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

Thomas Marquet[1] and Elisabeth Oswald[1,2][0000−0001−7502−3184]
first-name.last-name@aau.at

[1] Digital Age Research Center (D!ARC), University of Klagenfurt, Austria
[2] University of Birmingham, UK

Keywords: Side Channel Attacks · Masking · Deep Learning · Multi-Task Learning

Abstract. This paper investigates different ways of applying multi-task learning in the context of two masked AES implementations (via the ASCAD-r and ASCAD-v2 databases). Enabled by multi-task learning, we propose novel architectures that significantly increase the consistency and performance of deep neural networks in a context where the attacker can not access the randomness of the countermeasures during profiling. Our work provides a wide range of experiments to understand the benefits of multi-task strategies against the current single-task state of the art. We show that multi-task learning is significantly more performant than single-task models on all our experiments. Furthermore, such strategies achieve novel milestones against protected implementations as we propose a new best attack on ASCAD-r and ASCAD-v2, along with models that defeat for the first time all masks of the affine masking on ASCAD-v2.

1 Introduction

Deep learning techniques have fast become an alternative to the use of classical statistics in the context of profiled side-channel attacks, because of their unrivalled ability to efficiently utilise information across many trace points. The approach taken by many deep learning architectures still somewhat depends on the thinking found in traditional statistics-based attacks: a single intermediate target is learned at a time (thus a single learning task is performed).

Recent publications have begun to move beyond this single-task learning paradigm towards a multi-task learning approach: Mahgrebi [4] explores a deep learning architecture to learn two intermediate values (bit-wise) on an AES implementation simultaneously; Masure and Strullu [7] revisit Mahgrebi's idea and learn many intermediate values simultaneously. They set a new record for a "non-dissecting" approach for the ASCAD-v2 dataset and successfully recover the key bytes with 60 traces when assuming knowledge of the masks during profiling. Their paper concludes by reflecting on the potential power of multi-task learning: "A further study of the advantages and drawbacks of such paradigm

is yet to be done. Still, this could lead to help the SCA practitioner towards new milestones against protected implementations." (p. 21, [7]). Marquet et Oswald [5] further entertain this idea in their work, where the authors claim that multi-task learning models have an edge over single-task models in a scenario where knowledge of the masks isn't assumed.

1.1 Summary of Contributions and Outline

We focus on the application of multi-task learning in the context of the masked AES-128 implementations that are the basis of the ASCAD-r and ASCAD-v2 databases introduced in Prouff et al. [11]. After providing some notation and background in Sect. 2, we introduce our multi-task designs along with the hyperparameter choices in Sect. 3, we present our experimental results in Sect. 4 on both datasets, and finally we discuss how to extract points of interest from traces using our designs in Sect. 5. Whereby our innovations are as follows:

Contributions

- we propose to leverage multi-task learning to enable collaboration between different intermediates, and/or different bytes of the same intermediates.
- we suggest that multi-task learning allows an attacker to leverage constraints to "guide" the learning of the model.
- we provide experimental evidence that such constraints are beneficial, on the overall performance of the model, but also its convergence speed.
- we explain how to recover points of interest from raw traces using our designs.
- we compare those architectures against state-of-the-art single-task designs.
- we provide new best attacks for ASCAD-r and ASCAD-v2, with the first model to defeat both multiplicative and additive masks in the affine masking scheme of ASCAD-v2.

1.2 Relevant related works

Masure and Strullu [7] introduce the ASCAD-v2 database and provide a first characterisation of the included traces. They provide the first attacks on the ASCAD-v2 dataset, showcasing multiple scenarios including experiments where part of the countermeasures are unknown to the attacker.

Hu et al. [3] explains that it can be beneficial to use the data from the processing of the AES state bytes to train a single model representing an intermediate value. This is possible in the case of many software implementations because each state byte undergoes the same operations (the same sequence of Assembly instructions) which means that their leakage is very similar. Ngo et al. [8, 9] shows a similar technique to reduce the size of the dataset to attack a masked Saber implementation.

Ngo et al. [8] and later on, Masure et al. [6] consider the possibility of assuming the presence of masking during the training of two models and propagating a

loss on the combined probabilities from both outputs. Such training relieves the network by giving it a better understanding of what it should learn. The first authors, however, present bit-wise designs, while the latter's designs are over one hot encoded byte.

Perin et al. [10] investigate the impact that the selection of points of interest has when training deep networks. They observe that working with raw traces is sometimes possible (i.e. no points of interest are selected), which leads to a black box attack scenario, where an adversary needs no information about randomness during the training. They provide the best results for the ASCAD-v1 database: they achieve key recovery with just a single trace in many scenarios.

Marquet and Oswald [5] propose extensive experiments to compare single-task and multi-task learning strategies. Solely focusing on a scenario where countermeasures are not assumed, the authors show that multi-task models yield more often models that converge towards successful attacks, however, the authors only explore scenarios where all targeted intermediate share the same mask.

2 Preliminaries

We consider side-channel attacks that operate in two stages: a leakage identification stage where (if necessary) points of interest are selected and deep nets are trained, and a leakage exploitation stage, where the trained nets are used as classifiers in the context of differential side-channel attacks.

We stick to as simple notation as possible and stay with the variable naming conventions of the ASCAD databases: upper case letters denote sets (which we overload and simultaneously use as random variables), and lower case letters denote realisations of the random variables (and equivalently elements of a set). All variable/set names are taken (without renaming) from the original papers (implementations/data sets), such that "matching up" of our work with these original implementations is straightforward. The index i refers to the ith state byte, and we generally drop any indexing referring to points within a trace from our notation. Occasionally we require to refer to the j-th trace, which we put as an index (alongside the index indicating the state byte) to a variable.

2.1 Profiling based on Deep Learning

For the purpose of building a classifier for newly observed traces during the exploitation phase, a deep learning approach uses one (or more) trained models, which output values that can be understood as likelihood scores. In the context of our work, we are interested in recovering information about key values. Thus, our networks are configured to return per-trace log-likelihood scores S_i for 8-bit chunks of an AES secret key. To derive the log-likelihood score for the ith key chunk given an attack set of N_a traces, we just compute the sum $d[k_i] = \sum_{j=1}^{N_a} S_{i,j}$.

Training Methodology We use the same methodology across all datasets. To enable meaningful comparisons, we use the same overall architecture for single models and multi-task models, with the same learning rate and optimizer. The only difference between the models is how the branches are connected.

As per good practice, we divide the available data into training data, validation data, and attack (=test) data. All training happens on the training data set. We validate a learned model on a validation set of size N_v. During this validation phase, we monitor the validation accuracy. Our best training model is selected based on the best validation loss, and we use a Tensorflow callback to retrieve this model. We then test the best model on the full N_a attack dataset, retrieving accuracy-based metrics.

2.2 Computing resources

We're using a single GPU Nvidia A6000 Ada with 48GB of dedicated memory. In addition to the GPU, we're using 4 cores of an AMD EPYC at 2.6GHz with 128 GB of RAM. All that is running on an Ubuntu 22.04.1 kernel, with Tensorflow 2.9.1 and cuda 11.2.

2.3 Data Sets and Corresponding Notation

Our work is based on the ASCAD datasets, which are both based on masked AES implementations. We assume familiarity with low-order masking, as well as typical software implementations of low-order masking on standard microcontrollers, as we keep the following text as short as possible.

ASCAD-r The original ASCAD database (v1) features one data set of a masked AES implementation (on a simple 8-bit microcontroller) with varying keys, which we utilise in our work. The database is generous, each side channel trace offers many data points for inclusion in training. We select in the dataset, 110k traces from the random key split for training (N_t = 100k traces) and validation (N_v = 10k traces), and 10k from the fixed key split for the attack dataset (N_a = 10k traces).

