One Network to rule them all. An autoencoder approach to encode datasets

Cristian-Alexandru Botocan[1]
[1]EEMCS, Delft University of Technology, The Netherlands
c.a.botocan@student.tudelft.nl

Abstract

Side-channel attacks are powerful non-invasive attacks on cryptographic algorithms. Among such attacks, profiling attacks have a prominent place as they assume an attacker with access to a copy of the device under attack. The attacker uses the device's copy to learn as much as possible about the device and then mount the attack on the target device. In the last few years, Machine Learning has been successfully used in profiling attacks, as such techniques proved to be capable of breaking implementations protected with countermeasures.

In the deep learning-based profiling attack, a core problem is finding efficient neural network architectures to evaluate an implementation's security correctly. Unfortunately, this process is time-consuming, and a different neural network configuration usually needs to be defined for every target. Hence, we propose the following process: train a separate autoencoder for each dataset obtained from different cryptographic implementations and devices to receive an encoded version for each one. After that, define a universal model that can break multiple (encoded) datasets. Thus, instead of finding dataset-specific neural network architectures, we reduce the effort to find autoencoders to encode the datasets and a single neural network to break them.

1 Introduction

The profiling side-channel attacks field represents an important branch of the applied cryptography domain. During this attack, it is assumed that an attacker has access to a copy of the device under attack. Additionally, he utilizes it to gather as much information as possible about the device. After profiling, the attacker use the gained information to launch the attack on a target device. Based on this approach, profiling attacks are considered one of the most powerful tools for breaking a cryptographic algorithm [3].

In the last period in applied cryptography, much research focused on improving profiling side-channel attacks' performance using different methods. For instance, it was observed that different Machine Learning (ML) techniques could be used in this domain. Their performance could not be neglected, especially when Deep Learning (DL) techniques are applied. What happens behind the scene is that firstly, the traces obtained during the side-channel experiments are added to a database such that an ML-based model can further process this information. For this procedure, the scientists use an oscilloscope to perform the differential power or electromagnetic analysis on the device that will be attacked. Recently, many papers presented various DL architectures that can perform well on a specific dataset gained from a single cryptographic algorithm. Since we need to find the best DL configurations every time we want to attack a cryptographic algorithm, serious concerns can be raised: What happens if the number of cryptographic algorithms and their implementation still increase and for each one, we have every time to find a DL architecture which is suitable for achieving good performance during the attacking process? How much time will be spent testing all the ML configurations for a single cryptographic algorithm?

The aforementioned questions are very important for providing powerful profiling attacks for most of the used cryptographic algorithms. For example, in the study by Picek et al. [7], it is mentioned that starting from the first issued works in 2016 exemplifies how convolutional neural networks (CNNs) can be used in this domain [5], the number of public datasets has been increasing every year. Thus, at some point, the idea of creating a specific ML-based model for each dataset will no longer be feasible - due to the large number of datasets which will continuously appear and the time required to perform this kind of investigation for a single ML algorithm.

As a solution, we propose including the Deep Fake concepts into the SCA domain, such that, for each dataset, we create a compressed version of it. In this manner, we use Autoencoders (AEs), which can encode the initial dataset of traces into a smaller one - a dataset that contains a smaller amount of features and thus reduces the initially required storage for the original traces. Moreover, we try to find a universal DL-SCA model such that it can obtain good performance regardless of the type and implementation of the cryptographic algorithm used in the attack process. As a remark, the universal DL-SCA model attacks the compressed version of the initial dataset. In summary, we try to find AEs that can transform the original datasets into universal ones - all the datasets have the same amount of features. Then, a single ML algorithm use these for the attacking part.

As an observation, it is important to mention that the new stated process helps us create compressed versions of the traces so that they are100% synthetic. Additionally, we do not modify the number of traces; we only change the number of features describing a trace.

To be more concrete, we set up beforehand a value Y which represents the number of features for the "universal" dataset, and later, given a dataset that has X traces and each trace is described by Z features, after the compression part, the new version of the same dataset will contain X traces, and Y features will describe each trace. After we have the

compressed versions of all datasets, we use them to train the DL-SCA models and then analyze their performance.

