ALPEN-ADRIA
UNIVERSITÄT
KLAGENFURT

Exploring Multi-Task Learning in the Context of Masked AES Implementations

Thomas Marquet and Elisabeth Oswald

April 10, 2024

European Research Council
Established by the European Commission

T. Marquet and E. Oswald have been supported in part by the European Research Council

(ERC Grant No. 725042)

Outline

UNIVERSITÄT KLAGENFURT

European Research Council

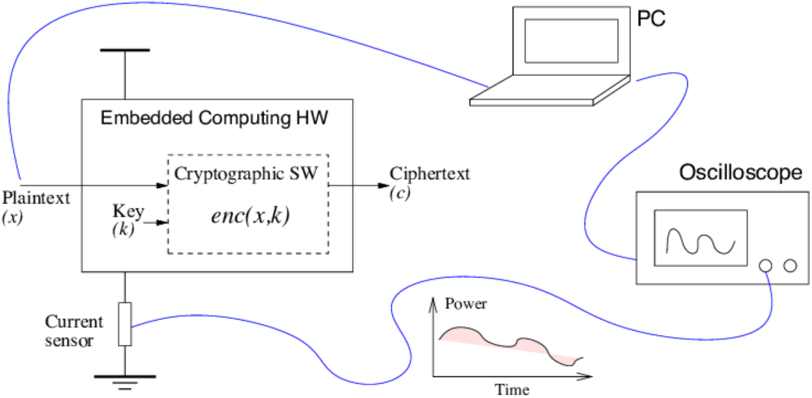


Figure: Power analysis classic setup [2]

Deep learning applied to side-channel attacks

Profiled attacks

- Require a clone of the target.
- Create a dataset with control of the inputs.
- Train deep networks on weak points (intermediates)
- Infer the key from the target.
- Straightforward in white-box

Challenges ?

Motivation

Plateau Effect

- Introduced in "Don't Learn What You Already Know", Masure et al. [3]
- Initial confusion due to the random values given as weights to the model
- Complexity of the attack => Passing the plateau

Plateau Effect

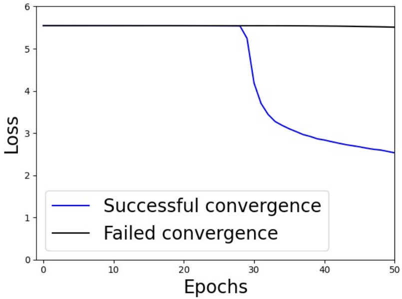


Figure: Example of the plateau effect on two training runs

How to build deep learning models that consistently break through the plateau?

Contributions

We discuss :

- The consistency of convergence of multi-task, and single-task models across seeds.
- Ways of sharing weights inside a deep learning model

We show that:

- Multi-task models are more consistent at breaking through the plateau
- Shared layers can introduce helpful constraints
- Multi-target strategies can be helpful even during training

Outline

Why does multi-task learning make sense?

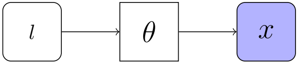
Introduced by Caruana in "Multitask Learning",[1]

Benefits

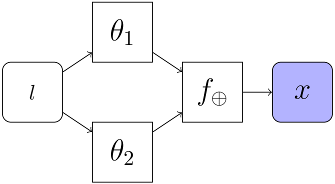
- Data amplification/augmentation.
- Input understanding.
- Eavesdropping between tasks.
- Representation bias.
- Better utilization of computing resources.

Drawbacks

- Competition of losses
- More expensive VRAM-wise



(a) Classic single-task model



(b) D-branch single-task model

Figure: Hard encoding of the masking scheme inside the network

Encoding the masking scheme in custom layers

Custom layers :

- $f_{\oplus}$: Xor
- $f_{\otimes}^{-1}$: Inverse multGF256

The calculation of $f_{\oplus}$, $f_{\otimes}^{-1}$ is a conditional probability:

$$\begin{cases} f_{\oplus}(x,y)[i] = \sum_{j=0}^{255} x[j] \times y[i \oplus j] \quad \forall\ i\ \in\ [0,255] & (1) \\ f_{\otimes}^{-1}(x,y)[i] = x[0] + \sum_{j=1}^{255} x[j] \times y[i \otimes j] \quad \forall\ i\ \in\ [0,255] & (2) \end{cases}$$

Outline

High-level parameter sharing

Classic design from Caruana [1] with shared layer $\theta_\forall$

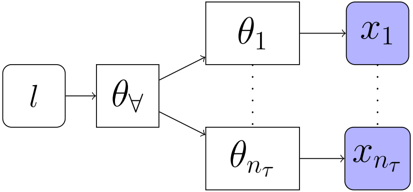


Figure: Multi-task design with high-level parameter sharing

Adapted design for two boolean shares with shared layer $\theta_\forall$

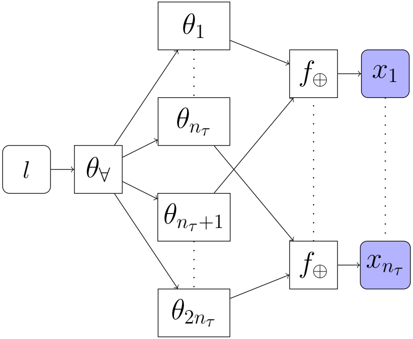


Figure: Multi-task design with high-level parameter sharing, d = 2

Layers $\theta_{n_{\tau+1}}$ to θ_{2n_τ} are collapsed into a single layer $\theta_{n_{\tau+1}}$