The datasets contain the information that relates to the masked computation of the AES SubBytes operation. The masking scheme is a simple two-share scheme, which precomputes a masked AES SubBytes Table $SubBytes^*$ prior to encryption. The masked SubBytes table is defined as $SubBytes^*[x] = SubBytes[x \oplus r_{in}] \oplus r_{out}$. During the computation of a masked encryption round, all state bytes t_i (i refers to the state byte index) are masked by a state mask r_i. Prior to the masked SubBytes step, the state bytes are remasked, so that the input to $SubBytes^*$ is masked by r_{in}, and because of the definition of $SubBytes^*$, the corresponding output is masked by r_{out}. The SubBytes output is then again remasked so that it is protected by the state mask r_i. The accompanying write-up for the database already performs an analysis to highlight the leakiest intermediate variables, which are the masked input and output of the SubBytes operation

$(t_i \oplus r_{in}, s_i \oplus r_i)$ as well as the two involved masks r_i and r_{in}. Whilst the output mask r_{out} and the masked intermediate $s_i \oplus r_{out}$ also leak, their leakage is weak and hence typically ignored.

There have been several papers that reported, for a variety of network architectures and approaches, results for this database. Our approach is to work with the raw traces (thus no points of interest selection take place). With this setting in mind, the best previous work is [10], which reach single trace success one some key bytes — culminating in 3 traces for the most resilient key bytes.

ASCAD-v2 The ASCAD-v2 dataset contains traces from a masked and shuffled AES implementation (on a more complex 32-bit architecture). The full dataset contains 800k traces with random keys and inputs. Each trace has 1 million sample points: therefore we extract only a subset of the available points for training/attack purposes. After shuffling all traces, we split the available data into training (N_t = 450k traces), validation (N_v = 45k traces), and attack (N_a = 5k traces) data sets.

The masking scheme is slightly more complex. It uses both a non-zero multiplicative mask β, as well as a Boolean mask α, i.e. each intermediate value x is represented by three shares: $(x \otimes \beta \oplus \alpha, \beta, \alpha)$ (the multiplication must be understood over the appropriate finite field). The SubBytes operation is based again on a pre-computed table. Shuffling happens throughout the encryption rounds: a permutation over 16 elements is used for all-round operations bar MixColumns, in which only the column elements are permuted. The permutation affects the index of the state bytes.

The specific notation for the intermediate values is akin to the notation in ASCAD-r and works as follows. The variable t_i denotes the i-th state byte before the SubBytes operation, s_i is the result of SubBytes. Key bytes are denoted by k_i. The multiplicative mask is called r_m (it is the same for all state bytes), and the additive masks are called r_{in} (before SubBytes), r_{out} (after SubBytes), and r_i (everywhere else).

The point selection strategy (based on computing the SNR of intermediate values) detailed in Masure and Strullu [7], even though sufficient to perform successful attacks, omits the inclusion, of what we found to be the leakiest part of the implementation: much leakage about the input mask r_{in} is not included. Therefore we extract our datasets.

We perform a subsampling, using a moving average in the way of Perin et al. [10] on ASCAD-r. This divides the number of total samples by 4. For the masks r_m, r_{in}, and r_{out}, we select an arbitrary number of points with the highest SNR. For the masked intermediates $r_m \otimes s_j \oplus r_{out}$ and $r_m \otimes t_j \oplus r_{in}$, we extract the S-box operation samples, which happens to be where both our targets are leaking. This corresponds to 93 samples per byte. **Permutations are disabled in this work**, as their access is assumed during profiling and attack.

This dataset has successful attacks in two situations where the countermeasures are toned down. The first one by Masure et Strullu. [7] has a successful attack without requiring the knowledge of the permutations, but also without

requiring the knowledge of the multiplicative mask. The paper Marquet et Oswald. [5], provides successful attacks when the additive mask r_{in} is unknown. The best results for a full key recovery attack depend on the scenario. For a scenario where knowledge of masks and permutations is assumed during profiling, but not during an attack, the best attack of Masure et Strullu. [7] takes 60 traces. The best attacks of Marquet et Oswald. [5] take 21 traces, assuming knowledge of permutations, and the multiplicative mask r_m during attack and profiling. Among the results we provide, such a scenario is explored in Sect 4.1.

2.4 Breaking free of the "Plateau"

Masure et al. [6], discuss the problem that the training of deep learning models faces when applied in a side-channel context, in the presence of masking. In the first epochs, the learning of the model is very slow and suddenly blows up. The gradient descent is stuck, as no single point in the trace is giving up information about the labels given to the model. Since no point is leaking the information it is looking for, the network first has to understand which points are useful to its task, and then how to combine them. We show examples of such plateaus based on our following experiments in the Figure 1.

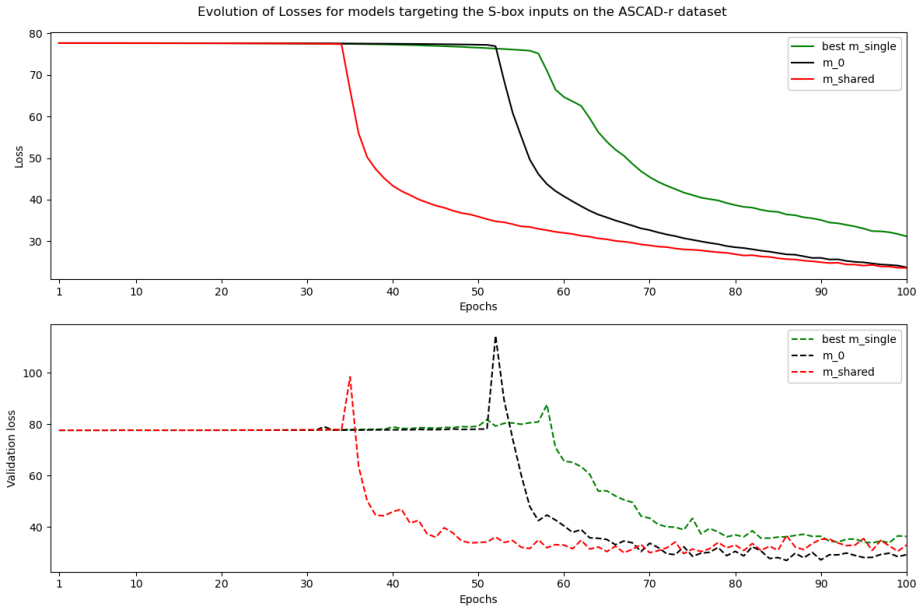


Fig. 1: Example of models breaking through the "plateau" during the experiments from Sect4.1

Overcoming this initial challenge is a precondition to a successful network during inference. Therefore, when defining deep learning architectures, we argue that the ability of a design to consistently break through the initial plateau should be a defining criterion to judge the quality of said design. With this idea in mind, we take a special interest at which epoch the model breaks through the plateau since it is the moment when the model gathered enough knowledge to understand how to overcome masking. This epoch can be clearly identified as seen in Figure 1, the losses vary significantly at those points.

2.5 Metrics

The metrics of interest are the following :

- **Number of traces to reach rank(key)** $< 2^1$ **.** We note it T_{win}.
- **Ratio of seeds leading to a full recovery of the key.** We note it $n_{\text{win}}/n_{\text{seeds}}$.
- **Epoch of convergence.**

2.6 Custom layers: Xor and inverse multGF256

Introduced in Masure et al. [6], as a custom layer performing conditional probabilities between the softmax layers of two models trained during the same process. Our iteration of this layer performs the following computation given two vectors x and y of size 256 :

$$f_{\oplus}(x,y)[i] = \sum_{j=0}^{255} x[j] \times y[i \oplus j] \quad \forall\, i \ \in \ [0,255] \tag{1}$$

$$f_{\otimes}(x,y)[i] = x[0] + \sum_{j=1}^{255} x[j] \times y[i \otimes j] \quad \forall\, i \ \in \ [0,255] \tag{2}$$

The function $f_{\otimes}$ has to discriminate the first case where $j = 0$, being a null element. We decided that in this case, the probabilities of x should be unchanged.