2 Related Works

Regarding the SCA part, we can observe a series of works that presented different DL models capable of obtaining good performances. One of them is the study by Benadjila et al. [2], where an entire process of obtaining MLP and CNNs architectures was conducted. Since this paper presented the layers in very small details (the parameters, number of layers, etc.), we use the best 2 DL models to see how they behave in our environment. More details are provided in Section 3. Additionally, the research by Wouters et al. [10] proposed a new approach to creating the architecture of the DL-SCA models. Since we also want to run new DL models, particularly ones that do not use a first convolutional layer and still behave as a CNN, we also use one model from this paper.

Furthermore, using the AEs for compressing some data is not new, and it was applied with success in other domains, especially in Imaging Science [1; 4; 8; 9]. Even if, in most cases, the images were encoded as 3D or 2D vectors, in work by Kuester et al. citekuester20211d the perspective of the used dataset is similar to SCA - the dataset contained instances that have only one dimension. The same situation applies to the Side-Channel traces. Thus, even though the study introduced a 1D-CAE (Convolutional AE) analysis, we think it is still an important sign that the compression using AEs can perform well on a 1D *images*. Thus, we can not neglect that applying this idea of compression in SCA represents a novelty.

In terms of creating synthetic traces from the original dataset using supervised learning, the research by Mukhtar et al. [6] used Generative Adversarial Networks (GANs) for generating synthetic traces which were appended to the original ones, focusing on data augmentation. In this manner, the new dataset became larger by containing 50% original traces and 50% synthetical ones, and afterward, a DL-SCA model was used to attack the obtained dataset. Hence, we can observe a big difference since we create compressed versions of the datasets that contain 100% synthetical traces generated by the Encoder part of the AE, which can be considered a significant element of novelty.

3 Experimental Analysis

In this section, we present the experiment design. We categorize it into five groups: datasets, configuration, results, create universal dataset formats and discussion.

3.1 Datasets

For the experiments conducted in this paper, we consider several open-source datasets. They all consist of software implementations of AES128.

ASCAD fixed key This dataset contains 60 000 side-channel traces collected from an ATMega8515 device that runs a masked AES128 implementation [2]. For this dataset, a fixed key was used for all measurements. Side-channel measurements from this dataset represent the power consumption of the first AES encryption round. We consider the trimmed version of the dataset, which contains 700 features representing the processing of the Sbox operation related to the third key byte (which is also the first masked byte). For that, the dataset is split into 50 000 traces used for the profiling phase, while the other 10 000 are used for the attacking part. For simplicity, we use the `ASCAD_fixed` notation for referring to this dataset.

ASCAD fixed key with desynchronization This dataset contains the same side-channel traces from `ASCAD_fixed` dataset, and they are artificially and randomly desynchronized. We consider the two versions of desynchronized datasets. The first contains trace desynchronization with a maximum window of 50 features. For the second desynchronized dataset, the maximum desynchronization is made for up to 100 features. For simplicity, we use the `ASCAD_fixed_desync50` and `ASCAD_fixed_desync100` notations for referring to these two datasets.

ASCAD variable key This dataset contains side-channel traces collected from the same masked AES128 of `ASCAD_fixed` dataset. The dataset consists of 300 000 measurements, where 200 000 traces contain a random key, which is used as profiling traces, and the remaining 100 000 traces were measured with a fixed key, which are considered for the attacking set. We again target the third key byte as this is the first masked byte. The target trace interval represents the processing of the third Sbox operation in the first encryption round and contains 1400 features. For simplicity, we use the `ASCAD_variable` notation for referring to this dataset.

