A Pre-Silicon Power Leakage Assessment Based on Generative Adversarial Networks

Aljuffri, Abdullah ; Saxena, Mudit ; Reinbrecht, Cezar; Hamdioui, Said; Taouil, Mottaqiallah

DOI
10.1109/DSD60849.2023.00022

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 26th Euromicro Conference on Digital System Design (DSD)

A Pre-Silicon Power Leakage Assessment Based on Generative Adversarial Networks

Abdullah Aljuffri, Mudit Saxena, Cezar Reinbrecht, Said Hamdioui, Mottaqiallah Taouil
Computer Engineering Dept. – EEMCS Faculty
Delft University of Technology
Delft, The Netherlands
{A.A.M.Aljuffri,M.saxena,C.R.WedigReinbrecht,S.Hamdioui,M.Taouil}@tudelft.nl

2023 26th Euromicro Conference on Digital System Design (DSD) | 979-8-3503-4419-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/DSD60849.2023.00022

***Abstract*—Security is one of the most important features that a system must provide. Depending on the application of the target device, different threats should be considered at design time. However, the attack space is vast. Hence, it is difficult to decide what components to protect, what level of protection they require and how efficient they are in the field. This paper tries to close this validation gap for power based side channel attacks by providing a fast and reliable leakage assessment at design time that can be used to perform design space exploration for security. To accomplish our goal, we use Generative Adversarial Networks (GAN) to generate reliable power traces for hardware implementations at design time that are subsequently used to assess the leakage of the design. As a case study, we validated our framework against three AES implementations (i.e., unprotected, masked-protected, and balanced protected). In comparison to CAD-based scenarios, our findings show that the GAN model creates extremely reliable power traces in terms of attackability and leakage assessment. In addition, it is approximately 120 times quicker than CAD tools with respect to trace generation.**

***Index Terms*—Countermeasures, design exploration, generative adversarial networks, side channel analysis, symmetric cryptography**

I. Introduction

There is no doubt that security has become a critical component of microelectronic devices. Software-only solutions are being complemented by hardware security solutions (e.g., ARM TrustZone [1] and Intel Boot Guard [2]) [3]. However, designing a hardware protection scheme is quite challenging as new vulnerabilities arise and, differently from software, they cannot be updated after deployment. One set of popular hardware attacks exploit the power consumption. For example, a malicious adversary can take advantage of power behavior to deduce secret information through side channel attacks [4]. One way of evaluating power-based vulnerabilities is to use off-the-shelf security tools and equipment (e.g., equipment of Rambus [5] or Riscure [6]). This approach can only be applied after the chip is manufactured. Consequently, this verification adds extra steps in the design process, which increases the production cost. Even worse, when the security is not satisfactory the chip has to be redesigned and manufactured, affecting not only the cost but also the design time considerably. Therefore, a pre-manufacturing power leakage assessment solution is needed.

There are currently a few options to evaluate countermeasures prior to manufacturing. These options can be divided into two categories: formal verification [7] and CAD tools [8]. The aim of formal verification-based solutions is to mathematically analyze the leakage of an implementation. Formal verification examples can be found in [7], [9]. Unfortunately, such solutions focus on analyzing randomness created by masks. As a result, they only operate on one type of countermeasure, i.e., masking countermeasures. CAD-based solutions, on the other hand, tend to produce the power behavior of the targeted implementation such as [10] and [11]. Unfortunately, creating simulated power traces is a time-consuming operation, and reducing the number of simulated traces cannot validate the protection against real attacks such as Correlation Power Analysis (CPA), which typically require a large number of traces. Additionally, we see in the industry that secure IPs are evaluated with a minimum of 10 million power traces [12]. Therefore, a CAD-based assessment solution would be an interesting solution only if it can generate millions of power traces in a timely manner.

This paper presents a novel method to speed up the CAD based assessment by generating reliable power traces at design. We first train a Generative Adversarial Network (GAN) based on CAD-based power traces and their corresponding switching activity. Subsequently, the GAN is used to generate new power traces using the switching activity as input only. In summary, the contributions of this paper are:

- Proposal of a novel methodology to generate power traces and a leakage assessment framework based on GANs that can be used at design time.
- Proposal of evaluation metrics to measure the quality of the generated traces.
- Evaluation of the framework by performing a design space exploration (i.e., search for a protected AES implementation with low leakage). The security is evaluated by performing attacks on the generated traces.
- Applying transfer learning to reduce training time of the GAN.
- Validation of the generated traces by comparing them to CAD-based power traces.

The remainder of the paper is organized as follows. Section II describes the related work. Section III provides a background on AES, side channel attacks, and Generative Adversarial Networks. Section IV explains the proposed leak-

2771-2508/23/$31.00 ©2023 IEEE
DOI 10.1109/DSD60849.2023.00022

age assessment framework. Section V validates the proposed framework against different AES implementations. Finally, Section VII concludes this paper.

