Diffuse Some Noise: Diffusion Models for Measurement Noise Removal in Side-channel Analysis

Sengim Karayalçin[1][0009−0000−1598−8400], Guilherme Perin[1][0000−0003−3799−7636], and Stjepan Picek[2][0000−0001−7509−4337]

[1] Leiden University, Leiden, the Netherlands s.karayalcin@liacs.leidenuniv.nl
[2] Radboud University, Nijmegen, the Netherlands stjepan.picek@ru.nl

Abstract. Resilience against side-channel attacks is an important consideration for cryptographic implementations deployed in devices with physical access to the device. However, noise in side-channel measurements has a significant impact on the complexity of these attacks, especially when an implementation is protected with masking. Therefore, it is important to assess the ability of an attacker to deal with noise. While some previous works have considered approaches to remove (some) noise from measurements, these approaches generally require considerable expertise to be effectively employed or necessitate the ability of the attacker to capture a 'clean' set of traces without the noise. In this paper, we introduce a method for utilizing diffusion models to remove measurement noise from side-channel traces in a fully non-profiled setting. Denoising traces using our method considerably lowers the complexity of mounting attacks in both profiled and non-profiled settings. For instance, for a collision attack against the ASCADv2 dataset, we reduced the number of traces required to retrieve the key by 40%, and we showed similar improvements for ESHARD using a state-of-the-art MORE attack. Furthermore, we provide analyses into the scenarios where our method is useful and generate insights into how the diffusion networks denoise traces.

Keywords: Side-Channel Analysis, Deep Learning, Diffusion Models

1 Introduction

While standard cryptographic algorithms are generally considered (or, at least, believed after sufficient public analysis and scrutiny) theoretically secure, as retrieving the secret key from only inputs and outputs in a reasonable time is impossible, their real-world deployment poses additional attack surfaces. Deployments of these algorithms will unintentionally leak some information about their computation to the outside world through power consumption, timing, or electromagnetic emanation. These information leakages, or side channels, can allow an attacker to recover secret information from a device efficiently. Since being introduced by Kocher [19], significant research has been done into side-channel

attacks (SCA) and their countermeasures. We can broadly categorize SCA into two categories: 1) non-profiled attacks, where an attacker collects side-channel leakages and uses statistical distinguishers to extract the secret key [20, 4] and 2) profiled attacks, where the attacker builds a model for the leakage using a copy (clone) device they have full control over [7]. From the machine learning perspective, we can divide the algorithms into generative and discriminative ones.[3]

While countermeasures for side-channel attacks exist, over recent years, a significant rise in deep learning-based SCA (DLSCA) has shown that these countermeasures can, in some cases, be circumvented, see, e.g., [24, 37, 36]. In the profiled setting, straightforward applications of discriminative models allow progressively more efficient attacks [26, 17, 53, 45]. Similar approaches based on discriminative models have also been applied in the non-profiled setting [42, 9]. To a lesser degree, there are approaches based on generative models that allow for pre-processing of side-channel traces to simplify/improve attacks [48, 44, 52].

While generative models can be a "natural" setting for SCA (for instance, the template attack [7] is generative), we see fewer developments with generative models-based SCA in the last years compared to the discriminative ones. A part of the reason for this is that discriminative models excel at distinguishing among classes, which is a common setup for SCA (since we commonly consider the classification task). On the other hand, generative models generate new data, which is a natural option for data augmentation, a direction already explored in SCA.

In this work, we propose a novel approach to denoise traces based on Denoising Diffusion Probabilistic Models (DDPMs). Using these models, we can effectively remove environmental (Gaussian) noise from side-channel traces without requiring a reference set of 'clean' traces or profiling labels (see Table 1 for an overview of comparable approaches). We experimentally validate our approach against several datasets and show improved attack performance for non-profiled collision attacks, non-profiled attacks using deep learning, higher order correlation power analysis (HO-CPA), and horizontal attacks. Additionally, when we consider profiling attacks, our technique can be used to improve the profiling complexity and ease the difficulty of finding good model architectures using hyperparameter search. Our main contributions are:

- We showcase the first use cases for DDPM models to pre-process traces in SCA.
- We provide an analysis of the trained DDPM models that explains how traces are denoised and gives insights into situations where denoising is (and is not) possible.
- We showcase improvements in attack performance for state-of-the-art non-profiled attacks after denoising using the proposed model. For collision attacks against ASCADv2, the required number of traces to retrieve the key

[3] A common division in machine learning-based SCA is into supervised and unsupervised machine learning, but that relates to the task and whether there are labels available and not how the algorithm works.

	Denoising	No Countermeasure Disabling	No Profiling Labels
DAE [48]	✓	×	✓
DAE [15]	✓	✓	×
DDPMs [51]	×	✓	×
Ours	✓	✓	✓

Table 1: Comparing necessary assumptions for denoising traces.

is reduced by approximately 40%, and for MORE attacks against ESHARD, we show similar gains in performance.
– We show significantly decreased difficulty in finding model architectures and hyperparameter configurations for profiling attacks, especially in settings with low numbers of profiling traces.

The source code to reproduce the experiments is available in the following repository[4].

2 Background

2.1 Side-channel Analysis

Side-channel attacks [19, 20] are a class of attacks aiming at the implementation of cryptographic algorithms. The idea is that (physical) side-effects, e.g., timing [19], power [19], or the electromagnetic emanation [1] of the execution of the algorithm can leak information about secret internal values. An attacker then captures a (large) number of traces of the algorithm's execution by measuring one of these side channels and utilizes these to mount the attack.

We can broadly categorize side-channel attacks into two threat models. First, non-profiled attacks where an attacker utilizes statistical distinguishers to differentiate the correct (sub)key candidate from the wrong ones. Techniques here generally compute the hypothetical intermediate value for all possible (sub)key candidates and attempt to find a connection between these labels and the side-channel traces [20, 4].

The second category includes profiled attacks. In this case, an attacker has access to (and full control of) an open copy of the device to be attacked. This allows the attacker to characterize the (physical) leakage using traces captured from the copy device, significantly improving the efficiency of attacks against the target [7, 41].

Both of these categories of attacks rely on the fact that values that are operated on during the algorithm's execution are related to the measured traces. This relation is modeled by using a leakage model. A leakage model $f : Y \mapsto R$ mapping from an intermediate value $y \in Y$ to the leakage is generally composed of a part that relates to the hypothetical leakage of the value and a noise part. Common ways to model the leakage of this value are the Hamming Weight (HW)

[4] https://github.com/Sengim/diff_release

(the number of ones in the binary representation of y) or the Hamming Distance (HD) (the Hamming weight of the bitwise difference between y and the value it overwrites in a register).

To leverage this leakage model for key retrieval, the intermediate value an attacker targets needs to be related to the key and some known values. For AES implementations, the Sbox output in the first round is commonly used (for the Hamming weight and Identity leakage models). In this case, $y = \text{Sbox}(p_i \oplus k_i)$ where p_i and k_i are the i-th byte of the known plaintext and secret key. As these values are bytes, it is computationally feasible to calculate the hypothetical values for all 256 possible values of k_i and "match" those to the measured leakage. In this way, each key byte can be attacked separately, eventually leading to the recovery of the full key.

Signal-to-Noise Ratio (SNR) SNR is a leakage assessment metric that quantifies the amount of leakage that is present in a random value. For a set of traces X with intermediate values Y at sample index i, it is defined as:

$$\text{SNR}(X_i, Y) = \frac{\text{Var}_{y \in \mathcal{Y}}(E(X_i|y))}{E_{y \in \mathcal{Y}}(\text{Var}(X_i|y))}.$$

Here, E is the mean, Var is the variance of a random variable, and $\mathcal{Y}$ is the set of possible values in Y. We generally compute SNR for secret shares that leak directly (e.g., masks or masked sensitive values). In this work, we always compute SNR with 20 000 traces and the Identity leakage model.

2.2 Algorithmic Noise vs. Measurement Noise

We consider algorithmic noise to be the parts of the computation that are happening in parallel with the intermediate values we target. For example, an optimized hardware implementation of AES might execute several Sboxes in parallel, resulting in the Hamming weight of all output bits leaking together. If we want to target only one byte, the contribution to the leakage of the other bytes is considered noise. Measurement noise is then the part of the trace that is part of taking the physical measurements. This could be due to imperfections in the measurement setup or environmental factors. We generally assume this noise follows or is similar to, a Gaussian distribution [27].
The main difference between these types of noise for the purposes of unsupervised pre-processing of side-channel traces is that the algorithmic noise is part of the signal and is, therefore, not removed. An illustrative example is that if we take several measurements during the computation of a larger state, the algorithmic noise will stay the same for each of these samples, while the measurement noise will vary.

2.3 Datasets

ESHARD. The ESHARD dataset[5] contains EM measurements of an AES implementation protected with first-order Boolean masking. The dataset contains 1 400 sample points corresponding to the loading of the mask values and 100 000 traces with a fixed key. We target the Sbox output in the first round for all attacks. Note that we use the non-shuffled variant for all our analyses, as the shuffling was implemented by manipulating plaintexts a posteriori.

