Trace-to-trace translation for SCA

Christophe Genevey-Metat, Annelie Heuser, Benoît Gérard

Trace-to-trace translation for SCA

Christophe Genevey-Metat[1], Annelie Heuser[1], and Benoît Gérard[1,2]

[1] Univ Rennes, Inria, CNRS, IRISA, France
[2] Direction Générale de l'Armement

Abstract. Neural Networks (NN) have been built to solve universal function approximation problems. Some architectures as Convolutional Neural Networks (CNN) are dedicated to classification in the context of image distortion. They have naturally been considered in the community to perform side-channel attacks showing reasonably good results on trace sets exposing time misalignment. However, even in settings where these timing distortions are not present, NN have produced better results than legacy attacks.
Recently in TCHES 2020, auto-encoders have been used as preprocessing for noise reduction. The main idea is to train an auto-encoder using as inputs noisy traces and less noisy traces so that the auto-encoder is able to remove part of the noise in the attack dataset.
We propose to extend this idea of using NN for pre-processing by not only considering the noise-reduction but to translate data between two side-channel domains. In a nutshell, clean (or less noisy) traces may not be available to an attacker, but similar traces that are *easier* to attack may be obtainable. Availability of such traces can be leveraged to learn how to translate *difficult* traces to *easy* ones to increase attackability.

Keywords: Side-channel analysis · Generative Adversarial Network · profiled attacks · neural networks · electromagnetic emanations · power consumption

1 Introduction

Physical side-channel analysis has been introduced at the end of the 90's in [9]. Since then, the interest of side-channel attacks has increased and new attacks and channels have been considered. Those attacks have been classified into two classes depending on the ability of the attacker to perform some prior training before attacking. Profiled attacks leverage the availability of training data to outperform non-profiled attacks by tuning the leakage model to best fit reality. Indeed profiled attacks are the most efficient ones since the model used is at worse as good as a generic model used in the non-profiled case. The reference for profiled attacks are Template Attacks (TA) [2] which are optimal as soon as the leakage fits the model used (usually a Gaussian model).

A few years ago, Deep Learning (DL) techniques have been introduced as new profiled attacks [11]. They have been shown to gain good performance even in the presence of countermeasures. As an example, the ASCAD database [14] contains

traces corresponding to a masked implementation with artificial desynchronization. The database has been used in numerous works leading to high-performance attacks compared to classical techniques.

Related work Most of the works using DL techniques are based on convolutional neural networks [14, 12, 10, 8, 20, 12]. Lately, some works go beyond this approach and have shown the advantage of using Generative Adversarial Networks (GAN)s for data augmentation [17] or Convolutional Auto-Encoders (CAE) for noise reduction [19]. In particular, in [17] authors investigated the use of Conditional Generative Adversarial Networks (CGAN) for data augmentation, which requires a labeled dataset. They show that increasing the amount of traces in the training dataset with traces generated by a CGAN can improve the performance of profiled attacks. As a result, they show that they can reach similar performances on the attack dataset using half of the original measured training traces.

Closer to our work, authors of [19] have investigated the capability of CAE to denoise/remove the effect of various hiding countermeasures. For this, they artificially added noise to the ASCAD datasets: Gaussian noise, shuffling, clock-jitter and random-delay. Their results show that auto-encoders are effective tools to improve the quality of traces and thus to increase the attack performance by targeting processed traces. This work is of interest since for the first time, NN are not considered for performing a full attack but only as a preprocessing step before applying a distinguisher. Actually, the approach does not need to be in a classical profiling context as we will detail in this paper and thus can be used as a prior process to any other classical side-channel attack.

To be able to learn how to *remove the noise*, the auto-encoder needs a training dataset containing the *noisy* traces and the corresponding *clean* traces. The main drawback of this approach is that the attacker must have access to those clean traces (traces without any countermeasure present, for instance). In many situations (e.g, when clock-jitter is present) the attacker might not be able to fulfill this requirement. We propose here to relax this constraint by considering an *easier* dataset instead of a *clean* one. That is, not using a noiseless version of the training dataset but some dataset obtained from a setup which is easier to attack. For this, we leverage on techniques designed to translate between two domains.

This Work We aim at investigating how far we could extend the research topic from [19] by going beyond noise reduction and toward trace translation. The main goal is shared with the aforementioned paper since the finality is to improve attack traces for reaching better attack performances. To this end, we investigate the use of GAN, as they are known in the field of computer vision to be effective to generate synthetic data. They are for instance used to generate art, improve astronomical images, or to perform image-to-image translation.

In this paper, we adjust and tune two GAN architectures over a set of hyper-parameters: (1) Speech Enhancement GAN (SEGAN) which has been designed

for speech enhancement [13], (2) Pix2Pix [5] designed in the context of image translation or generation, and select the best performing one for our experiments.

Our main idea is to translate traces measured on "difficult" (e.g., complex, noisy) sources to "easier" ones, without the restriction of having clean and noisy traces from the same device, source, and implementation. We considered our traces as paired datasets, even if they come from different devices, because these traces contain the same intermediate variable. To make a proof of concept of our approach, we perform the following translation experiments:

1. from (less informative) electromagnetic (EM) to power consumption traces,
2. from one target device to another (STM32F1 to STM32F2, STM32F0 to STM32F2, STM32F2 to STM32F4).

The first case is motivated by the practical limitations of an attacker that may not be able to introduce a resistor on the target PCB to attack while being able to measure the power on a training device. The second one has been chosen to see if the translation goes beyond what could be seen as a (simple) scaling/shifting and is actually able to transform the leakage model (e.g. leakage shape and location).

2 Preliminaries

2.1 Side-channel Analysis

In the side-channel adversary model one assumes that the attacker is able to measure side-channel information while knowing the plaintext or ciphertext plus some information about the internal structure of the implemented algorithm. Using this information the attacker is able to make predictions about parts of the secret key. Formally, a function of the cryptographic algorithm is processed taking as inputs (a part of) the secret key k^* and (a part of) the plaintext (or ciphertext) t. The attacker defines an internal state of the cryptographic algorithm y as a function of t and k^*, which is assumed to relate to the deterministic part of the measured leakage x. For example, in AES a common choice is the substitution output, ie. $y(t, k^*) = \text{SBox}[t \oplus k^*]$, where SBox$[\cdot]$ is the substitution look-up table. We denote $y(\cdot, \cdot)$ as the label.

