Can KANs Do It? Toward Interpretable Deep Learning-based Side-channel Analysis

Kota Yoshida[1⋆], Sengim Karayalcin[2⋆], and Stjepan Picek[3]

[1] Ritsumeikan University, Japan
y0sh1d4@fc.ritsumei.ac.jp
[2] Leiden University, The Netherlands
s.karayalcin@liacs.leidenuniv.nl
[3] Radboud University, The Netherlands
stjepan.picek@ru.nl

Abstract. Recently, deep learning-based side-channel analysis (DLSCA) has emerged as a serious threat against cryptographic implementations. These methods can efficiently break implementations protected with various countermeasures while needing limited manual intervention. To effectively protect implementation, it is therefore crucial to be able to interpret **how** these models are defeating countermeasures. Several works have attempted to gain a better understanding of the mechanics of these models. However, a fine-grained description remains elusive. To help tackle this challenge, we propose using Kolmogorov-Arnold Networks (KANs). These neural networks were recently introduced and showed competitive performance to multilayer perceptrons (MLPs) on small-scale tasks while being easier to interpret. In this work, we show that KANs are well suited to SCA, performing similarly to MLPs across both simulated and real-world traces. Furthermore, we find specific strategies that the trained models learn for combining mask shares and are able to measure what points in the trace are relevant.

1 Introduction

Since the seminal work of Kocher [17], side-channel analysis (SCA) has received significant attention from the research community. In such attacks, the relationship between secret dependant computations and (physical) leakages (e.g., timing [16] and power consumption [17]) from cryptographic implementations is exploited to extract secret keys. To assess the security of cryptographic implementations under worst-case assumptions, Chari et al. [7] proposed template attacks. For these attacks, the adversary uses a copy of the device they are attacking with known key(s), i.e., a profiling device, to create a model for the computation of the target device. This model can then be used to attack the target device more efficiently. However, classical profiling attack techniques still require significant manual intervention in the feature engineering phase [31,2,20].

⋆ equal contribution

To address this limitation, deep learning-based profiling attacks have recently emerged with the promise to automate much of the labor-intensive aspects of side-channel attacks [23,6,14,22,27,11]. In these works, neural networks are trained to directly predict sensitive intermediate values from traces of protected implementations leading to state-of-the-art attacking results. While these models result in (more) automated and efficient attacks than classical profiling methods, there are still several open challenges that limit the reliability of deep learning-based SCA in practice [30].

One of the key limitations of using neural networks is that the (trained) networks are a black-box. In the context of SCA, a network resulting in key retrieval clearly indicates that exploitable leakage is present in the traces, but actionable feedback about what information is leaking and where is difficult to extract. To address this challenge, several works have explored input attribution methods to visualize what parts of the trace are contributing to the neural networks predictions [12,26,24,45]. Further works attempted to gain a better understanding of the internal operations of the neural network [41,27]. Finally, in [42], a specific type of network architecture is used that is more interpretable. While these works make significant progress towards understanding aspects of the trained networks, only [42] provides explanations on the internals that reflect on how networks are combining secret shares to avoid masking countermeasures. However, this requires specialized architectures and results in significantly degraded attack performance.

To address this gap, we propose using the recently introduced Kolmogorov-Arnold Networks (KAN) [21]. KANs are a novel alternative to the widely used multilayer perceptron (MLP) with several key benefits for our application. KANs can achieve similar performance to MLPs with much fewer trainable parameters on several tasks and result in (qualitatively) far more interpretable networks. Considering the challenges related to explaining how neural networks effectively break implementations discussed above and the fact that (relatively) small models often work quite well in SCA [45,27], exploring whether KANs can provide competitive performance while improving interpretability is relevant. With simulation and real traces, we demonstrate that KANs can achieve comparable performance to MLPs and can interpret what profiling models learned in DLSCA.

The main contributions of this paper are as follows:

1. This is the first report on applying KANs in SCA. We demonstrate that KAN performs similarly to MLP in DLSCA. KANs can effectively retrieve the sub-key from (the truncated) ASCAD traces in ≈ 300 traces, which is comparable to early, less-optimized[4] MLPs [3].
2. We simulate side-channel leakage and show the function learned by KANs. More precisely, we simulated Hamming weight (HW) leakage from a software-implemented cryptographic algorithm with a masking countermeasure. Focusing on LSB labeling, we showed that KANs learned an expectation maxi-

[4] We note that better performance has been achieved for this target with more optimized architectures [45,38] or hyperparameter search [32,1], but similar performance improvements should be possible for KANs with more research.

mization function for each leakage pattern. This function can achieve 53.74% accuracy on the 2-share masking countermeasure, which indicates that more shares are needed to mitigate SCA risks.
3. We evaluated the interpretability of KANs with the ASCAD dataset, which contains power traces acquired from software-implemented AES with a 2-share masking countermeasure. We showed that the same result as the simulation can be obtained with real traces.

In this paper, we first introduce relevant SCA background in Section 2. In Section 3, we introduce KANs. In Section 4, we discuss relevant related work on interpretability in DLSCA. We then test whether KANs can be effectively used on simulated and real traces in Section 5 and Section 6, both in terms of performance and interpretability. Then, we discuss the results in Section 7. Finally, in Section 8, we provide conclusions and discuss avenues for future work.

2 Side-channel Analysis

2.1 Profiling SCA with Deep Learning

In the profiled SCA scenario, the adversary has access to a clone device on which they have access to the secret key(s) [7]. The adversary's first step is to create a model (with uncertainty) to predict a secret-dependant intermediate value of cryptographic operations from side-channel leakage. Since the intermediate value depends on a key, the adversary can reveal the key from the intermediate value. Here, we consider that the cryptographic algorithm is AES, and the side-channel leakage is the target device's power consumption. The intermediate value is generally set to the output of the AES Sbox, i.e., $SBox[p_i \oplus k_i]$ where p_i and k_i are the i'th bytes of the plaintext and key, respectively.

The adversary prepares a dataset that consists of measured power traces (input) $\mathcal{X}_p$ and corresponding intermediate values (label) $\mathcal{Y}_p$. These intermediate values can be transformed to effectively train the model using a leakage model representing how the value influences the physical measurements. The adversary trains a neural network-based classification model parameterized with trainable θ, $\mathcal{M}_\theta : \mathbf{X} \mapsto \mathbf{Y}$ with the dataset with gradient descent to minimize a loss function. This step is commonly denoted by the training or profiling step.

