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

Thomas Marquet[1] and Elisabeth Oswald[1,2][0000−0001−7502−3184]

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

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 ideas based on the encoding of relationships between the multiple learning tasks. Our work provides a wide range of experiments to understand the performance of multi-task strategies against the current state of the art. We show that multi-task learning benefits from the accumulation of constraints to guide the propagation of the gradient. Such strategies achieve novel milestones against protected implementations when the knowledge of randomness isn't assumed. We propose a new state of the art on ASCAD-r and ASCAD-v2, along with models that defeat for the first time all masks of the affine masking on ASCAD-v2.

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

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 unrivaled ability to efficiently utilise information across many tracepoints. 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. 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 wide range of 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 state-of-the-art results for ASCAD-r and ASCAD-v2, with the first model to defeat both multiplicative and additive mask 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. [2] 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] shows a similar technique in order 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 performs the conditional probabilities over one hot encoded bytes. Furthermore, Masure et al. [6] explain the impact of masking on the optimisation problem.

Perin et al. [9] 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 in order 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.

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. We note σ as the softmax function. Furthermore, we note the activated output of task x as $sigma_x = \sigma(\alpha_x)$, with α_x being the inactive output.

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 the number of fully connected branches and how the branches are connected. We design one branch per intermediate value, whereby an intermediate value may also refer to a mask value.

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 2.9nv.1 and cuda 12.2.

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, 60k traces from the random key split for training (N_t = 50k 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 most leaky 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 a number of 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 [9], which reach single trace success one some key bytes — culminating to 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: ($x \otimes \beta \oplus \alpha$,$\beta$, α) (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. [9] 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 a state-of-the-art in two situations where the difficulty is reduced. 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 Table. 2.

2.4 Metrics

The metrics of interest are the following :

- **Accuracy.** We use the minimum accuracy and maximum accuracy of a subkey k_i, noted respectively $min(k_i)$ and $max(k_i)$.
- **Number of traces to reach rank(key)** $< 2^1$ **.** We note it $T_{rank<1}$.

2.5 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.

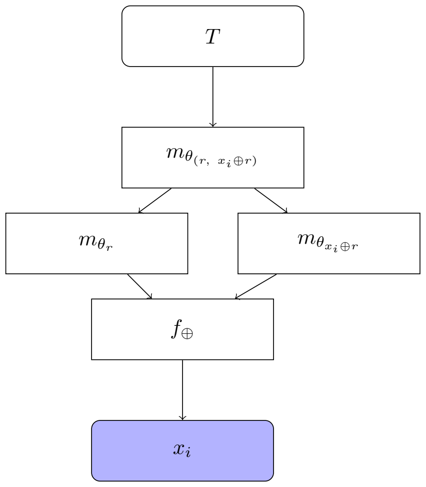
– Eavesdropping : x_a might share features with x_b. Let's say that x_a has a stronger signal and is easier to learn than x_b. Then, training both at the same time is beneficial for x_b
– Regularisation effect: The overall gradient being composed, it will rather go in valleys that are beneficial for 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 [10]. 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.

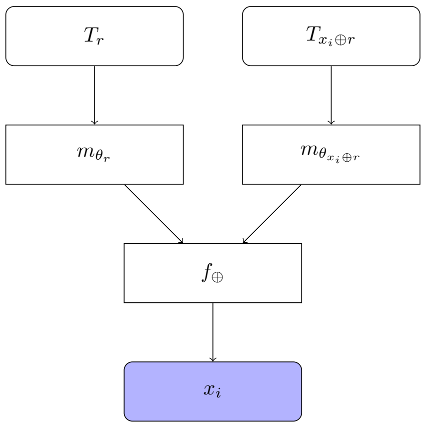
With profiled attacks, the most challenging setting is the one where knowledge of the countermeasures is not assumed. In the context of masked implementations, we would then assume that —because of a lack of access to internal randomness— the training data cannot be labeled with masks or masked values, 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.


T
$m_{\theta_{(r,\ x_i \oplus r)}}$
m_{θ_r}
$m_{\theta_{x_i \oplus r}}$
$f_\oplus$
x_i

(a) Single-task models on ASCAD-r



(b) Single-task models on ASCAD-v2

Fig. 1: 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, as it was done previously in the white-box scenario. 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 2. 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].