(Non-)Profiled attacks When performing a side-channel attack, an attacker has access to an attack dataset. That is, a set of traces which is obtained with some fixed secret that the attack aims at recovering. To extract information from these measurements, the attacker has to make some hypotheses about the least constrained leakage model. On top of that, profiled attacks have access to a training dataset that is a set of traces with known inputs/secrets obtained in similar conditions. The attacker may, for instance, acquire a clone device in order to extract such a training dataset.

Correlation Power Analysis (CPA) The most common non-profiled side-channel analysis technique is based on the Pearson correlation coefficient [1].

The attacker iterates over all possible values of the key k from the key-space and, knowing the ciphertext or plaintext t, computes $y(t, k)$ for each possible key prediction k. Given a certain amount of side-channel measurement traces x with corresponding t, he then calculates the Pearson correlation between x and $y(t, k)$. The most likely key prediction is the one that maximizes the correlation.

Neural Network Attacks In our experiments, we will use three different networks for exploiting an *attack* dataset (that is, recovering the fixed key used to generate the traces). Those networks have been designed to target traces containing 700 time samples, and thus we reduced our traces to 700 points by selecting relevant windows based on Signal-to-Noise Ratio (SNR). The training is run for 100 epochs with batches of size 128. For each architecture, we selected the model reaching the best validation loss.

ASCAD Neural Network. Authors of [14] have introduced the first deeply studied neural network, that they refer to as *cnn_best*. In our experiments, we will refer to this network as ASCAD network (using its corresponding published parameters).

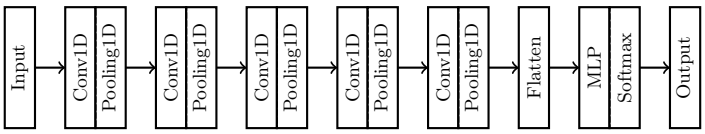


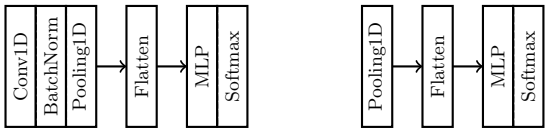
Fig. 1. ASCAD network

The ASCAD network, which is close to the architecture of VGG-16 [16] is depicted in Figure 1. It is composed of five blocks with one convolutional layer by block, a number of filters equal to (64, 128, 256, 512, 512) with kernel size 11 (same padding), ReLU activation function, and an average pooling layer for each block. The CNN has two final dense layers of 4 096 units.

Variations of the ASCAD network. Recently, lighter networks have been proposed in the side-channel context. To show that trace translation is not particular to one network, we additionally use:

1. the network proposed by Zaid et al. in [20] labeled as Zaid network,
2. the even lighter network presented by Wouters et al. in [18], labeled as No-Conv1 network.

The Zaid network (depicted in Figure 2(a)) is composed of one block with one convolutional layer, a number of filters equal to 4, with a kernel size 1 (same padding), SeLU activation functions, an batch normalization layer, and an average pooling layer. Finally, The Zaid network has two dense layers of 10 units.



(a) Zaid (b) NoConv1

Fig. 2. Additional networks

The NoConv1 network (depicted in Figure 2(b)) is composed of one block with one average pooling layer. The NoConv1 network has two dense layers of 10 units. Figure 2(b) gives an illustration of the NoConv1 network.

2.2 Datasets

We use the chipwhisperer light capture board combined with the CW308 UFO board to measure side-channel information from different STM32 devices. More precisely, we used the STM32F0 (Cortex-M0), STM32F1, STM32F2 (both Cortex-M3) and STM32F4 (Cortex-M4) target devices from the CW308 board[3].

On all devices, the beginning of an AES-128 encryption was measured, where we used the `TINYAES` implementation integrated in the chipwhisperer software. The chip frequency was set to 7.37MHz and the measurements are sampled at 4×7.37 Ms/s. For the chipwhisperer light setup, power consumption is collected through the measurement shunt on the CW308 UFO board. To capture EM signals we used a Langer near-field EM probe (RF-U 5-2) connected to a 20dB amplifier[4]. On each device and source, we measured 100k traces for training and validating (used in a ratio of 80:20 for learning the translation and training the neural network attack), and 25k traces as an attack dataset with 25 different keys.

2.3 Evaluation Metrics & Targeted Value

Experiments have been performed to first-order leakage from the output of the AES substitution box (SBox): $y = \text{SBox}(t \oplus k)$, To evaluate the amount of leakage, we use the SNR. Let X denote the captured side-channel measurement, let Y be the label that is determined by the plaintext and the secret fixed key, then SNR gives the ratio between the deterministic data-dependent leakage and the remaining noise, i.e. $SNR = \frac{\mathbb{V}(\mathbb{E}(X|Y)}{\mathbb{E}(\mathbb{V}(X|Y)}$, where $\mathbb{E}(\cdot)$ is the expectation and $\mathbb{V}(\cdot)$ the variance of a random variable.

To evaluate the ability to retrieve the key, we use the Guessing Entropy (GE), which is the expected ranking of the secret key k^* within a vector of key guesses. In particular, the vector of key guesses $g_{i,1}, \ldots, g_{i,|K|}$ for the i-th measurement

[3] https://rtfm.newae.com/Targets/UFO%20Targets/CW308T-STM32F.html

[4] We do not claim that the setup, nor the position of the EM probe on the device is chosen optimally.

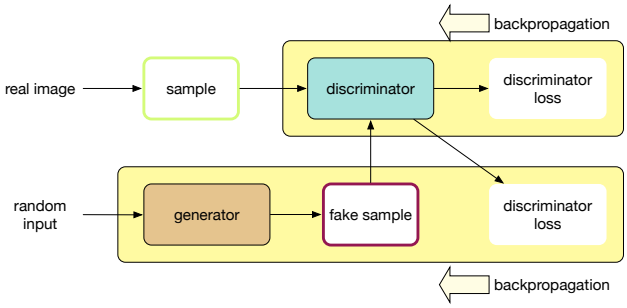


Fig. 3. General overview of GAN

is calculated by mapping each key guess k to a label j with probability $\hat{p}_{i,j}$ and applying the maximum-likelihood principle over 1 to m measurements (where m is the number of measurements considered for the attack). The guessing entropy is then the expected position of the secret key k^* in the sorted vector of key guesses, where the sorting is applied to the probabilities in descending order. In other words, the guessing entropy gives the average amount of key guesses an attacker needs to perform before he reveals the secret key. In case his first guess is the secret key $GE = 0$.