3 Multi-Task Learning

Multi-task learning has been introduced by Caruana [1], and has become the state of the art in many pattern recognition domains. Given two tasks x_a and x_b, a multi-task model can help in the following ways:

- Input explainability : Without the labels from x_a, it's signal will be noise to x_b, and this both ways. This might be a problem when the inputs are large and/or the training set is reduced.
- Noise cancellation: If x_a and x_b share features, the gradient will be averaged over both tasks, therefore reducing the noise.

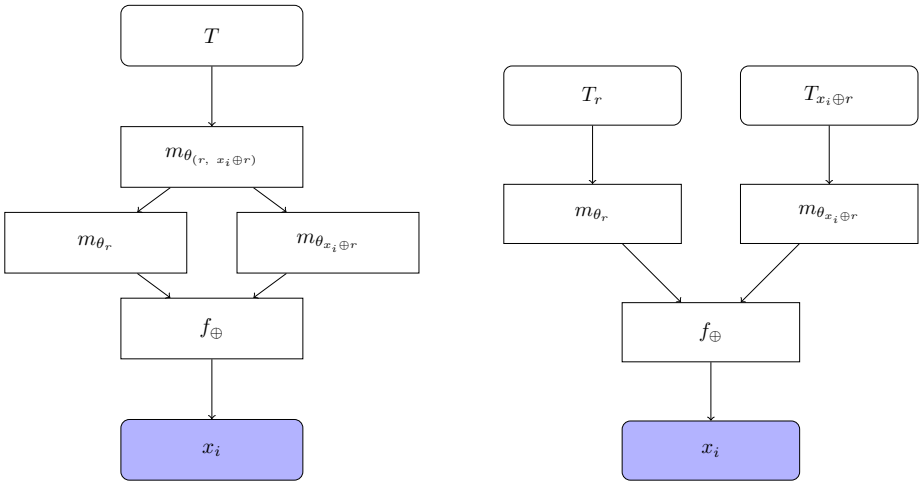
– Eavesdropping : x_a might share features with x_b. Let's say that x_a has a stronger signal and is easier to learn than x_b. Then, training both at the same time is beneficial for x_b
– Regularisation effect: The overall gradient being composed, it will rather go in valleys that are beneficial for all tasks. Effectively restraining the shared weights towards a representation that is good for all tasks.

In the deep learning community, Mahgrebi [4] was the first to pick up on the idea of multi-task learning. An improved design by Masure and Strullu [7] achieves impressive results for the ASCAD-v2 database. The core idea behind the existing architectures in these two previous works is that each intermediate value is learned by an independent branch of the deep net and that all branches are connected to several shared layers dealing with the higher-level features. This is the canonical design of multi-task networks, as summarised in [12]. Even though the work Masure and Strullu [7] introduces multi-task learning in a scenario where randomness is not known, their designs are not taking advantage of the idea of Masure et al. [6], which demonstrate the benefits of layers that perform combined probabilities between two branches of a network to encode the masking scheme in the architecture. Marquet et Oswald. [5] showcase the benefits of taking advantage of such principles in a multi-task architecture. Demonstrating the superiority of multi-task learning through the many advantages it has over single-task learning, at least in a scenario where masks are unknown.

With profiled attacks, the most challenging setting is the one where knowledge of the countermeasures is not assumed. In the context of masked implementations, we would then assume that —because of a lack of access to internal randomness— the training data cannot be labeled with masks or masked values, but only the (unmasked) intermediate values. Again, due to the absence of randomness information, a point of interest selection might not be feasible. Given that the application of multi-task learning to masked implementations is based on designing branches that learn masks and masked values, it is non-trivial to come up with a way to apply multi-task learning when masks are unknown. For this reason, we target multiple bytes at the same time to leverage common features between the masks of the targets. For example, a mask might be shared across bytes, but also, in the case of a state mask, the leakage of each byte of the mask might be related to the others.

3.1 Single-task designs

We define a single-task model as a model trained using the knowledge of only one label. In our scenario, where access to internal randomness is not assumed, this means a model labeled with the unmasked value of an intermediate. State-of-the-art single-task designs against masked implementations are composed of two branches networks in the like of Masure et al. [7] and Ngo et al .[8]. One branch is fed the mask leakages, while the other, the masked intermediate, and the branches are regrouped by a layer using conditional probabilities.



(a) Single-task models on ASCAD-r (b) Single-task models on ASCAD-v2

Fig. 2: Single-task architectures

On the ASCAD-r, we assume we cannot extract points and use the raw traces. To process the raw traces, the first layers are shared by both branches, then split apart to finally be combined at the end. On the ASCAD-v2 dataset, as the samples are extracted to reduce the dataset size and ease the problem, we leverage this and feed to each branch, only the samples related to the distribution expected to be learned. Such a principle will also be used in the multi-task designs on ASCAD-v2.

3.2 Multi-task designs

Naive modelling (noted m_0). The idea of splitting the network into two sets of layers respectively expected to fit respectively $x \oplus r$, r and then regrouping them with $f_\oplus$ can be naively applied to multi-task learning. However, it can be improved to maximise the sharing of features. In the very specific case where masks are shared among all bytes of the targeted intermediate, one can very successfully design a model such as in Figure 3. This model has an "expert" branch for the mask that is shared among all tasks and makes a bridge between them to allow collaboration. The $f_\oplus$ layer is acting as a constraint already in the single-task scenario, forcing each branch to take a very specific representation (conditional probabilities). The cumulative effect of those constraints, thanks to multi-task learning, is a natural improvement as showed in Marquet et Oswald [5]. However, in the case where masks are not shared, one has to find other ways to further leverage the multi-task process.

Hard-parameter sharing. Hard parameter sharing is simply when multiple tasks share a set of layers. By this logic, sharing convolutions or layers at the beginning of the network is already hard-parameter sharing. However, sharing

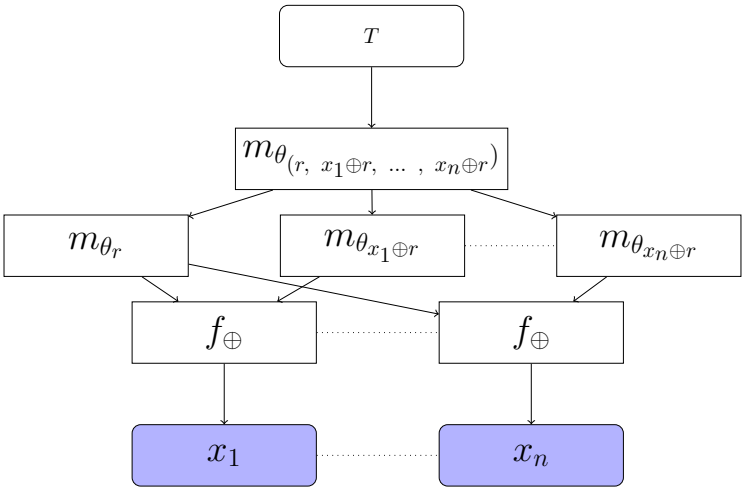


Fig. 3: Shared mask architecture

layers close to the input is usually made to share higher-level features. Going further down the network with shared layers means on the contrary sharing lower-level features. The difficulty of sharing lower-level features comes from the fact that the output must be different but obtained with similar input and the same weights. To do so, one has to separate the network into multiple channels, either by splitting the input inside the network, or through mid-level unshared layers. The design used in this paper is the latter and presented in Fig 5. Therefore, the network is built in a sequence of shared-unshared-shared layers.

