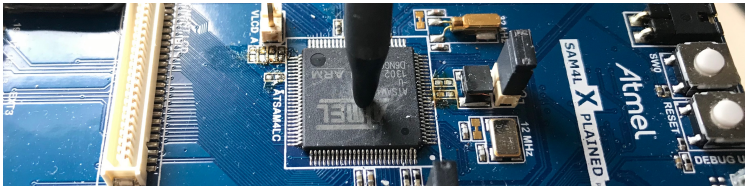


Don't Learn What You already Know

Scheme-Aware Modeling for Profiling Side-Channel Analysis against Masking

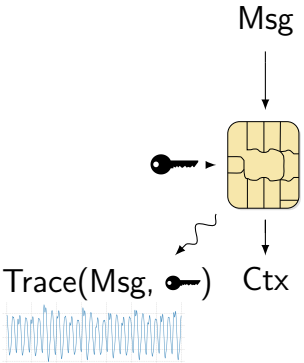
Loïc Masure, Valence Cristiani, Maxime Lecomte, François-Xavier Standaert

Prague, September 12th

Content

Context : Side-Channel Analysis (SCA)

"Cryptographic algorithms don't run on paper, they run on physical devices"



: N bits

Black-box cryptanalysis:
→ Exponential with N

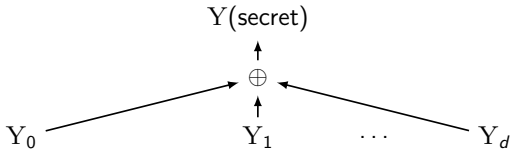
Side-Channel Analysis:
→ Exponential with bit-size
→ *Linear* with N

Trace : power, EM, acoustics, runtime, ...