ASCADf. The ASCAD fixed key dataset (ASCADf)[6] contains EM measurements from an AES implementation protected with first-order Boolean masking. The dataset contains 60 000 traces with 100 000 samples each. We focus on a pre-selected window of 700 samples containing leakages for the masked Sbox computation in the first round for the 3rd key byte (which is the first masked byte). The dataset has a fixed key for all traces.

ASCADv2. The ASCADv2 dataset[7] contains power measurements of an AES implementation protected with an affine masking scheme and shuffling. The dataset contains 800 000 traces with 1 million sample points each. We take smaller part of the 15 000 sample extracted dataset used in [31], which contain 2 000 samples corresponding to a concatenation of indices 0-1 000 (loading masks), 6 040-6 540 (processing masked Sbox for third byte), and 11 250-11 750 (removing additive mask). Note that for this analysis, we disable shuffling by manipulating plaintexts a posteriori.

AES_HD. The AES_HD dataset[8] is an unprotected AES implementation on an FPGA board. The dataset contains 500 000 power traces using a fixed key. Each trace consists of 1 250 sample points. We target the Hamming Distance of register writing in the last round ($\text{Sbox}^{-1}[C_i \oplus k*] \oplus C_j$).

AES_HD_MM. The AES_HD_MM dataset[9] is an AES implementation on an FPGA board protected with first-order Boolean masking. The dataset contains 5 600 000 traces using a fixed key. The measurements contain 3 125 samples per trace. We target the same intermediate value as for AES_HD.

ASCON. The ASCON dataset[10] is an unprotected software implementation of the ASCON cipher in authenticated encryption mode [10]. The dataset consists of 200 000 traces where 100 000 traces use random keys for profiling and 100 000 traces use a fixed key. Each trace consists of 772 sample points corresponding to the first round of the initialization phase of the authenticated encryption protocol.

2.4 Discriminative vs. Generative Models

Machine learning algorithms can be divided into two categories: generative and discriminative. Discriminative algorithms are primarily concerned with simulat-

[5] https://gitlab.com/eshard/nucleo_sw_aes_masked_shuffled
[6] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key
[7] https://github.com/ANSSI-FR/ASCAD/tree/master/STM32_AES_v2
[8] https://github.com/AISyLab/AES_HD_Ext
[9] https://chest.coe.neu.edu/?current_page=POWER_TRACE_LINK&software=ptmasked
[10] https://zenodo.org/records/10229484

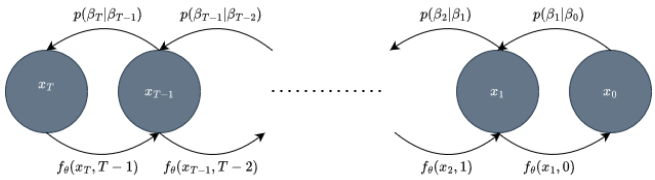


Fig. 1: Diagram illustrating the forward and backward process for training DDPMs, adapted from [14].

ing the conditional probability distribution of the output labels given the input features. The goal is to understand the decision boundary. On the other hand, generative algorithms are designed to simulate the joint probability distribution of the input features (possibly conditioned on labels). To create new samples, their goal is to learn the underlying data distribution.

2.5 Denoising Diffusion Probabilistic Models (DDPMs)

Denoising Diffusion Probabilistic Models were first introduced by Ho et al. [14]. Over the next few years, models based on the DDPM paradigm have outperformed state-of-the-art generative models on image generation and other tasks [50]. DDPM training is based on a relatively straightforward paradigm: during training, we iteratively add some noise to an image (or some other type of data) for T steps; this is referred to as the forward process (left direction in Figure 1). Then, for an image x_t where noise has been added t times, we train the model to predict x_{t-1} and thereby remove noise. This is called the backward process (right direction in Figure 1). The central idea here is that when we start from fully random noise and iteratively remove noise, we can generate realistic-looking images as the diffusion models try to 'amplify' patterns in the noise.

More formally, the forward process is defined using a Markov chain from x_0 (the original images) to x_T (Gaussian noise) and transitions $q(x_t|x_{t-1})$. We then have a noise schedule $\beta_0, \beta_1, \ldots, \beta_T$ and corresponding values $\alpha_0, \alpha_1 \ldots, \alpha_T$ (with $\alpha_0 = 0$ increasing to $\alpha_T = 1$). These α_i allow us to generate pairs x_t, x_{t-1} for arbitrary $1 < t \leq T$ using $x_t = (1-\alpha_t)x_0 + \alpha_t Z$ and $x_{t-1} = (1-\alpha_{t-1})x_0 + \alpha_{t-1} Z$ where $Z = \mathcal{N}(0, 1)$. These pairs can then be used to minimize the squared error of our diffusion model parameterized with weights θ, i.e., $\arg\min_\theta (f_\theta(x_t, t-1) - x_{t-1})^2$ using uniformly sampled t from $[0, T]$ for each mini-batch. For a more detailed description of diffusion models, see [14].

3 Related Work

The profiling side-channel analysis started with the template attack [7]. A few years later, the stochastic attack was also introduced [41]. Interestingly, both

of those attacks build generative models. With the introduction of "classical" machine learning in SCA, the community moved the attention to discriminative models. Still, deep learning-based generative models have been used in the last few years, with the primary goals to either pre-process the side-channel traces or generate synthetic traces.

3.1 Pre-processing using Neural Networks

While classical techniques for pre-processing side-channel traces have been explored, such techniques often require a significant domain expertise and error-prone manual intervention to achieve optimal results [23, 34, 38, 27]. As such, the focus of the SCA community has recently moved to automated techniques utilizing deep learning. A first approach to using denoising autoencoders for removing noise from side-channel traces was proposed by Yang et al. [49]. There, the authors used trace averaging to imitate a 'clean' set of traces, which can then be used to train an autoencoder to remove noise from the original traces. Subsequently, Wu and Picek [48] extended the approach to cover more hiding countermeasures like desynchronization and random delays. Berg et al. [2] further investigated hyperparameter configurations for these networks. Finally, Hu et al. [15] included additional training objectives to improve the performance of autoencoders for removing noise from traces. Beyond autoencoders, Wu et al. [44] utilized triplet networks to extract representations from traces that can be used to mount template attacks.

More recently, several studies have explored generative approaches for pre-processing. Genevey-Metat et al. [11] utilized a GAN to translate traces between side-channel domains. Cao et al. [6] used a GAN approach to tackle portability challenges by transforming measurements from the attack device to the profiling device. Krček and Perin [21] use autoencoders to map traces to the same dimensions to reduce hyperparameter tuning efforts. Karayalçin et al. [16] investigated a Conditional Generative Adversarial Networks (CGANs)-based framework to emulate feature selection for masked implementations without access to mask values. Finally, Zaid et al. [52] used variational autoencoders to model the physical leakage and subsequently leverage these models for attacks.

The main limitation of these approaches for pre-processing is that they only work in settings with additional assumptions over the standard non-profiled setting. The approaches using autoencoders in [49, 48, 2] require a set of 'clean' traces that serve as a target for the networks. When considering Gaussian noise, this clean set can be emulated by averaging, but for masked implementations, this requires access to mask values [49] which is not always feasible even for evaluators [29]. For the methods in [44, 6, 16, 52, 15], a labeled profiling set (or access to masks values for [52]) is necessary for training the models. The trace translation in [11] requires paired measurements in different side-channel domains, and it necessitates that the target side-channel domain is easier to attack, essentially mimicking the 'clean' set of traces in [48]. None of these approaches can effectively pre-process traces in a fully non-profiled setting.

3.2 Other Approaches using Generative Models in SCA

Several works have looked at applications of generative models for SCA. To generate additional traces, Wang et al. [43] considered CGANs to expand the size of the profiling set. Subsequently, Mukhtar et al. [33] improved upon the network architecture used in [43]. Yap and Jap [51] proposed the use of diffusion models to generate additional traces. Finally, Lu et al. [25] also use DDPMs to generate additional traces, but use a model architecture tailored for SCA. In all of these works, the authors relied on having access to a profiling device to label traces for training the networks. The resulting networks are then utilized to generate traces while providing label information to the network to control the trace generation. Note that while Transformers are often used in generative contexts in natural language processing, the Transformer architectures in [13, 5, 22, 12] are used as classification models (i.e., in a discriminative setting).

4 Denoising Diffusion Probabilistic Models for Removing Noise in SCA

While utilizing DDPMs for data generation to improve side-channel attacks is a straightforward direction, the results from Yap and Jap [51] suggest that there do not seem to be any significant advantages to using DDPMs over previously used (C)GAN methods [43, 33]. Additionally, using DDPMs for trace generation requires profiling labels (or even mask knowledge) to allow for useful trace generation, i.e., generating trace-label pairs, which limits the applicability to a profiled setting. On the other hand, we utilize unsupervised DDPMs to denoise traces. Notably, using unsupervised DDPMs means that our models cannot be used to generate new traces with corresponding labels. Conversely, utilizing the supervised DDPMs for denoising is possible, however, as these models require label information which means they cannot be used for unlabeled traces and have significant downsides for pre-processing (see Section 8.2).