2.4 GAN

GAN [4] falls into the class of generative modelling, meaning that it can be used to generate or output new synthetic data that could have plausibly been drawn from the original dataset. GAN is composed of two deep neural networks: the generator, and the discriminator. Generally speaking, the generator is used to generate new valid examples from the problem domain, whereas the discriminator is used to classify examples as real (from the domain) or fake (generated).

As illustrated in Figure 3, the generator takes a fixed-length random vector as input and generates a sample in the domain, i.e., the space of real data. The random vector is typically drawn from a Gaussian distribution and it seeds the generative process. The discriminator model takes either an input from the real data or the generated one from the generator and predicts a binary class label of real or fake. The discriminator uses the discriminator loss for back-propagation, whereas the generator takes the generator loss produced by the discriminator.

Since its introduction in 2014, the interest of GAN has grown and many other/specific GAN architectures have been proposed[5]. One particular class of GAN concentrates on the problem of image translation instead of pure image generation. One of the best-known architectures is Pix2Pix [6], it requires paired datasets.

[5] Collection of GANs from several domains: https://github.com/hindupuravinash/the-gan-zoo

Pix2Pix is trained to transform images from one domain into images that could plausibly belong to another domain. For example, a famous illustration in image translation transforms an drawing into a real image.

The interest of translation has been raised by different communities. For speech enhancement [13], authors implemented SEGAN to denoise audio waveforms. Contrary to the generator in standard GANs, the generator of SEGAN takes as input a random vector plus a noisy signal data to produce the enhanced signal.

3 Trace-to-Trace Translation

3.1 Approach

The idea proposed in this work is to use domain translation as a pre-processing technique to improve the quality of traces. This approach is enabled as soon as an attacker has access to paired datasets (that is, sharing the same intermediate target values) from two different settings. Indeed, one of the settings must correspond to the attack dataset and the other one corresponds to some settings for which the chosen attack performs better.

An attacker following the proposed approach will thus handle four different datasets. To smooth the description of the technique let us introduce some notation. The goal is to improve the attack on domain A traces by translating traces from the domain A to domain B. Datasets:

- the **original attack dataset** contains the traces from which the attacker wants to extract a secret key (thus from domain A);
- the **translation training dataset A** contains traces from domain A;
- the **translation training dataset B** contains traces from domain B that should be paired with the translation training dataset A;
- the **translated attack dataset** containing the translation from domain A to domain B of the *original attack dataset*.

Should this technique be used in a profiled setting, an additional fifth dataset would be added, namely

- the **training dataset** that contains training (thus labeled) traces from domain B used to train the profiled attack.

We want to bring the reader's attention to the fact that the only labeled dataset is the optional fifth one since translation training datasets only need to be paired (depending on the context, it may not necessarily imply being labeled).

Short Description With these datasets in mind, the technique can be simply described in a very compacted form.

1. Train a *translator* to translate traces from domain A to domain B using *translation training datasets A and B*.
2. Use the trained *translator* to generate the *translated* attack *dataset* from the *original attack dataset on domain A*.
3. (optional) Train a profiled side-channel attack on the *training dataset* from domain B.

4. Attack the *translated attack dataset* to recover the key using an independent (un-)profiled attack.

Instantiating the Translator As a first investigation of trace-to-trace translation, we propose to use GAN networks as a translator since they have shown to be efficient in several domains such as image translation [5], style transfer [7] or audio processing [13]. We do not claim that this architecture is an optimal choice for translation but its wide use makes it a relevant one for a first investigation. More details on the used GAN architecture are given in Section 3.3.

3.2 Experimental Methodology

For all our experiments on translation, we assess the improvements brought by the proposed preprocessing technique by applying both a profiled and a non-profiled attack. Let us recall that the attacker has an attack dataset from domain A and the improvement proposed in this paper is to translate them to domain B prior to the attack.

For the non-profiled scenario, the improvement consists in using the GAN to translate traces from a domain A to a domain B then apply CPA on translated (synthetic) traces. We compare this to the direct application of CPA to traces of domain A and thus generate two different graphs illustrating the attack success as a function of the number of traces used to attack.

Regarding the profiled scenario, the improvement consists in using the GAN to translate traces from a domain A to a domain B then apply a DL-based attack on translated (synthetic) traces. Obviously the DL-based attack should use a model trained on a training dataset from domain B.

For further comparison, we performed a complementary attack that consists in directly applying the DL-based attack using a model trained on domain B to the original attack dataset (from domain A). This additional experiment is only there as a witness experience and to demonstrate that the translation is indeed successful as both approaches use the same trained model.

3.3 Used GAN Architecture

We compared two architectures of GAN: Pix2Pix and SEGAN. Pix2Pix is well known for image translation, whereas SEGAN was designed to denoise audio-waveforms. We tuned both architectures over a set of hyperparameters, and selected the best performing model (see Table 1).

During the tuning phase, we used a paired dataset composed of 2×100k traces (100k for each domain). These traces were split: 80k for the training and 20k for the validation. Since using the GE as a validation metric was too expensiveon time, we saved the best model over training epochs based on the SNR

[6] For SEGAN we tried four potential batch sizes, but as Pix2Pix is deeper than SEGAN, and we were limited by GPU memory in our setup, we could only test batch size of 64.

		SEGAN	Pix2Pix
Hyperparameter	**Range**	**Selected**	**Selected**
Optimizer	{ Adam, RMSProp, SGD }	RMSProp	RMSProp
Activation function	{ Tanh, LeakLy ReLU, PReLU }	Tanh	Tanh
Batch size	{ 64, 128, 256 }	128	64^6
Epochs	{ 25, 50, 100, 200 }	200	200

Table 1. Hyperparameter tuning

obtained on translated traces. SNR can be directly computed on the validation dataset while using GE would require to translate a large attack dataset (with a few fixed keys). We kept the model providing the highest SNR peaks for each set of hyperparameters. A first analysis of the obtained SNR curves has shown that the best performances of Pix2Pix and SEGAN were obtained with Tanh as activation function and RMSprop as optimizer. To better compare the obtained models, we additionally used 25k attack traces (translated traces with few fixed keys). Using them we computed the GE for all final models and chose the hyperparameters leading to the smallest one as summarized in Table 1.

The SEGAN architecture we selected for our experiments (after fine-tuning it) is close to the architecture of the original SEGAN [13] that is used for audio waveform translation.