The second step is to reveal the most likely secret key using the trained model $\mathcal{M}_\theta$ and newly acquired traces $\mathcal{X}_a$ from the target device. The adversary can leverage to score key candidates $k \in \mathcal{K}$ according to the log-likelihood of the intermediate values y_k the model predicts:

$$score(k) = \sum_{\mathbf{x}_i \in \mathcal{X}_a} log(\mathcal{M}(\mathbf{x}_i)_\theta(\mathbf{x}[y_{k,i}])). \tag{1}$$

These scores can then be ordered, which allows us to rank the key candidates accordingly. As only the actual key $k*$ should generate intermediate values processed in the traces, the highest rank key should be the correct one if the

model is well-trained. By simulating different attacks with varying random subsets of attack traces, we can effectively simulate different attacks and estimate the guessing entropy of the correct key by taking the average rank of the correct key over these simulated attacks [35].

2.2 Leakage Models and Labeling Techniques

The Hamming weight (HW) leakage model is generally assumed when attacking software implemented AES [30]. In this model, we can observe power consumption correlated to the HW of the transferred value on the data bus.

A template attack, a typical profiled SCA technique, sets the HW of the intermediate value (i.e., Sbox output) as the prediction target. On the other hand, the HW labeling can be problematic for SCA using machine (deep) learning because of label imbalance problems [29]. As such, Identity[3] and bitwise[46] labelings are frequently used. The Identity labeling sets the target intermediate value directly as a classification target. The bitwise labeling sets a specific bit of the target intermediate value as a classification target. Specific leakage models that assume bitwise leakage are the least significant bit(LSB) and most significant bit(MSB) leakage models. This label can be regarded as focusing on a specific wire on the data bus, which can occur if there is some physical bias towards that wire in the power/EM measurements.

3 Kolmogorov–Arnold Networks

Kolmogorov–Arnold Networks (KANs) are a novel type of neural network architecture introduced by Liu et al. [21] based on the Kolmogorov-Arnold representation theorem [18]. In this section, we aim to give a (relatively) brief overview of the main concepts behind KANs and discuss the benefits concerning interpretability. For a more comprehensive introduction and discussion of benefits/-downsides, we refer the reader to [21].

3.1 Kolmogorov-Arnold Representation Theorem

The Kolmogorov-Arnold representation theorem [18] states that any multivariate continuous function $f(X) : [0,1]^n \mapsto \mathbf{R}$ can be represented by a finite superposition of univariate functions:

$$f(x_1, \cdots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right). \tag{2}$$

Here, $\phi_{q,p}$ are univariate functions s.t. $\phi_{q,p} : [0,1] \mapsto \mathbf{R}$ and $\Phi_q : \mathbf{R} \mapsto \mathbf{R}$. As these are all univariate functions, we can parameterize each as a basis spline curve [8] composed of local learnable basis spline functions. The spline function has two hyperparameters: the grid size and the order. Larger grid size/order allow more complex functions to be represented, but the functions may face optimization problems.

3.2 KAN Architecture

The key insight in [21] is then that these functions Φ_q can be used as a network layer, analogous to fully connected layers in multilayer perceptrons and that these layers can be stacked to create deeper networks. A KAN layer with n_{in} input and n_{out} output variables is expressed as:

$$\Phi = \{\phi_{q,p} | p \in \{1, \cdots, n_{in}\}, q \in \{1, \cdots, n_{out}\}\}. \tag{3}$$

The function in Eq. (2) is then a composition of two of these layers where the first has n inputs and $2n+1$ outputs and the second has $2n+1$ inputs and one output. Deeper networks are a further concatenation of an arbitrary number of these layers. Hereafter, we define a representation of KAN's network architecture by an integer array $[n_0, n_1, ..., n_L]$ where n_i is the number of nodes in the i-th layer of the KAN's graph by following the paper [21].

3.3 Interpretability of KANs

In this section, we discuss why KANs are more interpretable than other types of networks. The key attributes that result in easier interpretation are that the number of parameters to achieve similar accuracy is lower and that each of the learned activation functions $\phi_{q,p}$ can be easily visualized. This allows for a more intuitive understanding of the information flow in a network when compared to the more opaque nature of (large) weight matrices in MLPs.

To illustrate, Figure 1 contains a KAN[2,1,1] with two inputs x, y, which can be either 0 or 1. In the figure, we can see a plot of the learned activation function on each edge. We can see that the initial two activation functions are linear, i.e., 0 if the input is 0 and 1 if the input is 1, and these are summed together in the first node. Then, the final activation with inputs ranging from (rescaled) 0-2 is 1 if its input is around 1 and 0 otherwise. This results in a function that is 1 if $x \neq y$ and thus implements an XOR operation on these two inputs. For more examples, we again refer to [21].

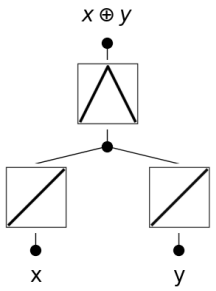


Fig. 1: Simple KAN implementing $x \oplus y$.

4 Related Work

As deep learning-based side-channel analysis (DLSCA) has grown in popularity over recent years [30], several works have investigated the interpretability/explainability perspectives. In general, we can divide the works into those that concentrate on the input (features) and those that concentrate on the inner workings of the neural network model.

Earlier works introduced input attribution methods that identify which parts of the input traces influence the network's predictions [12]. Works in this area include using gradient visualization methods [24], heatmaps [45], Layerwise Relevance Propagation [12,26], and saliency maps [12]. Wouters et al. used gradient input to understand the impact of the filter size on desynchronized traces [38]. Further research in this direction directly compared these feature attribution methods with classical leakage visualization tools [10]. Schamberger et al. introduced the concept of n-occlusion to examine how the window of occlusion impacts key recovery [34]. Later, Yap et al. developed a novel approach based on occlusion to find minimal sample points for a neural network key recovery [43]. Besides works that investigate the importance of features for a neural network, there are also works that concentrate on feature engineering techniques (selection or construction of points of interest). Rioja et al. considered using metaheuristics known as Estimation of Distribution Algorithms to help automate the selection of the PoIs in both unprotected and masking settings [33]. Wu et al. used a triplet network to transform and reduce the dimensionality of the traces into relevant embeddings that consist of important leakage information for a better attack [39]. Further works on features engineering utilize autoencoders to remove the effects of hiding countermeasures [40] and to transform traces to other targets to facilitate transfer learning [19]. (Conditional) generative adversarial networks have also been used to translate traces between modalities [9] and for feature selection [13].