Shared branches models (noted m_{shared}). Using the idea presented in the last paragraph, we design models that maximise the sharing of weights for all tasks. Such networks reduce the overall number of weights in the model while keeping the same number of weights per task. We believe this idea is key to reducing the difficulty to pass the "plateau" induced by the masking countermeasures. As the difficulty increase exponentially with the number of shares, it is crucial to maximise the propagation of the knowledge that one can assume. Furthermore, this reduces the amount of representation weights can take, as they have to satisfy all the bytes instead of just one, which increases consistency between different initialisations. Such low-level sharing works with minimum impact on total performance as the same intermediate bytes share a certain amount of low-level features. Strategies, where one leverages the common features between bytes, are very successful in a single-task scenario [3, 8, 2, 9]. Our strategy is the direct adaptation of such a technique in a multi-task learning scenario. However, our modeling does not need the extraction and alignment of each byte leakage in the trace, as it is done by the network instead of being a pre-processing step.

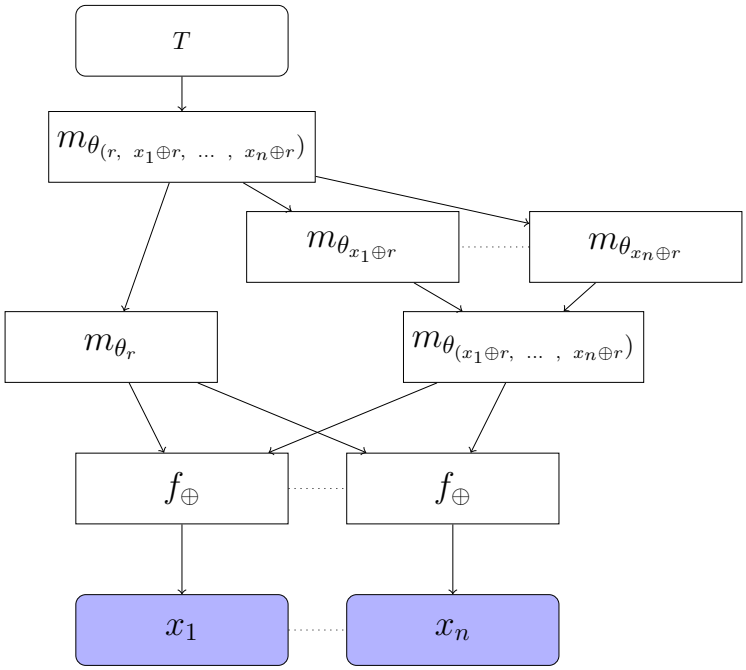


Fig. 4: Shared mask architecture using hard-parameter sharing

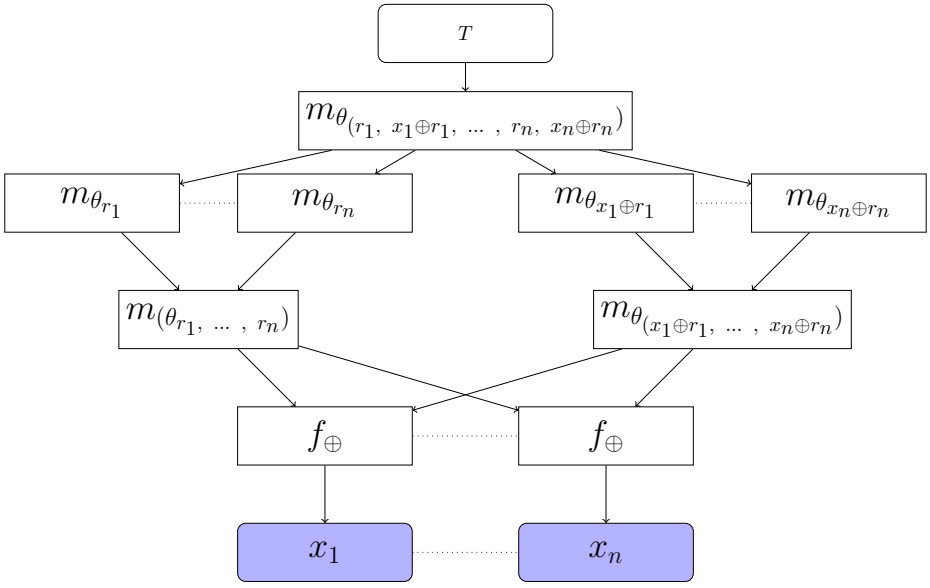


Fig. 5: Not shared mask with hard parameter sharing on both sides

3.3 Hyperparameters

ASCAD-r We build different models for the ASCAD-r database. In order to have a meaningful comparison, we chose the same core hyperparameters from the single-task models to the most complex multi-task architectures. One can see the single-task models as submodels from the corresponding multi-task models. Or the other way around, the multi-task models are simply single-task models branched together. The chosen core hyperparameters are listed in the following:

- **Weighted Pooling** Inspired by Perin et al. [10] we are using custom layers to perform a weighted average pooling on the raw traces to reduce the size of the following network. We are pooling a total of 4 times to reduce the size from 250k samples to 15625 points. After each average pooling, we perform a batch normalization and an alpha dropout.
- **Convolution block.** We use the convolutions from a CNN proposed by Perin et al.[10]. It consists of only one convolution layer (kernel 34, strides 17, and filters 4) followed by average pooling (pool size 2) and batch normalization.
- **Dense block.** Each prediction branch possesses 2 dense layers of 200 units and one output layer followed by a softmax. The units in the dense layers are activated using a SeLu function.
- **Training hyperparameters.** All models are trained for 100 epochs, using a fixed learning rate of 0.001 with an Adam optimiser.

ASCAD-v2 The ASCAD-v2 database is considerably newer and therefore much less analysed. In Masure et Strullu [7] the authors provide an excellent characterisation of the traces, and we took full advantage of this information in our work. Since the dataset is extracted, we choose a multi-input architecture as defined in 2b. Each branch x of the network will learn from a different input with a different set of hyperparameters θ_x. We define the most basic hyperparameters for the single-task models and by extension the multi-task models in the following :

- **Mask branches.** Leakages from the masks are strong and dispersed on many samples, therefore we choose very simple CNNs composed of a single convolution/pooling/batch-normalisation block. Kernel size is fixed to 32, filters to 16, and pooling size to 5 for all of the branches, while strides are chosen to scale based on the input size of each branch, respectively 10, 5, and 2.
- **Intermediate branches.** The input of the intermediate branch is the full execution of the S-box operation from the first round as 32-bit leakages are present. Therefore it is necessary to include the computation of the next bytes to capture the most information. The intermediate branches are 3 layers MLP with (64,8,8) units and a batch normalisation after each layer. The branch ends on a fully connected layer of 256 units.

– **Cyclic training.** All models are trained with 3 cycles of respectively 25, 4, and finally 1 epochs, with a decreasing learning rate at each cycle. The learning rates are 0.001, 0.0001, and finally 0.00001 with an Adam optimiser. This allows us to annihilate overfitting and maximise the performance of our models.

4 Experimental results

Throughout our experiments, we wish to discuss the performance of our designs against single-task learning but also the improvement from sharing weights at a lower level of the network. To do so, we train our designs 10 times using a different seed of initialisation. Effectively observing the resilience of the different architecture against multiple starting points. Each design possesses the same number of weights for each task, the difference is the total amount of weights as some weights are used for different targets. Multi-task models m_0 are single-task models that share high-level weights. While m_{shared}, are the same models sharing also low-level weights. We hope to entertain a discussion in the community about the importance of collaboration across many signals. As it is a way to efficiently utilise all the information available in one trace, to maximise the chances of breaking through the plateau effect.

4.1 Leveraging shared masks across bytes of the S-box operation

The idea with this model is to leverage the leakage of masks that are shared between bytes of a targeted intermediate. Fortunately, on both ASCAD datasets, such weaknesses are found. On ASCAD-r, all bytes of the SubBytes inputs share a strongly leaking mask r_{in}, and on ASCAD-v2, all bytes share r_{in} for the S-box inputs, and r_{out} for the S-box outputs. We train an architecture with a common branch for the mask we need the model to learn. This shared branch will act as an "expert" specialised to fit the distribution of the mask and is connected to all the others with a xor-like ($f_{\oplus}$) layer. The idea is that while learning $((x_i \oplus r) \oplus r)$, it might be beneficial to learn at the same time $((x_{i+1} \oplus r) \oplus r)$. This repeated for all the attackable bytes.