The key idea here is to take a diffusion model parameterized with θ, $f_\theta : X^m \times T \mapsto X^m$, where X^m is a side-channel trace with m samples and $T = \mathbf{Z}_n$ that we train using standard diffusion model training (see Section 2.5) on our measured traces. After training, we then input actual traces (or x_0) and try to remove noise (or predict x_{-1}) from these traces. This then results in every original trace being transformed into a denoised version. We note that using diffusion models directly to take real samples as input and output a cleaner version has (to the best of our knowledge) not been done before. Image super-sampling using DDPMs has been done but generally requires specialized training setups and architectures [32].

The reasoning here is that we can consider the side-channel traces as a combination of signal S and environmental/measurement noise Z. If we then write $x_i = q_i * x_i + (1 - q_i) * Z$, where q_i is some schedule for the forward process (note that $q_i < q_{i-1}$), the optimization objective becomes $min_\theta(f_\theta(x_i, i) - x_{i-1})^2$. As the trained model has been trained to remove Gaussian noise, the model should

then be able to remove some of the actual noise Z present in the original trace. Notably, this assumes that the distribution of the environmental/measurement noise Z is (somewhat similar to) a Gaussian distribution, but this is a common assumption in the SCA domain, see, e.g., [7, 27, 48].

The main advantage of this approach is that the models are trained to be agnostic of implementation specifics. The only requirement is that the noise we are trying to remove follows a Gaussian distribution. As the diffusion model tries to reconstruct the trace, its output will also still follow the original trace's structure in terms of intermediate values. While this also holds for the denoising autoencoders in [49, 48], the training procedure of the autoencoders necessitates access to a reference set of 'clean' traces, which requires the ability to disable countermeasures on a profiling device. Note that while in [48], the authors also emulate this reference set by averaging traces, this still requires a large number of additional measurements, complicating the process and making it only possible for unmasked implementations. Indeed, in masked implementations, an attacker cannot know in which traces the same intermediate values are processed (i.e., which traces to average) without access to masking randomness [49].

4.1 Network Architecture

To keep the focus of this work on the viability of DDPMs for denoising traces in an unsupervised context, we only use synchronized traces. This allows us to restrict our architecture to shallow MLPs as these have been shown to be effective for processing synchronized side-channel traces [37]. As such, the design choices for our architecture are relatively simple: we follow the general structure of a U-Net [39] where we first downsample for several layers, then keep the same dimensions for some layers to induce a compressed latent representation, and finally upsample again to the original trace dimensions. We utilize batch-normalization layers in the downsampling section of our network to stabilize training. The network architecture can be seen in Figure 2.

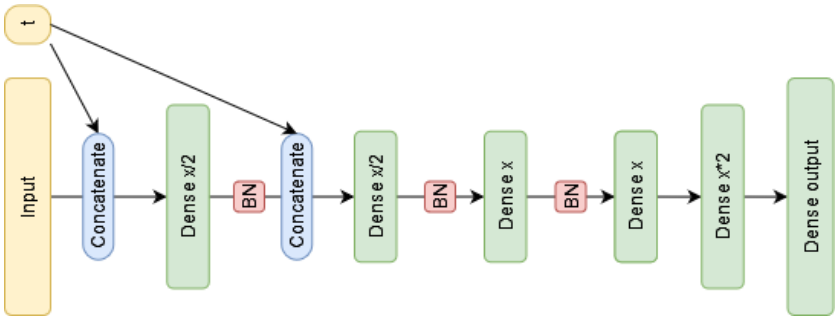
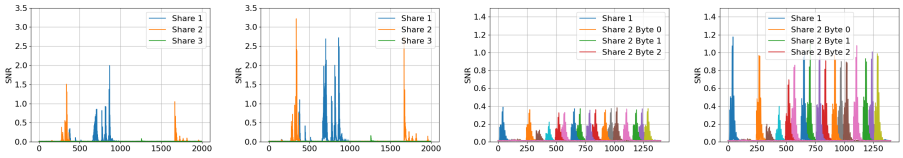


Fig. 2: General model architecture for the input of size X.



(a) Original (b) Diffused (c) Original (d) Diffused

Fig. 3: SNR values for secret shares for ASCADv2(left) and ESHARD(right).

4.2 Hyperparameter Setup

To train the DDPM models across all experiments, we use the Adam optimizer [18]. The learning rate is scheduled according to an exponential decay[11] schedule with an initial learning rate of 0.001, the decay rate of 0.96, and 10 000 decay steps. We train all models for 200 epochs using batch size 200. We use the *tanh* activation function for every intermediate layer and the linear activation for the output. We use $T = 16$ for all of the experiments with a linear noise-schedule [14]. These hyperparameters perform well and allow for reasonably effective denoising against the considered targets. We arrived at these values after preliminary testing. Note that these are not optimal, but we refrain from further optimization. We consider finding optimal architectures/hyperparameters to be out of the scope of this work. We provide further experiments to show the effects of some hyperparameter variations in Appendix C, which indicate that hyperparameter optimization is not as difficult for these models as it is for conventional DLSCA.

For the denoising of the traces after training, we observe in the initial set of experiments that predicting traces with $t = T - 1 = 15$ ($f_\theta(x_0, 15)$) works significantly better than using $t = 0$ ($f_\theta(x_0, 0)$). As such, we use $t = 15$ for all experiments unless otherwise specified. Our intuition here is that for higher t, the model gets noisier inputs during training, which forces it to find patterns in its input data more aggressively, resulting in better denoising performance. However, we do not claim that this is always best, and further work exploring inference time uses/augmentations of diffusion models could result in further improvements.

4.3 Proof of Concept

To provide a proof of concept for our method, we first look at testing against (relatively) noisy trace sets from software targets. First, we consider the ASCADv2 target as this is currently the most difficult public software target. There, the leakage of the masked output is noisy (SNR around 0.08), which potentially allows for significant benefits. Second, the ESHARD target provides measurements of a software implementation where both the mask and masked Sbox output leak

[11] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/schedules/ExponentialDecay

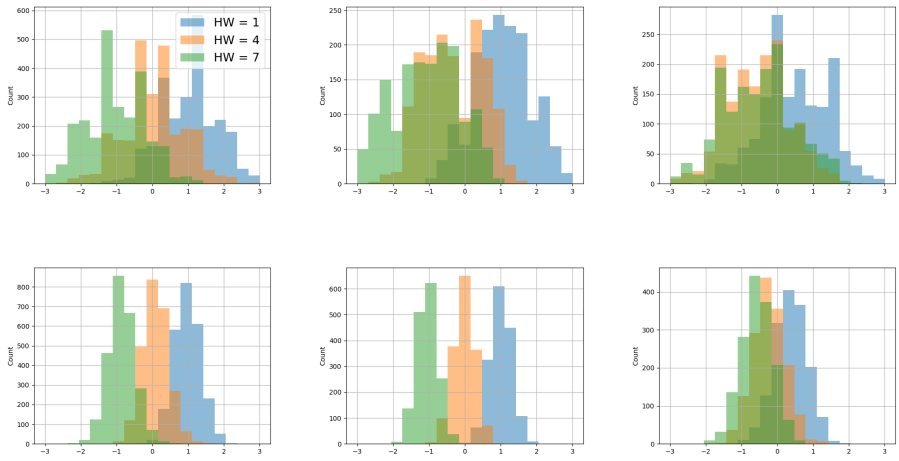


Fig. 4: Histogram for the HW values in the highest SNR samples for Original (top) and Diffused (bottom) traces for (left-right) ESHARD, ASCADv2 share 1, and ASCADv2 share 3.

with relatively low SNRs (see original ESHARD SNRs in Figure 3). Note that in this section, we only consider software targets where the measurements contain at least a moderate amount of measurement noise to highlight success cases of our method.

We train the models using all available traces and subsequently use the trained model to obtain denoised traces. We use the SNR of the secret shares in the traces (using the Identity leakage model) as a measure of how successful the noise removal was. In Figure 3, we see the SNR results for a trimmed version of the ASCADv2 dataset. The results clearly show that the SNR peaks for all three shares are significantly improved. For ESHARD, we also see significant improvements in SNR in Figure 3. These results show that our networks are effective at amplifying the side-channel signal. However, there are significant differences in the magnitude of the improvement for various secret shares and features. In Section 4.4, we aim to explain these differences using simulated traces.

In Figure 4, we showcase histograms of the values in the highest SNR point for the secret shares. We can observe that, especially for share 1 of ASCADv2 and for ESHARD, the distributions are visibly much smoother, and the separation between classes is clearer for the diffused traces. These results indicate that the diffusion networks can effectively smooth out the noise from side-channel traces while maintaining the leakages.