CHESCTF This dataset was released as part of the Capture-The-Flag competition from Cryptographic Hardware and Embedded Systems (CHES) 2018. It consists of power side-channel measurements from four different STM32 micro-controller devices running a masked AES128 implementation. From each device, 10 000 measurements were collected. Thus, our profiling set consists of 30 000 traces, while the other 10 000 are used for the attacking part. The original measurements contain 650 000 features, and here we extract a trimmed version of the resampled dataset (with a window of 10) composed of 4 000 features. This interval is a concatenation of two main parts of the measurements: the processing of mask share at the beginning of the measurements and the processing of the Sbox operation during the first encryption round. For simplicity, we use the notation `CHESCTF` for referring to this dataset.

DPAV42 This dataset consists of power measurements obtained from an 8-bit microcontroller running a masked AES128 implementation. In total, 80 000 traces were collected, and a different key was used for each 5 000 measurements. For that, the dataset is split into 70 000 traces used for the profiling phase, while the other 5 000 are used for the attacking part, all of them generated using a window with the dimension of 800 features. Before extracting a trimmed interval, the dataset was resampled with a window of 10, and the 800 features resulted in a concatenation of two 400 feature intervals. These two intervals represent the processing of two secret shares, which allows us to create a second-order profiling attack. For simplicity, we use the notation `DPAV42` for referring to this dataset.

3.2 Setting

For the research, we create the following setup. We use the supervised learning technique to search for the most suitable AE model. Using a non-normalized version of the data, we explore two types of AEs - Vanilla AE and Convolutional

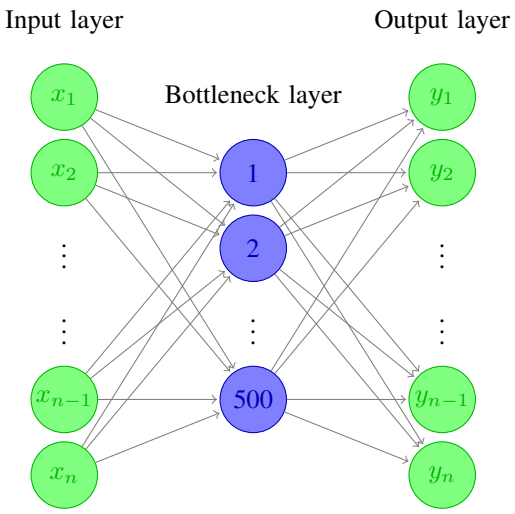

Figure 1: Vanilla AE architecture

Table 1: Hyperparameters space for the AE architecure

Hyperparameters	Training values
Optimizer	Adamax
Epochs	500, 700, 1000
Learning_rate	0.01, 0.001, 0.0025
Batch_size	200, 400
Shuffle	True

AE. Since for the second type of model, we find it remarkably sensible to the input data, and it can not perform well on all the datasets, in this work, we focus on the first type of AE, which can also be visualized in Figure 1. Additionally, we use a grid-search technique to find the best hyperparameters for this architecture. Table 1 offers an overview of the hyperparameter values used for finding the best Vanilla AE configuration.

After the grid-search process, we select the top-5 best MSE scores computed on an attacking dataset - we rank the AE configurations' performances based on how accurately they can reproduce the attacking version of the dataset. Then, for each top-5 best configuration, we take the **Encoder** part of the AE and use it to create a compressed version of the initial dataset. For our study, it is important to mention that an arbitrary dimension of 500 features is chosen as the bottleneck of AE. However, more research can be done to find additional good dimensions which can be used for that compression process. Therefore, we convert all trace sets into a universal format dataset containing 500 features.

Once we have the compressed version of each dataset, we apply the z-score normalization and proceed with the profiling and attacking phases. For our research, we use four different DL-SCA models:

1. `MLP Basic` - it represents a simple MLP with 4 Dense layers, each one containing 200 neurons. The parameters for the training and testing part are: **batch_size = 400**, **epochs = 10** and the **optimizer = Adam** with a **learning_rate = 0.001**.
2. `MLP Best` - this architecture performed well on the ASCAD datasets during the foregoing studies [2]. Along with the model, we reutilize from the paper the parameters' values for which the best performance was obtained: **batch_size = 200**, **epochs = 100** and the **optimizer = RMSprop** with a **learning_rate = 0.0001**.