ASCAD-r On this dataset, the targeted leakage pair is $(t \oplus r_{in}, r_{in})$. Using the raw traces, we train a multi-task model based on the design in Figure 3 noted m_0, and a multi-task model based on Figure 4 that we note m_{shared}. Finally, we train 14 single-task models (m_{single}) according to the design in Figure 2a. We show a scatter plot of the epochs of convergence for each target byte and all approaches. Then, we perform a full key recovery attack, 1000 times over 100 randomly picked raw traces from the attack dataset, and note the results in Table 4.1. We include in this table the ratio of seeds $n_{\text{win}}/n_{\text{seeds}}$ leading to a successful attack, along with the average performance of successful seeds $\overline{T_{\text{win}}}$, and finally the performance of the best seed *best* T_{win}.

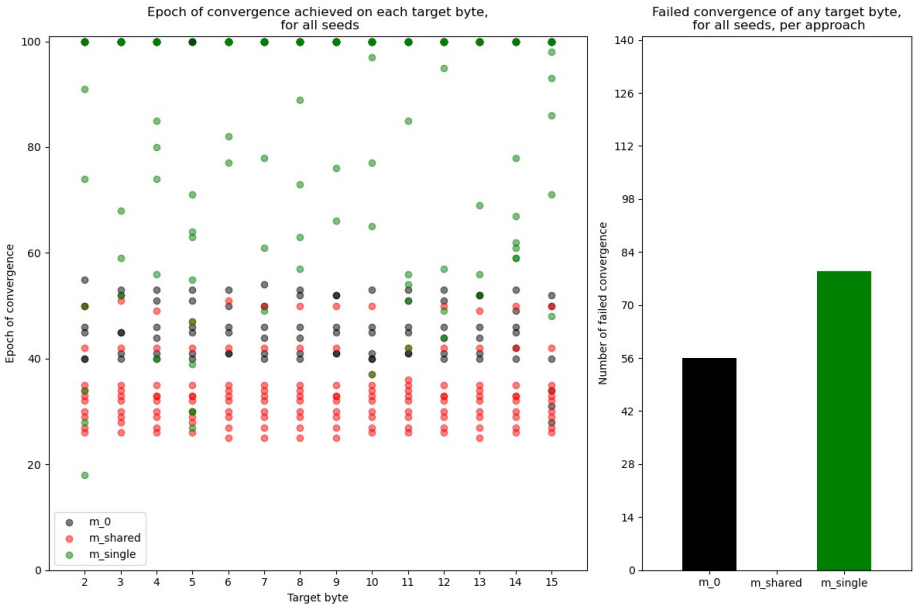


Fig. 6: Epoch of convergence and failure rate, per target byte and seed, for models targeting the S-box input pair $(t \oplus r_{in}\ ,\ r_{in})$

Observing the epoch where each model breaks free of the plateau caused by masking, we first see that the learning of single-task models varies greatly. Depending on the seed, the m_{single} converges around the epoch 20, 30, or 70, or not at all. We can see that such erratic behavior is erased from multi-task models. Many multi-task-induced effects can be the cause of such consistent behavior. The first one is the regularisation from the shared mask. All bytes collaborate to learn the mask, and therefore the branch of the mask benefits from the average of multiple gradients, effectively canceling noise in the inputs. In addition, the weights from individual bytes are not free to explore representations that do not benefit others. This effect is further reinforced by the sharing of low-level weights. As the model m_{shared} improves significantly the baseline model m_0, respectively reaches convergence for most seeds at epoch 25-30, and 40-50. Moreover, while the baseline model m_0 often fails to converge, the model using hard-parameter sharing consistently converges toward significant learning.

No seed allowed the single-task models to converge on all bytes. As training a deep net is a stochastic procedure, depending on the initialisation some bytes might benefit from a better starting point to pass through the increased complexity of masking. Multi-task procedure makes it more resilient to such problems because of the effect of multiple gradients. Among the multi-task models, we see that hard-parameter sharing impacts overall negatively the performance in this experiment. As the model m_{shared} possesses fewer weights, the maximum

Table 1: Performance metrics for the experiment leveraging the S-box input leakage pair $(t \oplus r_{in}\ ,\ r_{in})$ on ASCAD-r

Model type	$n_{\text{win}}/n_{\text{seeds}}$	$\overline{T_{\text{win}}}$	*best* T_{win}
m_{single}	0.0	>100	>100
m_0	0.6	**4.33**	4
m_{shared}	**1.0**	5.8	**2**

amount of knowledge it can contain is smaller than the baseline model m_0. Even though, one outlier run of the model m_{shared} recovers the full key in two traces consistently over all experiments.

ASCAD-v2 To further investigate the impact of constraints on multi-task models, we experiment with a scenario where only the additive masks r_{in} and r_{out} are unknown. We give the knowledge of r_m to the network during profiling and attack, reducing the masking scheme to first order. Therefore, we investigate two intermediates, with two different masks. The first target is the pair, masked S-box inputs $r_m \otimes t \oplus r_{in}$, with the mask r_{in}. The second is the pair of the S-box outputs, $r_m \otimes s \oplus r_{out}$ with the mask r_{out}. To increase the difference in performance between each approach, we reduce the size of the training dataset to only 225k traces. The architectures used in this experiment are the same as in the previous one, with a multi-input design since the dataset is extracted. This corresponds to the difference presented in 2. Again, we show a scatter plot of the epochs of convergence for each target byte and all approaches in Figures 8 and 7, and note the performance metrics in Table 4.1.

Looking at the Figure 7, we can observe another example of the inconsistency of single-task models m_{single}, as different seeds lead to drastic differences in terms of convergence. Even in this heavily simplified scenario, some seeds prevent single-task models to learn the given target byte. On the other hand, all multi-task models successfully converge, with most of them fitting properly the leakage at epoch 2-3. Interestingly, the m_{shared} seem to struggle on some seeds to converge toward their goal and only baseline multi-task models consistently do so at epoch 2-3. On the other intermediate however, Figure 8, we observe a similar scenario as on the ASCAD-r dataset, where oftentimes the baseline model does not converge at all, struggling to make sense of the samples. The performance of the m_{shared} model also coincide with the previous dataset, as it consistently outperforms the baseline model, and manages to converge on all seeds. Finally, on this intermediate, no single-task model managed to learn its target byte. Another example of the superiority of multi-task approaches is in this scenario where masks are shared among different intermediates.

Single-task models are successful on all seeds when performing the key recovery targeting the S-Box inputs. However, their performance are worse than both multi-task models. This is in line with the work in the same setup from Marquet

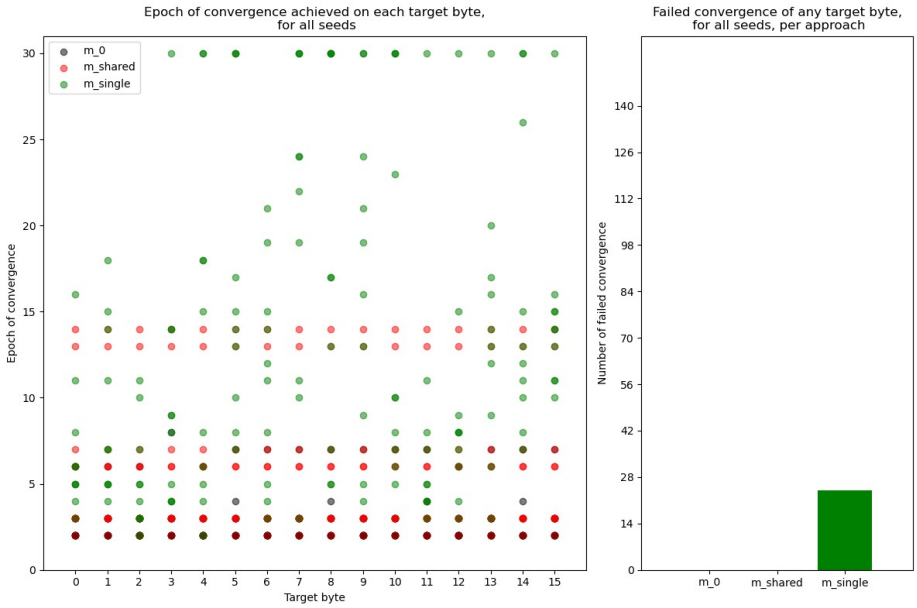


Fig. 7: Epoch of convergence and failure rate, per target byte and seed, for models targeting the S-box input pair $r_m \otimes t_j \oplus r_{in}$, r_{in})

