TU Delft

Delft University of Technology

Fake It Till You Make It

Data Augmentation Using Generative Adversarial Networks for All the Crypto You Need on Small Devices

Mukhtar, Naila ; Batina, Lejla; Picek, Stjepan; Kong, Yinan

DOI
10.1007/978-3-030-95312-6_13

Publication date
2022

Document Version
Final published version

Published in
Topics in Cryptology - CT-RSA 2022

Citation (APA)
Mukhtar, N., Batina, L., Picek, S., & Kong, Y. (2022). Fake It Till You Make It: Data Augmentation Using Generative Adversarial Networks for All the Crypto You Need on Small Devices. In S. D. Galbraith (Ed.), *Topics in Cryptology - CT-RSA 2022 : Cryptographers' Track at the RSA Conference, 2022, Proceedings* (pp. 297-321). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 13161). Springer. https://doi.org/10.1007/978-3-030-95312-6_13

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Fake It Till You Make It: Data Augmentation Using Generative Adversarial Networks for All the Crypto You Need on Small Devices

Naila Mukhtar[1(✉)], Lejla Batina[2], Stjepan Picek[3,4], and Yinan Kong[1]

[1] Macquarie University, Sydney, Australia
naila.mukhtar@ieee.org, yinan.kong@mq.edu.au
[2] Radboud University, Nijmegen, The Netherlands
[3] Radboud University, Nijmegen, Nijmegen, The Netherlands
lejla@cs.ru.nl
[4] Delft University of Technology, Delft, The Netherlands

Abstract. Deep learning-based side-channel analysis performance heavily depends on the dataset size and the number of instances in each target class. Both small and imbalanced datasets might lead to unsuccessful side-channel attacks. The attack performance can be improved by generating traces synthetically from the obtained data instances instead of collecting them from the target device, but this is a cumbersome and challenging task.

We propose a novel data augmentation approach based on conditional Generative Adversarial Networks (cGAN) and Siamese networks, enhancing the attack capability. We also present a quantitative comparative deep learning-based side-channel analysis between a real raw signal leakage dataset and an artificially augmented leakage dataset. The analysis is performed on the leakage datasets for both symmetric and public-key cryptographic implementations. We investigate non-convergent networks' effect on the generation of fake leakage signals using two cGAN based deep learning models.

The analysis shows that the proposed data augmentation model results in a well-converged network that generates realistic leakage traces, which can be used to mount deep learning-based side-channel analysis successfully even when the dataset available from the device is not optimal. Our results show that the datasets enhanced with "faked" leakage traces are breakable (while not without augmentation), which might change how we perform deep learning-based side-channel analysis.

Keywords: Deep learning-based side-channel attacks · ASCAD · Elliptic curve cryptography · GANs · Data augmentation · Signal processing

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
S. D. Galbraith (Ed.): CT-RSA 2022, LNCS 13161, pp. 297–321, 2022.
https://doi.org/10.1007/978-3-030-95312-6_13

1 Introduction

Profiling side-channel attacks (SCAs) are the class of attacks in which the adversary is assumed to have access to the target device's open copy. Then, the attacker uses that copy of a device to build a strong profiling model. In the second phase of the attack, the attacker infers the secret information, e.g., the secret key from the target device based on the profiling model and measurements of some physical activity of the target device while cryptographic implementation is running on it. Recently, deep learning-based attacks have been extensively studied for improving the profiling SCA, see, e.g. [1–3]. The deep learning model's performance can suffer if enough data is not provided during the training phase. What is more, using deep learning may not even make sense if not enough data is available. This lack of availability could be a consequence of implemented countermeasure prohibiting collecting a large number of traces or due to the evaluation setup [4]. Additionally, it is common to use side-channel leakage models like the Hamming weight or Hamming distance, which will result in an imbalanced class scenario [5].

Deep learning is data-hungry. Not providing enough data can mean that either we do not reach the full potential of a certain method or, in more extreme cases, that the method shows very poor performance. One common reason for it is overfitting, where the deep learning model learns how to model the noise, making it difficult to generalize to the previously unseen examples. To fight against it, researchers commonly use techniques like 1) hyperparameter tuning - to find architectures that are better tuned for the task and thus have less tendency to overfit, 2) regularization - to lower the complexity of a neural network model during training, and thus prevent the overfitting, and 3) data augmentation - to provide additional (synthetic) examples, utilizing the capacity of a model better, but also regularize the model with noise inherent to the synthetic examples. Each of those techniques has its advantages and drawbacks, and they are also successfully applied to the side-channel domain. Interestingly, while hyperparameter tuning and regularization (e.g., dropout, L2 regularization) are commonly used in SCA, data augmentation received less attention, despite very good preliminary results. One possible reason lies in the difficulty of clearly visualizing how a successful synthetic side-channel measurement should look (something much simpler in, e.g., image classification domain).

Cagli et al. proposed the first data augmentation setup for deep learning-based SCA to counter the effects of clock jitter countermeasure [6]. Still, the authors do not consider scenarios where the number of measurements is significantly limited. Picek et al. presented the results with "traditional" machine learning data augmentation techniques and concluded that Synthetic Minority Over-sampling Technique (SMOTE) could aid in data generation, resulting in improved attack performance [5]. Differing from us, the authors used the Hamming weight leakage model, resulting in an imbalanced dataset. In our work, we used intermediate values resulting in more classes and more challenging analyses. Luo et al. used a mixup data augmentation technique where new synthetic examples are based on randomly selected combinations of existing examples [7].

The authors conducted experiments for several datasets and leakage models and obtained mostly similar behavior for the original and mixup traces. Generative Adversarial Networks (GAN) is another popular data augmentation technique that is widely used in the image processing domain for generating fake images [8], which significantly improves the machine learning model's performance [9]. Only one existing study presents the realization of using GAN-generated fake signals as leakage signals for deep learning-based side-channel analysis [10]. However, the authors use more profiling traces and the Hamming weight leakage model (which will result in fewer classes), making their work significantly different from ours. What is more, this work misses providing a detailed analysis of the GAN network before using it for the fake data generation.

GAN's performance for generating fake images/signals depends on the generator's progressive learning based on the discriminator's response. The design and selection of a GAN play an important role in generating realistic leakage signals. A well-convergent GAN network will generate traces carrying relevant/significant features similar to the original data samples. Designing a GAN-based model with optimum convergence or equilibrium point is one of the greatest challenges for generating fake signals [11] that contain the characteristics of real leakage traces. Several techniques, presented for fake image generation, can help achieve convergence, including feature matching [12], conditional GAN (cGAN) [13], and semi-supervised learning [12].

In our work, we generated 50% traces for each class ($256 \times 150 \times 2$ real and fake traces for AES and $16 \times 150 \times 2$ real traces for ECC), making the setting very challenging.

Our approach is inspired by the fake image generation presented in [14]. Our presented generic model can generate fake data for various leakage datasets, including symmetric and public-key algorithm implementations. We provide a comparative analysis with the existing simple dense layer-based GAN used for leakage generation.

Specifically, we list our contributions as follows:

- We present a layered approach for generating the 1-dimensional fake signals for deep learning-based side-channel analysis (DL-SCA). Our approach combines Siamese network and conditional GAN (cGAN) characteristics with an extra model loss monitoring layer introduced to detect the model convergence. The visual representation of the loss function of the proposed data augmentation technique helps analyze the well-converged GAN model. A well-converged network helps in generating indistinguishable fake traces that will give the same insights to the data as that of the original signals. With this, we showcase the relevance of "real vs. fake" data and exhibit successful attacks with synthetic data that could impact various real-world use cases and security applications.
- We provide a comparative analysis exhibiting the fake leakage trace datasets' effect on the side-channel model training performance. These fake leakage traces are generated from various converging points during the proposed Siamese-cGAN model training, which helps to analyze the importance of the well-converged models.