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. Sharing layers close to the input is usually made to share higher-level features. Going further

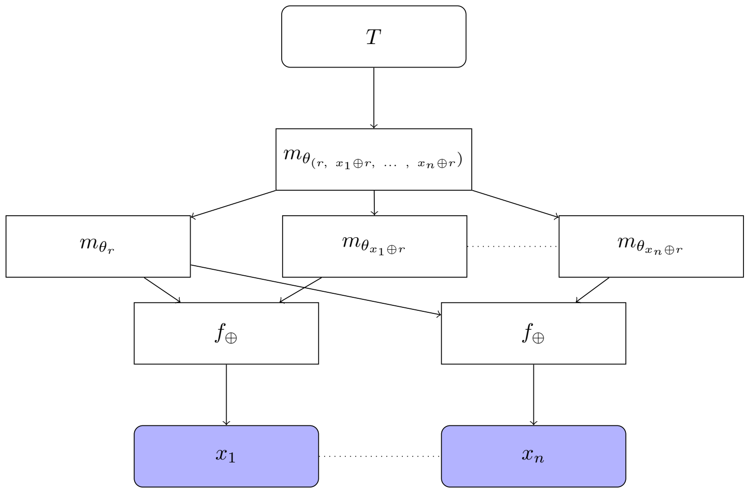

T
$m_{\theta_{(r,\ x_1 \oplus r,\ \dots\ ,\ x_n \oplus r)}}$
m_{θ_r}
$m_{\theta_{x_1 \oplus r}}$
$m_{\theta_{x_n \oplus r}}$
$f_\oplus$
$f_\oplus$
x_1
x_n

Fig. 2: Shared mask architecture

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 the same weights. To do so, one has to separate the network into multiple channels, either by splitting the input or inside the network, through mid-level unshared layers. The technique used in this paper is the latter and presented in Fig 4. The network is built in a sequence of shared-unshared-shared layers.

Shared branches models (noted m_{shared}). Using the trick presented in the last paragraph, we can design models that maximise the sharing of weights for all tasks. The benefit of such networks is to reduce 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 constrain the network further, reducing the amount of representation weights can take, as they have to satisfy all the bytes instead of just one. This technique 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 [2, 8]. Our strategy is the direct adaptation of such a technique in a multi-task learning scenario. However, our adaptation is superior as the extraction and alignment of each byte leakage in the trace, is done by the network instead of being a pre-processing step.

Balancing tasks through regularisation. The obvious problem of hard-parameter sharing is that weights might tend to take a representation biased against the less leaky bytes. We introduce a regularisation loss to counter such negative interaction. We note the default loss as $\mathcal{L}_{m_0}$, and is the sum of all cross-entropy losses. Our loss is noted $\mathcal{L}_{m_{\text{reg}}}$, and the use of the loss is signified in the name of the model (i.e. m_{reg} and $m_{\text{reg}-\text{shared}}$).

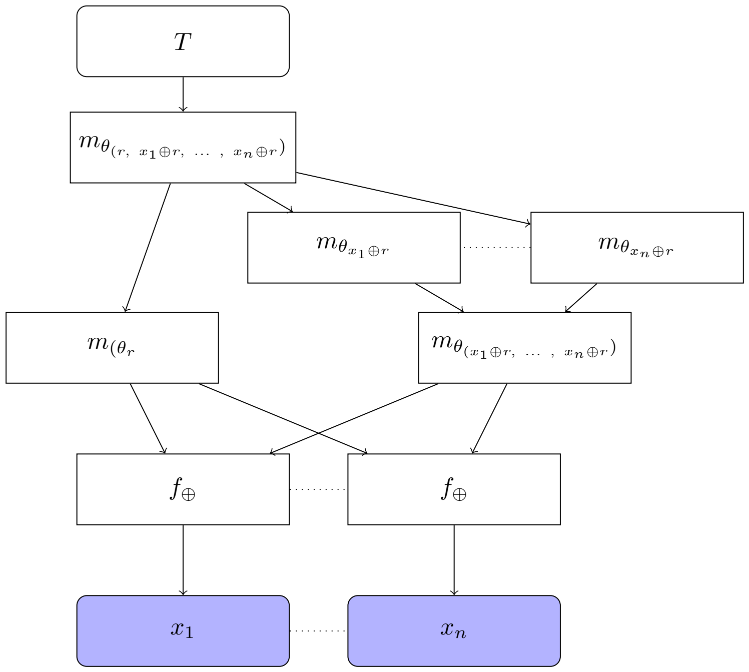

T
$m_{\theta_{(r,\ x_1\oplus r,\ \dots\ ,\ x_n\oplus r)}}$
$m_{\theta_{x_1\oplus r}}$
$m_{\theta_{x_n\oplus r}}$
$m_{(\theta_r}$
$m_{\theta_{(x_1\oplus r,\ \dots\ ,\ x_n\oplus r)}}$
$f_\oplus$
$f_\oplus$
x_1
x_n