Table 2: The # traces used for each phase for different datasets

Dataset	# traces for: profiling	validation	attacking
`ASCAD_fixed`	50 000	3 000	3 000
`ASCAD_variable`	200 000	3 000	3 000
`CHESCTF`	30 000	3 000	3 000
`DPAV42`	50 000	1 000	1 000

3. `CNN Best` - in the same manner as the previous architecture, it was used with good success in previous papers [2]. The parameters for the training and testing part are: **batch_size = 200**, **epochs = 75** and the **optimizer = RMSprop** with a **learning_rate = 0.0001** - parameters used for obtaining the best performance in the paper [2].
4. `No Conv` - this model was used in the study by Wouters et al. [10], with the purpose of getting as good performance as a CNN, but without using a first convolution layer, which is well time-consuming, especially when a lot of experiments have to be done. The parameters for the training and testing part are: **batch_size = 50**, **epochs = 50**, and the **optimizer = RMSprop** with a **learning_rate = 0.005**, as used for obtaining the best performance in [10].

3.3 Results on Original Dataset

To see the improvements in the attacks on the chosen datasets, we need to see the score if the compression operations are not used. For that, we use the DL-SCA models presented previously, and we use the z-score normalized version of the original dataset. The rest of the paper refers to these results as baseline scores.

As a metric for evaluating the attack, we use **GE - Guessing Entropy** and the **SR - Success Rate**. As an explanation, the SR has values between [0, 1] while the GE has values greater than 1.0. Additionally, to declare that an attack is successful, we must have the GE and SR = 1.0. Even though the GE and SR $\neq$ 1.0, we still consider improving the attack if the NT - number of traces used for the attacking becomes smaller than the maximum set value. Hence, depending on the dataset, we establish upper bounds, shown by Table 2.

To see the improvements in the attacks on the chosen datasets, we need to see the score if the compression operations are not used. For that, we use the DL-SCA models presented previously, and we use the z-score normalized version of the original dataset. The rest of the paper refers to these results as baseline scores.

As a metric for evaluating the attack, we use **GE - Guessing Entropy** and the **SR - Success Rate**. As an explanation, the SR has values between [0, 1] while the GE has values greater than 1.0. Additionally, to declare that an attack is successful, we must have the GE and SR = 1.0. Even though the GE and SR $\neq$ 1.0, we still consider improving the attack if the NT - number of traces used for the attacking becomes smaller than the maximum set value. Hence, depending on the dataset, we establish upper bounds, shown by Table 2.

In Figure 2, we can see how the `MLP Basic` architecture performs the attacks on all the original datasets. Since it is an MLP architecture, we can observe that it performs better on `ASCAD` datasets that do not use countermeasures. Moreover, for the datasets that use the clock jitters, the GE can not converge to 1.0, and its values fluctuate between 100-140 depending on the case. However, for the `CHESCTF` dataset, we