- The proposed Siamese-cGAN model can be generalized to any leakage dataset (from varying cryptographic algorithm implementations). To demonstrate this, we trained our proposed model on datasets containing either symmetric or public-key algorithm implementations, using two different neural networks for generator and discriminator. Best performing neural networks are further selected for analysis. While several results show the benefits of data augmentation for symmetric-key implementations, data augmentation is significantly less explored in the context of attacks on public-key implementations (we are aware of only one work [15]).
- We provide a comparative analysis of our proposed Siamese-cGAN model with the existing used cGAN model [10] (named as $cGANModelA$ in this study) for generating fake leakage traces.[1]
- The performance of the Siamese-cGAN data augmentation model is evaluated by applying the actual deep learning-based side-channel attack on the generated leakage traces using four different neural network architectures (one multilayer perceptron (MLP) and two Convolutional Neural Networks (CNNs) for symmetric algorithm implementation leakages and one CNN for public-key implementation leakages). Our results show that the fake data samples generated from the well-convergent model combined with 50% real data successfully recovered the secret with similar efficiency as real data traces alone. What is more, for the ASCAD dataset case (AES measurements), the key rank suggests even improved results for the dataset consisting of real and fake traces. We emphasize that the goal of our approach is not only to improve the attack performance in scenarios where there are enough real measurements for a successful attack. Rather, we envision it for constrained settings where more measurements than available are needed to break the target. We think here of implementations that randomize the secret (key) after a number of algorithm's execution such that the adversary can collect only a limited number of traces for the analysis.

2 Preliminaries

2.1 Profiled Side-Channel Attacks

The profiled attack represents the most powerful side-channel attack where the adversary has access to the target device's open copy. There are two phases of the attack: the profiling phase and the attack phase. In the profiling phase, the adversary creates a profile of the device with all the possibilities for the data leakages and then uses that profile (template in the case of template attack [16] or machine learning model) in the attack phase to distinguish/predict the unknown secret information [17].

Commonly, profiled attacks are divided into classical ones like template attacks [16] and stochastic model [18], and machine learning-based attacks [1,19].

[1] We note that the provided details were not sufficient to ensure the reproducibility of the results, so we did our best to infer the used architecture from the description.

Machine learning-based side-channel attacks (ML-SCA) follow the same steps as the classical profiling attacks. More precisely, they have two phases: training phases (profiling phase) and test phase (attack phase). The adversary can train the model with the leakage examples collected from the identical copy of a target device and then evaluate the trained model using previously unseen examples. When using deep learning instead of other machine learning techniques, we denote such attacks as deep learning-based side-channel attacks (DL-SCA).

2.2 Generative Adversarial Networks (GANs)

The Generative Adversarial Networks were first introduced by Goodfellow et al. in 2014 [9]. Since then, many variations of GAN have been proposed, including conditional GAN (cGAN), Deep Convolutional GAN (DC-GAN), Information Maximizing (InfoGAN), and Stacked GAN (StackGAN) [13,20–23].

A Generative Adversarial Network (GAN) is a neural network architecture for training a generative model, which can generate plausible data. It consists of two neural networks adversarial models, discriminator D and generator G. Real data is labeled as '1' and artificially generated (fake) data is labeled '0'. Generator G network generates the fake data with random noise input z, and discriminator D network discriminates between real and fake data.

GANs are based on the concept of a zero-sum non-cooperative game where one network (discriminator) is trying to minimize the loss, and the other network (generator) is trying to maximize the loss (this min-max problem as given by Eq. (1), where x and z represent the real and fake generated trace, respectively).

The discriminator's task is to distinguish the real and generated data instances, whereas the generator's task is to improve the model based on the feedback from the discriminator. However, the generator is trying to generate data traces that are alike. This makes it hard to find a good convergence point. GAN converges when both D and G reach a Nash equilibrium, meaning that one (D/G) will not change its actions anymore, no matter what opponent (D/G) does. This is the optimal point that adversarial loss in GANs aims to optimize.

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]. \tag{1}$$

2.3 Conditional Generative Adversarial Networks (cGANs)

GANs help generate plausible data, but there is no way to control what random data they will generate. GAN can be conditioned with extra data to control the generated information to handle that issue. GAN with an extra condition is called conditional Generative Adversarial Network (cGAN). The extra data in cGANs is the class label of the data samples. Using class labels for cGAN training has two main advantages: firstly, it helps in improving the GAN performance, and secondly, it helps in generating the specific target class samples.

In the training process of the cGAN model, the generator network generates data based on the label and the random input and tries to replicate the actual distribution in the real data samples. The generated samples are of the same structure as the input labeled data. In the next step, the real and the

generated/fake data are given as input to the discriminator. The discriminator is first trained with the original/real labeled data samples and is then trained with the fake generated data samples. Similar to GANs, the discriminator's task is to separate the fake from real data samples, which helps the generator generate better (realistic) samples [13].

2.4 Data Augmentation

Data augmentation is an umbrella term representing various techniques used to increase the amount of data by adding (slightly) modified copies of already existing examples or newly created synthetic examples from existing data. Data augmentation can act as a regularization technique and will help reduce overfitting. While data augmentation can be applied to any domain, most of the results and techniques were developed for data augmentation in image classification [24].

2.5 Deep Learning Algorithms

Based on the deep learning-based side-channel attacks (DL-SCA) performance on various cryptographic algorithms [6,25,26], we tested our newly generated datasets using two state-of-the-art deep learning approaches: multilayer perceptron (MLP) and Convolutional Neural Network (CNN). MLP and two variations of CNN [3,27] are used to evaluate the AES dataset, and one CNN architecture [25] is used to evaluate the ECC dataset.

2.6 Siamese Neural Network

Siamese neural network (also called twin/identical neural network) is an artificial neural network architecture consisting of two similar neural networks (with the same weights and network parameters). It is capable of processing two different input vectors to produce comparable output vectors [28]. The two neural networks are feedforward multilayer perceptrons that work in tandem and are trained in a backpropagation manner. The idea behind the Siamese neural network is not to learn to classify the labels but to discriminate between the input vectors. Hence a special loss function, contrastive loss, or Triplet Loss is used for the training of the network [29]. For training the network, the input vector pairs (x_i, x_j) are prepared; a few pairs consist of similar vectors, and a few pairs consist of dissimilar vectors. The similar vector pair is labeled as $y =$ '1', whereas the dissimilar pair is labeled as $y =$ '0'. Each pair is fed to the Siamese network, and distance is computed to check the similarity. The output vectors from each network are compared using cosine or Euclidean distance and can be considered as a semantic similarity between projected representation of the input vectors [30].

2.7 Cryptographic Algorithms Under Evaluation

For our analysis, we investigate the performance of our proposed model on two publicly available datasets. One dataset corresponds to the implementation of

Advanced Encryption Standard (AES) and the other to the Elliptic Curve Cryptography (ECC) implementation.

- The ASCAD dataset [27] is the first dataset that acts as a basis for comparative analysis of deep learning-based side-channel analysis (DL-SCA). The traces are collected from the masked AES-128 bit implementation on an 8-bit AVR microcontroller (ATmega8515). The leakage model is first-round S-box (Sbox$[P(i)_3 \oplus k^*]$) where the third byte is exploited (as that is the first masked byte). There are 60 000 total traces along with the metadata (plaintext/ciphertext/key). These traces are further split into two datasets, one for training (profiling) consisting of 50 000 and the other for the test (attack), consisting of 10 000 traces. Each trace consists of 700 features. The labels are stored in a separate file.
- The publicly available ECC dataset [25,31] consists of power consumption traces collected from a Pinata development board (developed by Riscure). The board is equipped with a 32-bit STM32F4 microcontroller (with an ARM-based architecture), running at the clock frequency of 168 MHz and having Ed25519 implementation of WolfSSL 3.10.2. The target is profiling a single EC scalar multiplication operation with the ephemeral key with the base point of curve Ed25519. The 256-bit scalar/ephemeral key is interpreted as slices of four nibbles. Hence, there are 16 classes/labels in the dataset. The dataset has a similar format as ASCAD. The database consists of two groups of traces; profiling traces and attack traces. Each group further consists of "TRACES" and "LABELS". Each raw trace consists of 1 000 features, representing the nibble information used during the encryption. Profiling and attack traces groups consist of n_p and n_a tuples, with a corresponding label for each trace. In total, there are 6 400 traces, out of which 80/20 are used for profiling ($n_p = 5\,120$) and attacking ($n_a = 1\,280$).