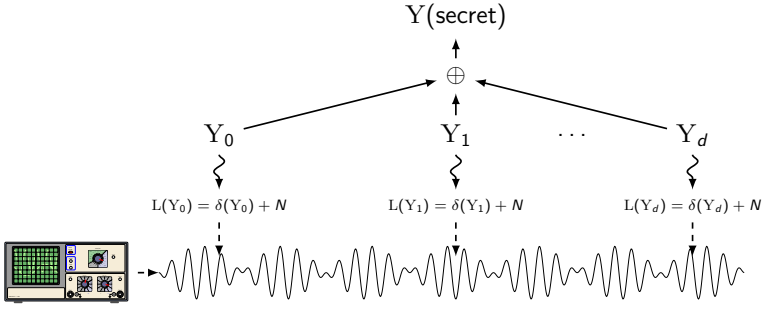
The Counter-Measure: Masking

$Y(\text{secret})$

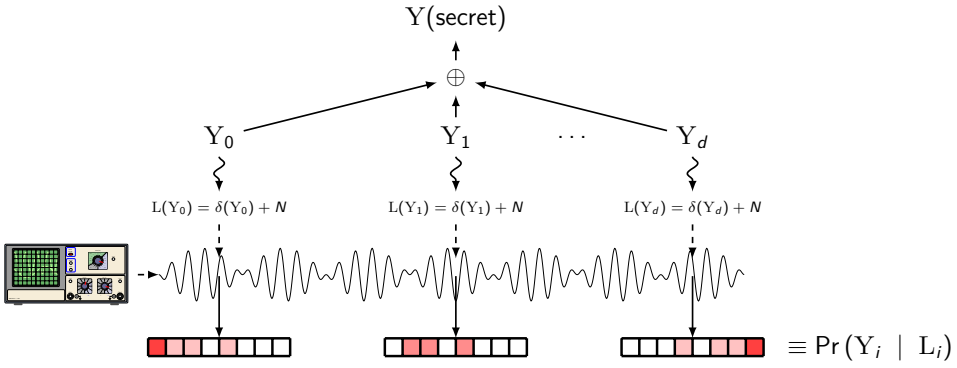
The Counter-Measure: Masking



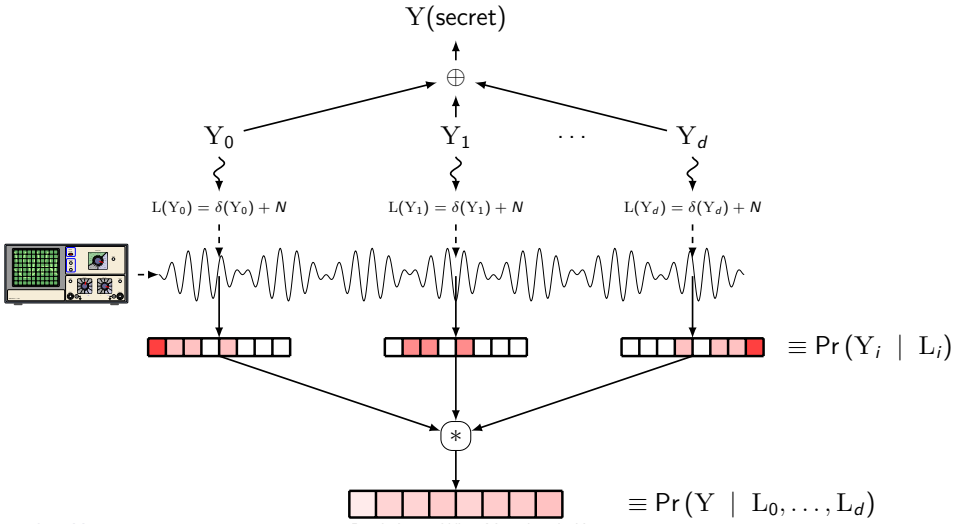
The Counter-Measure: Masking



The Counter-Measure: Masking

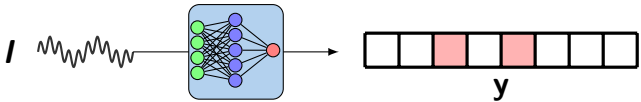


The Counter-Measure: Masking



Content

Deep Learning (DL) for SCA



Deep Neural Network (DNN)

General way to modelize, *i.e.*, to convert leakage into probabilities

$$F : \left| \begin{array}{ccc} \mathcal{L} & \longrightarrow & \mathcal{P}(\mathcal{Y}) \\ \boldsymbol{I} & \longmapsto & \mathbf{y} = F(\boldsymbol{I}) \approx \Pr\left(\mathrm{Y} \;\middle|\; \mathbf{L} = \boldsymbol{I}\right) \end{array} \right. \tag{1}$$

$F(\boldsymbol{I})$: output of a Directed Acyclic Graph (DAG) of computation:

Each node: elementary function $f_i(\cdot, \theta_i)$

θ_i: *parameters* fully describing f_i

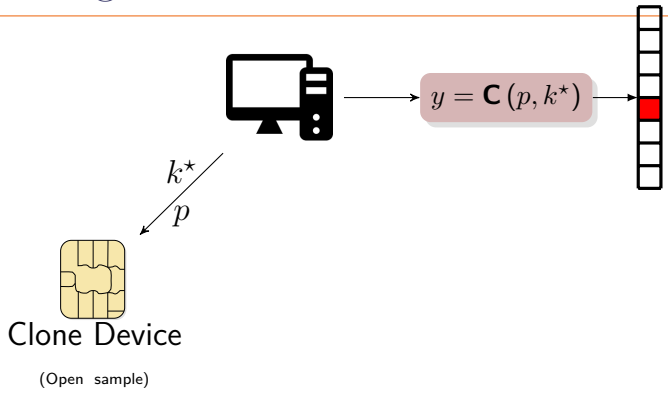
Shape of the DAG, nature of the classes of functions: *architecture* of the DNN.