While these works effectively highlight/select which parts of the traces are utilized by the networks, the internal processes of the network largely remain a black-box. Van der Valk et al. first explored using Vector Canonical Correlation Analysis to compare neural network internal representations. Subsequently, Wu et al. [41] explored ablating layers of trained networks to investigate what specific layers are doing in the network, focusing on challenges related to the portability of the networks. Perin et al. provided a metric based on the Information Bottleneck theory to visualize the information the deep neural network is learning for each epoch [25]. Perin et al. [28] employed probes on networks trained to circumvent the masking countermeasure and visualize where in the network secret shares are recombined to the target values. Yap et al. [42] proposed using an adapted network structure to be able to extract SAT equations from trained networks and precisely show the internal workings of networks. Zaid et al. designed a generative model by combining it with a stochastic attack using an autoencoder called Conditional Variational Autoencoder, providing equations of the leakage in the trace through the autoencoder's weights [44].

The above papers show several methods that can provide a broad understanding of the 'behavior' of the network by 1) showing important input features [12,26,24,45] or 2) in which layers certain information is processed [41,27]. Still, only the work in [42] attempts to gain a more precise understanding of the internals of the networks, but this requires a specialized architecture, resulting in reduced attack performance and computationally costly training procedures. This makes applying these models in practice much more difficult, especially when considering practical targets with long traces.

5 Simulations

5.1 KAN Parameters

In this section, we always use batch size 256 and train for 3000 steps. The optimizer is Adam [15] with a learning rate of `1e-3`. The grid and k parameters are both set to 3. The outputs of the KAN are transformed using a softmax function to rescale to a probability function, and the loss function is categorical-crossentropy. Based on the recommendations by Liu et al. that KAN's training starts from a simple setup, especially a small KAN shape and small grid size[5], we run some basic experiments with minimal KANs (i.e., from KAN[2, 1, 2] to KAN[2, 3, 2]) and determine that the additional nodes do not improve accuracy. As such, we use KAN[#inputs, 1, 2] unless mentioned otherwise.

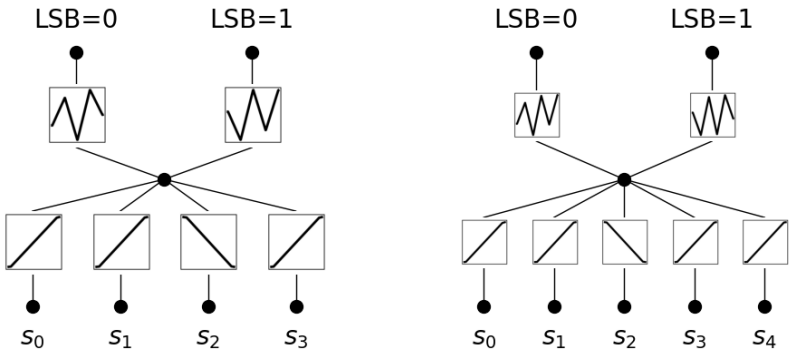
5.2 Bitwise Leakage

We first examine how KANs fit a Boolean masking scheme where a single bit for each share is leaked. In this case, we simulate n points corresponding to n shares and include the leakage for 1 bit in each of these. We add noise with 0 mean and varying standard deviations to each of these points to simulate more or less noisy side-channel traces. The label for each of these simulated traces is then the recombination (or bitwise XOR) of these shares. The number of KAN's output is set to 2 and activated by the softmax function. The output represents the confidence value corresponding to the prediction output of 0 or 1.

The results with low noise in Figure 2 show a pattern for increasing security orders (which we verify also holds for higher/lower orders). The basic algorithm that we can visually identify in the graph and activations in Figure 2 is to first linearly add the leakage for several shares together in a single node, and subsequently apply a periodic activation with a number of highs/lows that correspond to the masking order. To illustrate, imagine recombining 3 shares with perfect information. We first sum the 3 shares together, i.e., $1 + 1 + 0$. Clearly, if the result is even, the 1-bits cancel each other out, and the result should be 0. Conversely, if the result is odd, i.e., $1 + 0 + 0$, the result should be 1. More generally, we can compute xor for n-shares, $s_1, \cdots s_n \in \{0, 1\}$ by taking $(\sum_{i=1}^{n} s_i)$ `mod` 2. To achieve these results, KAN can fit sinusoid-like activations (top activations in

[5] https://github.com/KindXiaoming/pykan

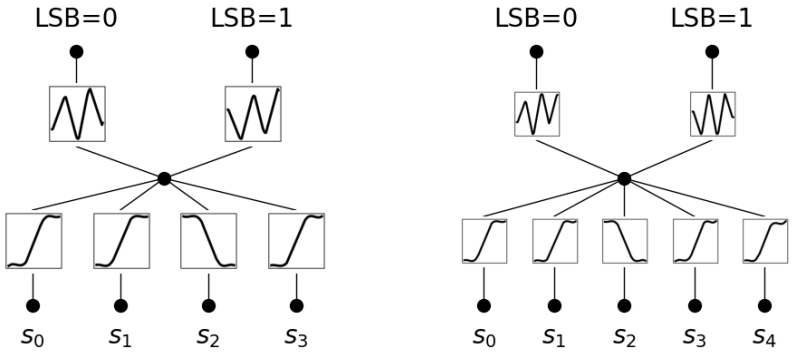
Figure 2 and Figure 3) that are on/off (1/0) when inputs are even or odd. Note that intermediate values can be rescaled while keeping the algorithm identical; if inputs are scaled by 0.5, then as opposed to even/odd, the functions can fit on these ranges (whole vs. not whole numbers) similarly.



(a) 4 shares (b) 5 shares

Fig. 2: Low-noise KAN models for bitwise leakage.

With higher noise values added to the inputs, the pattern of additional peaks/valleys for higher masking orders remains. In Figure 3, we see that the learned activations in the output layer closely resemble those in Figure 2. The main difference between these scenarios is in the input layer, where we see that the activation remains 1/0 for inputs corresponding to the distributions of the leakage.



(a) 4 shares (b) 5 shares

Fig. 3: Higher-noise KAN models for bitwise leakage.