The profiling traces from both datasets are used to train the cGAN models. Half of the real traces per class are kept in the final dataset, along with the generated data samples. The attacking traces are used for evaluating the performance of the DL-SCA model trained with the new dataset.

It should be noted that the purpose of this research is to analyze the effect of artificially generated features in data traces for environments where the adversary has an additional constraint on collecting leakage traces to form a dataset. The presented methodology can be extended to produce and test the fake leakage traces for any other cryptographic algorithms.

3 Related Works

Data augmentation represents a set of techniques to reduce overfitting and improve the supervised machine learning task (commonly, classification). Data augmentation is a well-researched topic, mostly framed in the context of image data augmentation [24]. While there are multiple ways to divide the data augmentation techniques, a common one is on 1) techniques transforming the input, 2) deep learning techniques (where our work also belongs), and 3) meta-learning.

Data augmentation in SCA is used to improve side-channel attack performance and can be put in the same general direction as, e.g., works exploring how to improve hyperparameter tuning. As data augmentation increases the amount of data, it is commonly considered in the deep learning perspective as there, very large datasets are beneficial. There are significantly more works considering symmetric-key cryptography and deep learning-based SCA than public-key cryptography.

The first investigation that uses convolutional neural networks for side-channel attacks on AES is conducted by Maghrebi et al. [1]. This work represents a significant milestone for the SCA community as it demonstrated how deep learning could be used without feature engineering and efficiently break various targets.[2]

Cagli et al. investigated how deep learning could break implementations protected with jitter countermeasures [6]. This work is highly relevant as it introduced data augmentation to the SCA domain. The authors used two data augmentation techniques: shifting (simulating a random delay) and add-remove (simulating a clock jitter). Picek et al. investigated how reliable are machine learning metrics in the context of side-channel analysis. Their results showed that machine learning metrics could not be used as sound indicators of side-channel performance [5]. Additionally, as the authors used the Hamming weight leakage model that results in class imbalance, they utilized a well-known data balancing technique called SMOTE, showing that the attack performance can be significantly improved. Kim et al. explored how to design deep learning architectures capable of breaking different datasets [2]. Additionally, they used Gaussian noise at the input to serve as a regularization factor to prevent overfitting. Luo et al. investigated how mixup data augmentation can improve CPA and deep learning-based side-channel attacks [7].

Next, several works aimed at improving neural network performance by providing a systematic approach for tuning neural network architectures. Zaid et al. were the first to propose a methodology to tune the hyperparameters related to the convolutional neural network size (number of learnable parameters, i.e., weights and biases) [3]. Wouters et al. [32] further improved upon the work from Zaid et al. [3] by showing how to reach similar attack performance with even smaller neural network architectures. Rijsdijk et al. used reinforcement learning to provide an automated way to construct small convolutional neural networks that perform well [33]. Following a different approach to improving the attack performance, Wu and Picek showed how denoising autoencoder could be used to reduce the effect of countermeasures [34]. The first work that considers the usage of GANs (more specifically, cGANs) for the SCA domain is made by Wang et al. [10]. While this work shows the potential of GANs in SCA, the results indicate that a large profiling set is required to construct useful synthetic data.

[2] We note earlier works are also using neural networks like multilayer perceptron, but the results were in line with other machine learning techniques, and researchers commonly used feature engineering to prepare the traces.

There are several works using template attack (and its variants) to attack public-key cryptography, see, e.g., [35–40]. Lerman et al. used a template attack and several machine learning techniques to attack an unprotected RSA implementation [19]. Carbone et al. used deep learning to attack a secure implementation of RSA [41]. The authors showed that deep learning could reach strong performance against secure implementations of RSA. Weissbart et al. showed a deep learning attack on EdDSA using the curve Curve25519 as implemented in WolfSSL, where their results indicate it is possible to break the implementation with a single attack trace [25]. Weissbart et al. considered deep learning-based attacks on elliptic curve Curve25519 implementation protected with countermeasures and showed that even protected implementations could be efficiently broken [42]. Perin et al. used a deep learning approach to remove noise stemming from the wrong choice of labels after a horizontal attack [15]. The authors showed that protected implementations having an accuracy of around 52% after a horizontal attack could reach 100% after deep learning noise removal (note that the authors also used data augmentation to reach 100% accuracy). Zaid et al. introduced a new loss function called ensembling loss, generating an ensemble model that increases the diversity [43]. The authors attacked RSA and ECC secure implementations and showed improved attack performance.

4 Proposed Approach

4.1 Data Splitting

The leakage data traces L are collected from the device while AES or ECC algorithms encryptions E are performed using the secret key K or scalar/ephemeral key K, respectively. The labeled collected traces are then divided into two sets: Training ($D_{Training}$) and Testing ($D_{Testing}$). The training set is used to train the Siamese-cGAN model, which produces fake traces. The newly generated dataset (real+fake traces) is used to train the DL-SCA model, and the test set is used to evaluate the SCA model's performance. For a fair evaluation of the trained Siamese-cGAN model, the test set is never shown to the network during the Siamese-cGAN model training process.

For the rest of the paper, "cGAN models" or "Siamese-cGAN models" refer to the model used for generating data. However, "DL-SCA models" refer to the deep learning models, which are used to evaluate the performance of the generated data by applying profiling side-channel attacks.

4.2 Siamese-cGAN Model for Data Augmentation

In contrast to the standard GANs, conditional GANs (cGANs) perform conditional generation of the fake data based on the class label rather than generating signals blindly. The labels c of the data traces/instances are used to train GANs in a zero-sum or adversarial manner to improve the learning of the generator (G). As mentioned before, the generator's G task is to generate the leakage signals

that carry similar properties as the original traces using the random noise z and the latent space input ls. The discriminator's D task is to distinguish real and fake signals. In the cGAN training process, first, the discriminator D is trained with the labeled real data traces T_{real}, and then the discriminator D is trained with the fake generated signals $G(z)$ or T_{Gen}. The objective function for cGAN is given by Eq. (2).

$$E_{x\sim T_{Real}}[log(D(x|c))] + E_z[log(1 - D(G(z|c)))]. \tag{2}$$

In our proposed design of the Siamese-conditional Generative Adversarial Network ($Siamese - cGAN$), we combine cGAN with the Siamese network concept. Siamese network is an architecture in which two identical/twin networks carrying the same weights are trained with two different inputs. In the proposed model, two generators $G1$ and $G2$ take two random input noise vectors $z1$ and $z2$, generating fake signals $G(z1)$ and $G(z2)$, respectively, in a Siamese fashion. Both the generators share the same network weights, and only the input is different. The discriminator D is first trained with the labeled real data T_{Real} and then with the fake data T_{Gen}, originating from the two twin generator networks $G1$ and $G2$.

As mentioned before, convergence is a challenging issue in training GANs. In some cases, the model converges and then starts diverging again; that is, it forgets its learned examples. Several techniques include memory-based learning, to handle such scenarios [44]. Training the model simultaneously, in a Siamese setting, with the random noise from two sources can help obtain a better-converged model in fewer epochs by combining output from both. Moreover, to analyze the impact of convergence, we introduced another layer in the two-step cGAN model. This layer monitors the real traces loss L_{Real}, generated traces loss L_{Gen}, and GAN model Loss L_{GAN}.