Training a DNN for Profiled SCA

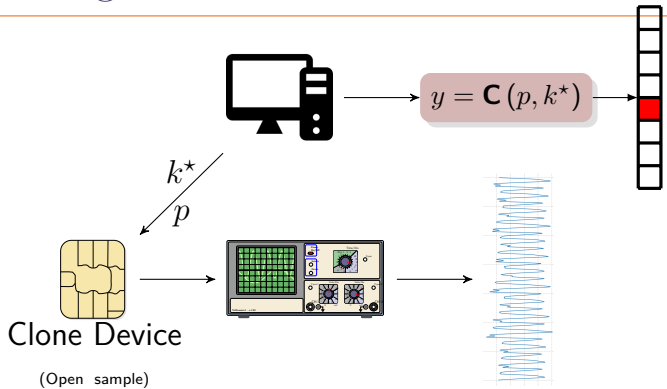
Clone Device

(Open sample)

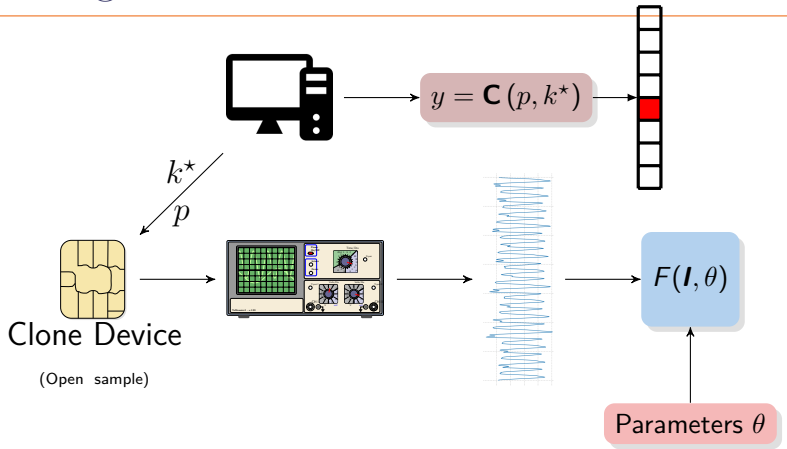
Training a DNN for Profiled SCA



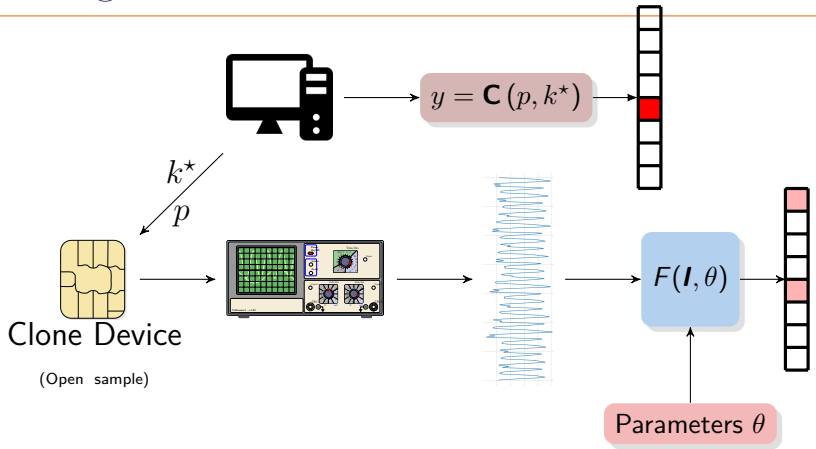
Training a DNN for Profiled SCA



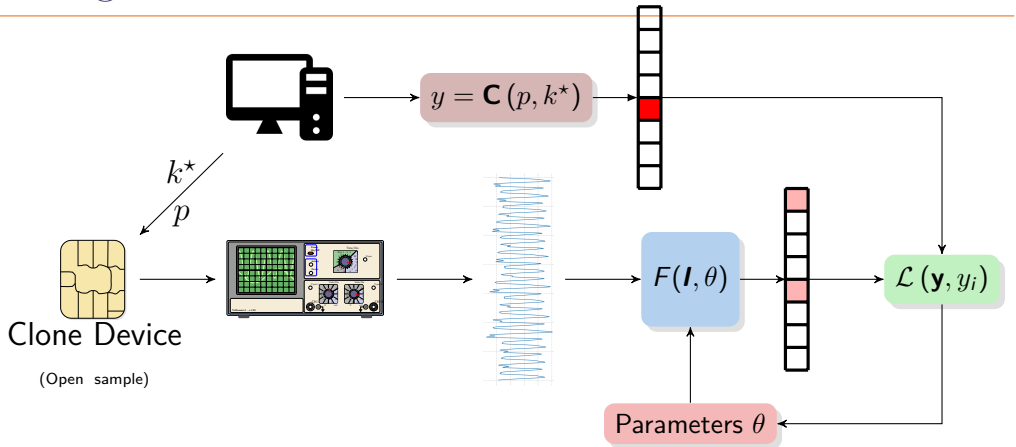
Training a DNN for Profiled SCA



Training a DNN for Profiled SCA

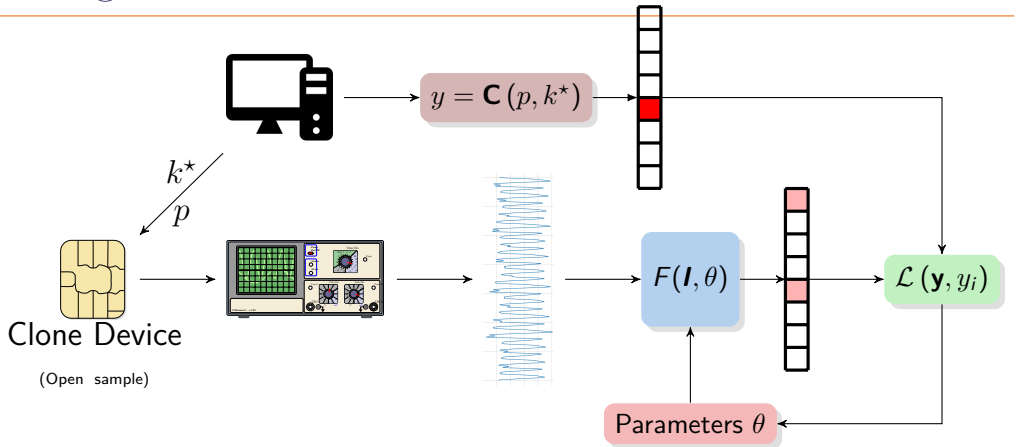


Training a DNN for Profiled SCA



$\mathcal{L}()$: loss function to minimize, with *gradient descent*

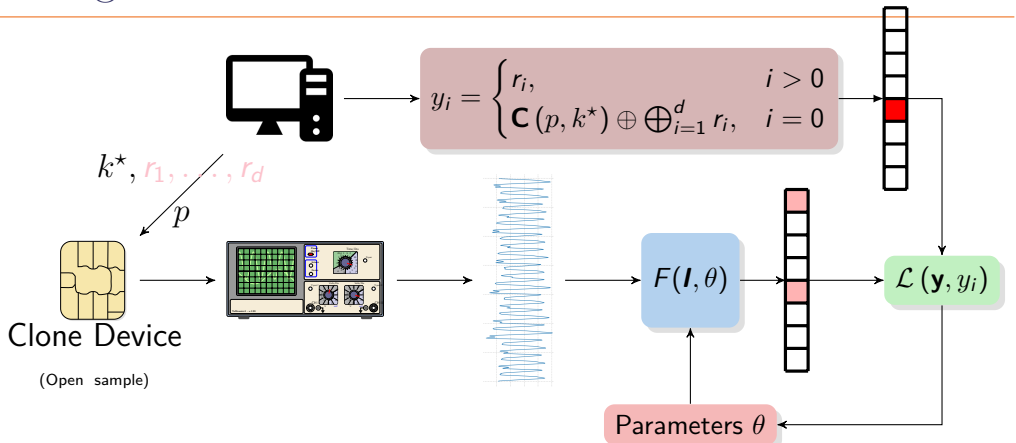
Training a DNN for Profiled SCA



$\mathcal{L}()$: loss function to minimize, with *gradient descent*