4.4 Simulations

Next, we explore in what situations we can improve the SNR of side-channel measurements using our DDPMs. To accomplish this, we utilize simulations

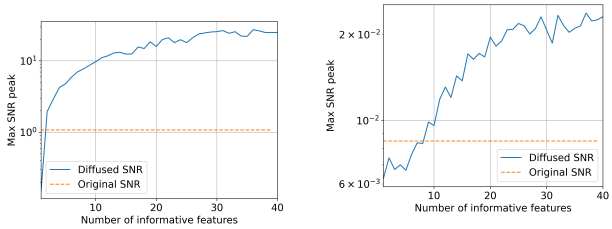


Fig. 5: Maximum SNR value for diffused traces simulations for varying numbers of informative points with low(left) and high(right) added noise.

with varying noise levels and a varying number of informative points. We follow the procedure:

1. we generate traces of 100 points noise following a normal distribution, and then,
2. for $0 < n \leq 40$ of these points (to allow different settings), we include the Hamming weight of an 8-bit intermediate value y uniformly sampled from the range $[0, 255]$.

The main purpose of varying the number of leaky points is to determine how the DDPMs are amplifying the side-channel signal. While the results in Section 4.3 show clear improvements in terms of SNR for individual features, the networks cannot create more information than what is present in the original trace. As such, the increased SNR in individual features must come from other trace points. Intuitively, combining information from several points leaking the same value is a straightforward way to amplify the signal in each of these points. As can be seen in Figure 5, there is a clear link between the number of leaky features and the level of SNR achieved. Notably, for both the high and low noise scenarios, the model does not increase the SNR if only one leaky feature is present. In the low-noise scenario, the model already shows significant improvement over baseline SNR when two leaky features are included, improving further with more leaky features. In the high-noise scenario, more leaky features are required before the SNR levels are improved over the baseline.

Overall, these results strongly suggest that diffusion models learn to differentiate the side-channel signal from noise by looking for correlated features in the trace. By finding and combining information from those related points, the model can decrease the error in its output. This is relevant for real-world side-channel traces when we take several measurements during an operation that leaks some sensitive value, e.g., the oscilloscope has a high sampling rate or some sensitive value is manipulated in several trace points.

4.5 Gradient Visualization

To validate that the DDPMs learn to combine information across correlated features in real-world settings, we visualize what features contribute to one of the output features. We use gradient visualization techniques that have been

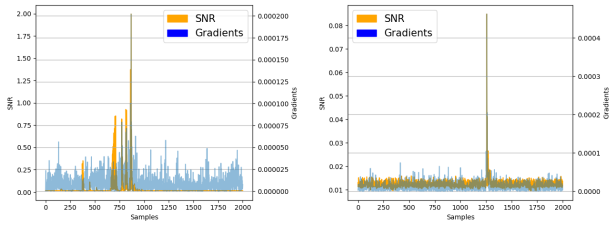


Fig. 6: Gradients vs. SNR values share 1 and 3 of ASCADv2.

previously used in SCA [30]. In Figure 6, we provide the gradient visualizations for the maximum SNR feature. We clearly see that sample points correlated with the most informative sample influence the model's output. Notably, these results explain the significant differences in the magnitude of the SNR increases for different secret shares of ASCADv2 we saw in Figure 3. As can be seen in Figure 6, the diffusion model can only utilize a small number of samples to combine information for share 3, while in the case of share 1, there are significantly more samples to learn from. These differences follow the demonstrated trends in Section 4.4. Note that the results for other targets are qualitatively the same using this analysis.

5 Experimental Results

5.1 Non-profiled Attacks

Multi Output Regression Enhanced (MORE) This section provides results for state-of-the-art non-profiled attacks using DL [40]. The basic idea of this attack is to train one model labeled for every possible key and conduct the regression task. As the labels generated using the correct key are the only ones that are related to the trace, the model should then most accurately predict labels of the correct key. A ranking for key candidates can then be created by measuring the network error for each candidate. We only show results against ESHARD as breaking the ASCADv2 target is still infeasible using the MORE methodology, while for ASCADf and the hardware targets, diffusing traces does not make a difference in terms of attack performance. We generate a distribution of key ranks using 40 separate random models following the hyperparameter ranges used in [40]. We choose this method as it allows us to assess the impact of diffusing traces on the difficulty of defining an appropriate model configuration, and it reflects directly on the effectiveness of the ensemble-based attacks that use these random models. The diffusion model in this case is the same as in Appendix A using 10 000 traces. We use the HW leakage model and target the third key byte. The results in Figure 7 showcase that diffusing the traces helps significantly. Attacks using diffused traces perform similarly at 20 000 traces to the attacks using 50 000 original traces. This indicates that diffusion models sig-

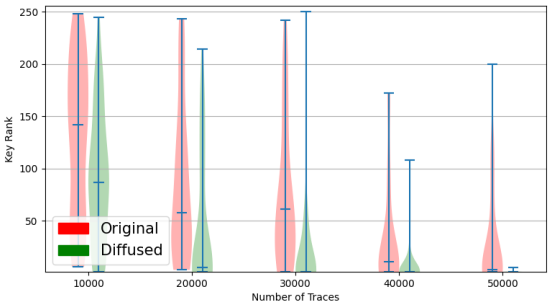


Fig. 7: MORE results for ESHARD.

nificantly help the consistency of training discriminative models, especially in more restricted settings.

Collision Attack against ASCADv2 To demonstrate the practical relevance of our approach, we first showcase attacking results in a non-profiled context. We focus on the collision attacks as described by Wu et al. [46], which aim to recover the bitwise difference between sub-keys (key-deltas). These key-deltas can then be used to brute-force one key byte, leading to full key recovery (given correct key-deltas). We include this attack as it is the only attack that can break the ASCADv2 dataset in a non-profiled setting.[12] Note that Cristiani et al. [8] also showcased successful attacks against the same implementation, but they require a different acquisition campaign with significantly more traces.

For training the DDPM, we use the intervals given in [46]. Note that for these attacks, the shuffling countermeasure is disabled, and we simulate a fixed attack key for the profiling set (see [46] for details). To simplify the analysis, we concatenated the used 100 sample intervals into one 1 600 sample trace and trained the diffusion model on 20 000 such traces to limit computational overhead. We then executed 50 runs on randomly sampled traces from the 500 000 profiling traces and averaged key-delta ranks to achieve a Guessing Entropy(GE) estimate. The results in Figure 8 clearly favor the diffused traces. In fact, using diffused traces can successfully reduce GE for all key-delta candidates below 1 using 60 000 traces, while three of the deltas are not fully recovered using 100 000 traces for the original traces.

Horizontal Attacks against Public Key Implementation To illustrate that our method is generally useful for analyzing side-channel traces, we showcase improvements to the horizontal attack from [35] which targets individual bits of the ECC key by classifying trace segments. In this attack, initial labeling that is only slightly better than random guessing (around 52%) is iteratively improved

[12] With shuffling disabled.

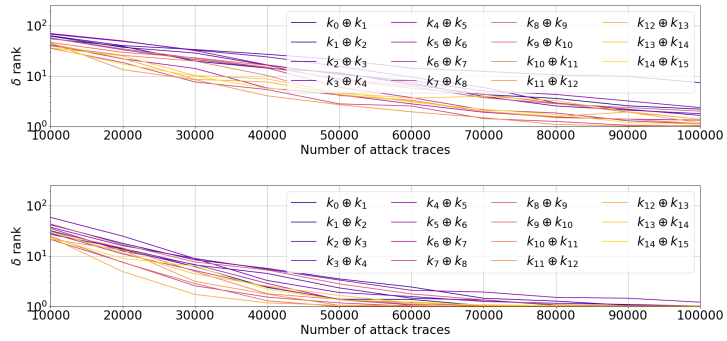


Fig. 8: ASCADv2 collision attacks for Original (top) and Diffused (bottom) traces.

upon using CNNs. In this work, results are presented using both an optimized fixed CNN architecture and a new random CNN at each iteration (for more details, see [35]). Note that in subsequent work, it was shown that similar attack performance could be achieved in some cases using only one neuron instead of larger CNNs [3]. We only show results on the cswap_arith dataset as the cswap_pointer is significantly easier to attack, and almost every network setup achieves 100% accuracy on both the original and diffused traces. The diffusion model is trained in the standard way using the 63 750 traces. For this dataset, we use batch size 1 000 and obtain denoised traces with $t = 0$ instead of $t = 15$ as this achieved better results.

The results in Table 2 again show significant improvements to accuracies by using diffused traces. In all cases, the maximum accuracy using diffused traces is higher than using original traces. Especially in the case where only the simplest perceptron from [3] is used, we obtain 99.2% maximum accuracy using diffused traces while only 80% using the original traces. For CNNs, we see the fixed CNN improves between 10% and 15%, and using random CNNs, we find the only attack achieving 100% maximum accuracy uses diffused traces. Still, average accuracies are not always improved. This is mainly due to the number of iterations optimized for the original traces. For the diffused traces, the average accuracy is maximized in earlier iterations and subsequently decreases sooner.

