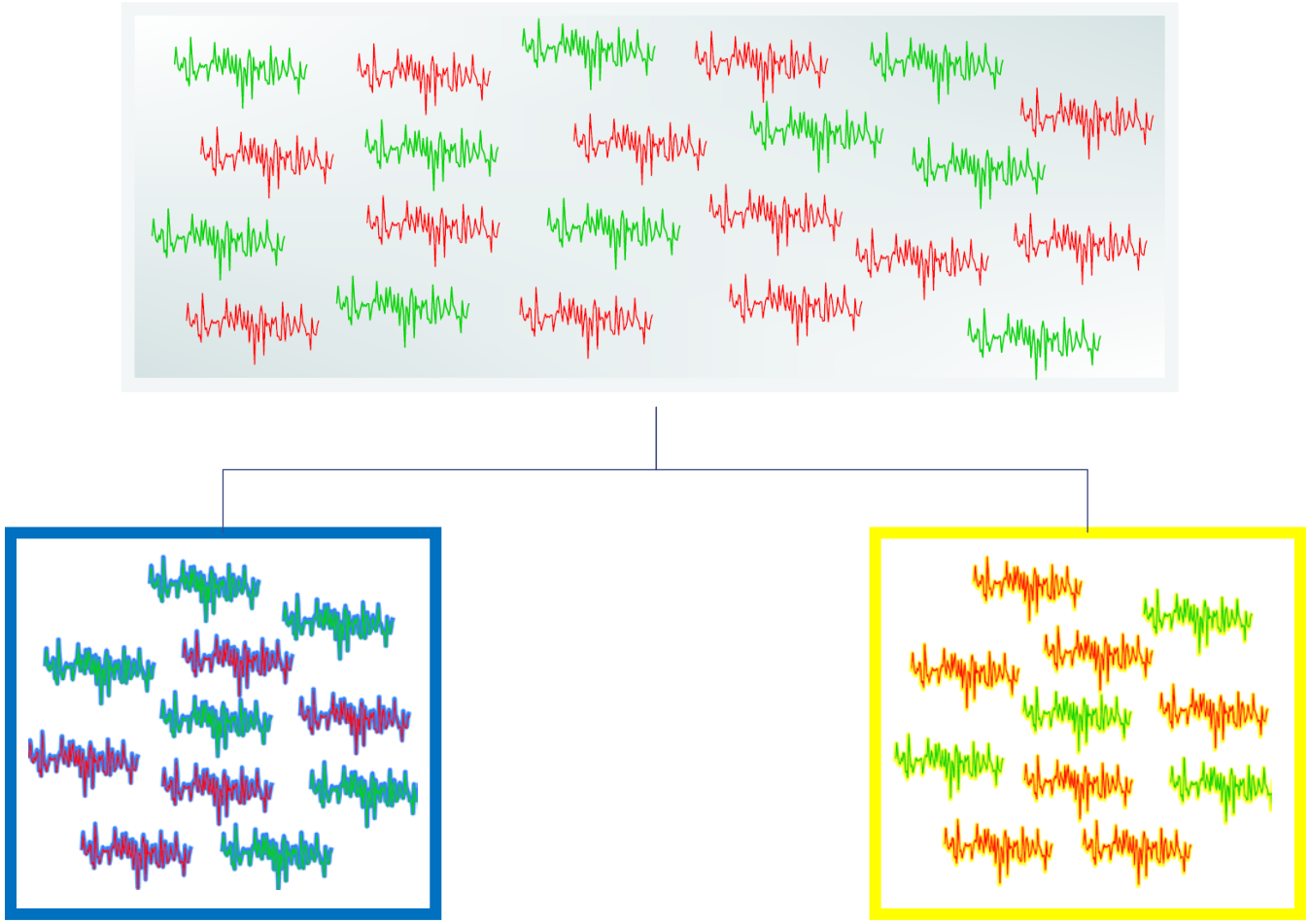
Overall attack methodology

Corrective phase – Presentation of the