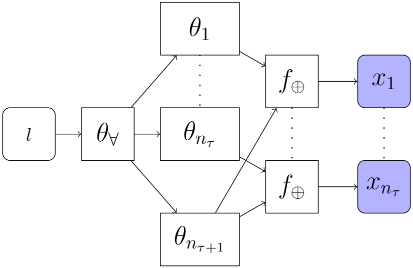


Figure: Multi-task design with high-level parameter sharing and shared randomness

Low-level parameter sharing
Not shared randomness

$\theta_{i*n_\tau+1}, \ \dots \ , \theta_{i*n_\tau+n_\tau}$ are fed to the respective prediction head $\theta_{d_{i+1}}$

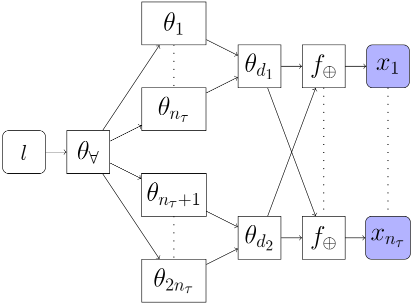


Figure: Multi-task design with high-level and low-level parameter sharing

Outline

Datasets and experiments

Datasets

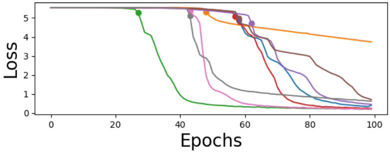
- ASCAD-r : raw traces with 250k samples
- ASCAD-v2 : extracted PoIs, permutations "disabled"
- Spartan-6 : extracted cycles of interest

Experiments

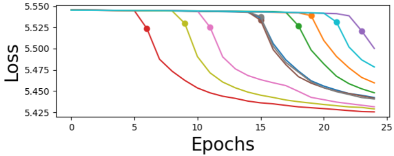
Collection of epoch of convergence with 10 different seeds

- Expert layer with shared mask (ASCADr, ASCADv2)
- Low-level parameter sharing (ASCADr, Spartan-6)
- Multi-target with shared mask (ASCADv2)

Collection of epoch of convergence



(a) ASCAD-r



(b) ASCAD-v2

Figure: Examples of the acquisition of the epoch of convergence for all seeds of one model type.

CHESCTF 2023, Spartan-6

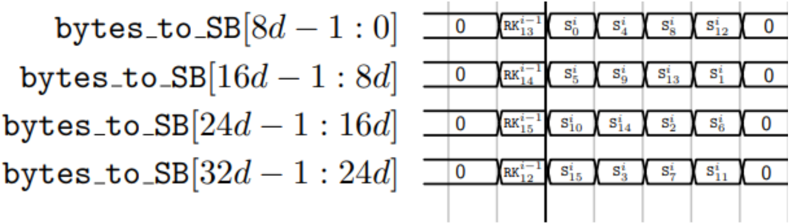


Figure: Dataflow on the Subbytes inputs wires

Leakage : Hamming distance

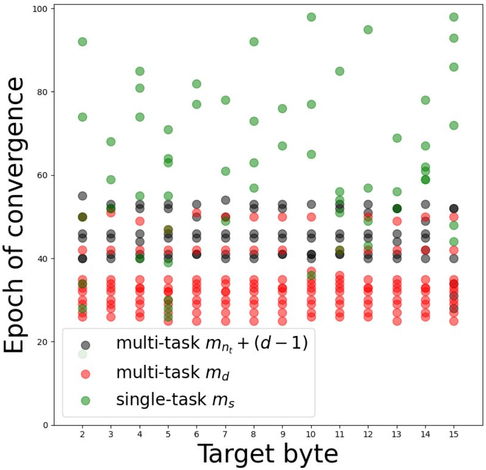
ASCAD-r : Expert layer with shared mask

Target : $t_i = (t_i \oplus r_i) \oplus r_i$

Models :

- Single-task: m_s
- Multi-task with expert layer: $m_{n_t+(d-1)}$
- Multi-task with expert layer, low-level parameter sharing: m_d

ASCAD-r : Expert layer with shared mask
Epoch of convergence



UNIVERSITÄT KLAGENFURT
erc European Research Council

ASCAD-r : Expert layer with shared mask
Performance metrics

Model type	f_r	$n_{\mathrm{win}}/n_{\mathrm{seeds}}$	$\overline{T_{\mathrm{win}}}$	*best* T_{win}
m_s	0.56	0.0	$>$100	$>$100
$m_{n_t+(d-1)}$	0.4	0.6	**4.33**	4
m_d	**0.0**	**1.0**	5.8	**2**

- f_r = failure rate of a training run across all bytes
- n_{win}/n_{seeds} = ratio of seeds leading to full key recovery
- T_{win} = Trace at which the full key is recovered