	One neuron	CNN	CNN + Dropout	Random CNN	Random CNN + Dropout
Original	70.9/79.2%	63.6/73.7%	55.2/75.7%	71.7/80.0%	98.6/99.6%
Diffused	96.3/99.2%	70.8/87.1%	50.1/83.5%	62.6/81.1%	99.6/100%

Table 2: Comparison of Average/Max single trace accuracy for key bits using the one neuron perceptron from [3] and CNN setups from [35].

6 Datasets with Algorithmic Noise

In previous sections, we have shown significantly improved attack performance against targets that have limited algorithmic noise. In these cases, it is clear from the results in Section 4.3 that the measurement noise is mitigated by using diffusion models. However, when we consider targets that process larger states, the denoising is less relevant.

In Figure 9, we see that the DDPMs cannot improve the peak SNR for any of the targets without added noise. Only for AES_HD_MM, we see that the SNR of the other samples is increased by a marginal amount. Note that this does not seem to affect attack performance; for profiling attacks using 100 random MLP models using the ranges from Table 4, GE is 60.78 ± 23.95 and 59.85 ± 21.00 for original and diffused traces, respectively. Looking at traces with added Gaussian noise, we see improved SNR for ASCON and AES_HD_MM. For AES_HD, SNR does not improve, presumably, as the noise level is relatively high and the intermediate value is only leaked in a very small number of samples following the results on simulations in Section 4.4.

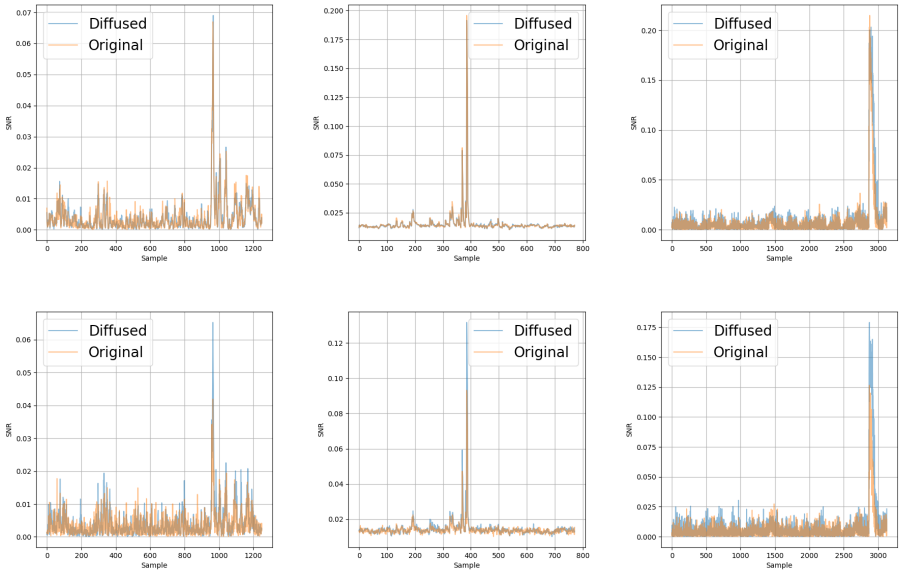
These results indicate that the diffusion models are not as useful for measurements with mostly algorithmic noise. However, even for hardware targets, we can see some improvements to SNR in specific samples, indicating that removing (some) measurement noise is still achievable in these cases. Additionally, when we add Gaussian noise, we see clear improvements in SNR for both the ASCON and the AES_HD_MM targets, reinforcing the usefulness of our diffusion models in scenarios with noisier measurements. Since noisier settings are more relevant from a practical perspective, and even the current results with deep learning perform well in scenarios with little noise, we consider our approach highly relevant and applicable in real-world settings.

7 Comparison with Denoising Autoencoders and Signal Processing

While several works have examined the use of techniques from the deep learning domain for pre-processing side-channel traces, almost none of these techniques are directly applicable in a non-profiled setting. As mentioned before, most either require profiling labels [44, 16, 15] or the ability to capture some 'clean' target traces [49, 48, 11]. However, as mentioned in Section 4.1 of [48], denoising autoencoders can be used for the same purpose.[13] We also include signal-processing techniques that can remove noise. We use moving averages and principal component analysis (PCA). For each of these, we try window sizes and number of components up to 100 and report the best results.

From the results in Table 3, we can see that autoencoders trained directly to reconstruct traces, i.e., without a 'clean' set, are not effective at removing measurement noise. Both the convolutional architecture from [48] and our DDPM

[13] Note that autoencoders trained in this way are equivalent to training DDPMs with one fixed noise step.



(d) AES_HD (e) ASCON (f) AES_HD_MM

Fig. 9: SNR values for intermediate values for various datasets for original traces (top) and traces with added noise (bottom).

architecture trained as autoencoders fail to increase SNR over the original traces. While tuning the architectures for each specific dataset could lead to improvements, we note that this is not necessary for our DDPMs. In addition, the autoencoders trained to reconstruct traces can easily overfit and memorize their training traces, increasing the difficulty of finding an appropriate architecture, especially in non-profiled contexts. Overall, it seems clear that in a scenario without a set of clean target traces (or label/mask knowledge), autoencoders are not appropriate for removing noise.

When we compare it to signal processing, we see that the performance is more competitive. In these cases, we mainly highlight that the optimal configuration differs for each setup. Notably, finding appropriate parameters depends on the physical leakage characteristics, and the appropriate parameters differ for different shares in the same traces (see ASCADv2 in Table 3). We note that it should generally be possible to use classical signal-processing techniques to achieve similar (or better) performance for individual shares but that doing this effectively in non-profiled settings (without access to masks/keys to verify the approach) is highly dependent on evaluator expertise.

	Original	DDPM	DAE CNN from [48]	DEA using DDPM MLP network	MA(n)	PCA(n)
ASCADf	1.30	1.24	0.76	1.17	1.63(20)	**1.63(21)**
Eshard	0.39	1.06	0.20	0.03	1.40(7)	**1.63(40)**
ASCADv2 share 1	2.00	**8.90**	1.35	1.93	2.69(14)	2.35(41)
ASCADv2 share3	0.08	**0.19**	0.06	0.03	0.09(5)	0.03(99)

Table 3: Comparison of max SNR peaks for denoising autoencoder (DAE) approach, moving averages (MA), and PCA. The (n) denotes the window size for MA and the number of components, respectively.

8 Discussion

Our results showcase that DDPM models can learn useful representations of side-channel traces in unsupervised contexts. Additionally, the analysis in Section 4.4 shows how the networks learn these representations and give an intuition for why the denoising can work. The mechanism is quite straightforward. To remove noise from a leaky sample point, the network needs more information about the leaking value. To accomplish this, it can find features that leak the same value and combine the information from these features to arrive at a less noisy version of the feature (see Section 4.5 for validation). In effect, we compress the information from several leaky samples into a singular sample.

The benefits of this for non-profiled attacks are clear. For these attacks, we often utilize only a single feature to represent each secret share [8]. In these contexts, our models allow for the implicit utilization of several leaky samples without having access to mask values. Additionally, for collision attacks, the stronger separation between classes can clearly reduce the number of traces necessary to detect key differences.

For profiled attacks, the benefits are less obvious. In principle, using LDA to reduce the dimension from a sufficiently large number of informative samples effectively compresses the information from all of those samples and thus eliminates the benefits of diffusing traces. Similarly, a well-trained neural network should implicitly combine the information available across a trace. However, in practice, we see significant benefits to denoising traces before training neural networks. Our results in Section 5.1(and in Appendix B) clearly indicate that the difficulty of finding appropriate hyperparameters for neural networks and the required number of traces for profiling is significantly reduced.

While our results show significant gains for the showcased attacks against some targets (specifically ESHARD and ASCADv2), it is clear that these benefits are not universal. Our method does not improve the SNR for datasets that contain mostly algorithmic noise. However, the SNR also does not seem to be harmed by pre-processing the traces using diffusion models, limiting the downside of using our method to the (limited) computational overhead required for training the diffusion model. The artificial addition of Gaussian noise in Section 6 also showcases that the method is still effective in more difficult scenarios when the measurements are more noisy.