Iterative Framework: Preliminaries



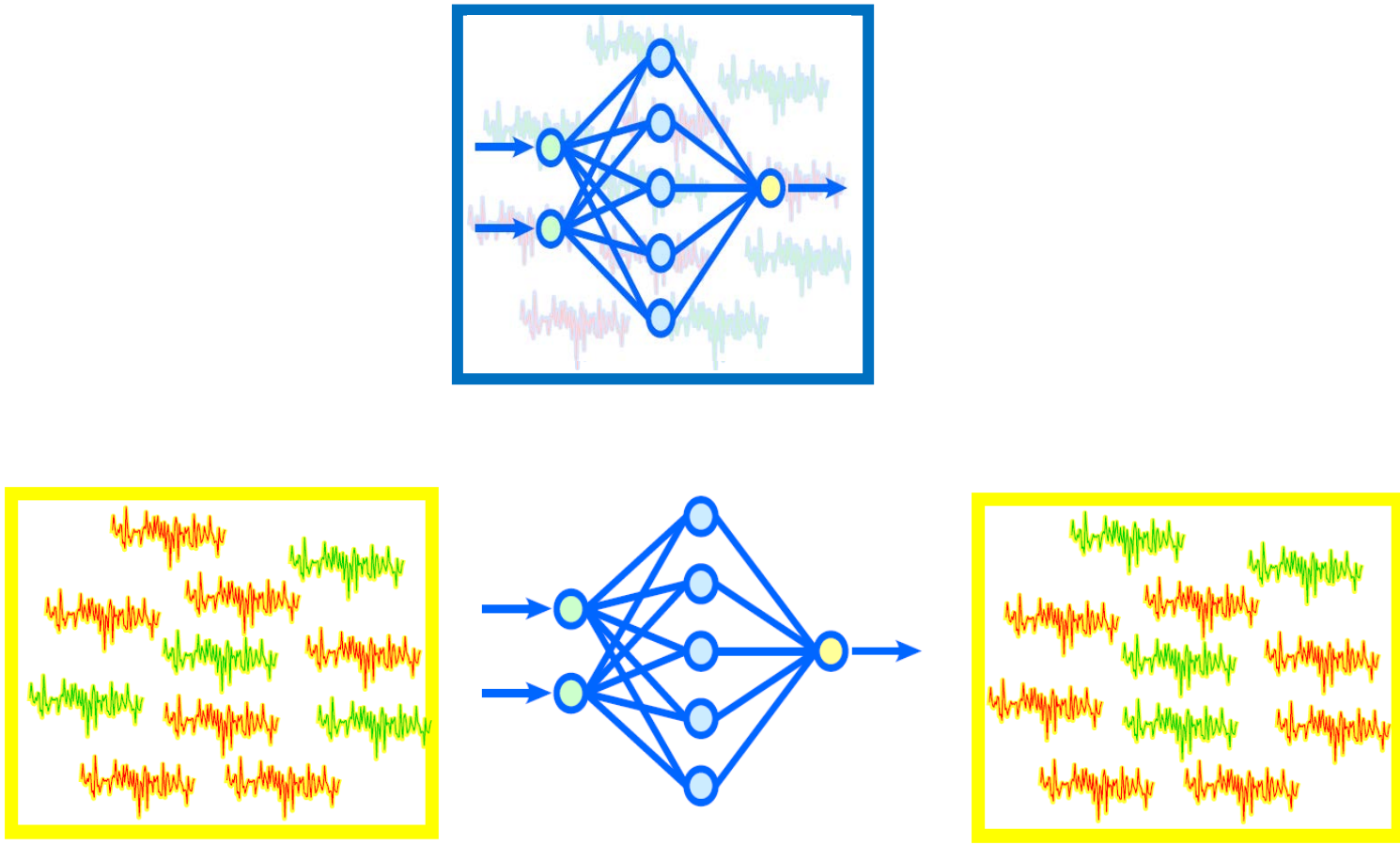
Creation of the dataset.