II. Related Work

Generative deep models have already been applied to numerous applications like images [13], audio [14], video [15] and medical data like ECG [16]. In addition to the Generative Adversarial Networks, there are also other methods that could be used to generate fake traces such as *Flow* [17] and *Auto-regressive* [18] based models. *Flow* models use a sequence of invertible transformations to learn the exact data distribution. As exciting as the possibility of exact likelihood computation might seem, *Flow* models usually have manifold times more trainable parameters and require an order of magnitude more processing (in GPU-based platforms) for the training as compared to progressive GAN models [19]. On the other hand, *Auto-regressive* models decompose the likelihood into a product of conditional distributions. However, since the prediction at every timestamp is dependent on all previous predictions, these models are implicitly slow. Hence, for this work, we prefer GANs over other deep generative models as GANs can be trained relatively faster, have a reasonable number of trainable parameters, and have been empirically proven to produce really good quality results. Additionally, taking the power leakage behavior into account, a GAN-based model is ideal since it is quick at inference time and thus can produce power traces swiftly. The existing Electronic Design Automation(EDA) tool-based approaches to generate artificial power traces are extremely slow to generate a large amount of traces. Hence, employing GANs for this task can potentially lead to an immense speed-up.

Generative adversarial networks have recently been introduced in the side channel analysis domain. In 2020, Wang et al. [20] proposed the usage of conditional GANs to enlarge the size of the profiling dataset for carrying out profiled side channel attacks. They use a traditional CGAN architecture with dense layers and train it using the Jensen-Shannon Divergence approach as proposed by Mirza and Osindero [21]. In the author's own words, their study was aimed as a proof of concept and not a robust methodology. Hence their proposed methodology has a few limitations. First, dense layers are best suited for shorter trace lengths, as the number of trainable parameters grows substantially as the input/output size of the GAN's layers increases. Second, they condition the GAN on only a few labels, namely the least significant bit and Hamming weight of the SBOX output. Finally, they only use the Jenson-Shannon Divergence loss and do not experiment with other loss functions. Since the inception of GANs, many architectural changes and loss functions have been proposed that help in alleviating this problem as well as enhancing training stability.

III. Background

This section provides the background needed for this paper. It first describes Advanced Encryption Standard. Thereafter, it presents the masking countermeasure. Finally, it discusses generative adversarial networks.

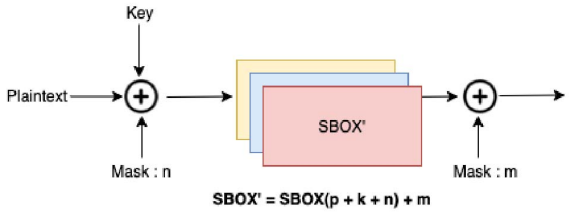

Fig. 1: Random Masking

A. Advanced Encryption Standard (AES)

Since 2001, AES, also known as Rijndael Cipher, has become the current standard for symmetric cryptographic algorithms. AES algorithm consists of four main functions: **AddRoundKey**, **SubByte**, **MixCoulmns**, and **ShiftRows**, and it supports three different keys length variants 128, 196, and 256 bits. For more details regarding AES we refer the reader to [22]. Among all AES functions, the **SubByte** leaks the most information, and hence is often targeted by side-channel attackers. The **SubByte** function (or so-called SBOX) is an essential nonlinear substitution function that attempts to obfuscate the correlation between the key and the plaintext/ciphertext. The **SubByte** function works as follows: the multiplicative inverse of an 8-bit input representing a polynomial is calculated using the finite Galois Field GF(2^8) and the irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$ followed by an affine transformation. To optimize the **SubByte** in terms of performance, a pre-calculated look-up table (LUT) containing all the possible values (256 values) of the 8-bit input is used, which replaces the computation of the multiplicative inverse and affine transformation. In addition to the two naive implementations already described (i.e., Non-LUT and LUT-based), the SubByte can also be implemented in a variety of different ways, mainly to prevent information leakage [23].

B. AES Countermeasures

For the past two decades, several countermeasures have been proposed to thwart power attacks. These countermeasures vary based on their implementation (i.e., software, hardware design, circuit implementation) and technique (i.e., obfuscating or balancing power consumption). One of the famous examples of obfuscating the power consumption is masking. Masking can be implemented in software [24], by modifying the hardware design [25] or using masked logic cells (i.e., masked dual-rail pre-charged logic) at circuit level [26].

In this paper, a simple form of masking implementation [27] is used in addition to the naive implementations. The countermeasure, as shown in Figure 1, works as follows: by selecting two random 8-bit values (mask m and mask n) the SubByte function can be calculated using Equation1. In Equation 1, P denotes a byte of the plaintext, while K represents a byte

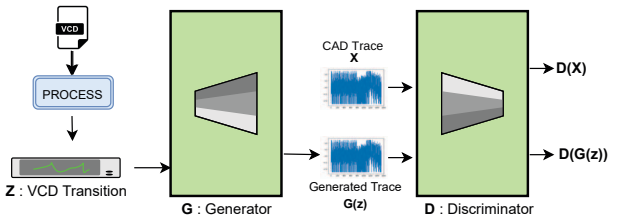

Fig. 2: Generative Adversarial Network

of the secret key. To ensure that the encryption output is correct, the SBOX also needs to be modified. Note that the SBOX performs a non-linear transformation, hence for a given plaintext P and mask n the sbox output $SBOX(P \oplus n)$ is not equal to $SBOX(P) \oplus SBOX(n)$. This countermeasure aims to reduce the leakage by invalidating the Hamming Weight and Hamming Distance leakage models.

$$SBOX[P \oplus K \oplus n] \oplus m \tag{1}$$

In addition to the masking countermeasure, which can be classified as an obfuscating countermeasure, the paper considers the balancing countermeasure known as duplicate design [28]. In this technique, two implementations of the encryption algorithm (e.g., AES) run simultaneously; one encryption uses the input message while the other one its complementary value. The aim is to prevent leakage by having overall the exact same switching activity, regardless of the message being encrypted.