The main takeaway from these results is that including diffusion models for pre-processing traces within the evaluation can significantly reduce the overhead caused by environmental noise while not requiring the same level of expertise to tune as other methods from the DL domain. The tuning of diffusion models in the SCA context seems relatively straightforward, and the pre-processing can be used to simplify any subsequent analysis. Especially in contexts where individual samples are used to represent the leakage from sensitive values, like (higher-order) CPA, our method allows for the combination of information from several samples without any additional access assumptions or alterations to the attack methodology. While the practical results presented in Section 5 showcase strong benefits in terms of attack performance, these are clearly influenced by our choice of datasets. Overall, the attack improvements are only present when the SNR of secret shares is similarly improved. For targets aside from ESHARD, ASCADv2, and cswap_arithmetic, utilizing our DDPMs does not seem to make a significant difference for practical attacks in our experiments unless we artificially inject Gaussian noise to simulate noisier measurement setups. Note that there may be some benefits (minor SNR improvements we see for AES_HD_MM and ASCON), but this did not make a difference for the considered attacks. Another consideration for interpreting our results is that the network and hyperparameter settings leave significant room for improvement. We aimed to present the denoising method using DDPMs, not to optimize the denoising performance for each specific scenario. In fact, even the results in Appendix C show that larger improvements than those presented in Section 5 are relatively straightforward to achieve by changing only one or two hyperparameters. Consequently, this suggests that future research looking into more complex network architectures and hyperparameter optimizations would still be beneficial in evaluating the full potential of our approach.

8.1 On Computational and Profiling Cost

While training generative models is often associated with significant computational and data requirements, our results showcase the opposite. We use relatively small models that require limited training data. The time required to train each of these models[14] requires between 10 minutes and 2 hours, impacted quite heavily by the number of profiling traces used. Note that the time required to train these models is almost identical for autoencoders with the same model architectures.

Furthermore, we see that the requirement for large numbers of training data for reasonable performance is not that strong. In the success cases, training DDPMs for denoising using significantly fewer traces than are required to successfully attack already improves the attack performance. Notably, this is because the models are not required to generate fully new traces and do not have to learn the relationship between traces and target values (see Section 8.2 for more discussion). In fact, as supported by simulation results in Section 4.4,

[14] On a workstation equipped with an NVIDIA RTX 4080 and 64 GB of RAM.

what the models need to learn to denoise effectively is that certain values leak in several samples. This then means the models can combine the information in these points to amplify the leakage if the noise in these points is (somewhat) independent as we discuss in Section 6.

8.2 On the Relevance of Unsupervised Pre-processing

It may seem somewhat counter-intuitive to neglect using labels during pre-processing for profiled attacks given their availability. However, from our perspective, pre-processing methods that use label information suffer from the key limitation that they rely on learning the relation between traces and labels, which is the core challenge for conventional classification models. If the labels help the pre-processing, it is clearly possible for the pre-processing network, and probably a conventional classification network, to learn this relationship without the pre-processing.

Furthermore, any network that takes label information as its input to pre-process a trace cannot be used to pre-process traces from an attack set where the key is not known. Note that in non-profiled settings, it is potentially possible to use plaintext as a stand-in for the intermediate value to mitigate this issue [47]. However, this still does not mitigate the above downsides of such a label-based approach.

9 Conclusions and Future Works

We have presented an approach for utilizing DDPMs to remove noise from side-channel traces. As shown in Section 4.4, our approach is effective at increasing SNR levels of traces when several samples in the side-channel trace leak the same information. In these cases, DDPMs can combine information from several samples to remove noise from each of the individual samples. Notably, this is, to the best of our knowledge, the first approach that can effectively denoise traces in a fully non-profiled setting without a "clean" set of target traces. Furthermore, we showed significant improvements in attack performance for several state-of-the-art non-profiled attacks and similar improvements in the profiling complexity of deep learning models for profiled attacks. One of the main limitations of our work is that we focus on aligned traces. While directly training our models on misaligned traces does not pose any technical difficulties, achieving satisfactory performance in such a context is more difficult. In future work, we plan to investigate mechanisms for effectively applying our method to misaligned traces, as well as to investigate more complex network architectures. Besides this, there are a number of use cases within the SCA domain for our models, which could potentially be interesting. Some initial ideas include pre-training (parts of) classification models since diffusion models could aid in the training of profiling models and exploring whether the implicit compression of leakage from several features can be used to limit the computational overhead of subsequent attacks.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side-channel(s). In: Jr., B.S.K., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2523, pp. 29–45. Springer (2002). https://doi.org/10.1007/3-540-36400-5“¨4, https://doi.org/10.1007/3-540-36400-5_4
2. van den Berg, D., Slooff, T., Brohet, M., Papagiannopoulos, K., Regazzoni, F.: Data under siege: The quest for the optimal convolutional autoencoder in side-channel attacks. In: International Joint Conference on Neural Networks, IJCNN 2023, Gold Coast, Australia, June 18-23, 2023. pp. 1–9. IEEE (2023). https://doi.org/10.1109/IJCNN54540.2023.10191779, https://doi.org/10.1109/IJCNN54540.2023.10191779
3. Boussam, S., Albillos, N.C.: Keep it unsupervised: Horizontal attacks meet simple classifiers. In: Bhasin, S., Roche, T. (eds.) Smart Card Research and Advanced Applications - 22nd International Conference, CARDIS 2023, Amsterdam, The Netherlands, November 14-16, 2023, Revised Selected Papers. Lecture Notes in Computer Science, vol. 14530, pp. 213–234. Springer (2023). https://doi.org/10.1007/978-3-031-54409-5“¨11, https://doi.org/10.1007/978-3-031-54409-5_11
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings. Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer (2004). https://doi.org/10.1007/978-3-540-28632-5“¨2, https://doi.org/10.1007/978-3-540-28632-5_2
5. Bursztein, E., Invernizzi, L., Král, K., Moghimi, D., Picod, J.M., Zhang, M.: Generalized power attacks against crypto hardware using long-range deep learning. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 472–499 (Jul 2024). https://doi.org/10.46586/tches.v2024.i3.472-499, https://tches.iacr.org/index.php/TCHES/article/view/11685
6. Cao, P., Zhang, H., Gu, D., Lu, Y., Yuan, Y.: AL-PA: cross-device profiled side-channel attack using adversarial learning. In: Oshana, R. (ed.) DAC '22: 59th ACM/IEEE Design Automation Conference, San Francisco, California, USA, July 10 - 14, 2022. pp. 691–696. ACM (2022). https://doi.org/10.1145/3489517.3530517, https://doi.org/10.1145/3489517.3530517
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Jr., B.S.K., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer (2002). https://doi.org/10.1007/3-540-36400-5“¨3, https://doi.org/10.1007/3-540-36400-5_3
8. Cristiani, V., Lecomte, M., Hiscock, T., Maurine, P.: Fit the joint moments: How to attack any masking scheme. IEEE Access **10**, 127412–127427 (2022). https://doi.org/10.1109/ACCESS.2022.3222760
9. Do, N.T., Le, P.C., Hoang, V.P., Doan, V.S., Nguyen, H.G., Pham, C.K.: Mo-dlsca: Deep learning based non-profiled side channel analysis using multi-output neural networks. In: 2022 International Conference on Advanced Technologies for Communications (ATC). pp. 245–250 (2022). https://doi.org/10.1109/ATC55345.2022.9943024

10. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight authenticated encryption and hashing. J. Cryptol. **34**(3), 33 (2021). https://doi.org/10.1007/S00145-021-09398-9, https://doi.org/10.1007/s00145-021-09398-9
11. Genevey-Metat, C., Heuser, A., Gérard, B.: Trace-to-trace translation for SCA. In: Grosso, V., Pöppelmann, T. (eds.) Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13173, pp. 24–43. Springer (2021). https://doi.org/10.1007/978-3-030-97348-3"˙2, https://doi.org/10.1007/978-3-030-97348-3_2
12. Hajra, S., Chowdhury, S., Mukhopadhyay, D.: Estranet: An efficient shift-invariant transformer network for side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(1), 336–374 (2024). https://doi.org/10.46586/TCHES.V2024.I1.336-374, https://doi.org/10.46586/tches.v2024.i1.336-374
13. Hajra, S., Saha, S., Alam, M., Mukhopadhyay, D.: Transnet: Shift invariant transformer network for side channel analysis. In: Batina, L., Daemen, J. (eds.) Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18-20, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13503, pp. 371–396. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-17433-9"˙16, https://doi.org/10.1007/978-3-031-17433-9_16
14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020), https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html
15. Hu, F., Shen, J., Vijayakumar, P.: Side-channel attacks based on multi-loss regularized denoising autoencoder. IEEE Trans. Inf. Forensics Secur. **19**, 2051–2065 (2024). https://doi.org/10.1109/TIFS.2023.3343947, https://doi.org/10.1109/TIFS.2023.3343947
16. Karayalçın, S., Krček, M., Wu, L., Picek, S., Perin, G.: It's a kind of magic: A novel conditional gan framework for efficient profiling side-channel analysis. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024. pp. 99–131. Springer Nature Singapore, Singapore (2025)
17. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 148–179 (2019)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
19. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996). https://doi.org/10.1007/3-540-68697-5"˙9, https://doi.org/10.1007/3-540-68697-5_9

20. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1"˙25, https://doi.org/10.1007/3-540-48405-1_25
21. Krček, M., Perin, G.: Autoencoder-enabled model portability for reducing hyperparameter tuning efforts in side-channel analysis. Journal of Cryptographic Engineering **14**(3), 475–497 (2024)
22. Kulkarni, P., Verneuil, V., Picek, S., Batina, L.: Order vs. chaos: A language model approach for side-channel attacks. IACR Cryptol. ePrint Arch. p. 1615 (2023), https://eprint.iacr.org/2023/1615
23. Le, T., Clédière, J., Servière, C., Lacoume, J.: Noise reduction in side channel attack using fourth-order cumulant. IEEE Trans. Inf. Forensics Secur. **2**(4), 710–720 (2007). https://doi.org/10.1109/TIFS.2007.910252, https://doi.org/10.1109/TIFS.2007.910252
24. Lu, X., Zhang, C., Cao, P., Gu, D., Lu, H.: Pay attention to the raw traces: A deep learning architecture for end-to-end profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems (2021)
25. Lu, Z., Longde, Y., Fei, W., Aidong, C., Ning, Y., Xiang, L., Jiancheng, Z., Yanlong, Z., Shuo, W., Jing, Z.: Scarefusion: Side channel analysis data restoration with diffusion model. Microelectronics Journal **156**, 106546 (2025). https://doi.org/https://doi.org/10.1016/j.mejo.2024.106546, https://www.sciencedirect.com/science/article/pii/S1879239124002509
26. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. pp. 3–26. Springer (2016)
27. Maghrebi, H., Prouff, E.: On the use of independent component analysis to denoise side-channel measurements. In: Fan, J., Gierlichs, B. (eds.) Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10815, pp. 61–81. Springer (2018). https://doi.org/10.1007/978-3-319-89641-0"˙4, https://doi.org/10.1007/978-3-319-89641-0_4
28. Mangard, S.: Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In: Okamoto, T. (ed.) Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2964, pp. 222–235. Springer (2004). https://doi.org/10.1007/978-3-540-24660-2"˙18, https://doi.org/10.1007/978-3-540-24660-2_18
29. Masure, L., Cristiani, V., Lecomte, M., Standaert, F.: Don't learn what you already know scheme-aware modeling for profiling side-channel analysis against masking. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(1), 32–59 (2023). https://doi.org/10.46586/tches.v2023.i1.32-59, https://doi.org/10.46586/tches.v2023.i1.32-59
30. Masure, L., Dumas, C., Prouff, E.: A comprehensive study of deep learning for side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 348–375 (2020). https://doi.org/10.13154/tches.v2020.i1.348-375, https://doi.org/10.13154/tches.v2020.i1.348-375
31. Masure, L., Strullu, R.: Side-channel analysis against anssi's protected AES implementation on ARM: end-to-end attacks with multi-task learning. J. Cryp-

togr. Eng. **13**(2), 129–147 (2023). https://doi.org/10.1007/S13389-023-00311-7, https://doi.org/10.1007/s13389-023-00311-7

32. Moser, B.B., Shanbhag, A.S., Raue, F., Frolov, S., Palacio, S., Dengel, A.: Diffusion models, image super-resolution and everything: A survey. CoRR **abs/2401.00736** (2024). https://doi.org/10.48550/ARXIV.2401.00736, https://doi.org/10.48550/arXiv.2401.00736
33. Mukhtar, N., Batina, L., Picek, S., Kong, Y.: Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices. In: Galbraith, S.D. (ed.) Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13161, pp. 297–321. Springer (2022). https://doi.org/10.1007/978-3-030-95312-6"˙13, https://doi.org/10.1007/978-3-030-95312-6_13
34. Oswald, D.F., Paar, C.: Improving side-channel analysis with optimal linear transforms. In: Mangard, S. (ed.) Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7771, pp. 219–233. Springer (2012). https://doi.org/10.1007/978-3-642-37288-9"˙15, https://doi.org/10.1007/978-3-642-37288-9_15
35. Perin, G., Chmielewski, L., Batina, L., Picek, S.: Keep it unsupervised: Horizontal attacks meet deep learning. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(1), 343–372 (2021). https://doi.org/10.46586/TCHES.V2021.I1.343-372, https://doi.org/10.46586/tches.v2021.i1.343-372
36. Perin, G., Chmielewski, L., Picek, S.: Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(4), 337–364 (Aug 2020). https://doi.org/10.13154/tches.v2020.i4.337-364, https://tches.iacr.org/index.php/TCHES/article/view/8686
37. 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
38. Pozo, S.M.D., Standaert, F.: Blind source separation from single measurements using singular spectrum analysis. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9293, pp. 42–59. Springer (2015). https://doi.org/10.1007/978-3-662-48324-4"˙3, https://doi.org/10.1007/978-3-662-48324-4_3
39. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., III, W.M.W., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III. Lecture Notes in Computer Science, vol. 9351, pp. 234–241. Springer (2015). https://doi.org/10.1007/978-3-319-24574-4"˙28, https://doi.org/10.1007/978-3-319-24574-4_28
40. Savu, I., Krček, M., Perin, G., Wu, L., Picek, S.: The need for more: Unsupervised side-channel analysis with single network training and multi-output regression. In:

Wacquez, R., Homma, N. (eds.) Constructive Side-Channel Analysis and Secure Design. pp. 113–132. Springer Nature Switzerland, Cham (2024)
41. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3659, pp. 30–46. Springer (2005). https://doi.org/10.1007/11545262“˙3, https://doi.org/10.1007/11545262_3
42. Timon, B.: Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(2), 107–131 (2019). https://doi.org/10.13154/tches.v2019.i2.107-131, https://doi.org/10.13154/tches.v2019.i2.107-131
43. Wang, P., Chen, P., Luo, Z., Dong, G., Zheng, M., Yu, N., Hu, H.: Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks. CoRR **abs/2007.05285** (2020), https://arxiv.org/abs/2007.05285
44. Wu, L., Perin, G., Picek, S.: The best of two worlds: Deep learning-assisted template attack. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(3), 413–437 (2022). https://doi.org/10.46586/TCHES.V2022.I3.413-437, https://doi.org/10.46586/tches.v2022.i3.413-437
45. Wu, L., Perin, G., Picek, S.: I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. IEEE Transactions on Emerging Topics in Computing pp. 1–12 (2022). https://doi.org/10.1109/TETC.2022.3218372
46. Wu, L., Perin, G., Picek, S.: Not so difficult in the end: Breaking the lookup table-based affine masking scheme. In: Carlet, C., Mandal, K., Rijmen, V. (eds.) Selected Areas in Cryptography – SAC 2023. pp. 82–96. Springer Nature Switzerland, Cham (2024)
47. Wu, L., Perin, G., Picek, S.: Weakly profiling side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 707–730 (Nov 2024). https://doi.org/10.46586/tches.v2024.i3.707-730, https://tches.iacr.org/index.php/TCHES/article/view/11901
48. Wu, L., Picek, S.: Remove some noise: On pre-processing of side-channel measurements with autoencoders. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(4), 389–415 (2020). https://doi.org/10.13154/tches.v2020.i4.389-415, https://doi.org/10.13154/tches.v2020.i4.389-415
49. Yang, G., Li, H., Ming, J., Zhou, Y.: CDAE: towards empowering denoising in side-channel analysis. In: Zhou, J., Luo, X., Shen, Q., Xu, Z. (eds.) Information and Communications Security - 21st International Conference, ICICS 2019, Beijing, China, December 15-17, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11999, pp. 269–286. Springer (2019). https://doi.org/10.1007/978-3-030-41579-2“˙16, https://doi.org/10.1007/978-3-030-41579-2_16
50. Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., Yang, M.: Diffusion models: A comprehensive survey of methods and applications. ACM Comput. Surv. **56**(4), 105:1–105:39 (2024). https://doi.org/10.1145/3626235, https://doi.org/10.1145/3626235
51. Yap, T., Jap, D.: Creating from noise: Trace generations using diffusion model for side-channel attack. In: Andreoni, M. (ed.) Applied Cryptography and Network Security Workshops. pp. 102–120. Springer Nature Switzerland, Cham (2024)
52. Zaid, G., Bossuet, L., Carbone, M., Habrard, A., Venelli, A.: Conditional variational autoencoder based on stochastic attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(2),

310–357 (2023). https://doi.org/10.46586/TCHES.V2023.I2.310-357, https://doi.org/10.46586/tches.v2023.i2.310-357
53. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(1), 1–36 (Nov 2019). https://doi.org/10.13154/tches.v2020.i1.1-36, https://tches.iacr.org/index.php/TCHES/article/view/8391

A Correlation Attacks

First, we investigate improvements to second-order CPA-based attacks. As these attacks utilize one sample for each secret share, the expectation is that we will see significant improvements as the diffusion models allow for the implicit utilization of information leaked across several correlated features. We showcase scenarios for ASCADf and ESHARD to show the effects of a diffusion model on the attack performance when SNR is improved (ESHARD) and when it is not (ASCADf). Note that as peak SNR is not improved for ASCADf, diffusing traces should have no impact on the performance of CPA-based attacks [28]. We select the highest SNR samples for each share in the original traces for these attacks and combine them using absolute difference as a shortcut to avoid testing all possible feature combinations.