Uninformed adversary: no knowledge of random shares during profiling

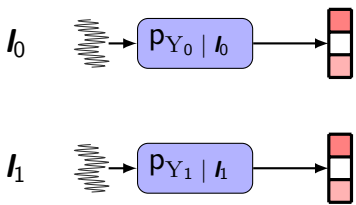
Training a DNN for Profiled SCA



$\mathcal{L}()$: loss function to minimize, with *gradient descent*

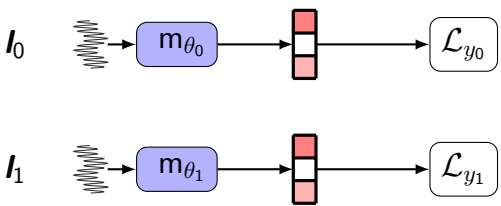
Worst-case adversary: knowledge of random shares during profiling

How to profile as Worst-Case Adversary?



The natural way: divide & conquer
$\rightarrow \Pr(\mathrm{Y} \mid \mathbf{L})$ decomposed as collection of $\Pr(\mathrm{Y}_i \mid \mathbf{L}_i)$

How to profile as Worst-Case Adversary?



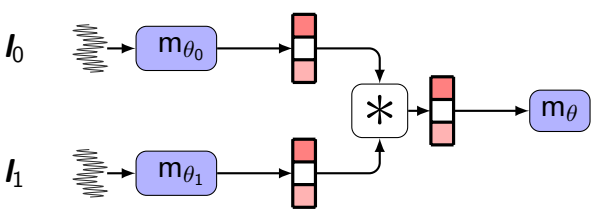
The natural way: divide & conquer