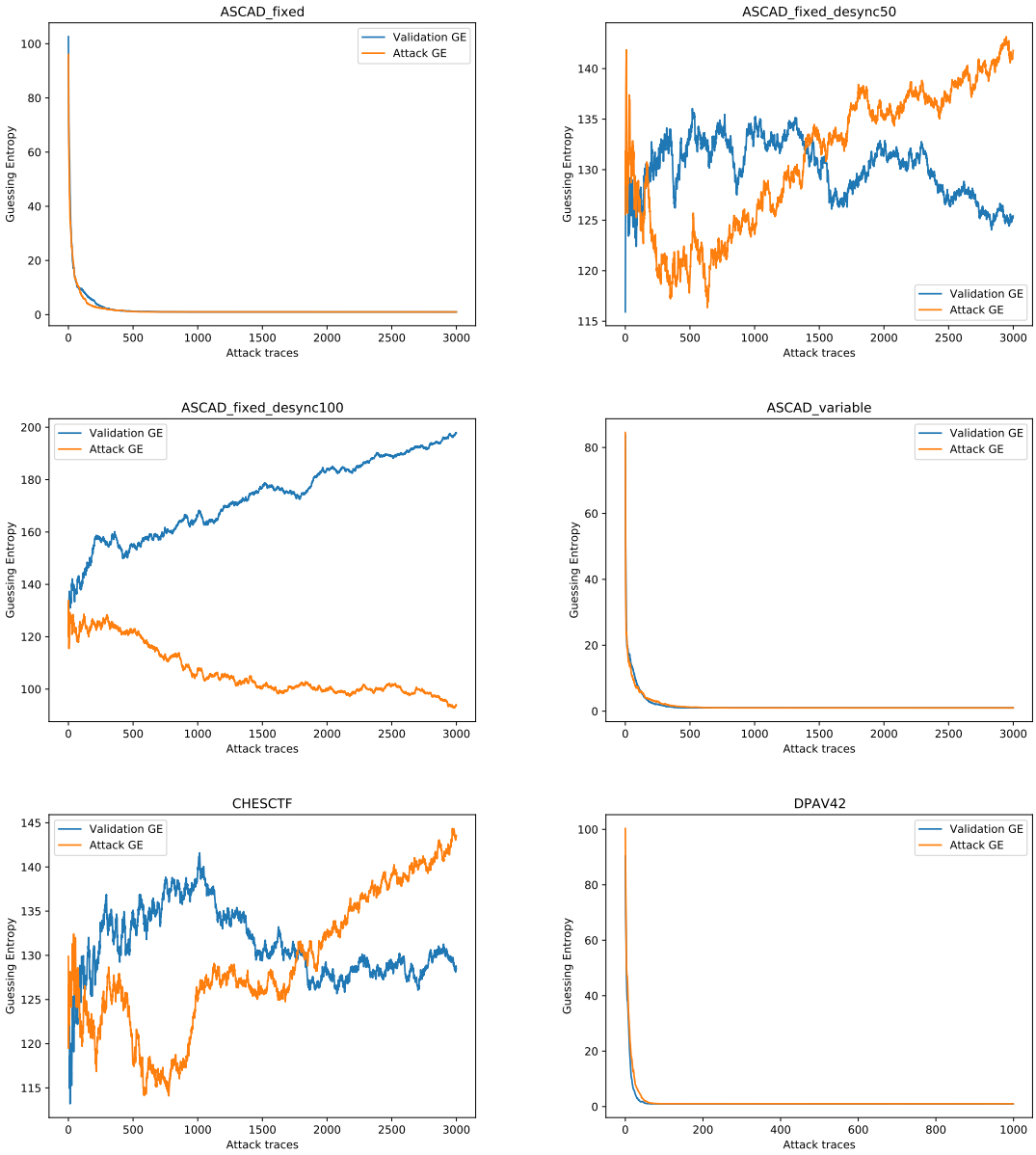

Figure 2: GE baseline results for `MLP Basic`

think that GE is not stable because the number of epochs used for training this model was insufficient to learn all 4000 features and therefore break the dataset. The exact opposite behavior happens to the `DPAV42` dataset, which uses just 800 features. Therefore the learning process is good enough for this model to perform a successful attack.

The `MLP Best` DL-SCA model was built to attack the `ASCAD` datasets, and it performs well only on the traces that do not use countermeasures. This claim can also be observed from Figure 3 - this type of architecture can perform very well on the datasets for which it is constructed; however, for the others, the GE could not reach 1.0, and its value continuously fluctuates.

In Figure 4, we can see that `CNN Best` is continuing the same pattern of having the GE = 1.0 for the `ASCAD_fixed` and `ASCAD_variable` datasets. Additionally, a slight convergence trend for the datasets using the counter measurements seems to be a slight convergence trend, especially for the `ASCAD_fixed_desync100`. However, it is still not enough to declare their successful attacks. On the other hand, the `DPAV42` and `CHESCTF`, the model has trouble processing good attacks with a GE value close to 1.0. We think this behavior happens since the CNN architecture was specifically created for breaking the `ASCAD` traces and was not tested on the other datasets.