Table 2: Performance metrics for two experiments leveraging first the S-box input leakage pair, and then the S-box output leakage pair on ASCAD-v2

	$(r_m \otimes t \oplus r_{in},\ r_{in})$			$(r_m \otimes s \oplus r_{out},\ r_{out})$		
Model type	$n_{\text{win}}/n_{\text{seeds}}$	$\overline{T_{\text{win}}}$	*best* T_{win}	$n_{\text{win}}/n_{\text{seeds}}$	$\overline{T_{\text{win}}}$	*best* T_{win}
m_{single}	**1.0**	29.5	25	0.0	>200	>200
m_0	**1.0**	**20**	**19**	0.6	134	98
m_{shared}	**1.0**	20.6	20	**1.0**	**118.1**	**92**

et Oswald .[5], as the regularisation effect on the mask branch improves greatly the network performance-wise. Similarly to the previous scenario, we see that baseline multi-task models m_0 are more performant than m_{shared} due to their greater number of weights. On the other hand, when information is scarcer due to lower leakages, the single-task models fail once again to recover the key, as was expected since no model converged. Moreover, the baseline multi-task model starts to show weaknesses, as its average, and best performance are slightly lower than m_{shared}. It has to be noted that the average performance is only calculated on the successful models, therefore the real delta is greater in favor of the shared parameters model.

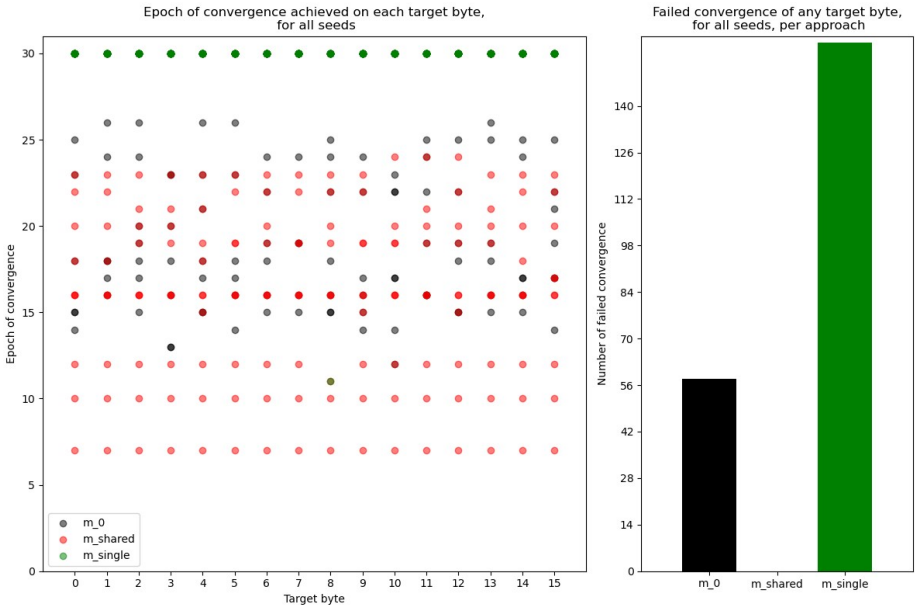


Fig. 8: Epoch of convergence and failure rate, per target byte and seed, for models targeting the S-box output pair $(r_m \otimes s_j \oplus r_{out} \;\; r_{out})$

4.2 Leveraging state masks with different values

When masks are not shared, it is not possible to train one expert shared among all tasks as in the previous section. However, it is still possible to use hard-parameter sharing, this time on both sides of the masking scheme in the manner of Figure 5. The idea is that sharing the weights on both branches will reduce the representations that can be taken by the model. This design is used by the model named m_{shared} in this experiment. The baseline multi-task model m_0 is naturally the same model but without hard parameter sharing. The single models (m_{single}) are again submodels of the latter design. The targeted leakage pair in this experiment is the S-box outputs with the state mask, i.e. $s \oplus r$, and r. We continue with the scatter plot of the epochs of convergence for each target byte and for all approaches in Figure 4.2, and note the performance of a full key recovery targeting the S-boxes output in the same setup as the previous experiment in Table 3.

Similar results can be observed in the previous scenario on ASCAD-r. Single-task models m_{single} are inconsistent and mostly converge after the multi-task models. Looking closely, one can see that not all bytes even converge in this experiment. Moving on to the multi-task models, we can see the epoch of convergence being a lot more inconsistent that in the previous experiments where the mask was shared by all bytes. This indicates that the shared mask had a strong impact on the training process. While for the single-task models, the

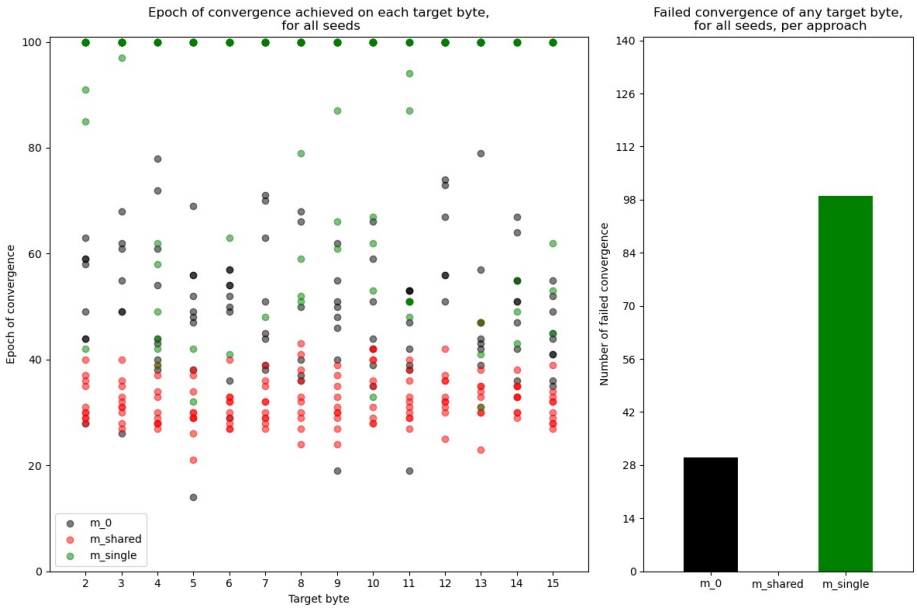


Fig. 9: Epoch of convergence and failure rate, per target byte and seed, for models targeting the S-box output pair $(s \oplus r_i \ , \ r_i)$

number of successful convergences is inferior that in the previous experiment, the baseline model m_0 learns overall seeds, more bytes. As the mask isn't shared anymore, individual branches are allowed to learn their target byte, when overall the model does not succeed in learning all of them. This problematic competition of losses is fixed in model design m_{shared}, as it successfully converges on all seeds. The sharing of weights managed to force collaboration between each byte, leading to consistent learning. We also observe a faster convergence for the latter model than any other one, once again hinting at its superior learning ability.