Let $D_{GAN\rightarrow R}$ represent the loss difference between L_{GAN} and L_{Real}, and $DGAN \rightarrow G$ represent the loss difference between L_{GAN} and L_{Gen}, then the average of loss differences over last t iterations will be given by:

$$\frac{1}{t}\sum_{i=1}^{t}(|D_{GAN\rightarrow R(i)}| + |D_{GAN\rightarrow G(i)}|). \tag{3}$$

The Siamese-cGAN model stops training when the average model loss $Loss_{Avg}$ over the last t iteration is less than the average loss over the last $t * 2$ iterations. The trained Siamese-cGAN model is then used to generate the n_g fake traces T_{Gen}, containing features similar to the original signals. T_{Gen} and T_{Real}, having n_g and n_r instances respectively, are combined together to form a resultant dataset T_{GAN}. This dataset is then used to train the deep learning model to analyze the generated dataset's performance with DL-SCA. A test set is set aside for a fair evaluation before adding the generated traces into the training dataset. The test set is never shown to the neural network during training. The proposed Siamese-cGAN specific for DL-SCA is shown in Fig. 1.

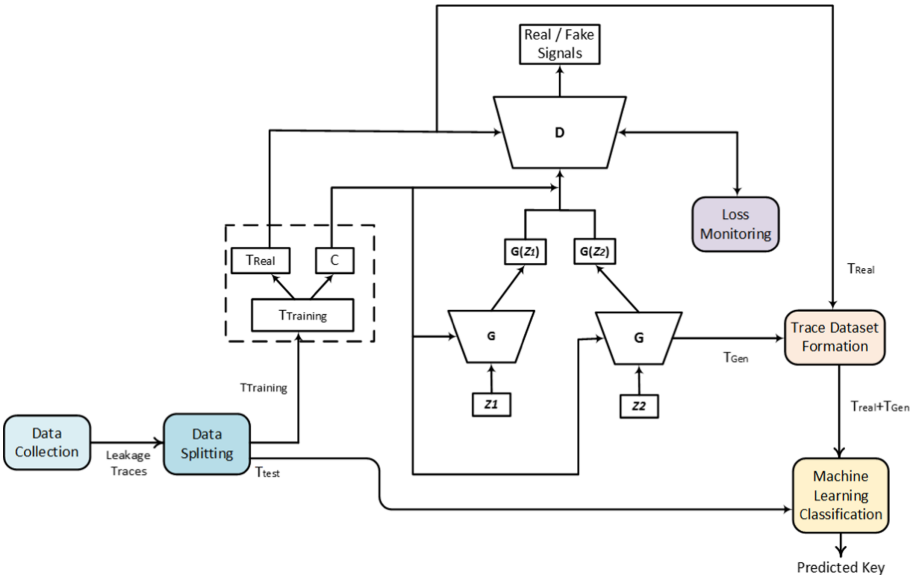

Fig. 1. Proposed Siamese-cGAN architecture for ML-SCA.

4.3 cGAN Models for Discriminator and Generator

In the proposed Siamese-cGAN model, the selection of generator and discriminator plays a vital role. The authors in [10] recommended using a generator and discriminator with fully connected dense layers only, without any complex layers. However, in this study, we explore the possibility of using fully connected dense layers and the convolutional layers in discriminator. The reason for evaluating CNN layer-based network is that it has provided better fake images generation in image processing [20]. Hence, for evaluating the trained Siamese-cGAN model performance, two different networks are used for the generator and discriminator. These networks are denoted as *Model A* and *Model B*. *Model A* is based on two fully connected layers for the generator and the discriminator. However, *Model B* (CNN-based) has a more complex architecture with four fully connected and one convolutional layer. Batch normalization, *LeakyRelu*, and dropout are introduced, which help achieve a more stable model that avoids overfitting. Additionally, we tested both *Model A* and *Model B* generators and discriminators with and without Siamese settings to analyze the improvements introduced by using the Siamese configuration.

Figures 8 and 9, given in Appendix A, show the structure of both the models. We use the same generator and discriminator model architectures to generate symmetric and public-key leakages data samples. The only parameter that needs to be changed is the size of dense layers, which changes based on the number of classes per target dataset. The size in each subsequent layer is doubled from the previous layer. That is, layer 1, layer 2, and layer 3 have a size equal to the

number of classes, the number of classes*2, and the number of classes*4, respectively. The convolutional layer has the *tanh* activation function. The hyperparameter details for the generator network are given in Table 1, while for discriminator, we use a dense layer (512), *LeakyReLU*, and dropout set at 0.4.

Table 1. Generator Architecture Details

Hyper-parameter	Value
Input shape	(700,1) for AES, (1 000,1) for ECC
Fully Connected layer 1	Number of classes
Fully Connected layer 2	Number of classes*2
Fully Connected layer 3	Number of classes*4
Dropout rate	0.4

5 Experiments and Results

5.1 DL-SCA Evaluation Model Architectures

As mentioned in Sect. 2.5, we use MLP and CNN architectures to evaluate the performance of the newly generated datasets using the cGAN model. The state-of-the-art DL-SCA model architectures for evaluating the original publicly available datasets [3,25,27] are used in the same setting except for the batch size and epochs, which are varied between 50–200 to see the impact on the results.

For evaluating the ASCAD dataset, the first architecture is an MLP-based network from [27]. The first CNN architecture is denoted as ASCAD-CNN1 [27], while the second one is denoted as ASCAD-CNN2 [3]. For evaluating the ECC dataset performance, we use the deep learning architecture presented in [25]. We use the existing state-of-the-art DL-SCA model architectures to allow fair comparison. What is more, the goal of this work is not to find new deep learning architectures but to enhance the performance of the existing architectures.

The performance of the DL-SCA trained models with the newly generated datasets is evaluated using two commonly used evaluation metrics: key rank and accuracy. Accuracy represents the number of correctly classified examples divided by the number of examples. Key rank is the position of the correct key guess in the key guessing vector. More precisely, this vector contains the key candidates in decreasing order of probability, and finding the position of the correct key indicates the effort required by the attacker to break the target.

5.2 Experimental Setup

The proposed GAN models have been developed and trained using Keras and Tensorflow libraries for our experiments. The models are trained to generate the fake data on a common computer equipped with 32 Gb RAM, i7-4770 CPU with

3.40 GHz, and Nvidia GTX 1080 Ti. The time required for fake data generation will vary depending on the number of classes and generated fake traces per class. The $Siamese-cGANModels$ training took less than 5 min to train and less than 5 min to generate 150 fake traces for 256 classes in total. For training a GAN model, real data traces are given as an input in the batches of 30 data traces, and the model is trained for 1 000 epochs.

We divide our experimental analysis into two sections.