Corrective phase - Presentation of the iterative Framework: Phase 1



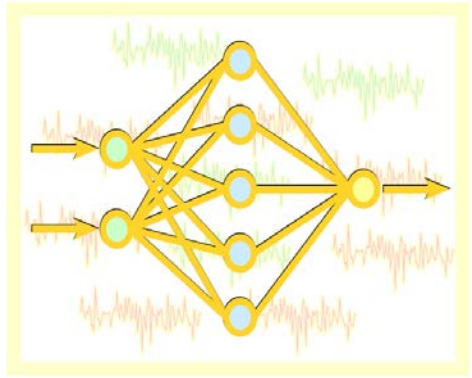
Phase 1 - Split of $\mathcal{D}$ into $\mathcal{D}_1$ (blue) and $\mathcal{D}_2$ (yellow).

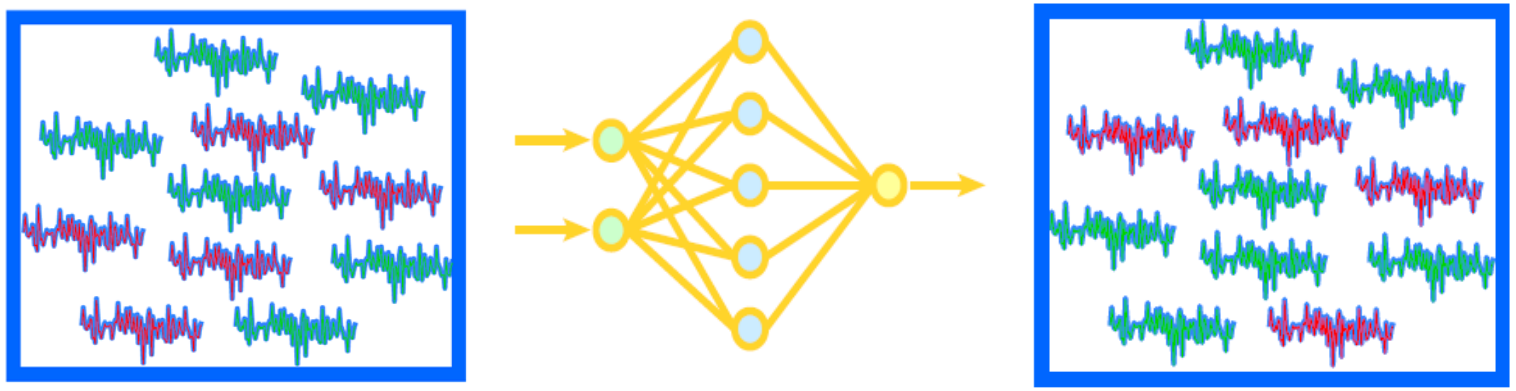
Corrective phase - Presentation of the iterative Framework: Phase 2



Phase 2 – Training on $\mathcal{D}_1$ (blue) in order to relabel $\mathcal{D}_2$ (yellow).