Table 3: Performance metrics for the experiment leveraging the S-box input leakage pair $(s \oplus r_i \ , \ r_i)$ on ASCAD-r

Model type	$n_{\text{win}}/n_{\text{seeds}}$	$\overline{T_{\text{win}}}$	*best* T_{win}
m_{single}	0.0	>100	>100
m_0	0.4	6	5
m_{shared}	**0.8**	**2.13**	**2**

Once again, no seed led to successful key recovery using the single-task models m_{single}. Furthermore, no seed allowed byte 12 to converge, and therefore even

by picking the best models across all seeds, a successful attack would not have been possible. The baseline model m_0 succeed on 4 seeds to recover the key, with an average performance of 6 traces. Finally, hard-parameter sharing successfully improved the success rate of multi-task models, as m_{shared} recovers the full key with around 2 traces on average. However, even though all seeds led to convergence, 100 traces was not enough for two seeds, as the learning suffered from too much overfitting.

4.3 Leveraging the S-box inputs and outputs to recover the shared multiplicative mask

On the ASCAD-v2 dataset, the affine masking scheme shares the multiplicative mask between $r_m \otimes s_j \oplus r_{out}$ and $r_m \otimes t_j \oplus r_{in}$. We design models that learn the unmasked S-box input and output at the same time, allowing us to take advantage of the shared multiplicative mask. We expect branches learning the different intermediates to collaborate on how to fit r_m. Based on this idea we train the two usual models, m_0 and its counterpart m_{shared} using hard-parameter sharing on the end layers. Finally, to understand the impact of training multiple intermediates, we additionally train a model without this "multi-target" approach. We note this model $m_{\text{t-shared}}$, as it learns only t_j through the triplet $(r_m \otimes t_j \oplus r_{in}\ ,\ r_m\ ,\ r_{in})$, using hard-parameter sharing. We note the main performance metrics after performing the usual full key recovery in Table 4.3, and plot the evolution of the losses in Figure 10.

The model $m_{\text{t-shared}}$, trained using only the labels from the unmasked S-box inputs, fail to converge even though the leakage from the triplet $(r_m \otimes t_j \oplus r_{in}\ ,\ r_m\ ,\ r_{in})$ is considerably higher than the second triplet linked to the S-box outputs. Moreover, the model m_0 using a multi-target strategy during training, also fails to converge even once. The only model converging multiple times is the model leveraging a multi-target strategy **during training** and low-level hard-parameter sharing m_{shared}. This feat, repeated 5 times while the other model never converges, is a testimony towards the importance of linking potential collaboration between intermediates even during training.

Table 4: Performance metrics against the full affine masking on ASCAD-v2

Model type	$n_{\text{win}}/n_{\text{seeds}}$	$\overline{T_{\text{win}}}$	*best* T_{win}
$m_{\text{t-shared}}$	0.0	>200	>200
m_0	0.0	>200	>200
m_{shared}	**0.5**	**17.6**	**16**

Observing the performances of the different models on a full key recovery attack, we see that every seed leading to convergence during training, also leads to successful attacks with good performances. Our best model, recovers the full

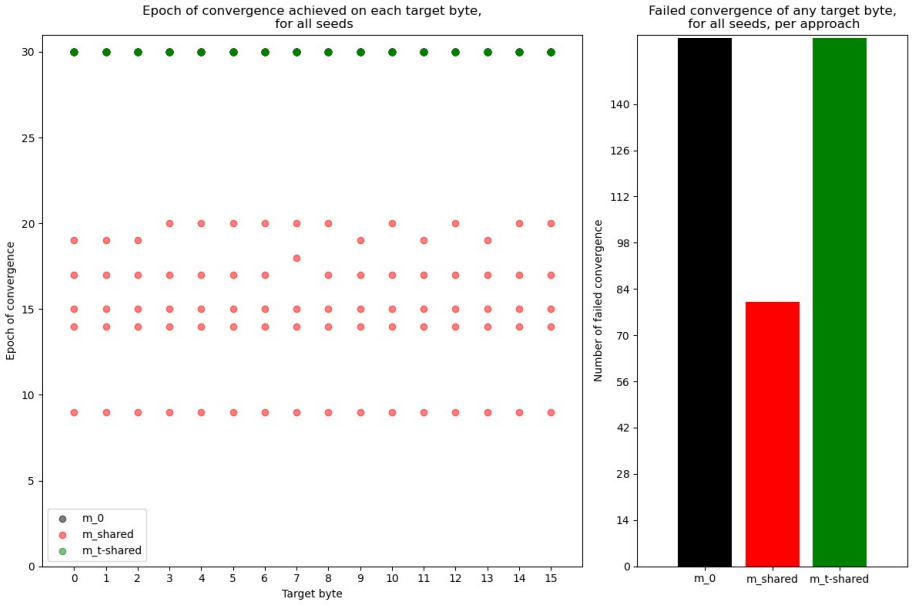


Fig. 10: Epoch of convergence and failure rate, per target byte and seed, for models targeting the full affine masking

key in only 16 traces, and is to the best of our knowledge, the best attack on ASCAD-v2, even in this simplified scenario, where PoIs from the masks are assumed, and permutations are disabled. We believe that strategies leveraging even more intermediates, for example the intermediates from the next rounds, would lead towards raw traces attack on ASCAD-v2. Even though one might not use those intermediates during the attack, the fact that the 10 rounds share the same masks would increase the regularisation factor by 10.

5 Explainability and point of interest extraction

Deep learning networks are often obscure boxes where explainability is given away in the hope of better performance. In side-channel analysis, a clear correlation between what is processed by the network and the input can be made. As the networks always leverage signals that could be observed if knowledge of the different shares was accessible. However, the classic designs struggled to be transparent as everything was processed by the same layers. Clever occlusion techniques have been designed to explain some of the mystery behind what has been learned. However, Masure et al. [6] showed that when training models with multiple branches such as our designs, one could recover at the output of those branches, a predicted probability distribution close enough to the true probability distribution of the targeted intermediates. Unfortunately, this distribution is

not giving away the values of the targeted intermediates as a δ is introduced on both sides of the xor operation, effectively canceling during the xor layer.

Using the model m_{shared} from the attack on the S-boxes input in Sect 4.1, we recover the points of interest of each targeted share. To do so, we extract the values going through the model at the end of each branch (i.e. inputs of the $f_{\oplus}$ layer). Those values are extracted after a softmax operation and therefore represent probability-like scores. Then, we "label" the traces using the value with the highest score. We evaluate using a simple SnR analysis with the predicted "labels" and plot in Figure. 11 the recovered correlations, along with the SnR analysis obtained with the real labels.

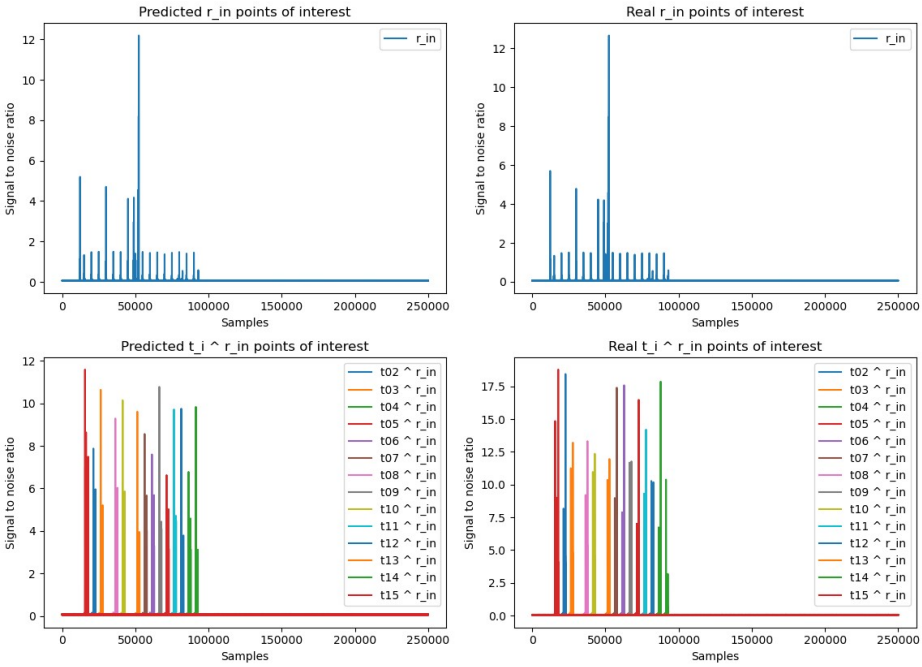


Fig. 11: Predicted SnR from the input mask r_{in}, and the corresponding masked S-Boxes inputs