- Analysis-1: first, in Sect. 5.3, we provide the visual representation of the trained cGAN models over 1 000 epochs. Visual representation helps identify the convergent model, which is then further selected for analysis in the second phase of analysis. We trained four cGAN models, out of which two are existing and two are newly proposed, based on the generator and discriminator as explained in Sect. 4.3. Details of the cGAN models are given in Table 2. We also provide the model convergence-based comparison of our proposed Siamese-cGAN-based model with the existing cGAN models [10].
- Analysis-2: second, in Sect. 5.4, we provide the deep learning-based side-channel analysis of two datasets; dataset containing real leakage signals only and the dataset consisting of both real and fake leakages, by training with two neural networks (MLP and CNN). We also compare generating leakage signals from the non-converging and converging networks for this analysis. To achieve this, we generated fake signals from various points while training the Siamese-cGAN network. More precisely, we generated the signals when the model converged the best and generated the signals when the model was the least convergent (initial epochs). Converging details of each model are given in the respective sections. This comparison is performed to highlight that not any cGAN can be selected blindly. *Only a well-convergent network will generate traces that are more alike in characteristics to that of original real traces.*

Table 2. cGAN Model Details

Model Name	Description
$cGAN\ Model\ A$	Model without CNN and Siamese network
$cGAN\ Model\ B$	Model with CNN but without Siamese network
$Siamese-cGAN\ Model\ A$	Model without CNN but with Siamese network
$Siamese-cGAN\ Model\ B$	Model with CNN and with Siamese network

5.3 Analysis-1: Existing and Proposed GAN-based Approaches

This section presents a convergence-based comparative analysis of our approach with the existing cGAN models. The existing cGAN networks (without layered Siamese-cGAN setting) are denoted as $CGAN\ Model\ A/B$, whereas our

proposed models are denoted as $Siamese - cGAN\ Model\ A/B$. For this analysis, all four cGAN networks are trained with the $D_{Training}$ dataset and the GAN model loss for both AES and ECC, as shown in Figs. 2 and 3, respectively. Figure 2 *a* and *b* presents the real, fake, and GAN loss for training with $cGAN\ Model\ A$ and $cGAN\ Model\ B$ without Siamese settings, respectively. Next, Fig. 2 *c* and *d* presents the real, fake, and GAN loss for training with $Siamese - cGAN\ Model\ A$ and $Siamese - cGAN\ Model\ B$ with the Siamese setting. Similarly, Fig. 3 presents all four cGAN training models' loss for the ECC dataset.

$Siamese - cGAN$ and $cGAN$ models, for the ASCAD and ECC dataset analysis, are trained for 1 000 epochs, and history is recorded every ten epoch. Hence x-axis is scaled by 10. It can be seen that the proposed $Siamese - cGAN\ Model\ B$ architecture (based on CNN) provides the best loss convergence of real, fake, and GAN models as the models converge around 100–150 epochs and 700–1 000 epochs for AES and ECC datasets, respectively.

This shows that the generator started generating traces similar to the real traces at this convergence point, making it harder for the discriminator to discriminate between real and fake. Existing $cGAN\ Model\ A$ and $cGAN\ Model\ B$ without Siamese configuration did not converge well in 1 000 epochs. Moreover, for $Siamese - cGAN\ Model\ B$, GAN loss is high and quickly decreases in initial epochs. Hence, the proposed $Siamese - cGAN\ Model\ B$ is the robust solution for generating artificial/fake leakage signals for both algorithms as it converges better and faster than other cGAN models. Interestingly, $Siamese - cGAN\ Model\ A$ performs relatively poorly, indicating that the more powerful neural network architecture was required for this task. In conclusion, from this analysis, $Siamese - cGAN\ Model\ B$ is further selected to generate fake data in analysis phase 2.

5.4 Analysis-2: Analysis of the Proposed Siamese-CGAN for DL-SCA

Based on the results from the previous analysis, $Siamese - cGAN\ Model\ B$ is selected for further experiments in this section. Now, we perform DL-SCA (using MLP, ASCAD-CNN1, and ASCAD-CNN2) on the newly generated T_{GAN} (real+fake traces) datasets, generated using $Siamese - cGAN\ Model\ B$. However, we also analyzed the real dataset (with reduced traces per class). Hence, two datasets are formed: one with real traces only and the other (T_{GAN}) with both real and fake traces. For T_{GAN} datasets, two further datasets are formed, one for the fake data generated from the well-converged model and the second from the non-convergent model. Finally, we analyze how all these different settings impact the key rank of a side-channel attack.

Analysis on Real Traces. In our experiments, we reduced the size of the ASCAD and ECC leakage datasets intentionally to analyze the effect of the artificially generated traces on the small-size datasets. We selected $n_r = 150$ (for

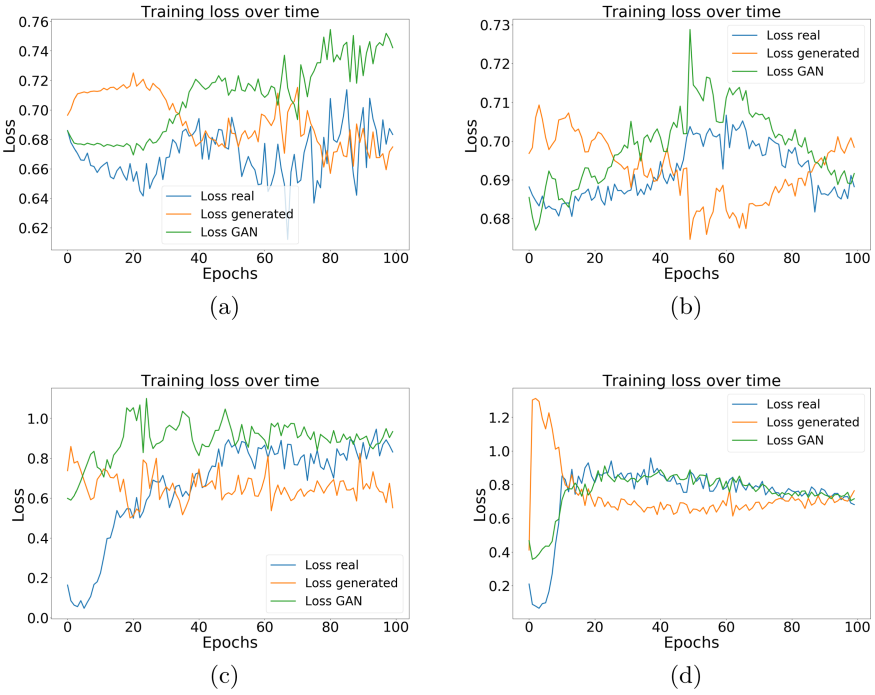

Fig. 2. CGAN Model Training Loss for the ASCAD dataset (a) *cGAN Model A*, (b) *cGAN Model B*, (c) *Siamese – cGAN Model A*, (d) *Siamese – cGAN Model B*

each class) leakage traces from the ASCAD and ECC datasets. The reason for selecting precisely 150 traces per class is because we wanted an equal number of real traces for all the classes. In the ASCAD dataset, class 213 has a minimum number of traces (154 traces); hence 150 is selected. No artificial/fake traces are included in the training dataset, so $n_g = 0$. Hence, total number of traces in AES and ECC datasets are 38 400 (150 traces $\times$ 256 classes) and 24 00 (150 traces $\times$ 16 classes), respectively. For deep learning-based attacks, we used the previously successful DL-SCA models for AES and ECC in respective studies [3,25,27].

The purpose of using the same deep learning-based models is to show that the artificially generated traces produce the same results as the real traces with the same model architectures. When considering the ASCAD-CNN1 architecture, everything stays the same as in previous studies' analysis except that we perform normalization on the training and test data. For normalization, each input variable feature is scaled in the range $[-1, 1]$ by using MinMaxScaler from the Sklearn library. For MLP, 10-fold cross-validation is performed. For ASCAD-CNN2 analysis, in addition to applying normalization, a standardization is added as per the proposed architecture in [3], and data is standardized around mean with a unit standard deviation (between 0 and 1) [45,46]. For ECC, the same model is used for training and test as proposed in [25].