→ $\Pr(\mathrm{Y} \mid \mathbf{L})$ decomposed as collection of $\Pr(\mathrm{Y}_i \mid \mathbf{L}_i)$

→ Each, modeled by m_{θ_i}

→ Separately trained with $\mathcal{L}_{y_i}$

How to profile as Worst-Case Adversary?



The natural way: divide & conquer

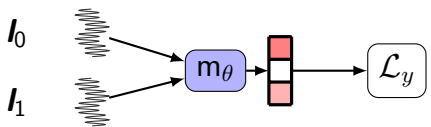
$\rightarrow$ $\Pr(\mathrm{Y} \mid \mathbf{L})$ decomposed as collection of $\Pr(\mathrm{Y}_i \mid \mathbf{L}_i)$

$\rightarrow$ Each, modeled by m_{θ_i}

$\rightarrow$ Separately trained with $\mathcal{L}_{y_i}$

$\rightarrow$ Then use $\circledast$ to recombine

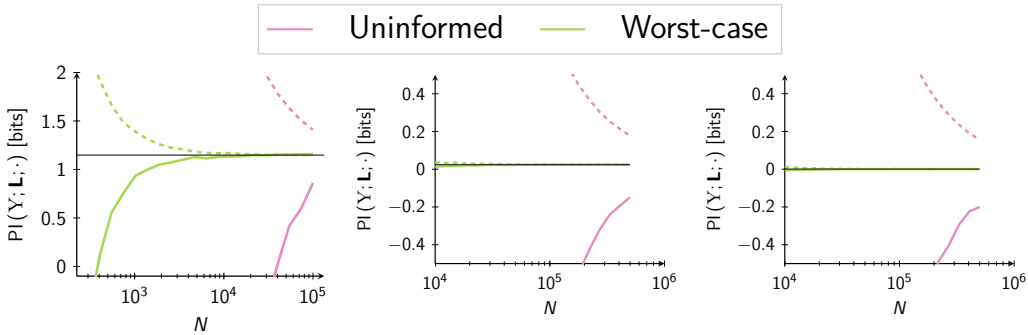
How to profile as Uninformed Adversary?