The reason behind creating the `No Conv` model presented in the study by Wouter et al. [10] was to obtain similar performance during the attacking phase as the CNNs but without using a first convolutional layer. In this manner, the time for the learning process is minimized. As we can see in Figure 5, this model behaves similarly to an MLP, obtaining GE = 1.0 for the `ASCAD_fixed` and `DPAV42` datasets. Nevertheless, for the desynchronized traces, there is an improvement compared with the `CNN Best` plots - having the GE values for `No Conv` are stable, and the attacking ones are smaller than the validation ones. This aspect also proves that the DL-SCA model does not make any overfitting. In the case of the `ASCAD_variable` dataset, a trend for GE value to be steady near 40 exists because these traces contain 1400 features. Hence, we conclude that for this type of architecture,

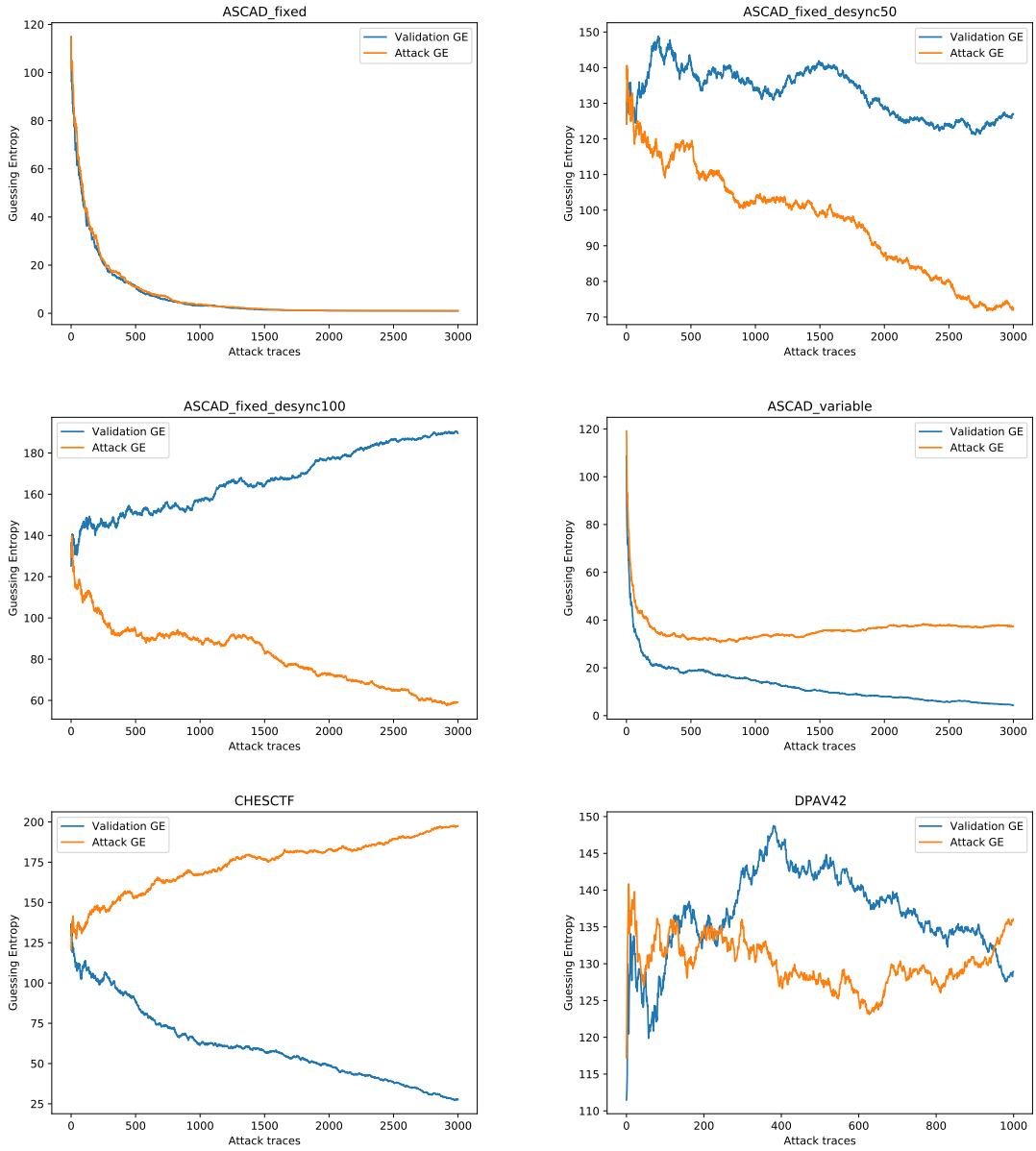

Figure 3: GE baseline results for `MLP Best`

we would need more data for launching successful attacks. In addition, the values for `CHESCTF` are more meaningful in this regard, since it has 4000 features, which are more than other datasets' features.

3.4 Results on universal dataset formats

After running all the required baseline experiments, as is shown in Section 3.2, we run the grid-search process on the AE and sort their performance based on the MSE metric.