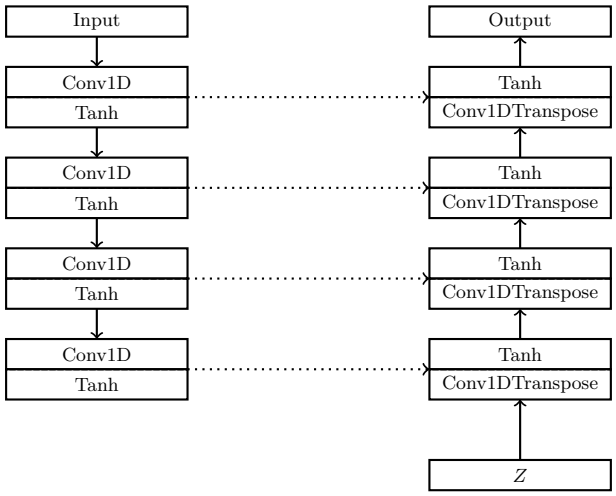


Fig. 4. Generator architecture (SEGAN)

The generator takes as input a 700-point trace coming from the original domain and outputs a 700-point synthetic trace, which is the translation to

the target domain. The generator is an auto-encoder that is composed of an encoding part and a decoding part. The output value of the encoding part (or equivalently the input of the decoding part) lies in the so-called latent space. The latent representation of the trace has a shape equal to (2, 256) and this is where the random noise Z is added. The generator is illustrated in Figure 4. The encoder is composed of four blocks with one convolutional layer per block, a number of filters equal to (32, 64, 128, 256) with kernel size 31 (same padding), followed by a Tanh activation function. The decoding part is composed of four blocks and one transposed convolutional layer per block, a number of filters equal to (128, 64, 32, 1) with kernel size 31 (same padding), followed by a Tanh activation function. Each block of the decoding part is concatenated with the output of each encoding block, represented by a dotted arrow in the fig 4, it is the principle of the U-Net architecture [15].

The discriminator takes as input a combination of two 700-point traces: one trace coming from the original domain and one trace coming from the target one. This last trace may directly come from the target domain (real trace) or it could be a generated trace (fake trace). The discriminator is trained to distinguish between real and fake. More precisely, the discriminator outputs the probability that a trace from the target domain is a translation from the original domain. The discriminator is illustrated in Figure 5. It is composed of four blocks, and one convolutional layer per block, a number of filters equal to (32, 64, 128, 256) with a kernel size 31 (same padding), a Batch normalization layer, and a Tanh activation function. The discriminator has two final dense layers of 256 and 128 units.

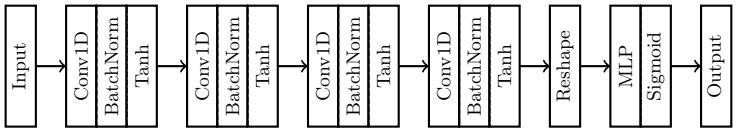
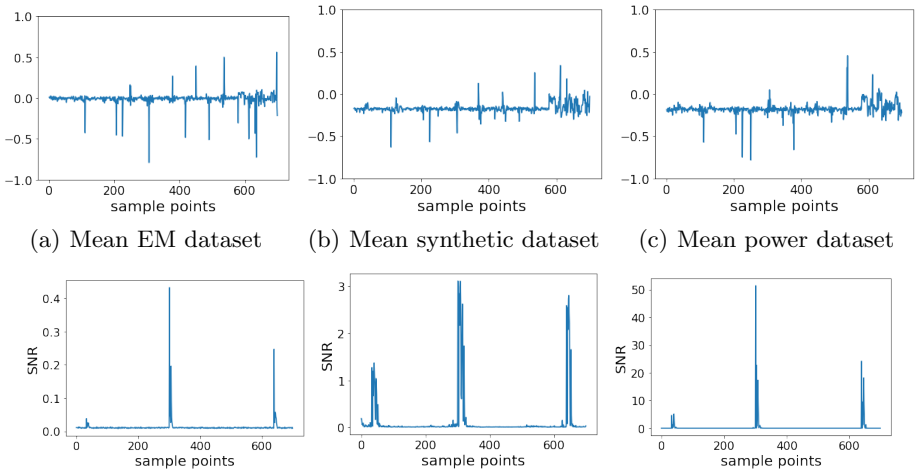


Fig. 5. Discriminator architecture (SEGAN)

As for SEGAN, we slightly adapted Pix2Pix to our datasets. The generator of the Pix2Pix takes as input a 700-point trace coming from the original domain and outputs a 700-point synthetic trace as SEGAN model. The generator is also an auto-encoder that is composed of an encoder part and decoder part. The Pix2Pix network didn't take latent representation between the encoder part and decoder part, but just one convolutional layer composed of 512 filters. The encoder part is composed of seven blocks with one convolutional layer per block, a number of filters equal to $(32, 64, 128, 256, 256, 256, 256)$ with a kernel size of 11 (same padding), followed by a Batch Normalization layer and an activation function (see Table 1). The decoder part is composed of seven blocks and one transposed convolutional layer per block, a number of filters equal to (256, 256, 256, 256, 128, 64, 32) with a kernel size 11 (same padding), followed by a Batch Normalization



(a) Mean EM dataset (b) Mean synthetic dataset (c) Mean power dataset

(d) SNR of EM traces (e) SNR of synthetic traces (f) SNR of power traces

Fig. 6. Evaluation of STM32F2: (a) - (c) illustrates the mean trace and (d)-(f) the SNR of the attack dataset for the EM, power channel, and the translated synthetic trace dataset

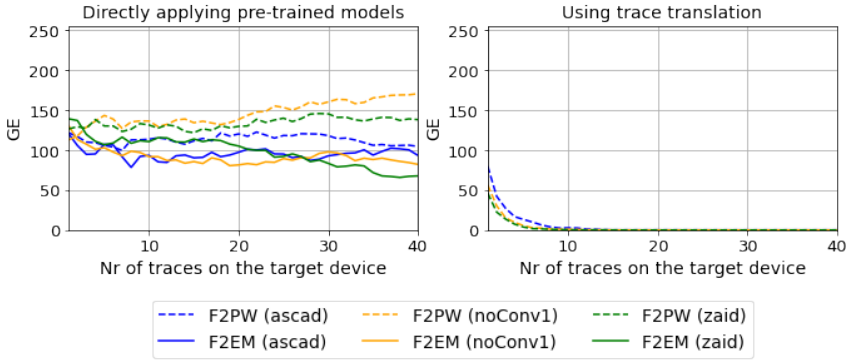
layer and an activation function. The discriminator of the Pix2Pix is composed of five blocks with one convolutional layer per block, a number of filters equal to $(32, 64, 128, 256, 1)$ with a kernel size of 11 (same padding). In the original paper [6] the discriminator part of the Pix2Pix is implemented as PatchGAN, which means that the discriminator will classify 70×70 patches of the input image as real or fake. In our case, we adapt the output of the discriminator to output one single value. During the tuning phase, Pix2Pix always provided lower performance (e.g., lower or less SNR peaks). Hence, SEGAN has been selected for our experiments on translation in the next sections.