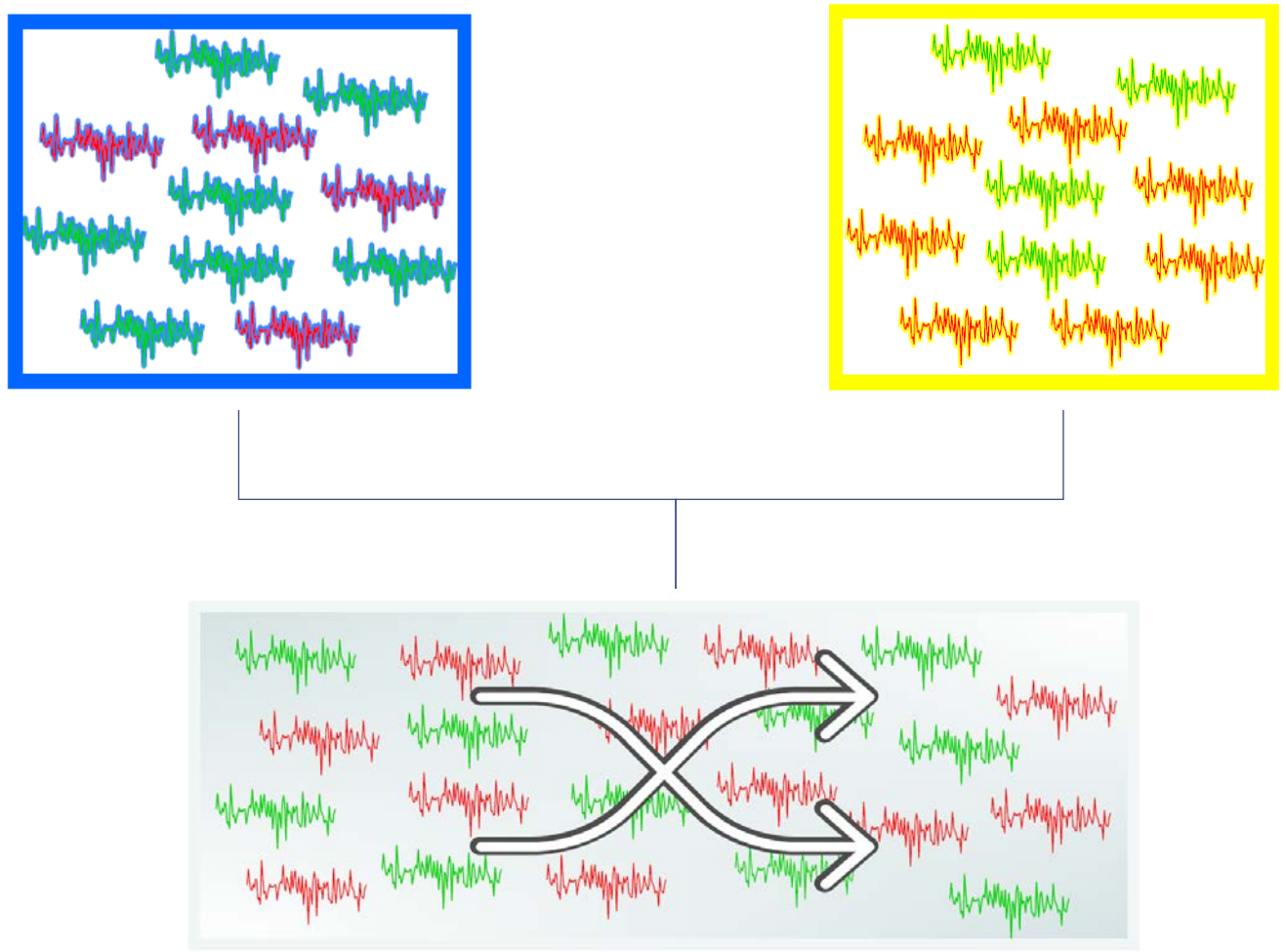
Corrective phase - Presentation of the iterative Framework: Phase 3





Phase 3 – Training on $\mathcal{D}_2$ (yellow) in order to relabel $\mathcal{D}_1$(blue).

Corrective phase - Presentation of the iterative Framework: Phase 4



Phase 4 – Merging of $\mathcal{D}_1$ (blue) and $\mathcal{D}_2$ (yellow) into $\mathcal{D}$.

Results achieved in [PCBP20]

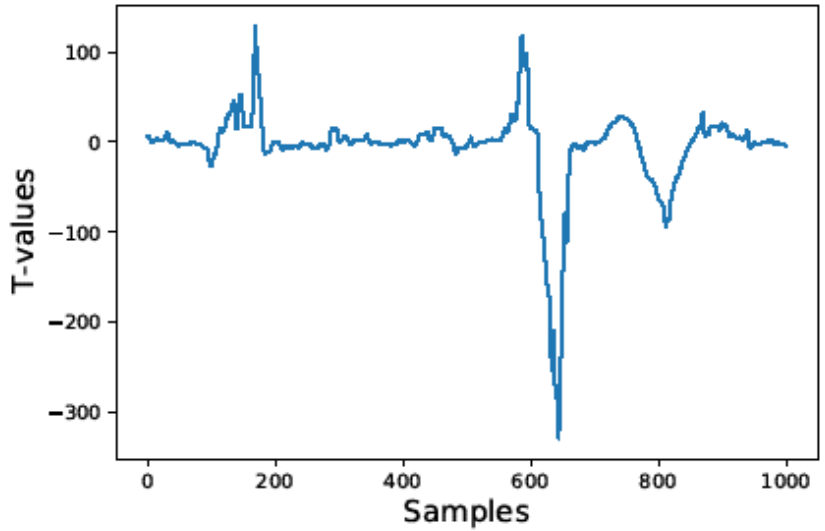
	cswap_pointer		cswap_arith	
	Average	Maximum	Average	Maximum
After prelabelling phase	52.24%	59.22%	52.44%	59.22%
CNN with no regularisation	85%	97.64%	52%	76.07%
CNN with regularisation (dropout + data augmentation)	91%	100%	83%	100%