Fig. 3: Shared mask architecture using hard-parameter sharing

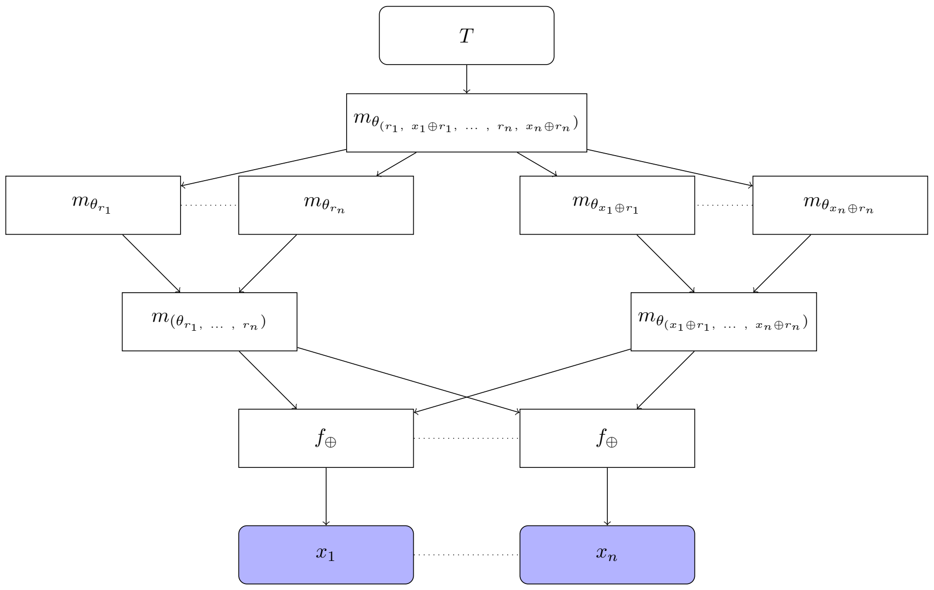

T
$m_{\theta_{(r_1,\ x_1\oplus r_1,\ \dots\ ,\ r_n,\ x_n\oplus r_n)}}$
$m_{\theta_{r_1}}$
$m_{\theta_{r_n}}$
$m_{\theta_{x_1\oplus r_1}}$
$m_{\theta_{x_n\oplus r_n}}$
$m_{(\theta_{r_1},\ \dots\ ,\ r_n)}$
$m_{\theta_{(x_1\oplus r_1,\ \dots\ ,\ x_n\oplus r_n)}}$
$f_\oplus$
$f_\oplus$
x_1
x_n

Fig. 4: Not shared mask with hard parameter sharing on both sides

$$\mathcal{L}_{m_0} = \sum_{i}^{n_t} CCE(\sigma_{x_i}, X_i) \tag{3}$$

$$\mathcal{L}_{m_{\text{reg}}} = \mathcal{L}_{m_0} + \frac{1}{n_t} \sum_{i}^{n_t} (CCE(\sigma_{x_i}, X_i) - \frac{\mathcal{L}_{m_0}}{n_t})^2 \tag{4}$$

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. [9] 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.[9]. 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 1b. 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 for all of the branches, while strides and pooling size are chosen to scale based on the input size of each branch.

- **Intermediate branches.** The input of the intermediate branch is the full execution of the S-box operation from the first round because 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 25 epochs, with a learning rate decreasing at each cycle. This allows us to annihilate overfitting as it can be seen in Figures 6, 7, 9 and maximise the performances of our models. The learning rates are 0.001, 0.0001, and finally 0.00001 with an Adam optimiser.

4 Experimental results

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. Since all bytes of those intermediates are masked with the same randomness, we train an architecture with a common branch for each 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 for the additive mask, or an inverse multiplication ($f_{\otimes}$) for the multiplicative mask. 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 2 multi-task models based on the design in Figure 2 noted m_0 and m_{reg}, and 2 multi-task models based on Figure 3 that we note $m_{0-\text{shared}}$ and $m_{\text{reg}-\text{shared}}$. Finally we train 14 single-task models (m_{single}), that are submodels of the design of the m_0 model, with only one branch $m_{\theta_{x_i \oplus r}}$. We plot the evolution of the losses during training of all models in the Figure 8, besides single-tasks where we plot only the best subkey rescaled to be easily compared. As the main performance metric, 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 3. We add to this table the worst (and the best) key byte k_i accuracy.