4 Translation from EM to Power

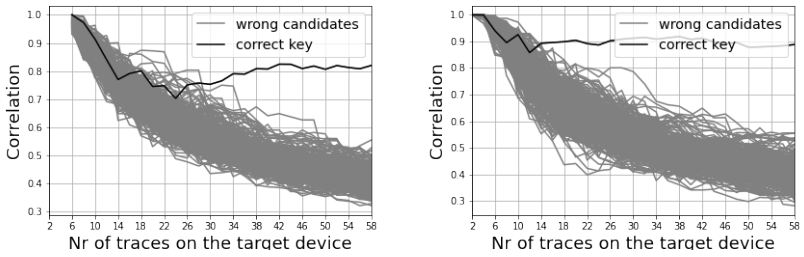
In this scenario, we investigate the capability of GAN to translate traces from EM to power consumption on the same device. Depending on the device, we obtained various different SNR levels. For each device, we observed that the SNR corresponding to its EM and power traces have similar shapes while being of different magnitudes. From a quantitative point of view, the SNR value obtained from power traces is higher than from EM traces: in our setup EM measurements contain more noise. For the experiments presented in this section, **Domain A** will thus correspond to EM traces (that are harder to attack) and **Domain B** to power ones.

STM32F2 Figure 6 shows the mean trace of the attack dataset of the EM data, the synthetic dataset, and the power consumption, as well as their SNR levels.

First, we observe a translation as indeed the mean translated synthetic trace is looking close to the mean trace of the power dataset, particularly for the second half. Next, the SNR value is low (close to 0.4) for the EM channel, whereas the SNR value is high (close to 50) for the power channel, but their shapes are similar. The synthetic dataset corresponds to the set generated by GAN, which was trained to translate traces from EM to the power channel. We observe that the shape of SNR is different compared to the shape of EM and power and that the leakage positions changed in time, while still being able to retrieve one (over three) leakage positions from the target domain. The SNR value of the synthetic dataset is close to 6, so the translation increases the magnitude and thus the amount of information. Still, the SNR value is lower than for the power channel, which means that some information could not be reproduced (as it may not be available in the EM trace).



(a) GE for three networks; left: original EM dataset, right: translated synthetic dataset



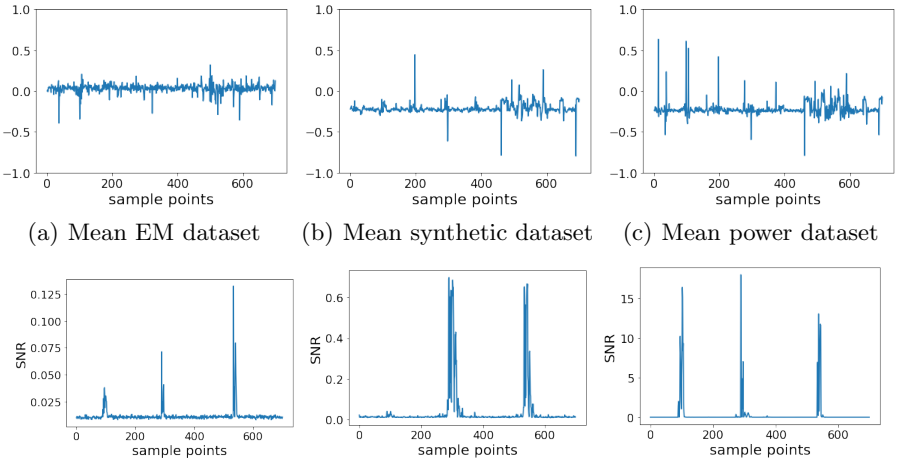
(b) CPA on EM dataset (c) CPA on synthetic dataset

Fig. 7. Attack evaluation on STM32F2 EM (original and synthetic translated traces)

In Figure 7 (top) we plot the GE in the profiled scenario when targeting the EM channel. First, we see that the performance is not specific to one network (ASCAD, Zaid, or noConv1). On the left side of the figure, one observes

that when using directly the EM traces for attacking (labeled as F2EM), the attack does not succeed for any network. Next, similar performance can be observed when the network is trained on the power channel (labeled F2PW). Contrary the right plot shows that using the model trained on the power channel (F2PW) while attacking the translated synthetic traces, we observe that the attack rapidly converges towards a GE of 0 using less than 10 attack traces. We further evaluate the outcome in the scenario of non-profiled attacks. Figure 7 (bottom) illustrates CPA using directly the EM dataset and when using the translated synthetic traces. Without translation the correct key is found with approximately 30 traces, whereas with the translated synthetic dataset the key can be found using below 15 traces.

These results confirm that GAN is able to translate between the EM and power domain (i.e. F2PW not working on EM dataset, but on translated dataset for the three networks), and further that the translation is increasing the exploitable side-channel information using a profiled DL-based attack or even when considering a classical non-profiled univariate attack.