C. Generative Adversarial Networks (GANs)

GANs are used to generate new data sets with similar characteristics as the training set, i.e., to approximate the training set's distribution. They consist of two main components, i.e., the generator G and discriminator D (see Figure 2. The generator's objective is to generate samples with a similar distribution to the actual dataset distribution. The discriminator's objective is to differentiate between real and fake traces, namely *x* and *G(z)*. Hence, the training process of a GAN takes place in an adversarial setting wherein the discriminator and the generator play a minimax game. The loss function L can be expressed as follows:

$$\begin{aligned} \min_G \max_D L(G,D) \quad &= \quad E_{x\sim p(x)}[\log D(x)] \\ &+ \quad E_{z\sim p(z)}[\log(1 - D(G(z)))] \end{aligned} \tag{2}$$

This loss function corresponds to the original GAN architecture as proposed by Goodfellow et al. [29]. However, as we want to condition the GAN on a categorical label corresponding to each encryption, the label y (which for example can correspond to the hamming weight/distance of the SBOX output) is also a part of the loss function. This conditional GAN loss function was originally proposed by Mirza and Osindero [21] and can be expressed as:

$$\begin{aligned} \min_G \max_D L(D,G) = \quad & E_{\boldsymbol{x}\sim \mathrm{p}_{\mathrm{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x} \mid \boldsymbol{y})] \\ + \quad & E_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z} \mid \boldsymbol{y})))] \end{aligned} \tag{3}$$

The non-cooperative game is what makes training GANs hard and unstable. To address this problem, a lot of research has been performed to find better loss functions and normalization techniques. In this work, in addition to the loss function presented in Equation 3, we experimented with the Least Squares GAN (LSGAN) [30] and the Wasserstein GAN Gradient penalty (WGAN-GP) loss function [31]. The LSGAN aims at increasing the quality of the generated samples by improving the discriminator's output by not only looking at the binary output decision but also the quality of the generated traces. Note that in the GAN proposed by Mirza and Osindero, the discriminator's purpose is to distinguish between the real and fake samples as shown in Equation 3, which is realized using a binary cross entropy loss. In LSGAN, however, we are not only concerned with the binary classification but also with how close or how far the fake traces are from the real ones. This can be seen in the loss function presented in Equation 4.

$$\begin{aligned} \min_D L_{\mathrm{LSGAN}}(D) &= \frac{1}{2}\mathrm{E}_{x\sim p_{\mathrm{dota}}(x)}\left[(D(x) - b)^2\right] \\ &+ \frac{1}{2}E_{\boldsymbol{z}\sim p_x(z)}\left[(D(G(z)) - a)^2\right] \\ \min_G L_{\mathrm{LSGAN}}(G) &= \frac{1}{2}\mathrm{E}_{z\sim p_x(z)}\left[(D(G(z)) - c)^2\right] \end{aligned} \tag{4}$$

During the training of the discriminator, we choose the value of *b* as 1 to signify real traces and *a* as 0 to signify generated traces. As the objective of the generator is to create fake traces that look real, we set the value of c during the training of the generator to 1 in order to attempt to fool the discriminator.

The WGAN-GP comes from the family of Wasserstein GANs (WGAN). The objective of WGANs is to minimize the earth mover (EM) distance. The EM distance represents the level of dissimilarity between the distributions of the generated and real traces. Minimizing the EM distance leads to smoother gradients even when the generator outputs unsatisfactory traces. The WGAN's discriminator tries to model a function that approximates the EM distance and not just distinguishing the real samples from the generated ones.

$$\mathrm{L}= \underbrace{\underset{\tilde{\boldsymbol{x}}\sim P_g}{E}[D(\tilde{\boldsymbol{x}})] - \underset{\boldsymbol{x}\sim P_r}{E}[D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}}\sim P_{\hat{\boldsymbol{x}}}}{E}\left[\left(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1\right)^2\right]}_{\text{gradient penalty}} \tag{5}$$

Calculating the EM distance is an intractable problem and the Kantorovich-Rubinstein duality [32] can be used to make the problem simpler. The Kantorovich-Rubinstein duality is used to transform the EM distance minimization problem in order to find a least upper bound. The transformed loss function is required to satisfy K-Lipschitz continuity. This continuity limits how fast a function can change. In the original WGAN paper [33], the Lipschitz constraint is enforced by weight clipping. However, the weight clipping method is

extremely sensitive to the clipping value hyperparameter and quite often reduces the network's ability to model complex functions. Instead, WGAN-GP adds a gradient penalty term to enforce the K-Lipschitz continuity as shown in Equation 5.

IV. Proposed Framework

This section presents our proposed framework. We introduce the methodology and subsequently explain each step in detail.