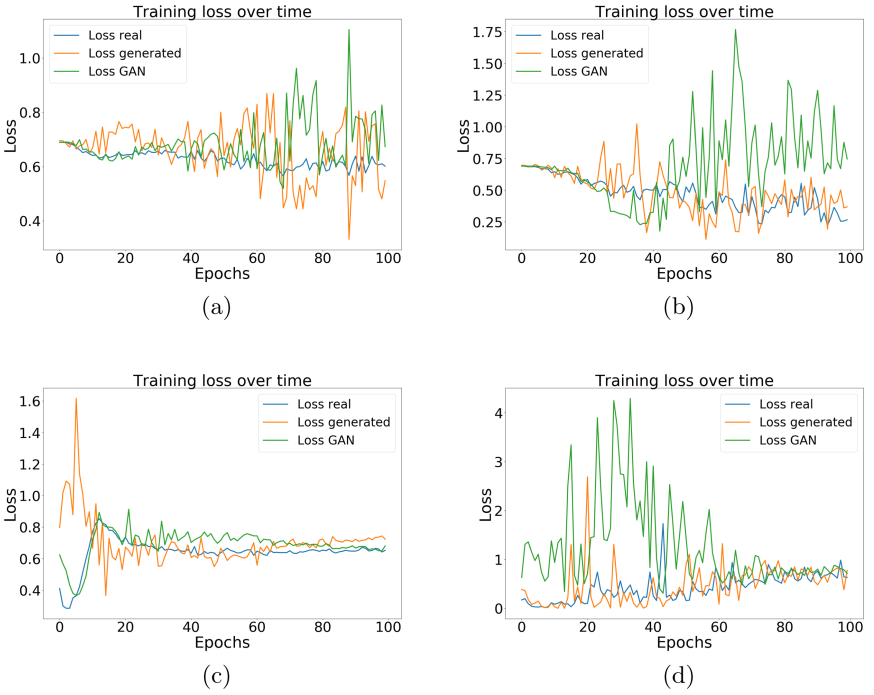

Fig. 3. cGAN Model Training Loss for the ECC dataset (a) *cGAN Model A*, (b) *cGAN Model B*, (c) *Siamese − cGAN Model A*, (d) *Siamese − cGAN Model B*

Figure 4 (a) shows the key rank for the real traces (38 400) dataset for the ASCAD dataset analysis using MLP and 10-fold cross-validation. We compare our results of reduced, original traces with the results of MLP_{best} reported in [27], which is plotted for the trained model on 50 000 traces. We can see a slight deviation though both figures are for the trained model on real traces. Figure 4 (b) shows the key rank on reduced ASCAD dataset trained using ASCAD-CNN2. It shows key rank not approaching zero in the first 1 000 traces. This confirms that the reduced dataset did not perform as expected with the existing models.

Figure 5 shows the accuracy for the real traces dataset for ECC dataset analysis using CNN architecture [3]. Raw, real data traces analysis for ECC shows that the private key can be recovered with 100% accuracy using CNN. It should be noted that preprocessing and alignment have not been applied to these datasets.

Analysis on Real and Generated Traces Dataset with the Maximum and Minimum Convergence. We introduce the terms maximum and minimum convergence for our analysis. Maximum convergence refers to the point (epochs) when a stable GAN model is achieved. Minimum convergence simply refers to the epochs when the GAN model is not stable, mostly in start epochs and often towards the end epochs as well. In certain failure modes or mode

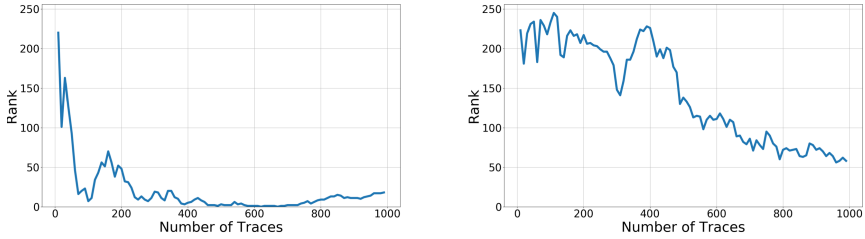

Fig. 4. Results for (a) Key rank for the ASCAD dataset having 38 400 profiling traces trained using MLP, and (b) Key rank for the ASCAD dataset having 38 400 profiling traces trained using ASCAD-CNN2

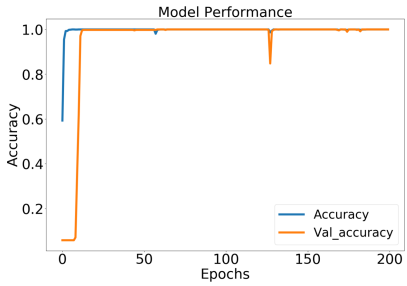

Fig. 5. Results for training and validation accuracy for the ECC dataset having 2 400 profiling traces

collapse scenarios, the GAN model stabilizes initially and might become unstable after a few epochs when the generator trains itself that it is hard to distinguish between the traces of different classes (meaning traces for all the classes look similar). The purpose of using these two types of analysis is to demonstrate that, in contrast to the non-convergent GAN model, the traces generated with the well convergent GAN model only produces traces similar in characteristics to the traces of the same class but different from the traces of the other classes.

For the GAN analysis, the training dataset consists of an equal proportion of the real traces and the artificially generated fake traces, that is, $n_r = 150$ and $n_g = 150$ per class, which means that in total, $300 \times 256 = 76\,800$ traces are in the dataset. Fake leakage signals are generated for the epochs during which the cGAN-Siamese model achieves maximum convergence. For the minimum convergence analysis, we combined the real traces with the artificially generated traces in equal proportion, the same as the maximum convergence case. However, traces are collected for the epochs during which the GAN model showed the minimum convergence.

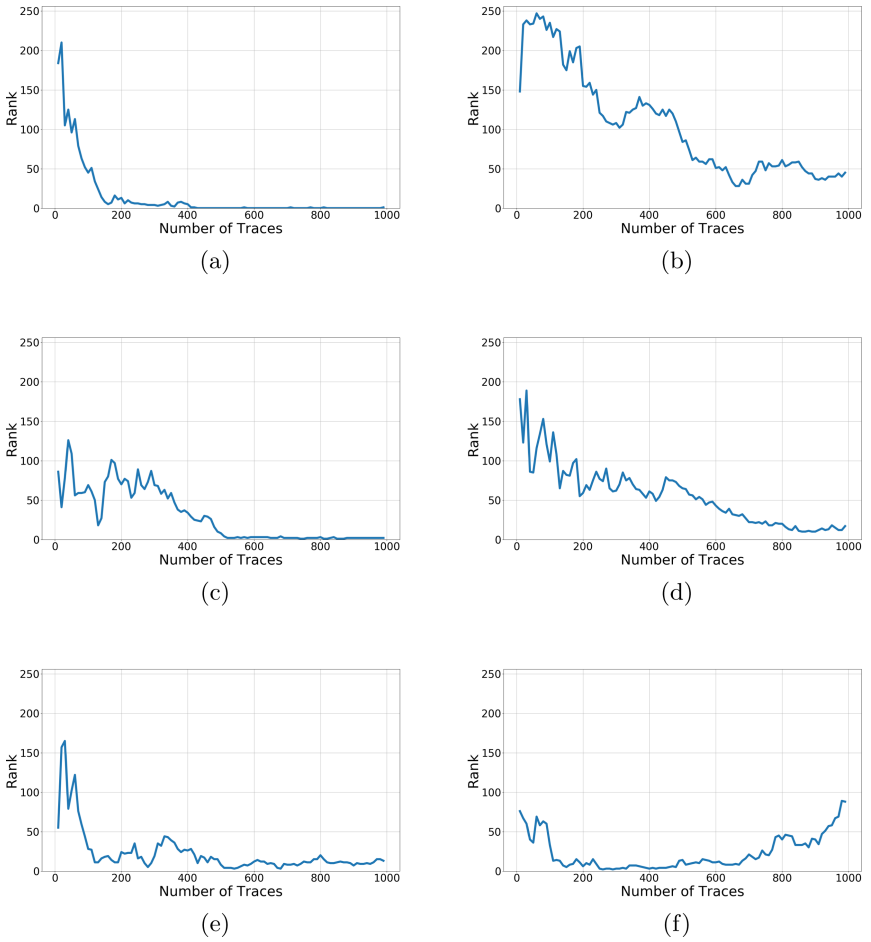

Fig. 6. Key rank for the ASCAD dataset for (a) Maximum convergence $cGAN - Siamese\ Model\ B$ using MLP, (b) Minimum convergence $cGAN - Siamese\ Model\ B$ using MLP, (c) Maximum convergence $cGAN - Siamese\ Model\ B$ using ASCAD-CNN1, (d) Minimum convergence $cGAN - Siamese\ Model\ B$ using ASCAD-CNN1, (e) Maximum convergence $cGAN - Siamese\ Model\ B$ using ASCAD-CNN2, (f) Minimum convergence $cGAN - Siamese\ Model\ B$ using ASCAD-CNN2