Finally, when we consider scenarios with additional uninformative samples in Figure 4, as is a common scenario in practical SCA, we see that the KANs still learn similar structures and effectively ignore additional samples.

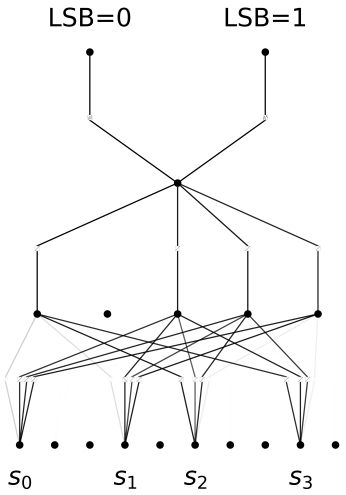


Fig. 4: 4 share KAN [10, 5, 1, 2] with uninformative samples.

5.3 HW Leakage

We next look at how KANs learn from 8-bit HW leakage on the 2-share Boolean masking scheme. We trained KAN with simulation traces without noise. The label for these simulated traces is then the LSB of XOR of these shares. The left of Figure 5 shows the trained KAN's graph. To obtain a more interpretable KAN graph, we set linear functions to activations belonging to input nodes, and the trained graph is shown in the center of Figure 5. Here, we considered sine, quartic, and other functions in addition to linear functions to make the input activation symbolic. As the error of the input activation can be corrected in the subsequent ones, we first adopted the simplest one: a linear function. Since it decreases the classification accuracy to 53.10%, we set a larger grid size to enhance the representation and re-train the KAN. We chose the grid size of 17 because the intermediate node's output $(HW(s_1) + HW(s_2))$ has 17 patterns. The last graph is shown on the right of Figure 5, and it achieves 53.26% accuracy. According to the graph, the intermediate node calculates $HW(s_1) + HW(s_2)$, and the activations belonging to output nodes determine the confidence of each class (LSB = 0 or LSB = 1).

Let us consider the occurrence distribution of 0 and 1 of LSB. The HW leakage model supposes all the lines on the data bus to contribute equally to

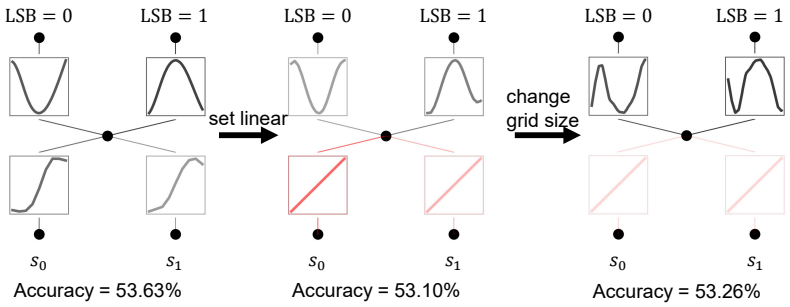


Fig. 5: Interpretation procedure for profiling model on 2-share Boolean masking using KAN.

power consumption. We cannot distinguish combinations with the same HW: e.g., both HW of "01" and "10" are 1. On the other hand, some combinations have a biased frequency of occurrence; we consider the aggregation based on $\mathrm{HW}(s_1)+\mathrm{HW}(s_2)$, which is the output of the intermediate node of the finally obtained KAN (right one) in Figure 5. Table 1 provides the number of occurrences of each LSB in all patterns of $\mathrm{HW}(s_1) + \mathrm{HW}(s_2)$. A reasonable way to predict LSB by using $\mathrm{HW}(s_1) + \mathrm{HW}(s_2)$ is by choosing ones with a higher expected value on each combination, such as the greyed cells in Table 1. Hereafter, we denote this approach by "expected value maximization (EVM) strategy". This strategy achieves 53.74% accuracy on 2-share masking; it is close to the KAN's accuracy. Figure 6 plots the results in Table 1 and their differences. Looking at the difference graph, we can see that it matches the graph learned by the KAN. From these results, we can consider that KANs have learned the EVM strategy because the activation in the KAN closely matches the graph of the EVM strategy, and their accuracy is comparable.

6 Real-world Targets

6.1 ASCAD Datasets

ASCADv1 dataset [3] is a public dataset for evaluating side-channel attacks. The dataset contains power traces acquired from software-implemented AES-128 running on ATMega8515, which has a masking countermeasure. The dataset provides plaintexts, ciphertexts, keys, masks, and traces. It is divided into profiling and attack subsets to evaluate profiling attack scenarios. It has two variants depending on how the key is provided during the trace acquisition. The fixed key dataset was acquired with a single key in both the profiling and attack subsets. There are 50 000 traces in the profiling and 10 000 traces in the attack subset. The variable key dataset was acquired with a random key in the profiling and a single key in the attack subset. There are 200 000 traces in the profiling and 100 000 traces in the attack subset. To evaluate elementary attacks, the first two

Table 1: Number of occurrences of each LSB in $\mathrm{HW}(s_1) + \mathrm{HW}(s_2)$

$\mathrm{HW}(s_1) + \mathrm{HW}(s_2)$	$\mathrm{LSB}(s_1 \oplus s_2) = 0$	$\mathrm{LSB}(s_1 \oplus s_2) = 1$
0	1	0
1	14	2
2	92	28
3	378	182
4	1092	728
5	2366	2002
6	4004	4004
7	5434	6006
8	6006	6864
9	5434	6006
10	4004	4004
11	2366	2002
12	1092	728
13	378	182
14	92	28
15	14	2
16	1	0