Average and maximum single trace accuracies obtained in [PCBP20] for fixed CNN hyperparameters (50 framework iterations).

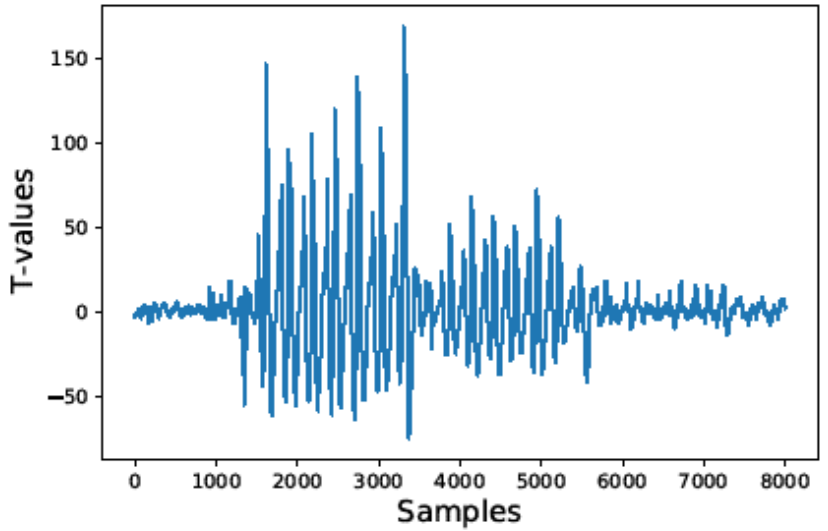
Our hypothesis: the model complexity is too high and leads to overfitting

3. In-depth analysis of the iterative framework

Data analysis: linear decision boundaries



(a) cswap_pointer dataset



(b) arith_pointer dataset

Welch's t-tests obtained on the two datasets using true labels.

K-means on interesting regions : High average accuracy → **Linear separability**

Simplification of the neural networks

Layers	-
Conv1D_1	8 filters
Conv1D_2	16 filters
Conv1D_3	32 filters
Dense_1	100 neurons
Dense_2	100 neurons
Softmax	2 neurons

CNN architecture for both datasets [PCBP20].

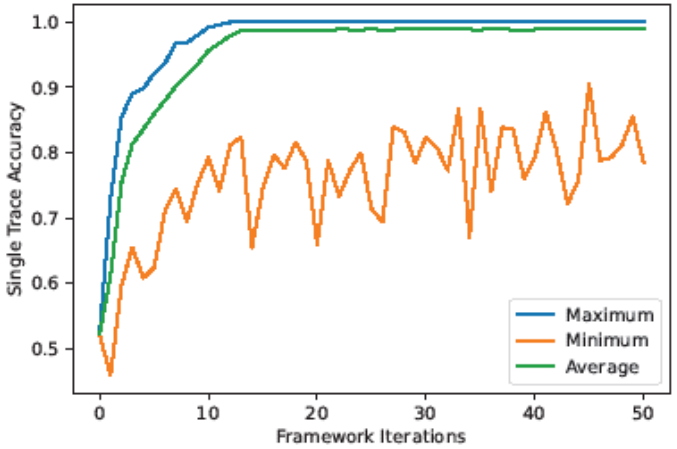
Layers	-
Sigmoid	1 neuron

Architecture proposed for both datasets - Perceptron, **a linear classifier**.