To continue on both datasets, and with all intermediates, we select the best models trained in our previous experiments, collect their predictions, and label each trace with the value with the maximum score. Then, using the real labels from each intermediate, we perform an SnR analysis and collect the n_{poi} samples that have the highest signal-to-noise ratio. This leaves some points of interest aside, but make sure that the points with the most information are captured. On the ASCAD-v2 dataset, as the samples related to the masks are already extracted using SnR analysis, we are discussing the quality of the recovery of

the 10% most leaky samples. As we create fake labels from predictions of the deep nets, we unfortunately often label the traces with the wrong value. This isn't a problem for intermediates that strongly leak such as on ASCAD-r, or the masks of ASCAD-v2. However, we found simple SnR analysis to be imprecise for the PoI recovery of $r_m \otimes s_j \oplus r_{out}$, and $r_m \otimes t_j \oplus r_{in}$. To reduce the loss of information during the creation of the "predicted" labels, we regroup predictions according to the hamming weight of their corresponding intermediate guess. After the SnR analysis using the predicted labels, we rank the n_{poi} samples with the highest SnR value. We then calculate the ratio r_x of correspondence between the samples. We average this ratio across all bytes of intermediate x, when it has multiple bytes, and note it in Table 5.

Table 5: Percentage of point of interest recovered with our best models, per intermediate

Dataset	ASCAD-r				ASCAD-v2				
x	r	r_{in}	$s \oplus r$	$t \oplus r_{in}$	r_m	r_{in}	r_{out}	$r_m \otimes s \oplus r_{out}$	$r_m \otimes t \oplus r_{in}$
n_{poi}	50	50	50	50	200	100	20	10	10
r_x (%)	81.86	100.0	81.71	91.57	99.0	99.0	95.0	79.38	88.13

From Table 5, we can observe a high recovery rate of the points of interest, especially of the shared masks. The shared branches are exceptionally accurate because of the regularisation imposed by the multiple branches depending on it. However, for the non-shared masks such as r on the ASCAD-r dataset, even though an overall higher accuracy is achieved for the network, the network shows a tendency to fit a hybrid distribution. In addition, it is impossible to know which side is the mask or the masked intermediate with this design. The PoIs of the masked intermediates of the ASCAD-v2 dataset are also accurately recovered. Those results have to be nuanced. Just as with the single-task models, one needs at least a minimum of information recovered by the model to identify any points of interest. However, one could imagine an iterative training where one would start with a very large number of samples, obtain an underperforming model, and retrain using a reduced set of points to improve performance.

6 Conclusion

Among all our experiments, we can observe a clear tendency: the more information per weight, the better. Hard-parameter sharing allows to focus the propagation of losses towards fewer weights. This reduces redundancy inside the network and increases the quality of the learning. However, one has to be assured of the collaboration between the targets of the network, as losses can compete as much as they can collaborate. Overall, multi-task learning seems to have a clear edge over single-task approaches, especially in the context of a side-channel evaluation. The key takeaways are the following :

– Multi-task learning is a natural improvement of single-task learning in a scenario where the knowledge of randomness cannot be accessed.
– Hard-parameter sharing allows multi-task learning to benefit from the learning of multiple bytes at the same time, even when the masks are not shared.
– Leveraging multi-tasking to add constraints on the network increases the chances of the attacker to build successful attacks.
– Multi-task learning allows an attacker to take advantage of multi-target strategies during profiling.
– Branch designs, along with multi-task learning can lead to accurate recovery of the shares distributions of a masking scheme.

Our results contribute to the research of multi-task deep learning models in the context of side-channel key recovery attacks. We extend previous results from Marquet et Oswald [5] to more challenging scenarios where masks are not shared by multiple potential targets. We show that linking potential common features and accumulating constraints on the network benefits the network by reducing overfitting and further enables models to lead successful attacks. In addition, we target the multiple masks of the ASCAD-v2 and successfully build an attack using the previously introduced concepts. We suggest that more complex architectures, adding helpful constraints on the network, would further improve the chances of an attacker finding successful attacks.

Acknowledgments Thomas Marquet has been supported by the KWF under grant number KWF-3520—31870—45842. Thomas Marquet and Elisabeth Oswald have been supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 725042).

References

1. 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
2. Dubrova, E., Ngo, K., Gärtner, J., Wang, R.: Breaking a fifth-order masked implementation of crystals-kyber by copy-paste. In: Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop. p. 10–20. APKC '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3591866.3593072, https://doi.org/10.1145/3591866.3593072
3. Hu, F., Wang, H., Wang, J.: Cross-subkey deep-learning side-channel analysis. Cryptology ePrint Archive, Report 2021/1328 (2021), https://eprint.iacr.org/2021/1328
4. Maghrebi, H.: Deep learning based side-channel attack: a new profiling methodology based on multi-label classification. Cryptology ePrint Archive, Report 2020/436 (2020), https://eprint.iacr.org/2020/436

5. Marquet, T., Oswald, E.: A comparison of multi-task learning and single-task learning approaches. Cryptology ePrint Archive, Paper 2023/611 (2023). https://doi.org/10.1007/978-3-031-16815-4, https://eprint.iacr.org/2023/611, https://eprint.iacr.org/2023/611
6. Masure, L., Cristiani, V., Lecomte, M., Standaert, F.X.: Don't learn what you already know: Scheme-aware modeling for profiling side-channel analysis against masking. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(1), 32–59 (Nov 2022). https://doi.org/10.46586/tches.v2023.i1.32-59, https://tches.iacr.org/index.php/TCHES/article/view/9946
7. Masure, L., Strullu, R.: Side-channel analysis against anssi's protected aes implementation on arm: end-to-end attacks with multi-task learning. Journal of Cryptographic Engineering **13**, 1–19 (03 2023). https://doi.org/10.1007/s13389-023-00311-7
8. Ngo, K., Dubrova, E., Guo, Q., Johansson, T.: A side-channel attack on a masked ind-cca secure saber kem implementation. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(4), 676–707 (Aug 2021). https://doi.org/10.46586/tches.v2021.i4.676-707, https://tches.iacr.org/index.php/TCHES/article/view/9079
9. Ngo, K., Wang, R., Dubrova, E., Paulsrud, N.: Higher-order boolean masking does not prevent side-channel attacks on lwe/lwr-based pke/kems. In: 2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL). pp. 190–195 (2023). https://doi.org/10.1109/ISMVL57333.2023.00044
10. Perin, G., Wu, L., Picek, S.: Exploring feature selection scenarios for deep learning-based side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(4), 828–861 (Aug 2022). https://doi.org/10.46586/tches.v2022.i4.828-861, https://tches.iacr.org/index.php/TCHES/article/view/9842
11. Prouff, E., Strullu, R., Benadjila, R., Cagli, E., Canovas, C.: Study of deep learning techniques for side-channel analysis and introduction to ascad database. IACR Cryptol. ePrint Arch. **2018**, 53 (2018), https://api.semanticscholar.org/CorpusID:41991837
12. Ruder, S.: An overview of multi-task learning in deep neural networks. CoRR **abs/1706.05098** (2017)