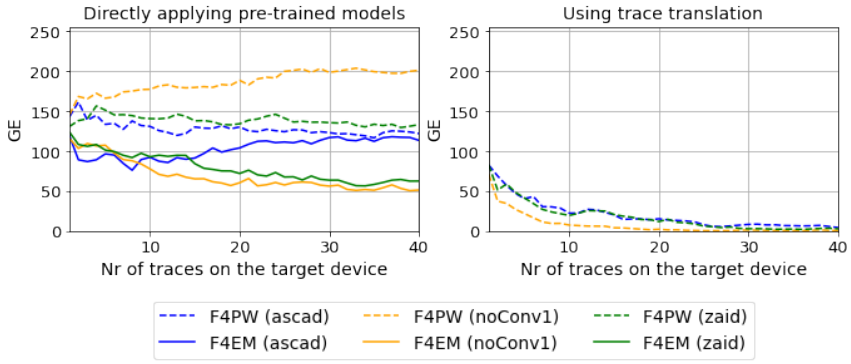
(a) Mean EM dataset (b) Mean synthetic dataset (c) Mean power dataset

(d) SNR of EM traces (e) SNR of synthetic traces (f) SNR of power traces

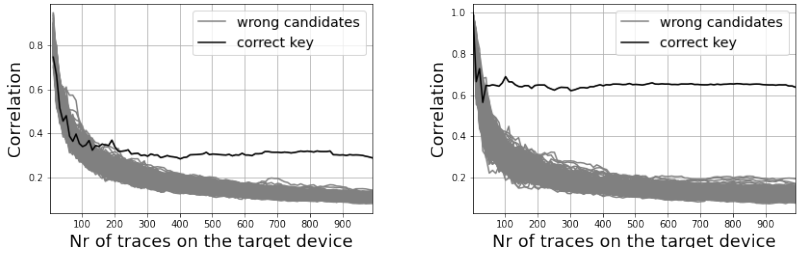
Fig. 8. Evaluation of STM32F4: (a) - (c) illustrates the (mean) trace and (d)-(f) the SNR of the attack dataset for the EM, power channel, and the translated synthetic trace dataset

STM32F4 Figure 8 (top) shows the mean trace obtained from STM32F4 of the dataset for EM/power measurements and the synthetic traces, where we visually observe a translation. In the bottom of the figure we plot the SNR obtained on different attack datasets. The maximum SNR peak from the EM channel is close to 0.12, whereas it is close to 17.5 for the power channel, showing

that the power channel is containing less noise. Again, the shape of the synthetic dataset is different compared to the original channels but in this experiment the translation preserved perfectly time locations. We observe that the SNR value of the synthetic attack set is improved by a factor 10.



(a) GE for three networks; left: original EM, right: translated synthetic dataset

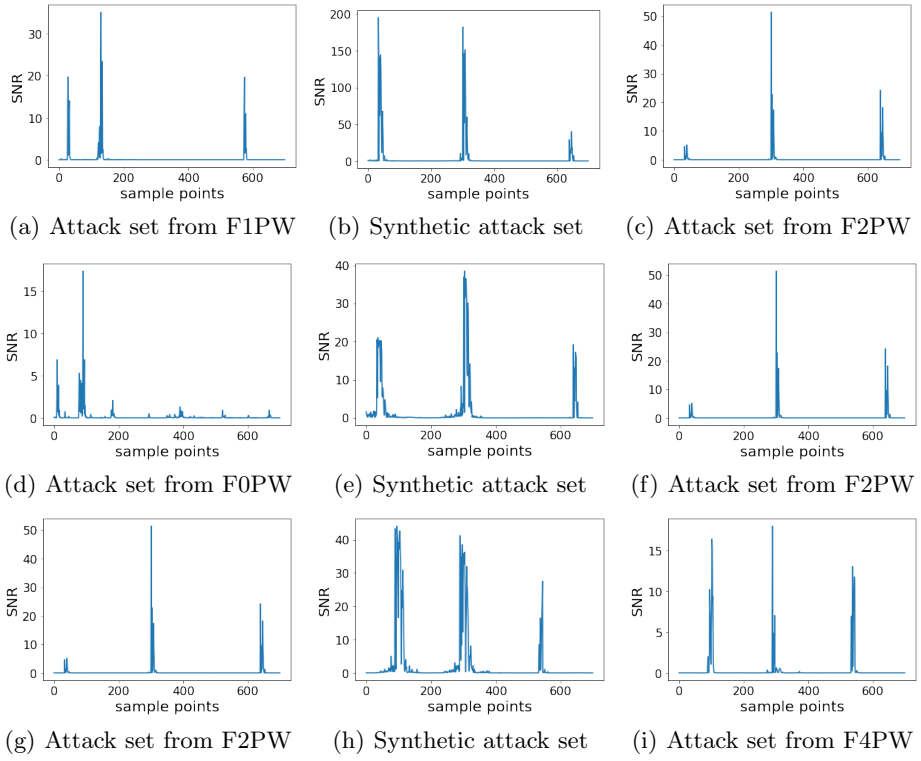


(b) CPA on EM dataset (c) CPA on synthetic dataset

Fig. 9. Attack evaluation on STM32F4 EM (original and synthetic translated traces)

Figure 9 (top) shows the GE obtained when targeting device STM32F4 with EM for all three networks. As for STM32F2, we observe on the left that attacking the EM channel using a model trained on EM (F4EM) or trained on power consumption (F4PW) does not converge within the given traces for any network. When using the translated synthetic traces and a model trained on the power channel (F4PW) GE reaches a mean rank equals to zero with only 20 traces for NoConv1 and around 30 traces for the other two networks. So, again by using the GAN translation, we could turn an unsuccessful attack into a successful one.

Figure 9 (bottom) shows that again the performance of CPA is improved as well. Using directly the EM dataset reveals the key using around 200 traces, while the difference of the correct key guess and the other key guesses is not significant. On the other hand, when using the translated synthetic traces succeeds using around 50 traces.



(a) Attack set from F1PW (b) Synthetic attack set (c) Attack set from F2PW

(d) Attack set from F0PW (e) Synthetic attack set (f) Attack set from F2PW

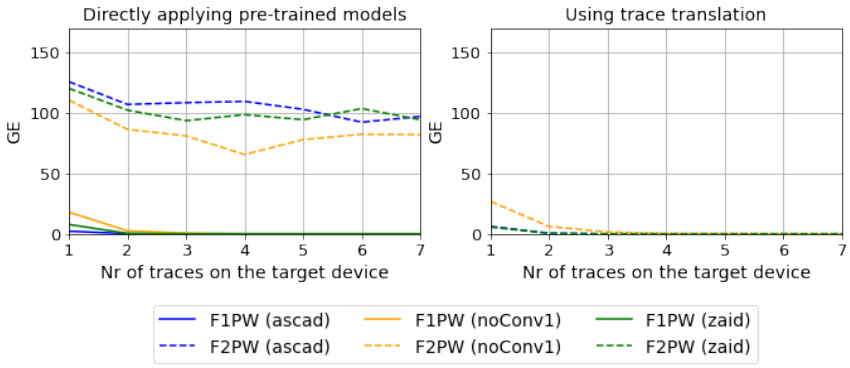
(g) Attack set from F2PW (h) Synthetic attack set (i) Attack set from F4PW

Fig. 10. SNR evaluation for each of scenarios considered: (a) - (c) F1PW translated to F2PW, (d)-(f) F0PW translated to F2PW, (g)-(i) F2PW translated to F4PW; left column domain A, middle column translated, right column domain B dataset.