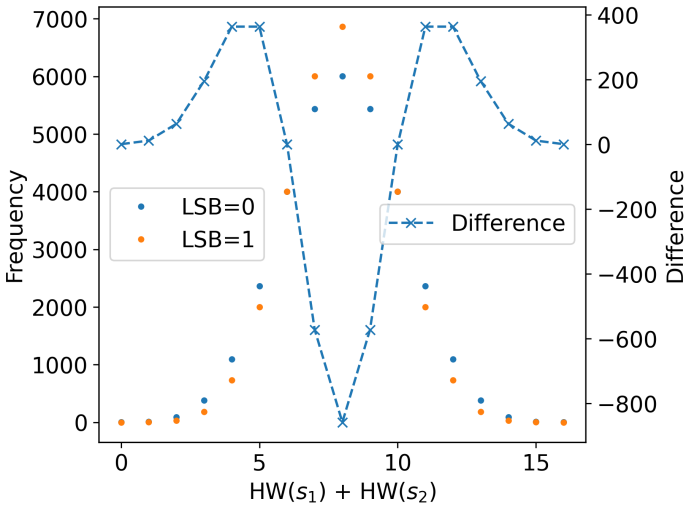
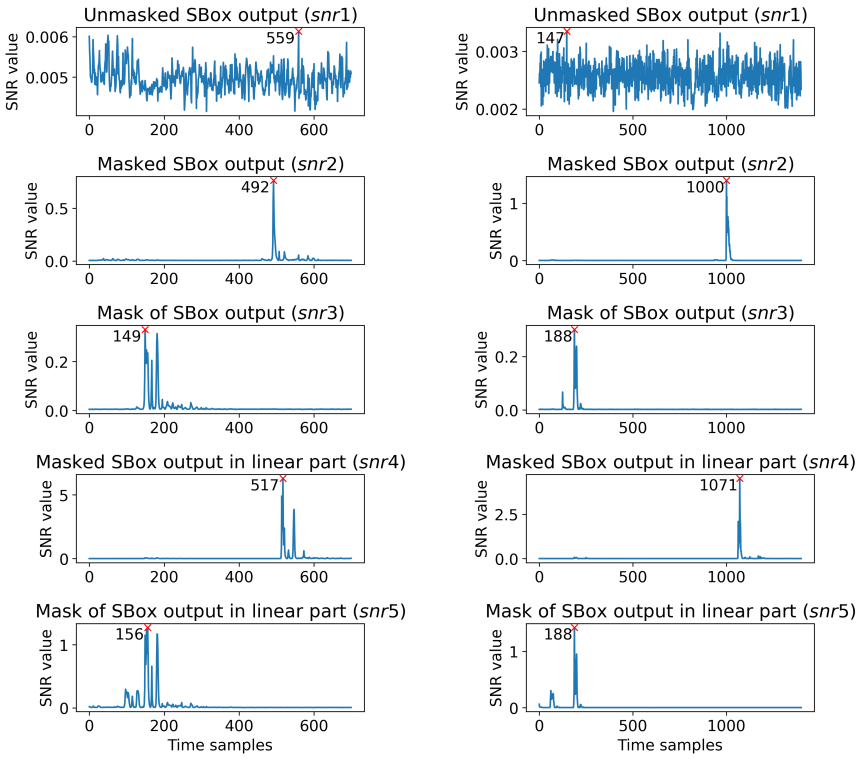


Fig. 6: Expectation value graph for the EVM strategy.

bytes of the AES state during the first round are not masked. Our attack experiments reported in the rest of the paper only target the third SBox processing during the first round, as it is the first masked byte. Each trace has 700 (fixed key) and 1400 (variable key) sample points, covering the calculation of the first round, as recommended by the authors of the datasets.

Table 2: List of known leakages about the third byte in the first round.

Name	Type	Definition of the target variable
$snr1$	unmasked SBox output	$\mathrm{SBox}(p[3] \oplus k[3])$
$snr2$	masked SBox output	$\mathrm{SBox}(p[3] \oplus k[3]) \oplus r_{out}$
$snr3$	common SBox output mask	r_{out}
$snr4$	masked SBox output in linear part	$\mathrm{SBox}(p[3] \oplus k[3]) \oplus r[3]$
$snr5$	SBox output mask in linear part	$r[3]$



(a) ASCADv1 fixed key dataset (b) ASCADv1 variable key dataset

Fig. 7: SNR value of each snr. The sample points highlighted in red indicate the largest peak and its location.

The countermeasure implementation has two masked states; the linear part is secured by 16 different masks with a table re-computation method, and the SubBytes processing is secured by the same input and output mask pair used for each state element. This means the leakage about an unprotected intermediate value (unmasked SBox output), which is strongly related to the key, is expected to be hidden. On the other hand, the leakage of shares made by the masking

procedure can be observed from traces. Hereafter, we call these known leakages snr1-5 following Benadjila et al. [3], as shown in Table 2. Each snr2-3 and snr4-5 pair is supposed to correspond to the pair of $share_1$ and $share_2$ in the simulation in the previous section. Figure 7 shows the SNR (F-test) value of each snr. The snr1 has small values overall; it essentially indicates no first-order leakage. The snr2-5 have some peaks, indicating leakage for each share. The snr4-5 are relatively high, respectively, and snr2-3 are smaller.

6.2 Revealing What Models Learned

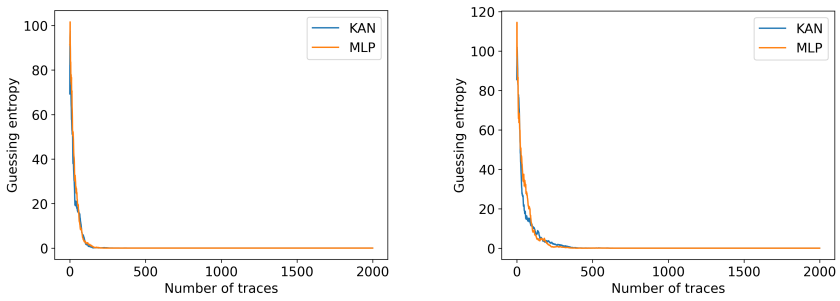
We trained MLP and KAN using the ASCAD dataset. Models receive all sample points: 700 on fixed key and 1400 points on variable key dataset. To select the MLP architecture, we considered model architecture MLP_{exp} from [36] and MLP_{best} from [3]. The former is designed for the LSB labeling, and the latter is for the Identity labeling. These achieved almost the same classification accuracy with LSB labeling, but the MLP_{exp} is significantly smaller architecture than the MLP_{best}; thus, we picked MLP_{exp} in this paper. The architecture of MLP_{exp} is [#inputs, 20, 10, 2] and has ReLU activation for each intermediate layer. KAN is set to [#input_samples, 5, 1, 2]. To design this architecture, we considered that the number of inputs significantly increased from the simulation settings in Section 5. We added a new layer after the input layer intended to aggregate and organize features from traces. The number of nodes in the layer was set to 5 as the minimum number that each node can represent snr1-5, respectively. Note that this does not guarantee each node actually represents a separate share in the trained KAN.