A. Methodology

Our proposed framework can be used to evaluate the efficacy and efficiency of countermeasures against side-channel attacks without the need to procure actual power traces for an ASIC design. The methodology to create this framework consists of two major phases as shown in Figure 3. In the first phase, the *Training Phase*, the generative adversial neural network is being trained for the targeted circuit using the switching activity from simulation and their corresponding CAD-based power traces. Once the GAN model reaches a desired accuracy, the second phase begins. In this phase, the *Generating Phase*, the trained neural network is used to generate the desired number of power traces to evaluate the security of the design. Here the GAN generates power traces solely from switching activity. In the following subsections, we describe each phase in more detail.

B. Training Phase

The first step in training the neural network to generate reliable power traces is to define the GAN's input and classification labels. As input data, we use the switching activity of the targeted circuit. One of the most common representations of the switching activity is the value change dump (VCD) file. VCD files can be generated during RTL or netlist simulations. To quantify the training accuracy, we need labels that represent the actual power traces. Hence, we use gate-level power simulations generated from a CAD tool. The CAD tool uses the switching activity and technology library to generate gate-level power traces. Next we configure the GAN network to be able to generate accurate power traces. Unfortunately, using the switching activity for the labels prevents us from using embedding layers in the GAN's architecture, as embedding layers expect integer numbers as input. The reason for this is that they are implemented as simple look-up tables. Hence, we remove the embedding layer as well as the noise vector and instead just use the VCD transition as a directed input to the GAN (see also Figure 2). A similar architectural choice was made by Kumar et al. in their MelGAN architecture [34], wherein they observe little perceptual difference in the generated waveforms when additional noise is fed to the generator. Mathieu et al. [35] and Isola et al. [36] demonstrated the capability of the noise vector's redundancy when using highly informative conditioning. Finally, we train the structured GAN until we reach a desired accuracy level.

C. Generating Phase

During this phase, the generator component of the trained GAN model (see Figure 2) is utilized to generate the power traces that can be used to evaluate the design or countermeasure. It generates these traces using the switching activity which can be obtained from RTL or gate-level simulations. Note that the amount of power traces that need to be generated can be significantly higher than those used in the training phase. Generally, more than 100k traces could be required for the evaluation. Several data sets can be constructed based on the evaluation method; examples are data sets with a random key and random plaintext, fixed key and fixed plaintext, and fixed key and random plaintext.

D. Evaluate Framework

The evaluation of the framework is performed through a generalization test. We evaluate if our model can provide reliable power traces. In this step, not only the plaintext varies but also the key value. In each scenario, the quality of the generated power traces is verified through leakage assessment techniques known as evaluation-test (i.e., Correlation Power Attack). Finally, both CAD-based and generated power traces are compared. In addition, for various different implementations, traces can also be generated and evaluated. In this context, the framework can be used to perform a design space exploration to find the most secure solution for a certain algorithm by quickly generating and evaluating traces for VCD files belonging to a different design. We limit our tests to three implementations only, which are: unprotected AES implementation, AES implementation with masking and AES implementation with blinding.

When the framework is completed, designers can generate as many traces as needed. For example, security components used in the industry (i.e., Intellectual Property blocks) require for their vulnerability assessment 10 million, 100 million or 1 billion power traces [12].

E. Framework Optimization

Each time the design changes, the generator model must be retrained to be able to generate reliable power traces. Training the generator is the most time-consuming phase of the proposed framework. Luckily, *transfer learning* can be used to reduce the training time. Transfer learning enables retraining of the neural network with much less effort, including the amount of training data required to achieve high accuracy [37]. There are two methods of applying transfer learning, namely feature extraction and fine tuning. In feature extraction, some of the layers in the trained networks are frozen and used for feature extraction, while others are retrained based on the new training data. In fine tuning, the weights of the trained network are used as initialization for the new network. Using the fine tuning technique in our framework, we were able to reduce the required number of traces from 10000 to 1000 to train the GAN for each newly developed countermeasure.

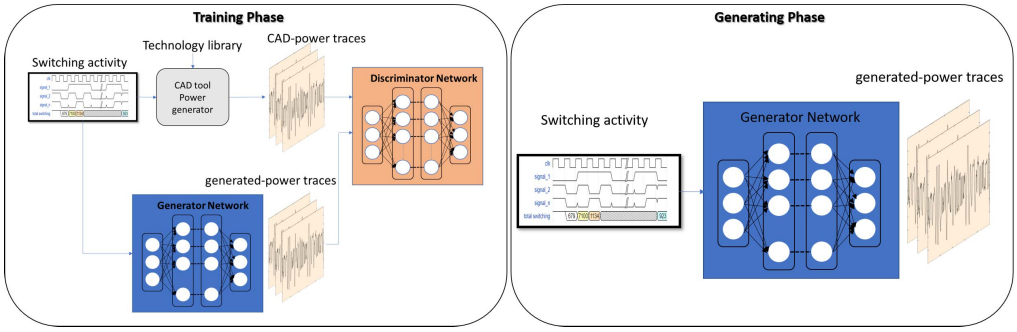

Fig. 3: Framework Methodology

V. Experimental Results

This section presents the experiments setup, performed experiments and evaluates the obtained results.