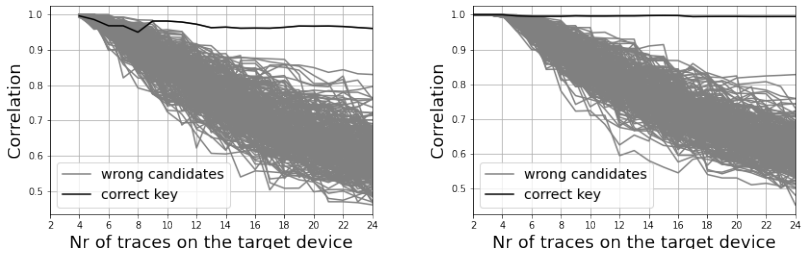
Summary In this section, we demonstrate that a translation from EM to power consumption is possible and that it reduces the number of traces needed for a successful attack. On both datasets, the classical approach (attacking directly the noisy channel) with DL failed and we observe that directly using a network trained on the power channel (but without translating the attack dataset) leads to poor performances. Our results show that GANs can be used to translate traces from EM to power channel, and the synthetic traces combined with a network trained on the power channel is successful. Additionally, our results show that the performance of CPA is greatly improved when attacking on translated instead of the original traces.

5 Cross-Device Translation

In this scenario, we investigate the capability of GAN to translate traces captured from one device to another. For the experiments presented in this section,



(a) GE; left: original dataset, right: translated synthetic dataset



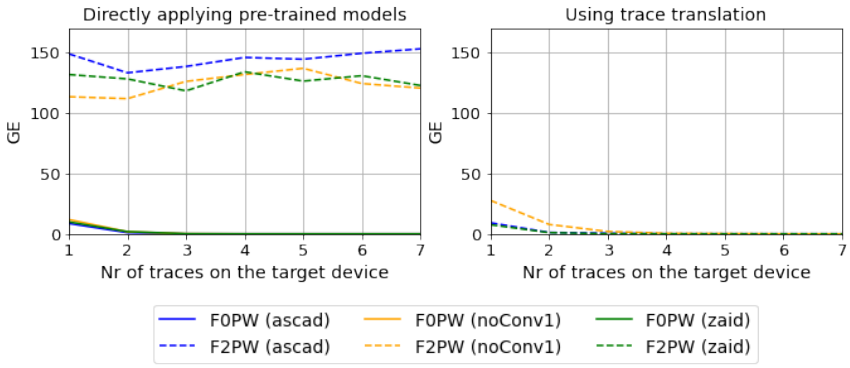
(b) CPA on STM32F1 (c) CPA on synthetic dataset

Fig. 11. Attack evaluation on STM32F1 (original and synthetic translated traces)

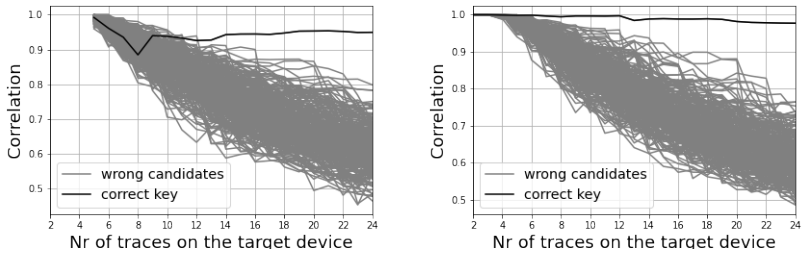
Domain A will thus correspond to traces from one chip and **Domain B** to traces from another chip.

STM32F1 power to STM32F2 power In Figure 10 (a)-(c), we plot the SNR evaluation obtained with different attack sets when translating F1 to F2 with power. The SNR value is close to 50 for F2, the synthetic attack set has an SNR value close to 200 (which is even larger than the target domain), and we can observe that GAN retrieved all three leakage positions from F2 which are at different time locations than F1.

In Figure 11 (top), we plot the GE obtained when we target F1 with power. Any of the DL-based attacks on the target domain (labeled F1PW) reaches a mean rank equal to zero below 2 traces. Using a model trained on F2 directly does not succeed, however, when translating to the domain F2, the ASCAD and Zaid networks succeed as well within 2 traces. Even though the performance using directly F2PW cannot be improved because of its high SNR by nature, this scenario shows that the translation into another domain is possible. For CPA (given at the bottom of the figure) we see that the performance can be



(a) GE; left: original dataset, right: translated synthetic dataset



(b) CPA on STM32F0 (c) CPA on synthetic dataset

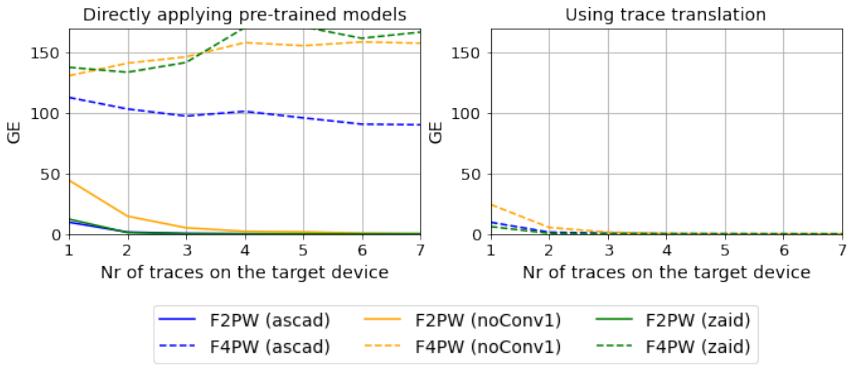
Fig. 12. Attack evaluation on STM32F0 (original and synthetic translated traces)

improved due to translation. Attacking the original traces is successful within 8 traces, whereas the correct key is found on the synthetic traces using 2 traces.