Table 3 lists the classification accuracy of models on the test set. The MLPs achieved 60.46% on the fixed key and 58.00% on the variable key datasets. The KANs achieved 61.18% on the fixed key and 59.17% on the variable key datasets. These results indicate that MLP and KAN achieved similar classification accuracy on each dataset. In addition, we calculated guessing entropy with these trained models (Figure 8). The number of traces for GE = 0 is listed in Table 3. The classification accuracy of MLPs was slightly lower than KANs. However, the number of traces to achieve GE = 0 was lower for MLP than for KANs. We assume this happened because the GE calculation is affected by both classification accuracy and confidence in each prediction.

Table 3: Attack results on ASCADv1 dataset with MLP and KAN

	Classification accuracy		#traces to achieve GE = 0	
Model	fixed key	variable key	fixed key	variable key
MLP	60.46%	58.00%	222	388
KAN	61.18%	59.17%	315	476

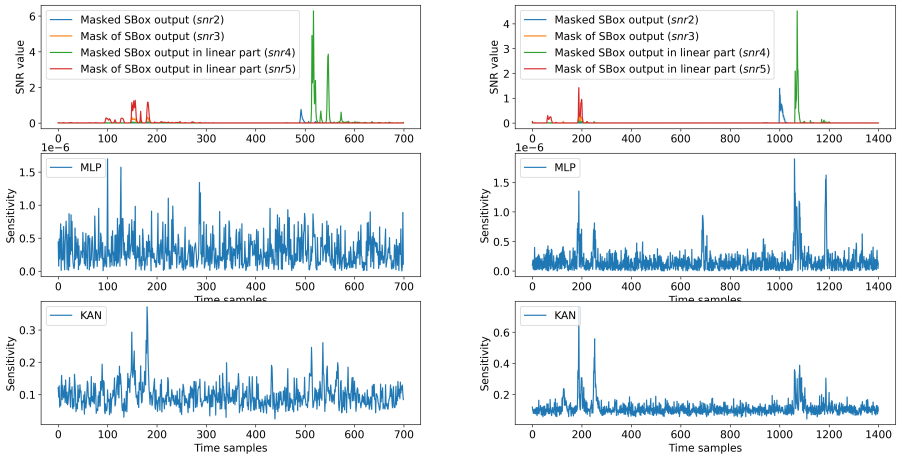
Figure 9 shows the models' input-based sensitivity. The sensitivity of the MLP model is the partial derivative of the model output with regard to the given



(a) ASCADv1 fixed key dataset. (b) ASCADv1 variable key dataset.

Fig. 8: Guessing entropy with MLP and KAN.

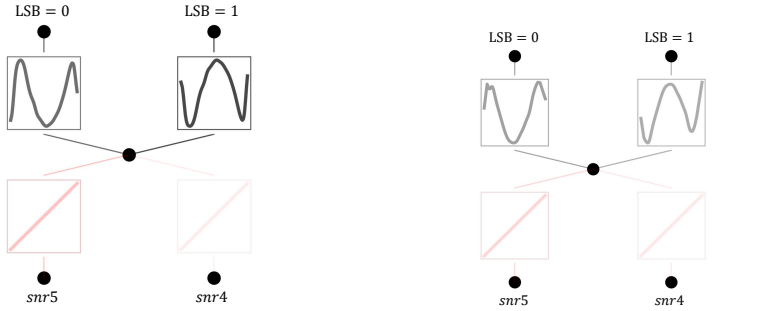
input, the same approach as [36]. The sensitivity of the KAN model is calculated by summing $output\_range/input\_range$ of activations belonging to each input sample, which is the score for the importance of each node (inputs) used for pruning KAN nodes [21]. Since KAN and MLP achieved similar classification accuracy and showed similar sensitivity distributions, this suggests that they learned a similar function. We can see sample points with high sensitivity that coincide with the points corresponding to the peaks of snr3, 4, and 5. Since the peak positions of snr3 and 5 are almost the same, and the model did not focus on snr2, it is reasonable to assume that the principal part of the learned function focuses on the pair of snr4 and 5.



(a) ASCADv1 fixed key. (b) ASCADv1 variable key.

Fig. 9: Sensitivity results.

Next, we trained KANs with the peak point of the $snr4$ and 5. The chosen sample point pairs are 517 ($snr4$) and 156 ($snr5$) on the fixed key dataset and 1071 ($snr4$) and 188 ($snr5$) on the variable key dataset. Since the number of input nodes is reduced to 2, the KAN's architecture is the same as the simulation: KAN[2, 1, 2]. The trained KANs' graphs are shown in Figure 10, which was trained using the same settings as the simulation, and the grid size is set to 17, where the input activations are set to the linear function for the same reasons as in the simulation. These KANs achieved 56.79% classification accuracy for the fixed key and 55.33% for the variable key dataset. Comparing KAN's graph on the right of Figure 5 and Figure 10, they have the same input activations and similar output activations. This indicates that KANs learned the same graph as the simulation, which is the EVM strategy, from real traces (ASCADv1 dataset). On the other hand, the classification accuracy of the KANs trained with real traces is higher than that of the simulation. In the next section, we discuss this is due to imbalance leakage from the data bus.



(a) ASCADv1 fixed key dataset. (b) ASCADv1 variable key dataset.

Fig. 10: Trained KAN's graph.

6.3 Imbalanced Leakage on Real-world Targets

To discuss why the classification accuracy of KANs trained with real traces is higher than that of the simulation, we first list the classification accuracy for each method and dataset for each target bit in Table 4. The classification accuracy of the EVM strategy was calculated by counting all combinations for each target bit. To calculate the classification accuracy for the KAN for the simulation dataset, we trained KAN[2,1,2] with the same setup as in Section 5 by changing the label to each target bit. For KAN with the ASCAD dataset, we trained KANs with two PoI settings; one used all sample points for input, and the other used only sample points of $snr4$-5. A typical HW leakage model considers

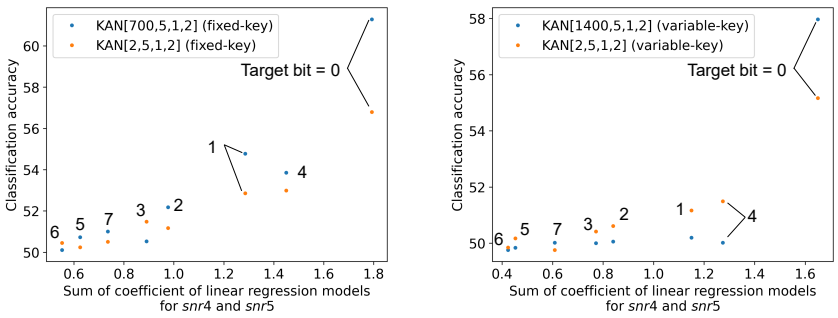
each wire (bit) on the data bus to contribute equally to power consumption. The EVM strategy and KAN on the simulation achieve almost the same accuracy for each target bit because they consider the typical HW leakage model. On the other hand, it can be seen from Table 4 that KANs' accuracies on the ASCAD dataset are biased towards the LSB.