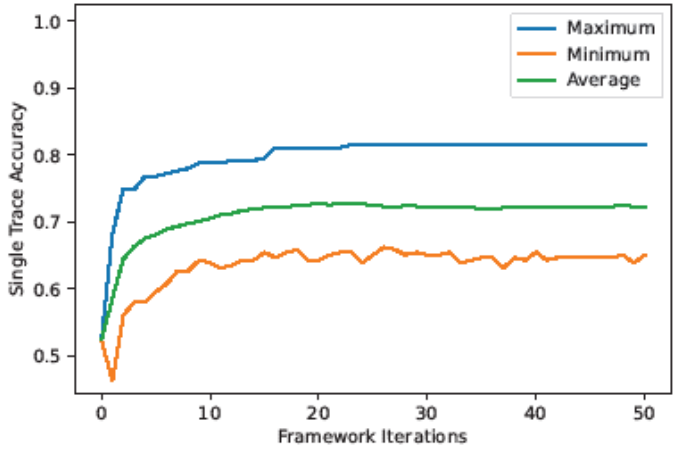
	Nb. of parameters			Attack duration		
	(a)	(b)	(c)	(a)	(b)	(c)
CNN [PCBP20]	45,978	3,069,820	669,820	75 min.	1,080 min.	300 min.
Perceptron	**1,001**	**8,001**	**2,001**	**8 min.**	**35 min.**	**10 min.**

too complex.

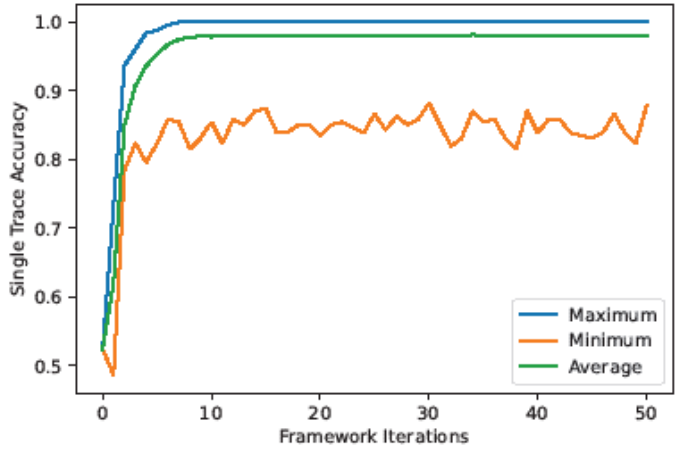
Simplification of the neural networks: results



(a) cswap_pointer dataset



(b) cswap_arith dataset

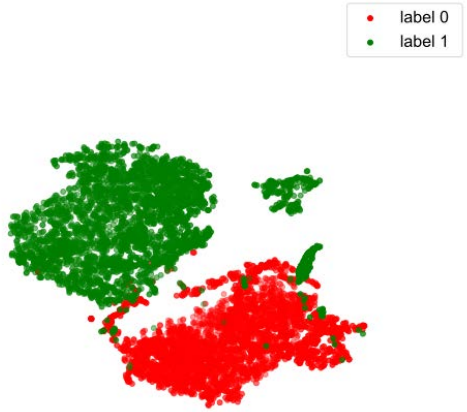


(c) cswap_arith_reduced dataset

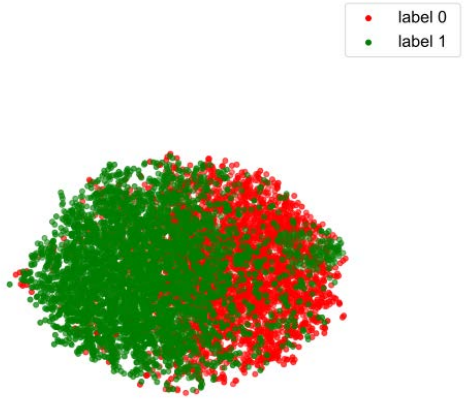
Average accuracy obtained using iterative framework with our neural network after 50 iterations (no regularization)