The End-to-End Way:

$\rightarrow \Pr(\mathrm{Y} \mid \mathbf{L})$ directly modeled by m_θ, trained with $\mathcal{L}_y$

Simulated Experiments



(a) SNR = 200 (b) SNR = 1 (c) SNR = 0.1

Figure: Learning curves: MI estimation vs. data complexity.

What Kind of Adversary for Evaluation ?

Adversary	Worst-case	Uninformed
Access to shares	Yes	No
Knowledge of scheme	Yes	No

Access to shares during profiling:

Easy to reach optimal attacks ✓

Too conservative ✗

Not realistic ✗

What Kind of Adversary for Evaluation ?

Adversary	Worst-case	Scheme-Aware	Uninformed
Access to shares	Yes	No	No
Knowledge of scheme	Yes	Yes	No

Access to shares during profiling:

Easy to reach optimal attacks ✓

Too conservative ✗

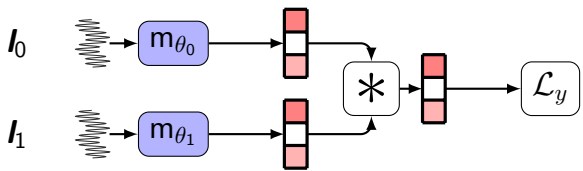
Not realistic ✗

How to leverage the knowledge of the masking scheme, without relying on the knowledge of the shares?

Content

Don't Learn what You Already Know !

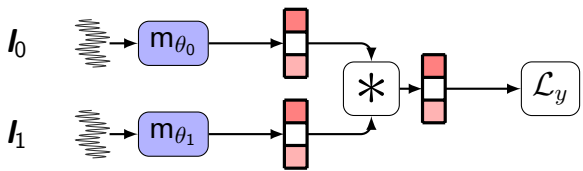
Can we find a trade-off between both approaches ?



$\rightarrow$ Model still decomposed as collection of $\Pr(\mathrm{Y}_i \mid \mathbf{L}_i)$

Don't Learn what You Already Know !

Can we find a trade-off between both approaches ?

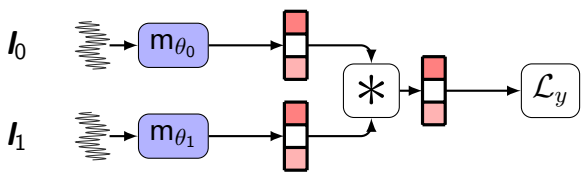


→ Model still decomposed as collection of $\Pr(\mathrm{Y}_i \mid \mathbf{L}_i)$

→ Still recombined with $\circledast$ but ...

Don't Learn what You Already Know !

Can we find a trade-off between both approaches ?



→ Model still decomposed as collection of $\Pr(\mathrm{Y}_i \mid \mathbf{L}_i)$
→ Still recombined with $\circledast$ but ...
→ ... Training done *jointly* with $\mathcal{L}_y$
→ Need to backprop gradients through $\circledast$