Figure 6 shows the key rank for both maximum and minimum convergence for all three DL-SCA models and the ASCAD dataset. We notice that generating fake traces from the maximum convergence point significantly impacts key rank. The maximum convergence is achieved around 100–150 epochs for $Siamese - cGAN\ Model\ B$. Hence, data traces are generated around those epochs for analysis. It is observed that with the generated traces, all models (MLP, ASCAD-CNN1, and ASCAD-CNN2) gave the best performance, and the secret key can be

obtained efficiently. *Thus, we can conclude that the artificially generated traces contain significant information that improved the ML-SCA performance.*

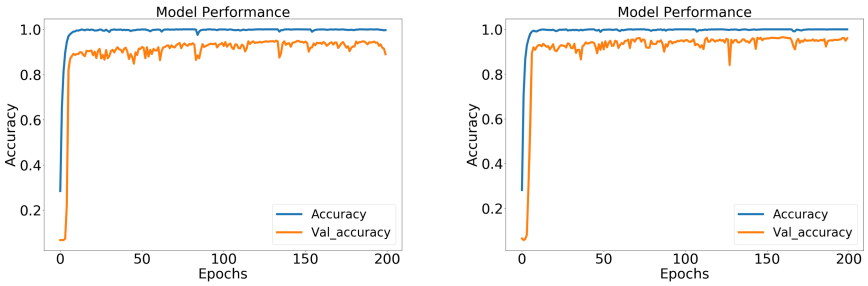

Fig. 7. Training and validation accuracy on the ECC dataset collected from (a) Maximum convergence point, (b) Minimum convergence point

The minimum convergence is observed around initial epochs, so artificial 150 traces per class are generated around this point and are combined with the real 150 traces to train the DL-SCA model. Observe that with minimum convergence, the DL-SCA attack model shows key rank is not stable, as it reaches zero in certain cases and starts increasing again as can be seen from Figs. 6*b* and 6*f*. However, for Fig. 6*d* (trained with ASCAD-CNN1), it appears to reach a key rank of zero near 1 000 traces, so more investigation is required to assess this case properly.

Figure 3 shows the GAN convergence curve for the ECC dataset. The model trained with the proposed $Siamese - cGAN\ Model\ B$ shows a better convergence than the other three cGAN models' losses. The traces with the maximum convergence analysis are generated around epoch 700–1 000 (70–100 scaled in Fig. 3), and traces for the minimum convergence are generated around 20–30 epochs. The performance accuracy is high after adding artificial traces. The trained model with the artificial traces generated with the convergent model (Fig. 7a) shows accuracy greater than 97% using CNN, which is nearly the same accuracy as achieved on the real traces. However, the trained model with artificial traces, with the least convergent model, shows around 90% accuracy. While this performance is still good, we note that it cannot be compared with the performance on the real dataset. *Hence, the maximum convergent model generates the artificial traces that are more alike in characteristics to the real leakage traces and helps in training an efficient model for profiling side-channel analysis.*

5.5 Discussion

Based on the conducted experiments, we draw some general observations:

- GANs (more precisely, conditional GANs) represent a viable option for constructing synthetic side-channel traces. To improve the performance of GANs, it is beneficial to use deeper architectures and convolutional layers.
- A combination of a Siamese network and a cGAN can further improve the quality of the obtained synthetic examples.
- The procedure of generating fake traces is efficient and can generate hundreds of traces in a matter of minutes.
- It is important to monitor the GAN loss carefully and use the model that minimizes it when crafting synthetic examples.
- The combination of fake and real traces performs well regardless of the applied deep learning-based model. What is more, we see that fake traces can improve attack performance.
- It is possible to construct synthetic examples for various cryptographic implementations with similar success, i.e., this technique is not limited to a specific cryptographic implementation.

6 Conclusions and Future Work

A dataset of leakage traces with insufficient traces can pose a significant problem for accurate attack modeling using deep learning-based side-channel analysis. Data augmentation using a Generative Adversarial Network (GAN) can be useful for such scenarios. This work proposed a layered architecture (Siamese-cGAN) based on cGAN and Siamese network that presents a well-convergent model to generate artificial traces similar to real traces. We performed two sets of analyses. In the first set of analyses, we run the experiments and present a visual comparative analysis between the performance of the proposed model and the existing cGAN based models for leakage signal generation. For this analysis, two neural network-based models (MLP and CNN) have been used for modeling the generator and the discriminator networks. The best model is selected based on the comparative analysis. The second set of analyses evaluated the generated fake dataset by applying the DL-SCA on the leakage datasets from the exiting AES and ECC algorithm implementations. Four state-of-the-art neural network architectures (one MLP and three CNNs) are used for this evaluation. We also provided a comparative analysis of the dataset's performance consisting of data generated from the well-convergent network and data generated from the non-convergent network.

The proposed Siamese-cGAN model performed better than the existing simple cGAN models for both existing symmetric and public-key datasets. The quantitative analysis results show that the well-converged Siamese-cGAN network produces fake leakage traces similar to the collected traces. Hence, they enable a better deep learning-based model for side-channel attacks. We also observed that the CNN-trained models performed better than MLP for the key

recovery. We conclude that leakage traces/instances with significant contributing features can be efficiently generated. However, selecting a fully converging model might vary for each cryptographic algorithm.

As future work, we plan to explore the limits of our approach from the perspective of the number of synthetic traces. Indeed, while our results indicate that 150 traces per class are more than sufficient to construct convincing synthetic data, understanding the minimum required number of traces would allow a proper evaluation of the method's viability. Furthermore, we used only the intermediate value leakage model, which results in more classes and balanced measurements per class. We plan to evaluate different leakage models like the Hamming weight model, resulting in imbalanced data and fewer classes. Finally, it would be interesting to explore if the GAN-based approach could generate measurements that would help reduce the effect of portability for deep learning-based SCA [47].