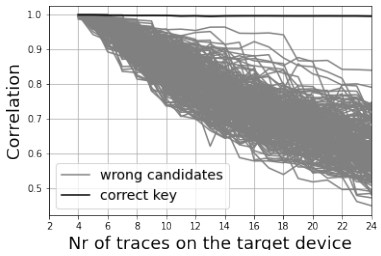
STM32F0 power to STM32F2 power In Figure 10 (d)-(f), we plot the SNR evaluation when we translate F0 to F2 device with power. The SNR value is close to 50 for the F2 device, and the synthetic attack set has an SNR close to 12 (which is even smaller than the target domain), but again we can observe that GAN retrieved the leakage positions from F2, while being different in time and amount for F0 and F2.

In Figure 12 (top), we plot the GE obtained when we target F0 with power. As before, all DL-based attacks on the target domain (labeled F0PW) are already efficient (as the SNR is high enough), whereas the model trained on F2PW does not succeed on the original traces. Even though with a tiny difference, we observe that the best performance was obtained by applying GAN and the Zaid network.

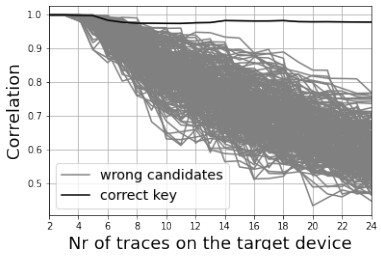
As in the previous scenario, we see that the translation improves the attackability with CPA. On the original dataset, the key can be found using around 12 traces, whereas on the translated synthetic traces we see that the correct key can be found immediately using 2 traces.



(a) GE for three networks; left: original EM dataset, right: translated synthetic dataset



(b) CPA on STM32F2



(c) CPA on synthetic dataset

Fig. 13. Attack evaluation on STM32F2 (original and synthetic translated traces)

STM32F2 power to STM32F4 power Unlike as in the previous scenarios, we now consider to translate from a device with higher SNR to another one with lower SNR. This is driven by several aspects, for example, the investigation if the translation between domains is still successful (even though it may give less attackability), or an attacker may only have a dataset available with a tuned network for a device with less SNR. In Figure 10 (g)-(i), we plot the SNR evaluation when we translate F2 to F4. The SNR value is close to 17.5 for the F4 device, the synthetic attack set has an SNR value close to 40 (which is still larger than the target domain), and we can observe that GAN has recovered all three leakage positions at the same time positions as F4.

Figure 13 shows the GE obtained when targeting F2 with power. First, we see that indeed a translation to a higher noise domain is possible, as all networks trained on F4PW succeed on the translated dataset, but fail on the original dataset. Second, one can observe that the attack performance of the three neural networks is not degraded, but rather slightly improved. The increase may be explained by the fact, that even though F2 has a higher SNR peak, F4 contains three SNR peaks that are of a similar magnitude. We can see a similar behavior in the translated synthetic attack dataset, which shows three SNR peaks with

comparable SNR levels. Indeed, when summing up all SNR values, the synthetic dataset achieves a higher value than F2.

The performance of CPA shows a degraded performance, which is expected as its an univariate attack, only considering one SNR peak at a time, where the magnitude of SNR is directly related to the success of CPA [3].

Summary Our experiments show that a trace translation between devices is possible (lower to higher and higher to lower SNR) and the leakage positions of the target domain are retrieved by translation. Concerning the SNR amplitudes, we observed various behaviors. In two of the experiments, the synthetic traces lead to a significantly higher SNR than the target domain, which we did not observe when translating EM to power consumption in the previous section. An intuitive explanation may be that the noise of the two devices (domains) is not related (i.e., random for the GAN). Accordingly, the translation will be mainly focused on deterministic leakage, which results to a higher SNR.

6 Conclusion & Future Work

This paper derives successful trace-to-trace translations using GANs in two different contexts, which opens a new door for adversaries. To attack a noisy setting, an adversary is able to select another type of device or side-channel source that is easier to attack and use trace translation to improve attack performances. Using three different state-of-the-art profiled neural network attacks on the synthetic translated traces, we show that the translation is not particular to a network. To learn the translation between datasets, the approach in this paper does not require knowledge of the label (and thus the secret key in some contexts), but only requires to have paired datasets. This makes it also interesting in the non-profiled attacker model, demonstrated by using CPA in our experiments. As our work is the first demonstration of trace-to-trace translation, we see several directions of future works. For example:

- further fine-tuning and architecture exploration of GAN networks for trace translation in the context of side-channel analysis (similarly as it has been achieved for CNNs in the state-of-the-art);
- exploration of the limitations of trace translation that may go beyond devices (of related types) and side-channel sources;
- further relaxation of the required datasets;
- applicability of protected implementations. We already applied our method to a hiding countermeasure (additional Gaussian noise) that showed similar success as the results presented in the paper. Another important research direction would be the investigation of masking countermeasures. However, this may need adaptions of the methodology to lead to successful translations due to the more complex alegbraic links between the leakage and the labels.

References

1. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) CHES. pp. 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
2. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: CHES. pp. 13–28. CHES '02, Springer-Verlag, London, UK, UK (2003)
3. Fei, Y., Luo, Q., Ding, A.A.: A statistical model for dpa with novel algorithmic confusion analysis. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2012. pp. 233–250. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
4. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014)
5. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. CVPR (2017)
6. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks (2018)
7. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks (2019)
8. 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. TCHES **2019**(3), 148–179 (May 2019)
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO '99. pp. 388–397 (1999)
10. Kubota, T., Yoshida, K., Shiozaki, M., Fujino, T.: Deep learning side-channel attack against hardware implementations of aes. In: 2019 22nd Euromicro Conference on Digital System Design (DSD). pp. 261–268 (Aug 2019)
11. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: SPACE 2016s. pp. 3–26 (2016)
12. Masure, L., Dumas, C., Prouff, E.: A comprehensive study of deep learning for side-channel analysis. TCHES **2020**(1), 348–375 (Nov 2019)
13. Pascual, S., Bonafonte, A., Serrà, J.: Segan: Speech enhancement generative adversarial network (2017)
14. Prouff, E., Strullu, R., Benadjila, R., Cagli, E., Dumas, C.: Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. Cryptology ePrint Archive, Report 2018/053 (2018), https://eprint.iacr.org/2018/053
15. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), http://arxiv.org/abs/1409.1556
17. 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 (2020)
18. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient CNN architectures in profiling attacks. TCHES **2020**(3), 147–168 (2020)
19. Wu, L., Picek, S.: Remove some noise: On pre-processing of side-channel measurements with autoencoders. TCHES **2020**(4), 389–415 (Aug 2020)
20. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient cnn architectures in profiling attacks. TCHES **2020**(1), 1–36 (Nov 2019)