Pytorch code available at
`github.com/uclcrypto/Scheme-Aware-Architectures`

Back to our Simulated Experiments

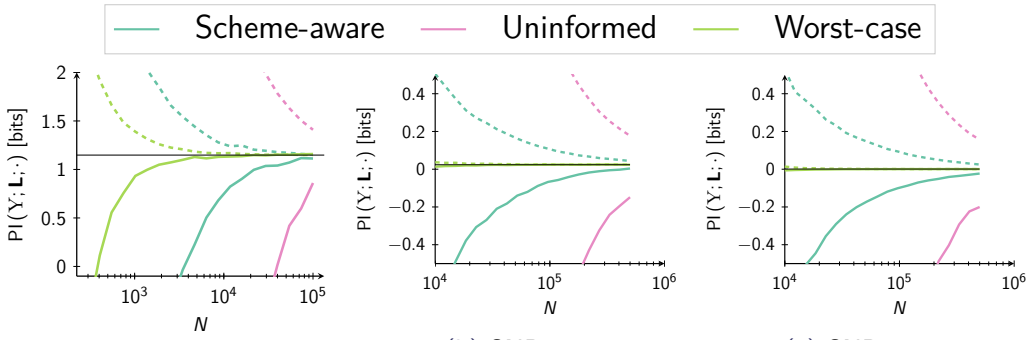
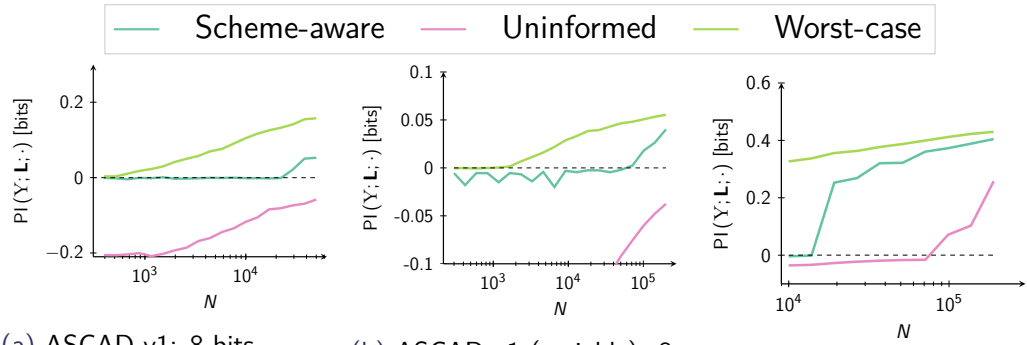


Figure: Learning curves: MI estimation vs. data complexity.

Scheme-Aware spares some data complexity

Application to Experimental Data



(a) ASCAD-v1: 8 bits, $d = 1$.

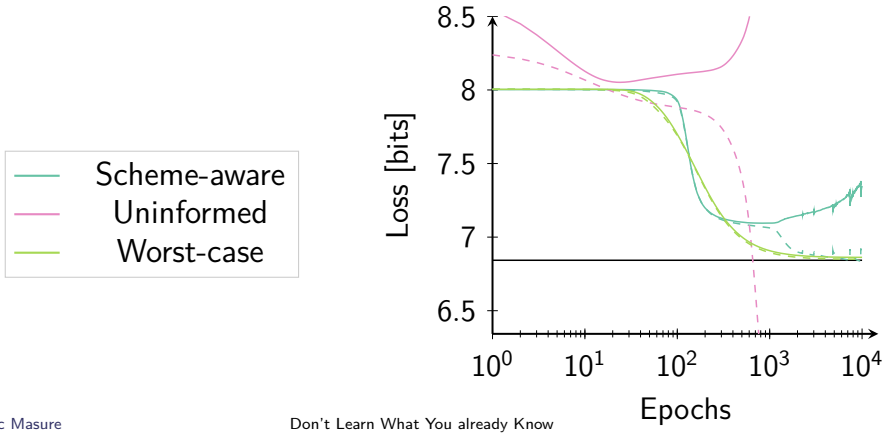
(b) ASCAD-v1 (variable): 8 bits, $d = 1$.

(c) Spook: 2 bits, $d = 2$.

Figure: Learning curves: MI estimation vs. data complexity.

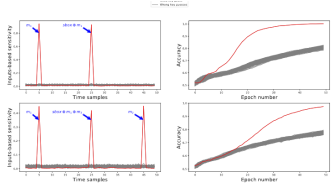
What about Higher Order Masking?