A Appendix

A.1 Siamese-cGAN Model Architectures

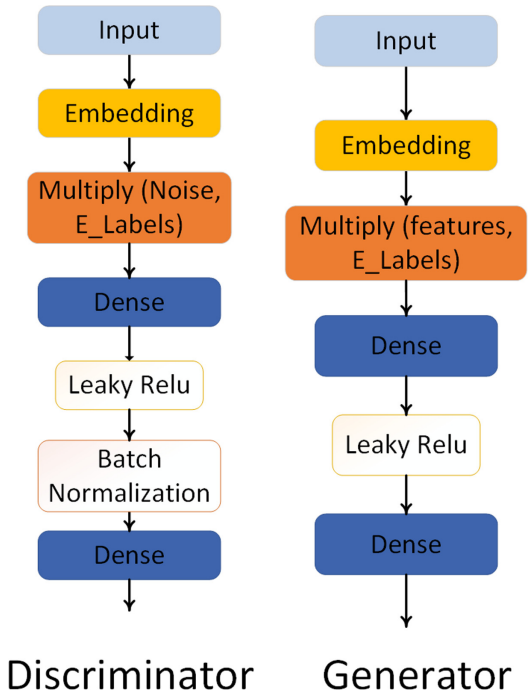

Fig. 8. *Siamese – cGANModel A*

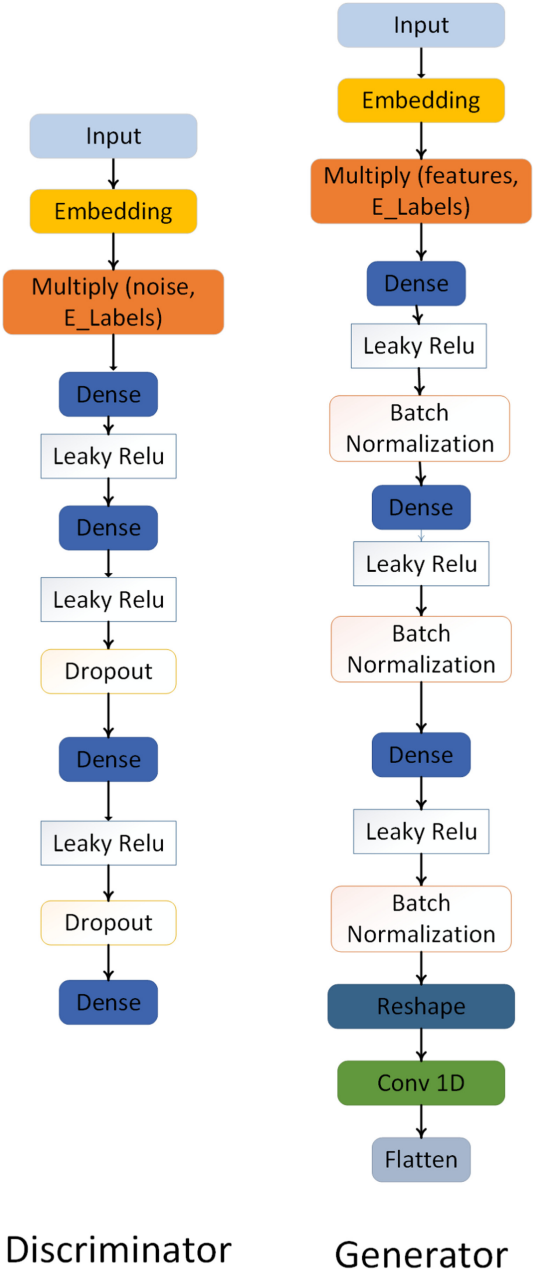

Fig. 9. *Siamese – cGANModel B*

References

1. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) SPACE 2016. LNCS, vol. 10076, pp. 3–26. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49445-6_1
2. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**, 148–179 (2019)
3. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 1–36 (2019)
4. Picek, S., Heuser, A., Guilley, S.: Profiling side-channel analysis in the restricted attacker framework. IACR Cryptology ePrint Archive **2019**, 168 (2019)
5. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(1), 209–237 (2018)
6. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 45–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_3
7. Luo, Z., Zheng, M., Wang, P., Jin, M., Zhang, J., Hu, H.: Towards strengthening deep learning-based side channel attacks with mixup. Cryptology ePrint Archive, Report 2021/312 (2021). https://eprint.iacr.org/2021/312
8. Liu, M.-Y., Huang, X., Yu, J., Wang, T.-C., Mallya, A.: Generative adversarial networks for image and video synthesis: algorithms and applications. CoRR, abs/2008.02793 (2020)
9. Goodfellow, I.J., et al.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS 2014, pp. 2672–2680. MIT Press, Cambridge (2014)
10. Wang, P., et al.: Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks (2020)
11. Kodali, N., Abernethy, J., Hays, J., Kira, Z.: On convergence and stability of GANs (2017)
12. Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. CoRR, abs/1606.03498 (2016)
13. Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR, abs/1411.1784 (2014)
14. Hsu, C.-C., Lin, C.-W., Su, W.-T., Cheung, G.: SiGAN: siamese generative adversarial network for identity-preserving face hallucination. CoRR, abs/1807.08370 (2018)
15. Perin, G., Chmielewski, L., Batina, L., Picek, S.: Keep it unsupervised: horizontal attacks meet deep learning. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(1), 343–372 (2021)
16. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
17. Whitnall, C., Oswald, E., Standaert, F.-X.: The myth of generic DPA...and the magic of learning. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 183–205. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_10

18. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
19. Lerman, L., Bontempi, G., Markowitch, O.: Power analysis attack: an approach based on machine learning. Int. J. Appl. Cryptol. **3**(2), 97–115 (2014)
20. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016, Conference Track Proceedings (2016)
21. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 29, pp. 2172–2180. Curran Associates Inc. (2016)
22. Zhang, H., et al.: StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5908–5916 (2017)
23. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. CoRR, abs/1809.11096 (2018)
24. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. J. Big Data **6**(1), 60 (2019)
25. Weissbart, L., Picek, S., Batina, L.: One trace is all it takes: machine learning-based side-channel attack on EdDSA. In: Bhasin, S., Mendelson, A., Nandi, M. (eds.) SPACE 2019. LNCS, vol. 11947, pp. 86–105. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35869-3_8
26. Mukhtar, N., Mehrabi, A., Kong, Y., Anjum, A.: Machine-learning-based side-channel evaluation of elliptic-curve cryptographic FPGA processor. Appl. Sci. **9**, 64 (2018)
27. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptogr. Eng. **10**(2), 163–188 (2019). https://doi.org/10.1007/s13389-019-00220-8
28. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition (2015)
29. Leyva-Vallina, M., Strisciuglio, N., Petkov, N.: Generalized contrastive optimization of Siamese networks for place recognition. CoRR, abs/2103.06638 (2021)
30. Chicco, D.: Siamese neural networks: an overview. In: Cartwright, H. (ed.) Artificial Neural Networks. MMB, vol. 2190, pp. 73–94. Springer, New York (2021). https://doi.org/10.1007/978-1-0716-0826-5_3
31. Database for EdDSA (2019). https://github.com/leoweissbart/MachineLearningBasedSideChannelAttackonEdDSA
32. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient CNN architectures in profiling attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(3), 147–168 (2020)
33. Rijsdijk, J., Lichao, W., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 677–707 (2021)
34. Lichao, W., Picek, S.: Remove some noise: on pre-processing of side-channel measurements with autoencoders. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(4), 389–415 (2020)

35. Medwed, M., Oswald, E.: Template attacks on ECDSA. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 14–27. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00306-6_2
36. Heyszl, J., Mangard, S., Heinz, B., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of cryptographic implementations. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 231–244. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_15
37. Batina, L., Chmielewski, Ł., Papachristodoulou, L., Schwabe, P., Tunstall, M.: Online template attacks. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 21–36. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13039-2_2
38. Batina, L., Chmielewski, L., Papachristodoulou, L., Schwabe, P., Tunstall, M.: Online template attacks. J. Cryptogr. Eng. **9**(1), 21–36 (2019)
39. Özgen, E., Papachristodoulou, L., Batina, L.: Classification algorithms for template matching. In: IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA (2016)
40. Roelofs, N., Samwel, N., Batina, L., Daemen, J.: Online template attack on ECDSA: In: Nitaj, A., Youssef, A. (eds.) AFRICACRYPT 2020. LNCS, vol. 12174, pp. 323–336. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51938-4_16
41. Carbone, M., et al.: Deep learning to evaluate secure RSA implementations. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(2), 132–161 (2019)
42. Weissbart, L., Chmielewski, Ł., Picek, S., Batina, L.: Systematic side-channel analysis of curve25519 with machine learning. J. Hardware Syst. Secur. **4**(4), 314–328 (2020)
43. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Efficiency through diversity in ensemble models applied to side-channel attacks: - a case study on public-key algorithms -. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 60–96 (2021)
44. Wu, C., Herranz, L., Liu, X., Wang, Y., van de Weijer, J., Raducanu, B.: Memory replay GANs: learning to generate images from new categories without forgetting (2019)
45. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
46. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient BackProp. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 9–48. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_3
47. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., Shrivastwa, R.R.: Mind the portability: a warriors guide through realistic profiled side-channel analysis. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, 23–26 February 2020. The Internet Society (2020)