We hypothesize that the reason for the bias is the power contribution of each wire is imbalanced in the ASCAD dataset. To confirm this, we trained linear regression models so that they predict trace amplitude from data on the bus. The target sample points are 156 ($snr5$) and 517 ($snr4$) for the ASCADv1 fixed key dataset and 188 ($snr5$) and 1071 ($snr4$) for the variable key dataset. For example, a linear regression model, which predicts $snr5$ on ASCADv1 fixed-key dataset, is represented as $y = \alpha_0 b_0 + \alpha_1 b_1 + \cdots + \alpha_7 b_7 + C$, where y is predicted value of the sample point (156), $b_0, ..., b_7$ are values (0 or 1) of each bit expected to be transferred on the data bus, $\alpha_0, ..., \alpha_7$ are coefficients corresponding to the bits, and C is a constant. Figure 11 plots the classification accuracy of each KAN and the sum of coefficients from the regression models ($snr4$ and 5) for each target bit. The plots show that there is a correlation between the classification accuracy and the coefficients. This indicates that the power contribution of each wire is imbalanced in the ASCAD dataset, and it affects the bias of the classification accuracy between the target bits. As the LSB wire contributes most to the power consumption (see Figure 11), neural network models trained on the LSB achieve higher accuracy than for other bits and models in simulation.

Table 4: Classification accuracy for each target bit.

			Target bit			
Method	Dataset	PoI	0	1	2	3
EVM strategy	Simulation	-	53.74%	53.74%	53.74%	53.74%
KAN[2,1,2]	Simulation	-	53.36%	53.36%	53.61%	53.70%
KAN[700,5,1,2]	ASCADv1 fixed key	All	61.29%	54.78%	52.18%	50.53%
KAN[2,5,1,2]	ASCADv1 fixed key	[156, 517]	56.80%	52.86%	51.17%	51.48%
KAN[1400,5,1,2]	ASCADv1 variable key	All	57.97%	50.20%	50.06%	50.00%
KAN[2,5,1,2]	ASCADv1 variable key	[188, 1071]	55.17%	51.17%	50.61%	50.42%

			Target bit			
Method	Dataset	PoI	4	5	6	7
EVM strategy	Simulation	-	53.74%	53.74%	53.74%	53.74%
KAN[2,1,2]	Simulation	-	53.72%	53.80%	53.85%	53.59%
KAN[700,5,1,2]	ASCADv1 fixed key	All	53.86%	50.73%	50.11%	51.01%
KAN[2,5,1,2]	ASCADv1 fixed key	[156, 517]	52.99%	50.24%	50.45%	50.51%
KAN[1400,5,1,2]	ASCADv1 variable key	All	50.02%	49.84%	49.76%	50.02%
KAN[2,5,1,2]	ASCADv1 variable key	[188, 1071]	51.49%	50.17%	49.85%	49.76%



(a) ASCADv1 fixed key dataset. (b) ASCADv1 variable key dataset.

Fig. 11: Correlation between classification accuracy and sum of coefficients in regression models.

7 Discussion

As showcased by our results in Section 5, KANs can effectively learn understandable mechanisms to combine mask shares in both bitwise and Hamming weight leakage models. In these smaller settings, the model plots clearly show activation patterns that match either an EVM strategy for HW simulations or a switching strategy for bit-leakage simulations. These explanations are significantly more fine-grained than those accomplished in the simulation experiments from [42]. Yap et al. can decompose the operations into a SAT equation where they can deduce what samples are being recombined, while our results show the precise strategy by which the models recombine shares. Further experiments with irrelevant input features also showcase that KANs learn to ignore those features quickly and that the standard model visualization functionality clearly shows what features are (ir)relevant. Altogether, these results indicate that KANs can generate explanations at the input level by attributing predictions to specific input features, matching the input visualization tools from related works [10], while also providing solid explanations of how models are actually recombining shares, improving over [42].

When we subsequently consider targeting real-world traces, we show with input visualization tools that the network learns from the correct leakage points. Moreover, when we select the most important points and retrain the KAN on those, we can match the learned combination scheme to the strategies found in simulations. While it is plausible that the same can be done directly for the models trained on full traces, selecting features makes it significantly easier as the models are much smaller, allowing for a much simpler visual inspection of the resulting network graphs. The attack performance of the model is also reasonable, being competitive with MLP models and significantly outperform-

ing architectures specialized for interpretability[6]. While the performance here is behind state-of-the-art models [45,38,32,1], further work in optimizing KAN training setups should help close this gap.

In general, KANs seem like a viable alternative to MLPs in terms of attack performance. On the side of interpratability, KANs clearly improve over MLPs. For now, the only real downside to using KANs seems to be the computational overhead of training them with larger input dimensions. In our computing environment[7], training KAN[700,5,1,2] in $\#steps = 5000$ and $batch\_size = 256$ takes 6 hours, and KAN[1400,5,1,2] in $\#steps = 7000$ and $batch\_size = 256$ takes 13 hours. In contrast, the training of MLPs is finished within 15 minutes. However, as mentioned in [21], no significant effort has been put into optimizing implementations and considering the interest of the ML community in utilizing KANs (see, e.g., [5,4,37]) it seems reasonable to expect the computational performance to be improved.

8 Conclusions and Future Work

In this paper, we propose using KANs as an alternative to MLPs for side-channel analysis. Overall, our results across both simulated and real-world traces indicate that KANs can be competitive regarding attack performance. Furthermore, we can reverse-engineer the method the networks use to recombine mask shares on simulations. Subsequently, we train KANs on the ASCAD dataset, resulting in a key recovery in ≈ 300 traces, performing competitively with a basic MLP from [3]. Moreover, we can find the share combination methods from simulations in the trained networks.

In future work, we plan to interpret profiling models using other labeling techniques, especially Identity labeling. We also plan to explore whether embedding KAN layers as part of larger networks, e.g., to replace the fully connected parts of CNNs from [45], can be an effective strategy to mitigate the computational overhead of training KANs on full traces.