A. Setup

The different AES designs are simulated in Questasim [38], which is additionally used to generate the VCD files that contain the switching activity. Their corresponding power traces are generated by Synopsys SpyGlass; using the RTL-code of the design and the technology library for the target ASIC design, SpyGlass can generate power traces at gate-level. The VCD files and power traces are used to train the GAN model. The GAN model is implemented with the PyTorch [39] deep learning library and the remaining analysis are performed in Python as well. All the experiments including the training of the GAN are performed on an Intel i7-10750H CPU running at 2.60GHz and the Nvidia GeForce RTX 2070 GPU.

To verify the accuracy of the framework, we test our findings using three implementations with different keys/plaintext combinations to ensure that our approach works regardless of the input to the target design. As weight initialization (known as transfer learning) for both the Generator and Discriminator, a data set is used containing 20k traces of the unprotected AES implementation based on random keys and plaintext values. Note this is only done once per technology library. Next for the evaluation of the target design, we start by training the generator with only 1k traces. Subsequently, the Generator is used to generate traces based on a VCD belonging to the trained target (e.g., unprotected AES, protected masked SBOX implementation or protected AES implementation based on balancing). The traces are validated against corresponding traces obtained from SpyGlass. Note that the GAN is only trained with random inputs (i.e., for both plaintext and keys).

B. Evaluation Metrics and Performed Experiments

In this subsection, we first present the metrics used to evaluate our results. Thereafter we describe to which experiments these metrics have been applied.

Evaluation-style Metric: In evaluation-style testing, power traces are tested using actual side-channel attack scenarios. They show whether the implementations are resistant to these attacks or not. The attacks can be performed in a profiled or unprofiled manner. Examples of profiled side-channel attacks are template-based [40] and deep learning attacks [41]. Examples of unprofiled side-channel attacks are Differential power analysis [42] and correlation power analysis [43]. In this paper, we limit our analysis to CPA as it is one of the most popular unprofiled techniques. Subkeys with highest correlation are most likely the correct key guesses. The results are represented using rank analysis of the correct, also referred to as partial guessing entropy.

Trace equivalency: To compare the similarity between the SpyGlass (referred to as CAD traces) and the GAN traces (referred to as generated traces), certain signal processing metrics like *dynamic time warping* and *power spectral density* can be used. *Dynamic time warping (DTW)* [44] finds an optimal alignment between two unmatched temporal sequences. This optimal alignment or the 'warping path' maps the two sequences such that the distance between them is minimized. The minimum distance can be used as a measure for the similarity between any such two sequences. Similarly, even the *Power spectral density (PSD)* of two signals can be used as a measure for the similarity between them. *Power spectral density (PSD)* is the measure of power distributed across different frequency components that compose a signal [45]. For measured and generated traces to be visually similar, the *DTW* distance should be low in value (the lowest it can be is zero) and the *PSD* should be very similar.

C. Framework Evaluation

As described previously, we compare the attackability of the generated traces with the CAD-based power traces. To make the comparison fair, we test the Generator's generalization ability on VCD files corresponding to the same AES implementation but with a different key. In addition, different AES implementations with masking and blinding countermeasures are tested as well. Note that the experiment was performed using the GAN architecture shown in Figure 4. Hyperparameter tuning for GANs is much more complex than hyperparameter tuning for other machine learning models since the two-model architecture of GANs does not easily fit into the popular hyperparameter search APIs. Our initial architecture was inspired by a popular GAN implementation [46] and then we randomly searched for optimal hyperparameters. For this work: the batch-size is 100, and the kernel size

for the convolutional layers is 8. The idea of using a slightly larger kernel size was inspired from [47], where the author demonstrates that deep learning based power side-channel attacks using a convolutional neural network (CNN) with larger kernel sizes perform much better than CNNs with small kernel sizes. As for the loss function, the visual appearance of the traces as well as the leakage behavior showed minor variations for different loss functions. This minor impact of loss functions on the generated traces is in line with [48], where the authors state that the quality of the GAN generated samples are not substantially dependent on the loss functions.

```
DiscriminatorModel(
  (layers): Sequential(
    (0): Conv1d(1, 48, kernel_size=(8,), stride=(2,))
    (1): LeakyReLU(negative_slope=2.0)
    (2): MaxPool1d(kernel_size=2, stride=2)
    (3): Conv1d(48, 24, kernel_size=(8,), stride=(2,))
    (4): LeakyReLU(negative_slope=2.0)
    (5): MaxPool1d(kernel_size=2, stride=2)
    (6): Conv1d(24, 12, kernel_size=(8,), stride=(2,))
    (7): LeakyReLU(negative_slope=2.0)
    (8): Conv1d(12, 6, kernel_size=(8,), stride=(2,))
    (9): LeakyReLU(negative_slope=2.0)
    (10): MaxPool1d(kernel_size=2, stride=2)
    (11): Conv1d(6, 3, kernel_size=(8,), stride=(2,))
    (12): LeakyReLU(negative_slope=2.0)
    (13): MaxPool1d(kernel_size=2, stride=2)
    (14): Conv1d(3, 1, kernel_size=(8,), stride=(2,))
    (15): LeakyReLU(negative_slope=2.0)
    (16): MaxPool1d(kernel_size=2, stride=2)
  )
)
GeneratorModel(
  (layers): Sequential(
    (0): ConvTranspose1d(1, 48, kernel_size=(8,), stride=(2,))
    (1): LeakyReLU(negative_slope=2.0)
    (2): MaxPool1d(kernel_size=2, stride=2)
    (3): ConvTranspose1d(48, 24, kernel_size=(8,), stride=(2,))
    (4): LeakyReLU(negative_slope=2.0)
    (5): MaxPool1d(kernel_size=2, stride=2)
    (6): ConvTranspose1d(24, 12, kernel_size=(8,), stride=(1,))
    (7): LeakyReLU(negative_slope=2.0)
    (8): ConvTranspose1d(12, 6, kernel_size=(8,), stride=(1,))
    (9): LeakyReLU(negative_slope=2.0)
    (10): ConvTranspose1d(6, 1, kernel_size=(8,), stride=(1,))
    (11): Sigmoid()
  )
)
```