After having these results, we take the top-5 best AE configurations for each dataset and use only the **Encoder** part for doing the compression operation. In practice, for each utilized Encoder configuration, we generate a compressed version of the dataset, such that its normalized version is fetched into DL-SCA models afterward. We organize the extensive results from all 4 DL-SCA models for evaluation in Tables 3, 4, 5 and 6.

In Table 3, we can see that for each dataset, there is at least one encoded dataset version used to launch improved versions of the attack. We do not obtain a significant difference between the compressed and the baseline data for the dataset, which contains many features (e.g., `CHESCTF`). However, for the rest of the dataset, there is a significant gap between those two scores, and we think that the number of features contained in that dataset is affecting the performance of the AE model since it could not learn all the features in a very precise way. Even for desynchronized versions of the dataset, we obtain at least one model which performs better than the baseline in terms of GE. Additionally, the ranking of the AE based on their performance does not respect the same ranking in terms of the DL-SCA evaluation part. For instance, for the `ASCAD_fixed` dataset, the 5^{th} AE configuration can improve the attack in terms of the NT. Therefore, we can see a reason for this kind of behavior. We obtain close results of MSEs between the top-5 best AE configurations. Since the learning technique of an ML is a non-deterministic operation, we do not obtain different MSE values even for the same input. Therefore, we consider that we can obtain different encodings for the initial dataset. After this step, the generated smaller dataset is fetched into another ML algo-