Single-task models do not succeed in recovering the full key, even though some bytes successfully managed to converge since the maximum subkey accuracy is relatively high. 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

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	$min(k_i)$	$max(k_i)$	$T_{rank<1}$
m_{single}	0.32	30.21	>100
m_0	19.85	43.14	8
$m_{0-\text{shared}}$	18.16	24.81	18
m_{reg}	**23.35**	**43.45**	**7**
$m_{\text{reg}-\text{shared}}$	16.49	25.49	18

it more resilient to such problems because of the effect of multiple gradients. Beyond this fact, the accuracies reached by the single models even after convergence are inferior to most multi-task models. Two factors can explain such differences. First is the ability of multi-task learning to make sense of the different signals with a smaller training set because of the multiple labels. Second, is the regularisation effect over the shared mask. Among the multi-task models, we see that hard-parameter sharing impacts the performance significantly in this experiment, as the models $m_{0-\text{shared}}$ and $m_{\text{reg}-\text{shared}}$ effectively possess fewer weights to learn the same amount of tasks. The model m_{reg} is the close winner against the baseline model m_0 recovering the full key respectively with 7 and 8 traces.

Looking at the evolution of the losses, we see that the first single-task model to pass the plateau is slower to converge than the multi-task models. Furthermore, it seems like the model doesn't completely pass the plateau, as the loss drops significantly multiple times. The same goes for the multi-task models that do not use hard-parameter sharing m_0 and m_{reg}. While the other multi-task models plunge towards the maximum learning of the model a few epochs after passing the plateau. Eavesdropping is the main reason why this sudden plunge happens. Since all bytes share features, when one byte manages to find the way to the goal, the other bytes quickly understand that this distribution is benefiting them. Finally, regularisation seems to decrease the number of epochs needed to converge, as models with extra regularisation are faster than the baseline model m_0. Finally, the regularisation loss $\mathcal{L}_{m_{\text{reg}}}$ seems to stabilise the hard-parameter sharing model, as the validation loss from $m_{\text{reg}-\text{shared}}$ is more stable than it's counterpart $m_{0-\text{shared}}$.

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 dataset available

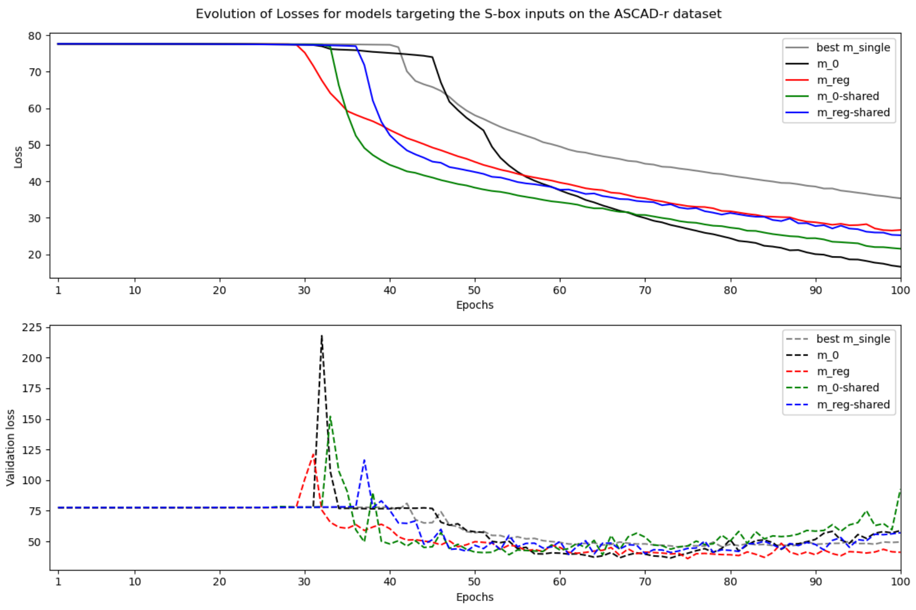


Fig. 5: Loss and validation loss evolution for models targeting the S-box input pair $(t \oplus r_{in}\ ,\ r_{in})$

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 1. We plot the evolution of the losses during training in Figures 6 and 7, and note the performance metrics in Table 2.