Affine masking: positive results on simulation, but not on experimental data.
Why ? **The Plateau Effect**

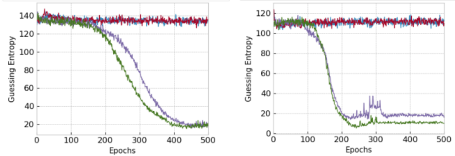


Content

The Elephant in the Room

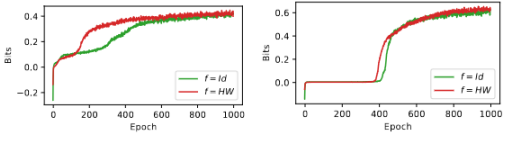


(a) Timon, Ches'19



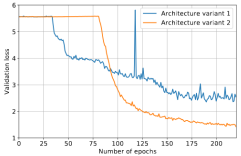
(a) ASCAD fixed key. (b) ASCAD random keys.

(b) Perin & Picek, SAC'20



(a) First order masking ($\sigma = 1$) (b) Second order masking ($\sigma = 0.5$)

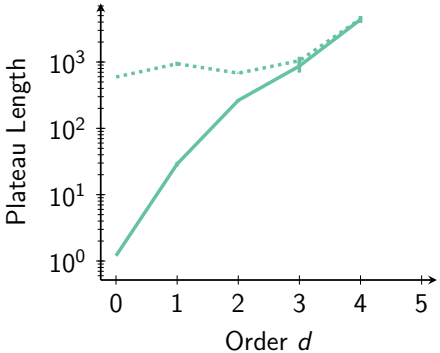
(c) Cristiani *et al.*, JoC'23



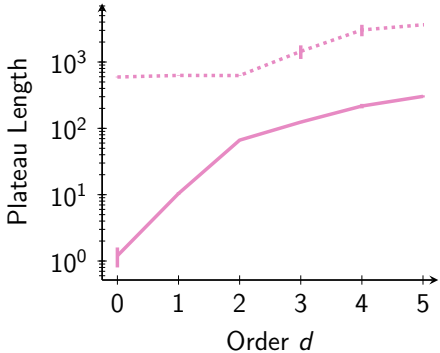
(d) Lu *et al.*, Ches'21

How Masking Affects the Plateau Length

Simulation with HW leakage model and *exhaustive* dataset (no profiling error)



(a) Scheme-aware.



(b) Uninformed.

An Explanation

Theorem (informal[1])

Assume that each $\mathbf{L}_i$ *is i.i.d. standard Gaussian in* $\mathbb{R}^p$*. Define the target function* $h_{\boldsymbol{u}}(\boldsymbol{l}) = \prod_{i=1}^{d} \operatorname{sign}(\boldsymbol{u}^\intercal \boldsymbol{l}_i)$*, for some normalized hyperplane* $\boldsymbol{u}$*. Let* m_θ *be a model, such that* $\mathop{\mathbb{E}}_{\mathbf{L}} \left[\|\nabla_\theta \mathrm{m}_\theta\|^2 \right] \leq G(\theta)^2$*. Then,*

$$\mathop{\mathbb{E}}_{\boldsymbol{u}} \left[\left\| \nabla_\theta \mathcal{L}(\theta) - \mathop{\mathbb{E}}_{\boldsymbol{u}} \left[\nabla_\theta \mathcal{L}(\theta) \right] \right\|^2 \right] \leq G(\theta)^2 \cdot \mathcal{O}\left(\sqrt{\frac{d \log(p)}{p}} \right)^d .$$

The gradient almost takes the same direction, regardless of $\boldsymbol{u}$!

[1] Shalev-Shwartz, Shamir, and Shammah, "Failures of Gradient-Based Deep Learning", p. ICML 2017.

Content

Open Problems

How to tackle higher orders with DL remains unclear: GD really not suitable ?

- Efficient surrogate to gradient descent ?

  $\rightarrow$ Then current evaluator run suboptimal attacks

- No efficient surrogate to GD (reduction to hard learning problem) ?

  $\rightarrow$Then intrinsic gap between worst-case approach and others

Worth investigating, no matter the answer !

References I


Shalev-Shwartz, S., O. Shamir, and S. Shammah. "Failures of Gradient-Based Deep Learning". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3067–3075. URL: http://proceedings.mlr.press/v70/shalev-shwartz17a.html.