Fig. 4: GAN architecture

Evaluation-style Metric: The generated traces behave similarly to the measured traces when we look at CPA ranking analysis. For example, in the case of the unprotected implementation, the number of attackable bytes was 16, which equals the 16 attackable bytes when CAD traces were used. Similarly, in the mask-protected implementation, the number of the attackable bytes was zero for the GAN generated traces, which matches the results obtained from the CAD traces. The same results have been observed for the balance-protected implementation. We further go into the depths of the ranking analysis by examining the rank behavior of the correct key for each of the three implementations as shown in Figures 5, 6, and 7. The figures show that for different implementations the rank analysis trends are similar both for generated and CAD traces. This shows that the generative model can be used with a wide range of VCD inputs. We believe that the GAN model is able to extract the relevant signals for the leakage effectively from the VCD and filter out the unneeded signals. As a result, the generated traces can be used for evaluating

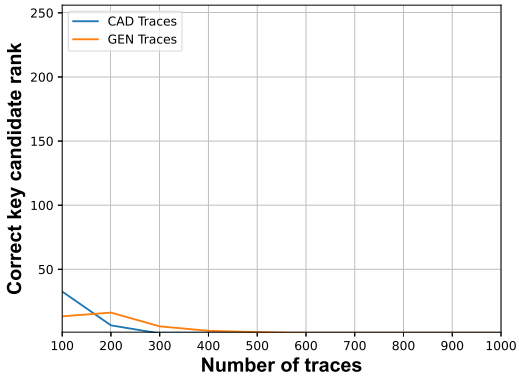

Fig. 5: CPA Results of Unprotected AES Implementation

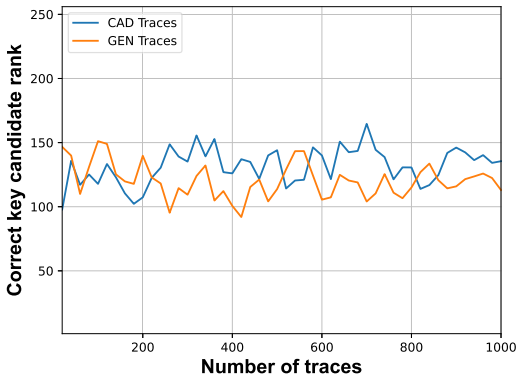

Fig. 6: CPA Results of Masked AES Implementation

countermeasures.

Trace equivalency: The distance between GAN and CAD traces was calculated using FastDTW [49] and we used Euclidean distance as the distance measure for DTW. We obtained values around 0.5 and noticed that the GAN provide good results when the DTW value is between 2.0 and 0.3