Table 2: Experimental results for S-box input leakage pair, and 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	$min(k_j)$	$max(k_j)$	$T_{rank<1}$	$min(k_j)$	$max(k_j)$	$T_{rank<1}$
m_{single}	1.00	1.74	32	0.24	0.54	>200
m_0	**1.78**	2.30	20	0.46	0.86	128
$m_{0-\text{shared}}$	**1.78**	**2.44**	**19**	0.48	**0.94**	**95**
m_{reg}	1.76	**2.44**	20	0.48	0.88	119
$m_{\text{reg}-\text{shared}}$	1.68	2.28	20	**0.58**	0.92	98

Single-task networks only manage to lead to successful attacks on the more leaky S-box inputs, even in this simplified scenario. As the leakages are extracted and fed directly to the network, the most significant effect that allows multi-task learning to outperform single-task learning is the extra regularisation from the shared mask branch. It is hard to distinguish between the multi-task models on the S-boxes inputs intermediate, as the difference performance-wise is non-significant. However, on the less leaky S-boxes outputs, we can observe a clear performance difference between the shared weights models and their counterparts. Regularisation seems to be again the cause of this performance gain, as the baseline model m_0 is outperformed by all models including m_{reg}.

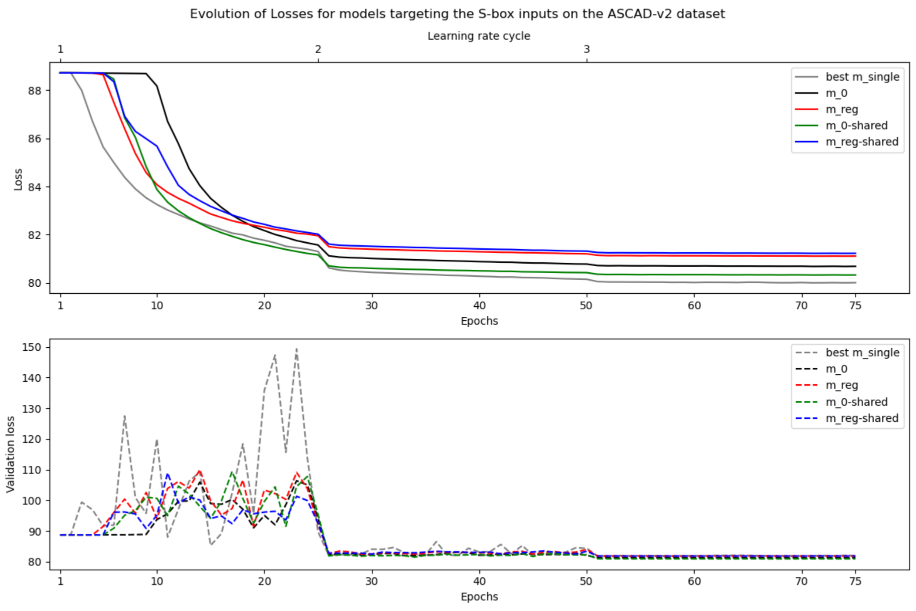


Fig. 6: Loss and validation loss evolution for models targeting $(r_m \otimes t_j \oplus r_{in}\ ,\ r_{in})$

The evolution of losses tells us a different story than in the raw traces scenario. While single-task models were slower to converge towards significant information on the ASCAD-r dataset, it is quite the opposite on ASCAD-v2, at least when the signal is strong enough. The training loss of the best single-task model is drastically faster on the S-boxes inputs but overfits because of the lack of regularisation. This overfitting is quite clear when looking at cycle 2, where the validation loss of the multi-task models is already flat, while the single-task loss is spiking from time to time. In the special case of multi-task models, the shared weights models $m_{0-\text{shared}}$ and $m_{\text{reg}-\text{shared}}$ are consistently converging faster than the baseline model m_0 by a significant margin. Finally, m_{reg} is also faster than its basic counterpart, hinting again at the importance of regularisation.