	(a)	(b)	(c)
Perceptron, no regularisation	**98.9%**	70.93%	**98%**
CNN [PCBP20], no regularisation	85%	52%	58%
CNN [PCBP20], dropout + data augmentation	91%	**83%**	**98%**

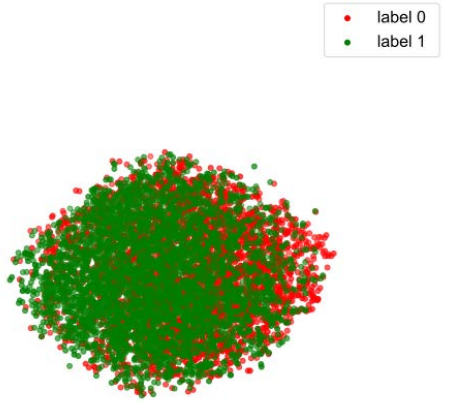
Correction capability of the iterative framework – Case study: noisy traces



Without noise



Adding a Gaussian noise ($\sigma = 30$)



Adding a Gaussian noise ($\sigma = 90$)

Visualization of `cswap_pointer` dataset's clusters

Correction capability of the iterative framework – Case study: noisy traces – Results

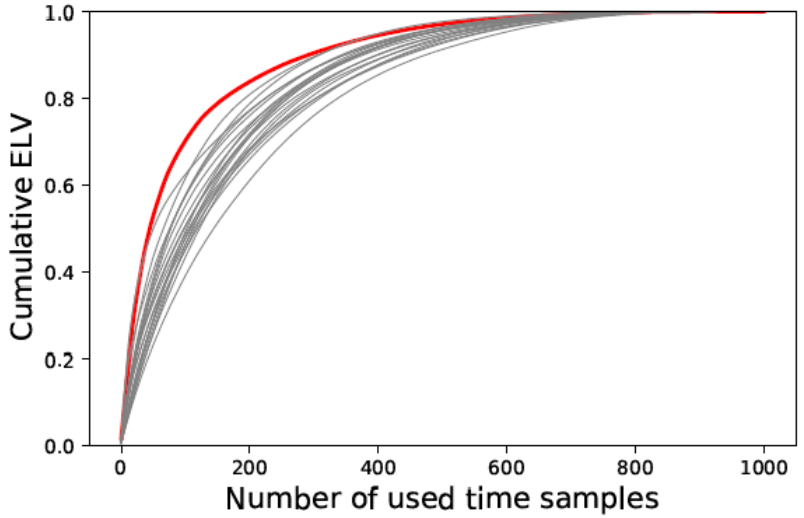
	$\sigma = 15$	$\sigma = 30$	$\sigma = 90$
CNN [PCBP20]	99.87%	98.83%	79.43%
CNN [PCBP20]	97.11%	71.25%	50.78%
Perceptron	98.87%	96.66%	77.1%
Perceptron	98.21%	67.15%	56.43%

Average single trace accuracy achieved for cswap pointer dataset in a **supervised**/**unsupervised** setting.