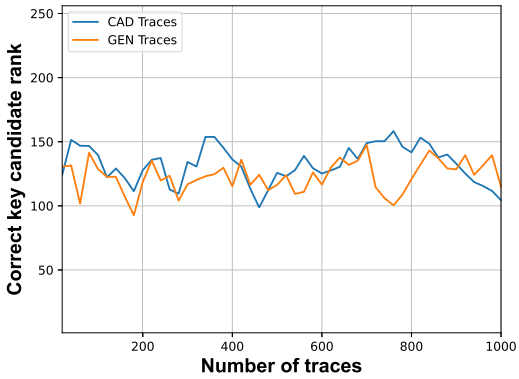

Fig. 7: CPA Results of Balanced AES Implementation

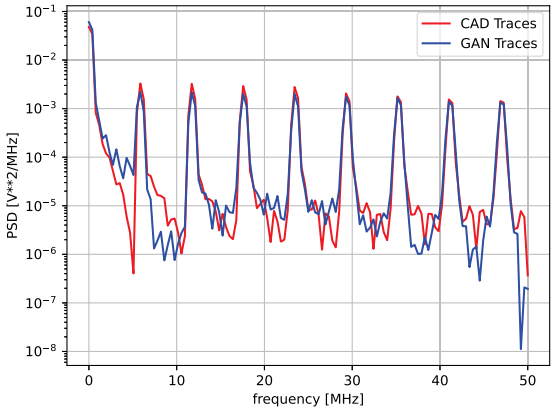

Fig. 8: PSD of CAD Versus GAN Traces

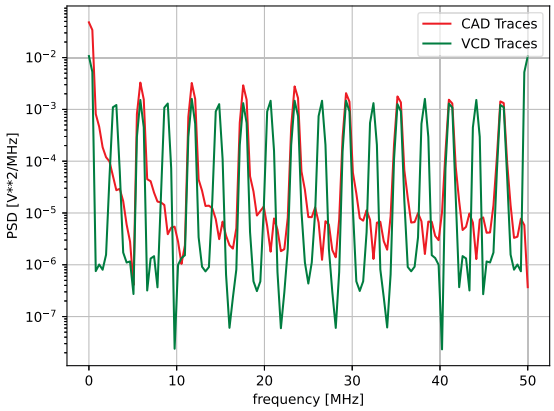

Fig. 9: PSD of CAD Versus VCD Traces

. Both these scores test to the similarity of the CAD and generated traces. The Power spectral density (PSD) of CAD and generated traces were also almost identical, as shown in Figure 8. Note however that the distance between the PSD of CAD and VCD traces is much larger (see Figure 9). This shows that using traces obtained from VCD only is not as accurate as generated from the GAN, even though the GAN uses the VCD as input.

VI. Comparison to State of the Art

Because we are only interested in hardware implementations, we have excluded from the comparison any trace generation methods that are based on formal verification, any methods that target software implementations at the instruction level, and any analytical-based approaches such as [50]. Methods that did not provide a discussion of the amount of time necessary to gather the traces were also excluded [51]. Instead, we compare our method solely to state-of-the-art pre-silicon techniques. We compare our approach to one method that is based on the layout level and three other methods that are based on the gate level, one of which is an approach based on artificial intelligence. For the purpose of this comparison, two criteria were used: first, the quality of the traces, which was evaluated depending on the level at which the traces were created (i.e., the layout, gate, or RTL level), and second, the speed at which those traces were generated. We presumed that traces produced at lower levels had a higher quality; nevertheless, they require more time to generate [52]. In terms of quality, we were successful in generating traces at the gate level, which is better than the RTL level and allowed us to obtain comparable results in terms of attackability. When it comes to speed, our proposed approach exceeded the state-of-the-art by 120 times as shown in Table I.

We are under the impression that our approach is likewise capable of generating layout-based traces at a quicker rate than the conventional way. However, because we need to train the GAN using several thousands of layout-based traces and because the generation of those traces takes a significant amount of time, this may be considered a shortcoming of the present technique. Still has the potential to be quicker than the standard approach (i.e., CAD tool) when the number of traces needed is in the hundreds of thousands or more.

VII. Conclusion

This study proposed a framework that is able to quickly generate traces that are very comparable to CAD-based traces. Our generative models were not only able to generate visually indistinguishable power traces from the training set, but were also able to learn the characteristics of the VCD transition array. According to our experiments, only a few thousands of CAD-traces are required to train GAN models using transfer learning in order to produce high-quality power traces by simply generating them from the switching activity. As a result, we significantly improve the performance.

References

[1] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," 2019.
[2] Intel, "A Security Model to Protect Intelligent Edge Devices." Available at: https://intel.ly/3CqyWS8. Accessed: 2021-05-08.
[3] G. Guez, "Why Hardware-Based Design Security is Essential for Every Application." Available at: https://bit.ly/2VAt9Zg. Accessed: 2021-12-23.
[4] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptogr.*, vol. 4, no. 2, p. 15, 2020.
[5] Rambus, "DPA Workstation Platform." Available at: https://www.rambus.com/security/dpa-countermeasures/dpa-workstation-platform/. Accessed: 2021-05-08.
[6] Riscure, "Inspector side channel analysis." Available at: https://www.riscure.com/security-tools/inspector-sca.
[7] R. Bloem *et al.*, "Formal verification of masked hardware implementations in the presence of glitches," in *EUROCRYPT 2018*, 2018.
[8] A. Nahiyan *et al.*, *CAD for Side-Channel Leakage Assessment*. 2021.
[9] H. Eldib *et al.*, "Formal verification of software countermeasures against side-channel attacks," *ACM Trans. Softw. Eng. Methodol.*, 2014.
[10] R. Sadhukhan *et al.*, "Count your toggles: a new leakage model for pre-silicon power analysis of crypto designs," *J. Electron. Test.*, 2019.
[11] A. Nahiyan *et al.*, "SCRIPT: A CAD framework for power side-channel vulnerability assessment using information flow tracking and pattern generation," *ACM*, 2020.

TABLE I: Comparison with State of the Art

Pre-silicon	Prepartion/Training time	Time (10k traces)	Trace Type
Gate-level (GAN)	35 minutes	$\sim 30seconds$	GAN generated
Gate-level (Transfer-learning)	15 minutes	$\sim 30seconds$	GAN generated
Gate-level PRIMAL [53]	3 hours	-	CNN generated
RTL-Level (RTL-PSC) [54]	N/A	1 hours	Simulated
Gate-Level (PATCH) [55]	N/A	10 hours	Simulated
Layout (RTL-PSC) [54]	N/A	days	Simulated

[12] RAMBUS, "Dpa resistant core - rambus." https://www.rambus.com/security/dpa-countermeasures/dpa-resistant-core/. (Accessed on 05/10/2021).

[13] A. Brock *et al.*, "Large scale GAN training for high fidelity natural image synthesis," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[14] J. Engel *et al.*, "Gansynth: Adversarial neural audio synthesis," 2019.