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-

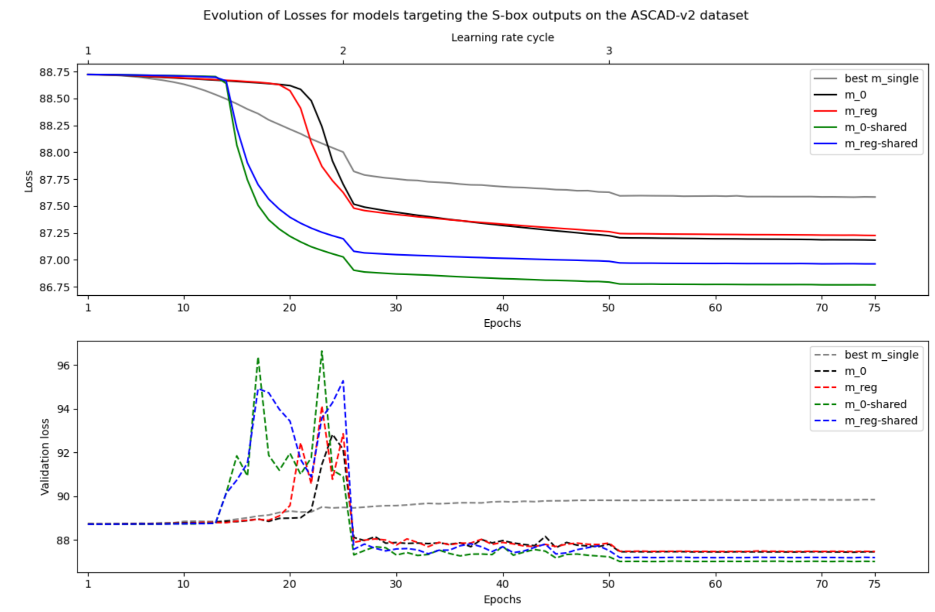


Fig. 7: Loss and validation loss evolution for models targeting $(r_m \otimes s_j \oplus r_{out}\ \ r_{out})$

parameter sharing, this time on both sides of the masking scheme in the manner of Figure 4. 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 two models in this experiment, named $m_{0-\text{shared}}$ and $m_{\text{reg}-\text{shared}}$. Their counterparts, m_0 and m_{reg}, are naturally the same models but without hard parameter sharing. The single models (m_{single}) are therefore 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 present the evolution of the losses of all multi-task models in Figure 8, along with the best single-task model, which was tasked to learn the 3rd byte of the S-boxes output. We perform a full key recovery targeting the S-boxes output in the same setup as the previous experiment on ASCAD-r.

As in the previous scenario on ASCAD-r, single-task models do not converge for all bytes. However, one model reaches a very high accuracy and recovers its designated key byte in only 1 trace. The signal is high, the difficulty comes from finding the leakage in the raw traces rather to utilise it. The baseline model m_0 and the regularised version m_{reg}, suffer from the competition of the different losses, as none of them collaborate by default. Since tasks learned in those models do not share any features. Finally, the two models with shared weights are the only ones recovering the key, both in 2 traces, setting a new state of the art in this specific raw traces setup. The extra regularisation present in the $m_{\text{reg}-\text{shared}}$ is demonstrated to be useful for the first time in this experiment, managing to increase accuracies reached, even though it doesn't improve on the number of

Table 3: Performance metrics for the S-box output leakage pair $s \oplus r$, r experiment on ASCAD-r

Model type	$min(k_i)$	$max(k_i)$	$T_{rank<1}$
m_{single}	0.30	97.77	>100
m_0	0.30	0.52	>100
$m_{0-\text{shared}}$	12.93	24.99	**2**
m_{reg}	0.26	0.50	>100
$m_{\text{reg}-\text{shared}}$	**53.89**	**88.05**	**2**

traces required to recover the full key. Looking at the loss evolution, however, it is likely that more epochs would improve the network, while the model $m_{0-\text{shared}}$ reached its peak.