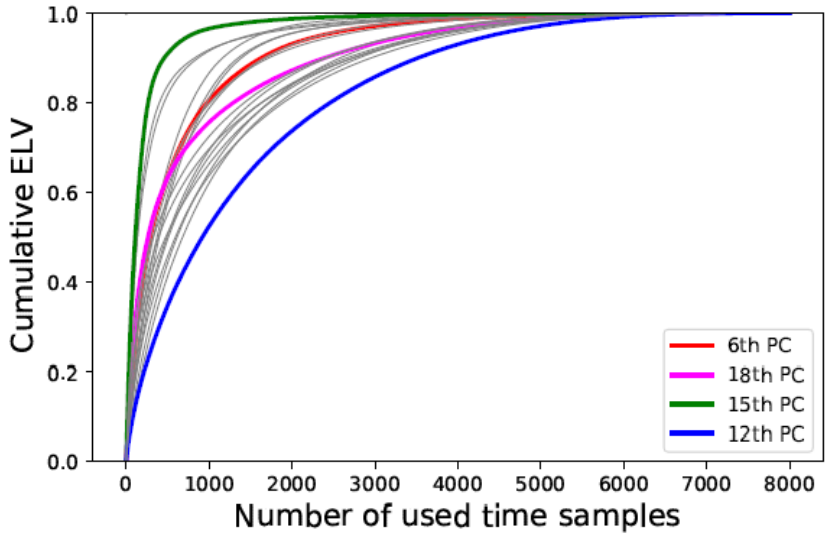
4. Deep Learning Free Approach

Deep Learning free attack

- Features extraction → PCA
- Component selection issue
  - Visual selection
  - Natural order (decreasing eigenvalues)
  - Explained Local Variance (ELV - [CDP15])
    - Denoising strategy
- Clustering → K-means



(a) `cswap_pointer`



(b) `cswap_arith`

Deep Learning Free Results

	Cswap_pointer			Cswap_arith		
	Average	Maximum	PC/Pols	Average	Maximum	PC/Pols
PCA+Virtual	94.1%	97.7%	8th	60.8%	72.2%	6th
PCA+Natural order	94.1%	97.7%	8th	76.1%	82.4%	18th
PCA + ELV + Denoising	98%	100%	8th	90.6%	96.1%	18th
SOST [PCBP20]	52.24%	55.22%	20 Pols	52.44%	55.22%	20 Pols
SOST+Framework [PCBP20]	91%	100%	20 Pols	83%	100%	20 Pols

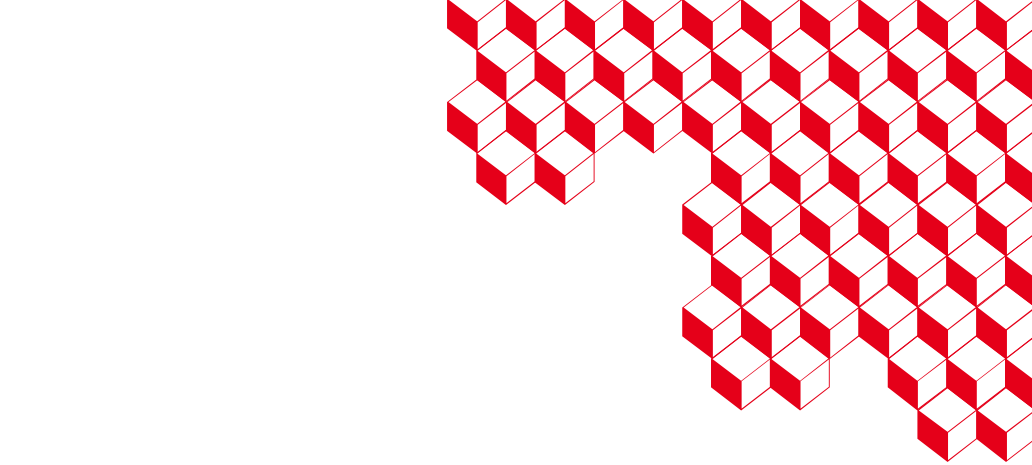
91% → 23 wrong bits on average → 2^{109} operations to end the attack
96,1% → 10 wrong bits on average → 2^{58} operations to end the attack
98% → 5 wrong bits on average → 2^{33} operations to end the attack

Deep Learning Free approach leads to **at least the same successfull attacks** than [PCBP20] in the studied case.

5. Conclusion

Conclusion

- The study proposed in [PCBP20] was biased by the simplicity of the used datasets (two linear separable classes)
- Such a simplicity was hidden by the huge complexity of the proposed deep-learning-based solution
- When moving to a more complex scenario, the iterative framework is strongly impacted.
  Open problem: finding a good strategy to correct mislabeled data, filling the supervised/unsupervised gap
- Is it suitable to use very complex machine-learning solution for general problems?
- Instead: focus on the simplest possible attacks → better insight on the actual threats
- In case machine-learning is needed → size it carefully!



Thank you