References

1. R. Y. Acharya, F. Ganji, and D. Forte. Information theory-based evolution of neural networks for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):401–437, 2023.
2. C. Archambeau, E. Peeters, F. Standaert, and J. Quisquater. Template attacks in principal subspaces. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[6] In [42] retrieving a key-byte from the ASCADv1 fixed key dataset requires 7222 traces.

[7] Intel Core i7-13800H CPU, Nvidia GeForce RTX 4060 Laptop GPU, and 64GB DDR memory

3. R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptographic Engineering*, 10(2):163–188, 2020.
4. Z. Bozorgasl and H. Chen. Wav-kan: Wavelet kolmogorov-arnold networks. *CoRR*, abs/2405.12832, 2024.
5. R. Bresson, G. Nikolentzos, G. Panagopoulos, M. Chatzianastasis, J. Pang, and M. Vazirgiannis. Kagnns: Kolmogorov-arnold networks meet graph learning. *CoRR*, abs/2406.18380, 2024.
6. E. Cagli, C. Dumas, and E. Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without preprocessing. In W. Fischer and N. Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.
7. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In B. S. K. Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
8. C. De Boor and C. De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.
9. C. Genevey-Metat, A. Heuser, and B. Gérard. Trace-to-trace translation for SCA. In V. Grosso and T. Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2021.
10. A. Golder, A. Bhat, and A. Raychowdhury. Exploration into the explainability of neural network models for power side-channel analysis. In I. Savidis, A. Sasan, H. Thapliyal, and R. F. DeMara, editors, *GLSVLSI '22: Great Lakes Symposium on VLSI 2022, Irvine CA USA, June 6 - 8, 2022*, pages 59–64. ACM, 2022.
11. S. Hajra, S. Chowdhury, and D. Mukhopadhyay. Estranet: An efficient shift-invariant transformer network for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(1):336–374, 2024.
12. B. Hettwer, S. Gehrer, and T. Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. In K. G. Paterson and D. Stebila, editors, *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*, volume 11959 of *Lecture Notes in Computer Science*, pages 645–666. Springer, 2019.
13. S. Karayalcin, M. Krcek, L. Wu, S. Picek, and G. Perin. It's a kind of magic: A novel conditional GAN framework for efficient profiling side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 1108, 2023.
14. J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.
15. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

16. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996.
17. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag.
18. A. N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114, pages 953–956. Russian Academy of Sciences, 1957.
19. M. Krcek and G. Perin. Autoencoder-enabled model portability for reducing hyperparameter tuning efforts in side-channel analysis. *J. Cryptogr. Eng.*, 14(3):475–497, 2024.
20. L. Lerman, R. Poussier, O. Markowitch, and F. Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *J. Cryptogr. Eng.*, 8(4):301–313, 2018.
21. Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Y. Hou, and M. Tegmark. KAN: kolmogorov-arnold networks. *CoRR*, abs/2404.19756, 2024.
22. X. Lu, C. Zhang, P. Cao, D. Gu, and H. Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021, Issue 3:235–274, 2021.
23. H. Maghrebi, T. Portigliatti, and E. Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.
24. L. Masure, C. Dumas, and E. Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020.
25. G. Perin, I. Buhan, and S. Picek. Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis. In S. Bhasin and F. De Santis, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 53–81, Cham, 2021. Springer International Publishing.
26. G. Perin, B. Ege, and L. Chmielewski. Neural network model assessment for side-channel analysis. Cryptology ePrint Archive, Paper 2019/722, 2019.
27. G. Perin, L. Wu, and S. Picek. 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.
28. G. Perin, L. Wu, and S. Picek. I know what your layers did: Layer-wise explainability of deep learning side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 1087, 2022.
29. S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019.
30. S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina. SoK: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11):227:1–227:35, 2023.
31. C. Rechberger and E. Oswald. Practical template attacks. In C. H. Lim and M. Yung, editors, *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, volume 3325 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2004.
32. J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):677–707, 2021.

33. U. Rioja, L. Batina, J. L. Flores, and I. Armendariz. Auto-tune pois: Estimation of distribution algorithms for efficient side-channel analysis. *Computer Networks*, 198:108405, 2021.
34. T. Schamberger, M. Egger, and L. Tebelmann. Hide and seek: Using occlusion techniques for side-channel leakage attribution in cnns. In J. Zhou, L. Batina, Z. Li, J. Lin, E. Losiouk, S. Majumdar, D. Mashima, W. Meng, S. Picek, M. A. Rahman, J. Shao, M. Shimaoka, E. Soremekun, C. Su, J. S. Teh, A. Udovenko, C. Wang, L. Zhang, and Y. Zhauniarovich, editors, *Applied Cryptography and Network Security Workshops*, pages 139–158, Cham, 2023. Springer Nature Switzerland.
35. F. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In A. Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.
36. B. Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.
37. C. J. Vaca-Rubio, L. Blanco, R. Pereira, and M. Caus. Kolmogorov-arnold networks (kans) for time series analysis. *CoRR*, abs/2405.08790, 2024.
38. L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):147–168, Jun. 2020.
39. L. Wu, G. Perin, and S. Picek. The Best of Two Worlds: Deep Learning-assisted Template Attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(3):413–437, Jun. 2022.
40. L. Wu and S. Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):389–415, 2020.
41. L. Wu, Y.-S. Won, D. Jap, G. Perin, S. Bhasin, and S. Picek. Ablation analysis for multi-device deep learning-based physical side-channel analysis. *IEEE Transactions on Dependable and Secure Computing*, 21(3):1331–1341, 2024.
42. T. Yap, A. Benamira, S. Bhasin, and T. Peyrin. Peek into the black-box: Interpretable neural network using SAT equations in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):24–53, 2023.
43. T. Yap, S. Bhasin, and S. Picek. OccPoIs: Points of interest based on neural network's key recovery in side-channel analysis through occlusion. Cryptology ePrint Archive, Paper 2023/1055, 2023.
44. G. Zaid, L. Bossuet, M. Carbone, A. Habrard, and A. Venelli. Conditional variational autoencoder based on stochastic attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):310–357, Mar. 2023.
45. G. Zaid, L. Bossuet, A. Habrard, and A. Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, Nov. 2019.
46. L. Zhang, X. Xing, J. Fan, Z. Wang, and S. Wang. Multilabel deep learning-based side-channel attack. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(6):1207–1216, 2021.