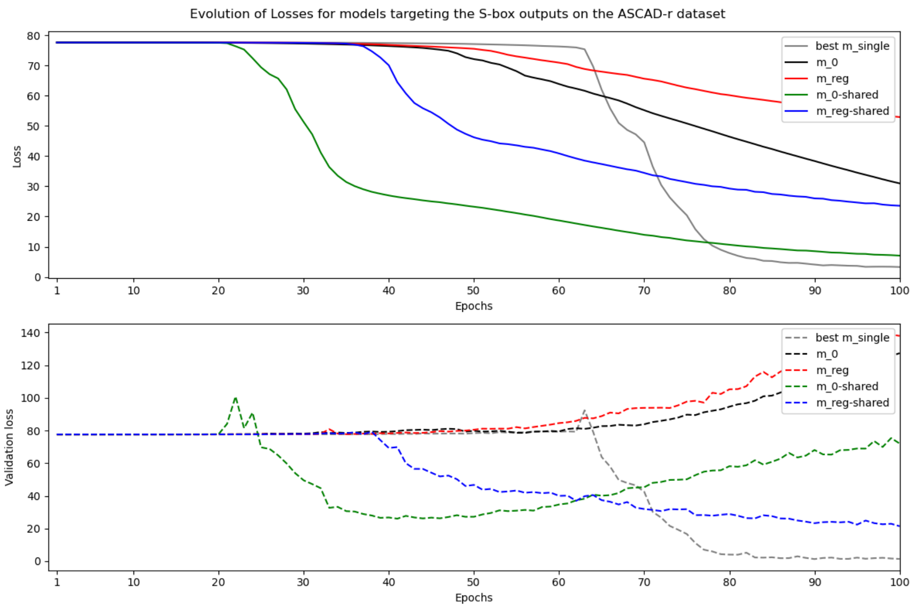


Fig. 8: Loss and validation loss evolution for all multi-task models

Focusing on the models that successfully converged, we can observe three successful convergences. The first one is also the best model $m_{0-\text{shared}}$, learning meaningful information, for all bytes, from epoch 20 onwards, as one can see with the spike on the validation loss. The best single-task model passes the plateau around epochs 63-64. Finally $m_{\text{reg}-\text{shared}}$ is slower than it's counterpart $m_{0-\text{shared}}$, but steadier. The forced collaboration through shared weights and regularisation losses implemented in the design of the model $m_{\text{reg}-\text{shared}}$, successfully intensifies the accumulation of knowledge about the different shares, and yields an almost linear evolution of the training and validation loss from epoch 50 onwards.

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 $r_m \otimes t_j \oplus r_{in}$. This allows us to design a model that learns the unmasked S-box input and output at the same time. Both learning tasks depend on the same branch, we expect the model to collaborate to understand how to fit the multiplicative mask. To better understand the impact of training multiple intermediates, we train the same models, but only without this "multi-target" approach. We note $m_{0-\mathrm{t}}$, the model learning only t_j through the triplet $(r_m \otimes t_j \oplus r_{in}\ ,\ r_m\ ,\ r_{in})$, and $m_{0-\mathrm{t-shared}}$, the same model using hard-parameter sharing. We note the main performance metrics after performing the usual full key recovery in Table 4, and plot the evolution of the losses in Figure 9.

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

Model type	$min(k_i)$	$max(k_i)$	$T_{rank<1}$
$m_{0-\mathrm{t}}$	0.28	0.70	>200
$m_{0-\mathrm{t-shared}}$	0.34	0.60	>200
m_0	0.24	0.56	>200
$m_{0-\mathrm{shared}}$	**2.42**	**3.34**	**16**

The two models $m_{0-\mathrm{t}}$ and $m_{0-\mathrm{t-shared}}$ trying to recover only the S-boxes 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-boxes. Interestingly, the only model converging towards a successful attack is the model leveraging a multi-target strategy **during training**, also learning the less leaky triplet. This is to the best of our knowledge, the first attack utilising such a strategy, along with the first model being able to learn all masks of the ASCAD-v2 dataset.

If one compares the evolution of the losses from Figure 9 with the ones from Figure 6 and 7, one can observe the effect of adding an extra share in the optimisation problem. The flat evolution of the losses and then the sudden drop and almost immediate convergence towards maximum information available is endemic to masking according to Masure et al. [6] and increases with the number of shares.

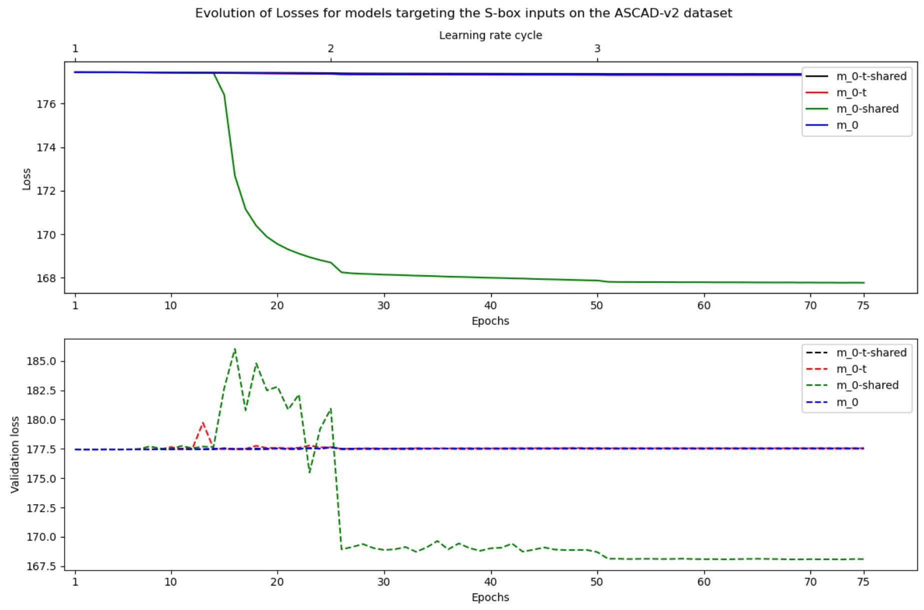


Fig. 9: Loss and validation loss evolution for all multi-task models

5 Explainability and point of interest extraction

Deep learning networks are often obscure boxes where explainability is traded against performances. 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. Even though this distribution is not giving away the values of the targeted intermediates, a δ is introduced on both sides of the xor operation, effectively canceling during the xor layer. When training single-task models, one does not have certainty about the intermediate that it recovers on each branch, as it could be either $x \oplus r$ or r. However, with our designs, such certainty is assured since branches collaborate to force one to fit a certain distribution, at least with the shared mask designs.