The results in Figure 10 indicate that clear improvements in attack performance are achieved. Note that for these attacks, the CPA results are not entirely representative of real-world attacks. In fact, we require more traces than are available in the attacks to train the diffusion models. Thus, to simulate representative attacks, we should train diffusion models for every subset of traces we attack in each of the attack simulations, which is impractical, especially when training diffusion models using low trace counts. However, for noisier targets where (very) large numbers of traces are necessary for key retrieval, this limitation is not an objection, as the diffusion model can be trained using the larger set. As such, the results in Figure 10 indicate that trained diffusion models provide significant benefits for improving CPA attacks (or other attacks that represent the leakage of a secret share using a single sample point).

B Profiling Attacks

In this section, we explore the impact of using diffusion models to denoise traces in a profiled setting. We report the distribution of the attack performance of random models to assess the impact of using diffused traces on the difficulty of finding good model configurations.

B.1 Experimental Setup

To investigate the impact of using denoised traces for profiling attacks, we will examine the profiling complexity of attacks against several datasets. To do this,

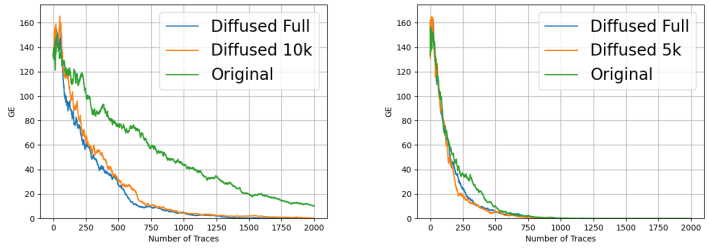


Fig. 10: CPA results for ESHARD (left) and ASCAD(right). Diffused 5k/10k refer to denoising with DDPMs trained with 5 000 or 10 000 traces respectively

we randomly search small MLP models using the ranges in Table 4. This search is run using a varying number of profiling traces for original and diffused traces. We use two diffusion models, one trained with the maximum considered number of profiling traces and one with the minimum considered number of traces (5 000 and 25 000, and 10 000 and 70 000 for ASCADf and ESHARD, respectively).

Table 4: Hyperparameter search ranges for MLP architecture as a profiling attack model.

Hyperparameter	Options
Dense layers	1, 2, 3, 4
Neurons	10, 20, 50, 100, 200, 300, 400, 500
Activation Function	selu, relu,
Learning Rate	0.005, 0.001, 0.0005, 0.0001
Optimizer	Adam, RMSprop
Batch Size	100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
Weight Initialization	random uniform, he uniform, glorot uniform

B.2 Results

Figure 11 shows the distribution of attacking results for the 100 random MLPs against ESHARD. The attack performance is significantly improved by utilizing diffused traces. For all the tested settings, we see that more of the models trained on diffused traces result in successful attacks. In fact, the distribution of attack performances at 30 000 diffused profiling traces is already better than the distribution using 70 000 original profiling traces. Additionally, in settings with lower numbers of profiling traces, only attacks using diffused traces can successfully recover the key in 2 000 traces.

In Figure 11, the results for ASCADf are less impressive. In fact, in this case, there does not seem to be any difference between using diffused and original traces. These results indicate that for datasets where the diffusion models

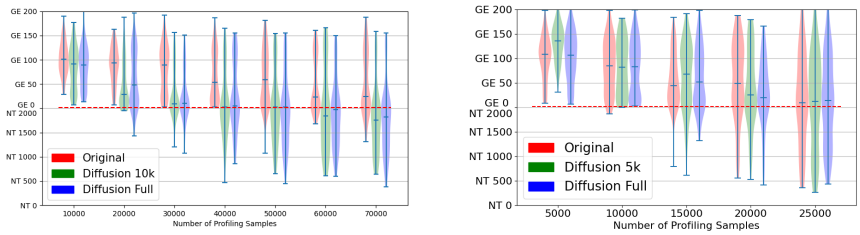


Fig. 11: Distribution of Guessing Entropy(GE)/number of traces to reach $GE = 0$ for 100 random MLPs for ESHARD(top) and ASCAD(bottom). Diffused 5k/10k refer to denoising with DDPMs trained with 5 000 or 10 000 traces respectively

successfully improve SNR, it becomes significantly easier to define and train profiling models that can retrieve the key, while for cases where the models do not improve SNR, the difficulty remains the same.

C Hyperparameter Evaluation

As we only utilize one architecture and hyperparameter configuration to achieve the results, we provide insights into the effects of varying this architecture/hyperparameter setup. All of these results are evaluated on the maximum SNR peaks of the first secret share of ESHARD (the mask) and the third share of ASCADv2 (the masked Sbox output). We chose these two secret shares as our standard architecture works well on these targets, and they are rather different in terms of the number of features that leak the secret shares (which can be seen in Section 4.3). Note that, we make the default values bold to improve readability.

C.1 Noise Schedules

The noise schedule is a key hyperparameter, and in Table 5, we see that switching to different schedules harms performance. However, for ESHARD, each schedule still significantly improves over the original SNR. For ASCADv2, none of the other schedules performed better than the original SNR, but the SNR is also not lowered. We note that other schedules may work better for different numbers of steps or other hyperparameter changes. Our results here do not mean the quadratic schedule is always best, and more specific work optimizing noise schedules for different side-channel setups could be an interesting direction for future work.

C.2 Architecture

The effects of varying the number of downsampling (and corresponding upsampling) layers are quite different for different targets. When only one downsam-

	quad	linear	cos	sig
ASCADv2	0.18	0.08	0.09	0.08
ESHARD	1.13	0.92	0.97	0.92

Table 5: SNR peaks for varying noise schedules.

pling layer is used, Table 6 shows that the difference from the original SNR is limited for both targets. When the number is increased, we see that for ESHARD, the SNR increases over our standard configuration, while for ASCADv2, the SNR decreases. In Table 7, varying the activation function results in somewhat decreased performance for ASCADv2, while for ESHARD, there does not seem to be much of an effect. Overall, defining an appropriate architecture is mainly the question of defining an appropriate depth. Using two blocks seems like a reasonable middle-ground for the network, not ignoring certain leakages that only contribute to a small number of features while still providing enough expressive power in the network to remove noise effectively.

	1	**2**	3	4
ASCADv2	0.08	0.18	0.08	0.02
ESHARD	0.63	1.13	1.61	1.48

Table 6: SNR peaks for varying numbers of downsampling blocks in the network.

	tanh	relu	selu	linear
ASCADv2	0.18	0.15	0.09	0.08
ESHARD	1.13	1.43	1.21	1.24

Table 7: SNR peaks for varying activation functions.

C.3 Training Time Hyperparameters

As can be seen in Table 8, the initial learning rate is quite an important factor. Only the standard 0.001 can effectively denoise ASCADv2. For ESHARD, it matters significantly less, and while the performance is best for our standard case, varying it still results in significant improvements over the original traces. Varying the number of steps T in Table 9 shows significant room for improvement over our baseline model. Especially for ASCADv2, we achieve another 50% improvement in peak SNR by optimizing T. Overall, this indicates that the specifying batch size and T are not too sensitive, and values in a broad range are effective. On the contrary, defining learning rates that are inappropriate can quickly result in models that do not learn anything for some targets.

	0.01	**0.001**	0.0001	0.0005	1e-05	5e-05
ASCADv2	0.02	0.18	0.05	0.08	0.02	0.04
ESHARD	0.02	1.13	0.98	0.99	0.98	0.97

Table 8: SNR peaks for varying initial learning rates.

	4	8	**16**	32	128	512	1024
ASCADv2	0.10	0.18	0.18	0.20	0.27	0.09	0.02
ESHARD	1.27	1.28	1.13	1.15	1.16	1.53	1.56

Table 9: SNR peaks for varying number of steps T.

	10	25	50	100	**200**	400
ASCADv2	0.03	0.08	0.11	0.15	0.18	0.20
ESHARD	1.09	1.08	1.18	1.20	1.13	1.13

Table 10: SNR peaks for varying numbers of epochs.

	10 000	20 000	30 000	**40 000**	50 000	60 000
ASCADv2	0.16	0.15	0.17	0.18	0.20	0.24
ESHARD	0.89	1.02	1.04	1.20	1.13	1.26

Table 11: SNR peaks for varying numbers of traces.

C.4 Epochs and Number of Traces

Table 11 shows that the models are surprisingly effective when given only a relatively small number of training traces. In fact, while using more traces obviously does not hurt, the benefits are only marginal, and for the fairly difficult case of ASCADv2's third share, we can already see a doubling of the SNR using only 10 000 traces. The number of epochs is somewhat more sensitive. In Table 10, for ESHARD, the model performance is already good with only 10 epochs, while for ASCADv2, good performance starts at 100 epochs.