UNIVERSITÄT KLAGENFURT

erc European Research Council

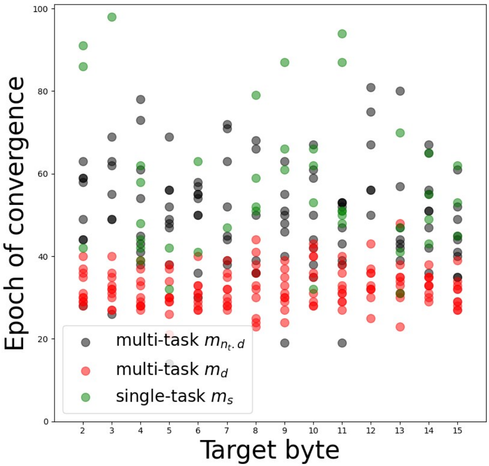
ASCAD-r : Low-level parameter sharing

Target : $s_i = (s_i \oplus r_i) \oplus r_i$

Models :

- Single-task : m_s
- Multi-task : $m_{n_t.d}$
- Multi-task with low-level parameter sharing : m_d

ASCAD-r : Low-level parameter sharing
Epoch of convergence



UNIVERSITÄT KLAGENFURT

	ASCAD-r			
Model type	f_r	$n_{\mathrm{win}}/n_{\mathrm{seeds}}$	$\overline{T}_{\mathrm{win}}$	*best* T_{win}
m_s	0.69	0.0	>100	>100
$m_{n_t.d}$	0.21	0.4	**6**	5
m_d	**0.0**	**1.0**	2.13	**2**

- f_r = failure rate of a training run across all bytes
- n_{win}/n_{seeds} = ratio of seeds leading to full key recovery
- T_{win} = Trace at which the full key is recovered

Spartan-6 : Low-level parameter sharing

Target : $t_i = (t_i \oplus r_i) \oplus r_i$

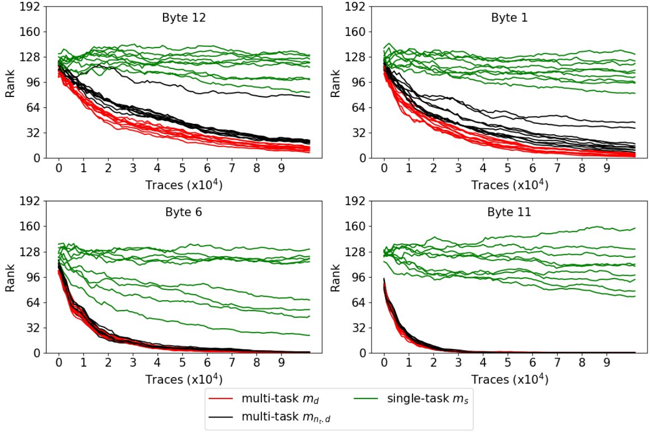


Figure: True rank evolution for all targeted bytes across all seeds and models

ASCAD-v2 : Multi-target with shared mask

Targets :

- $t_i = r_m \otimes^{-1} ((r_m \otimes t_i \oplus r_{in}) \oplus r_{in})$
- $s_i = r_m \otimes^{-1} ((r_m \otimes s_i \oplus r_{out}) \oplus r_{out})$

Models :

- Multi-task, single target trained only for t_i: m_{st-d}
- Multi-task, multi target: $m_{n_t+(d-1)}$
- Multi-task, multi target, and low-level parameter sharing : m_d

ASCAD-v2 : Multi-target with shared mask
Performance metrics

Model type	$n_{\mathrm{win}}/n_{\mathrm{seeds}}$	$\overline{T_{\mathrm{win}}}$	*best* T_{win}
m_{st-d}	0.0	>200	>200
$m_{n_t+(d-1)}$	0.0	>200	>200
m_d	**0.5**	**17.6**	**16**

- f_r = failure rate of a training run across all bytes
- n_{win}/n_{seeds} = ratio of seeds leading to full key recovery
- T_{win} = Trace at which the full key is recovered

Conclusion

What I am not saying

- Multi-task is always more performant than single-task
- Sharing layers is always beneficial

What can be concluded

- Multi-task needs less profiling traces to converge consistently
- Careful constraints are helpful to improve the latter point
- Multi-target strategies can be implemented even during training
- Multi-task make evaluations faster

Acknowledgments

- Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 725042)

Caruana, R.: Multitask learning. In: Thrun, S., Pratt, L.Y. (eds.) Learning to Learn, pp. 95–133. Springer (1998). https://doi.org/10.1007/978-1-4615-5529-2_5, https://doi.org/10.1007/978-1-4615-5529-2_5

Eldib, H., Wang, C., Taha, M., Schaumont, P.: Qms: Evaluating the side-channel resistance of masked software from source code. In: Proceedings of the 51st Annual Design Automation Conference. p. 1–6. DAC '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2593069.2593193, https://doi.org/10.1145/2593069.2593193

Masure, L., Cristiani, V., Lecomte, M., Standaert, F.X.: Don't learn what you already know: Scheme-aware modeling for