Using the model m_{reg} from the attack on the S-boxes input in section 4.1, we try to recover the points of interest of each targeted share. To do so, we recover the values going through the model at the end of each branch. Those values are extracted after a softmax operation and therefore represent probability-like scores. We take the values with the highest score to label the traces for the distinguisher. We evaluate and plot in Figure. 10 the recovered correlations using a simple SnR analysis with the predicted "labels" and compare against the SnR analysis obtained with the real labels.

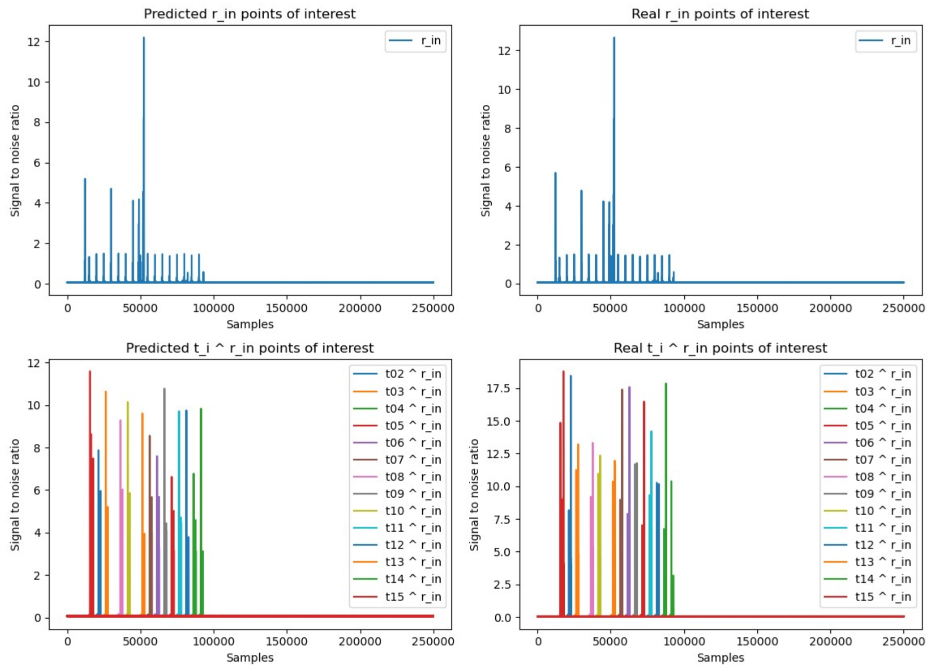


Fig. 10: Predicted SnR from the input mask r_{in}, and the corresponding masked S-Boxes inputs

To confirm such results on both datasets, with the most 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.

We rank the n_{poi} samples using either a simple SnR analysis or a Kruskal-Wallis [3] distinguisher for $r_m \otimes s \oplus r_{out}$ and $r_m \otimes t \oplus r_{in}$. 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.

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

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 (%)	73.28	98.00	75.50	89.28	97.50	98.00	75.00	81.25	87.50

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 with the KW distinguisher. 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 recover any points of interest. But such a technique could lead to a kind of cyclic training where one would start with a very large number of samples, and ultimately reduce to only a few points to improve performances.

6 Conclusion

The key takeaways of our experiments 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, as it increases the speed of convergence.
– 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 into using 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 while 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 in a given time.

Acknowledgments Thomas Marquet has been supported by the KWF under grant number KWF-3520—31870—45842. Elisabeth Oswald has 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. 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`
3. Kruskal, W.H., Wallis, W.A.: Use of ranks in one-criterion variance analysis. Journal of the American statistical Association **47**(260), 583–621 (1952)
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: Grey-box modeling for profiling side-channel analysis against masking. Cryptology ePrint Archive, Report 2022/493 (2022), `https://eprint.iacr.org/2022/493`
7. Masure, L., Strullu, R.: Side channel analysis against the ANSSI's protected AES implementation on ARM. Cryptology ePrint Archive, Report 2021/592 (2021), `https://eprint.iacr.org/2021/592`
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. 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`
10. Ruder, S.: An overview of multi-task learning in deep neural networks. CoRR **abs/1706.05098** (2017)