[15] A. Sagar, "Hrvgan: High resolution video generation using spatio-temporal gan," 2020.

[16] K. Antczak, "A generative adversarial approach to ECG synthesis and denoising," *CoRR*, vol. abs/2009.02700, 2020.

[17] R. Prenger *et al.*, "Waveglow: A flow-based generative network for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pp. 3617–3621, IEEE, 2019.

[18] A. van den Oord *et al.*, "Wavenet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, p. 125, ISCA, 2016.

[19] A. Odena, "Open questions about generative adversarial networks," *Distill*, 2019. https://distill.pub/2019/gan-open-problems.

[20] P. Wang *et al.*, "Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks," 2020.

[21] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[22] NIST, *Announcing the Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication 197, 2001.

[23] E. Oswald *et al.*, "A side-channel analysis resistant description of the AES s-box," in *Fast Software Encryption: 12th International Workshop, FSE 2005*, 2005.

[24] M. Masoumi, P. Habibi, and M. Jadidi, "Efficient implementation of masked aes on side-channel attack standard evaluation board," in *2015 International Conference on Information Society (i-Society)*, pp. 151–156, 2015.

[25] H. Maghrebi *et al.*, "A first-order leak-free masking countermeasure," in *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, vol. 7178 of *Lecture Notes in Computer Science*, pp. 156–170, Springer, 2012.

[26] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints," in *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, vol. 3659 of *Lecture Notes in Computer Science*, pp. 172–186, Springer, 2005.

[27] H. S. Kim and S. Hong, "New type of collision attack on first-order masked aess," *ETRI Journal*, 2016.

[28] M. Doulcier-Verdier *et al.*, "A side-channel and fault-attack resistant AES circuit working on duplicated complemented values," in *IEEE International Solid-State Circuits Conference, ISSCC 2011*, 2011.

[29] I. J. Goodfellow *et al.*, "Generative adversarial networks," 2014.

[30] X. Mao *et al.*, "Least squares generative adversarial networks," 2017.

[31] I. Gulrajani *et al.*, "Improved training of wasserstein gans," 2017.

[32] G. Basso, "A hitchhikers guide to wasserstein distances." https://bit.ly/3lFixDe, June 2015. (Accessed on 05/12/2021).

[33] M. Arjovsky *et al.*, "Wasserstein GAN," *CoRR*, vol. abs/1701.07875, 2017.

[34] K. Kumar *et al.*, "Melgan: Generative adversarial networks for conditional waveform synthesis," 2019.

[35] M. Mathieu *et al.*, "Deep multi-scale video prediction beyond mean square error," 2016.

[36] P. Isola *et al.*, "Image-to-image translation with conditional adversarial networks," 2018.

[37] Y. Wang *et al.*, "Transferring gans: generating images from limited data," *CoRR*, vol. abs/1805.01677, 2018.

[38] Siemens, "Questa Advanced Simulato." Available at: https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator/. Accessed: 2021-05-08.

[39] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019.

[40] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems*, 2002.

[41] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *IACR Cryptology ePrint Archive*, 2016.

[42] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO '99*, 1999.

[43] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *CHES*, 2004.

[44] M. Müller, *Information retrieval for music and motion*. Springer, 2007.

[45] P. Stoica and R. Moses, "Spectral analysis of signals," *Prentice Hall*, 01 2005.

[46] A. Radford *et al.*, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[47] R. Tubbing, "An Analysis of Deep Learning Based Profiled Side-channel Attacks," Master's thesis, Delft University of Technology, the Netherlands, 2019.

[48] M. Lucic *et al.*, "Are gans created equal? A large-scale study," in *Advances in Neural Information Processing Systems 31*, 2018.

[49] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," 2007.

[50] A. V. Lakshmy, C. Rebeiro, and S. Bhunia, "FORTIFY: analytical pre-silicon side-channel characterization of digital designs," in *27th Asia and South Pacific Design Automation Conference, ASP-DAC 2022, Taipei, Taiwan, January 17-20, 2022*, pp. 660–665, IEEE, 2022.

[51] P. SLPSK, P. K. Vairam, C. Rebeiro, and V. Kamakoti, "Karna: A gate-sizing based security aware EDA flow for improved power side-channel attack protection," in *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2019, Westminster, CO, USA, November 4-7, 2019* (D. Z. Pan, ed.), pp. 1–8, ACM, 2019.

[52] I. Buhan, L. Batina, Y. Yarom, and P. Schaumont, "Sok: Design tools for side-channel-aware implementations," in *ASIA CCS '22*, 2022.

[53] Y. Zhou, H. Ren, Y. Zhang, B. Keller, B. Khailany, and Z. Zhang, "PRIMAL: power inference using machine learning," in *Proceedings of the 56th Annual Design Automation Conference 2019, DAC 2019, Las Vegas, NV, USA, June 02-06, 2019*, p. 39, ACM, 2019.

[54] M. T. He *et al.*, "RTL-PSC: automated power side-channel leakage assessment at register-transfer level," in *37th VTS 2019*, 2019.

[55] V. S. Bokharaie and A. Jahanian, "Power side-channel leakage assessment and locating the exact sources of leakage at the early stages of ASIC design process," 2022.