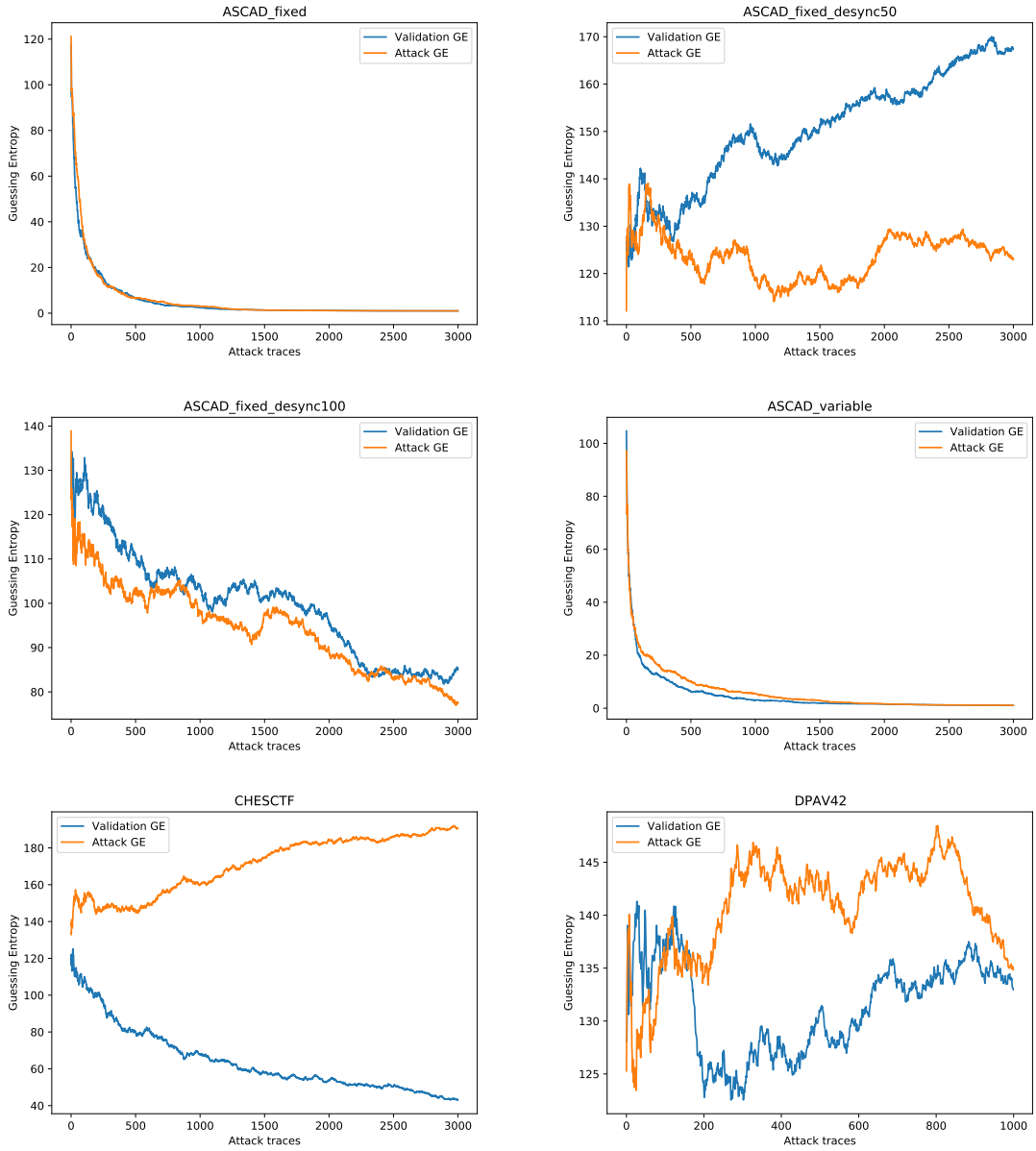

Figure 4: GE baseline results for `CNN Best`

rithm, and therefore the initial input affects the results of the DL-SCA model's performance.

In Table 4, there is a significant improvement, in terms of launching successful attacks (GE = 1, SR = 1, and the NT attacking of the baseline is greater than the one gained from the encoded version of the traces) is obtained for the `ASCAD_fixed` and `ASCAD_variable` datasets. This performance was expected since this MLP architecture was specifically created to attack these datasets [2]. However, we can not neglect the progress of enhancing the GE values for the rest of the datasets.

Similar to `MLP_Best`, the reason for building `CNN_Best` was to launch much better attacks on all ASCAD datasets [2]. However, we can see that in Table 5, we do not obtain a very significant improvement for most of the datasets. For example, for the `ASCAD_fixed` dataset, none of the AE models can help in launching the attacks with GE = 1.0 and SR = 1.0. For the `CHESCTF` dataset, most of the values from the baseline are close to the values obtained using the encoded version of the traces. We think this chaotic behavior of the CNN model happens because of the higher amount of traces needed for a very successful learning process.

In Table 6, we can see the results which we obtain using the `No_Conv` model. The results are promising and offer a significant improvement in the attacking phase. A small exception is represented by the `ASCAD_fixed` traces, where the amount of the NT was better just for the 5^{th} AE configuration. However, since the numbers are stable, the setup of experiments requires a single run, and the learning process is a non-deterministic technique, we can still consider an improvement for this dataset.

3.5 Discussion

As a first remark, if we compare the baseline results with those obtained using the compressed version of the dataset, we can see some promising results, which denote the fact that we improved the attack. For the `MLP_Basic`, `MLP_Best`, and `No_Conv`, for each dataset, we obtain at least an AE configuration such that it is capable of creating an encoded ver-

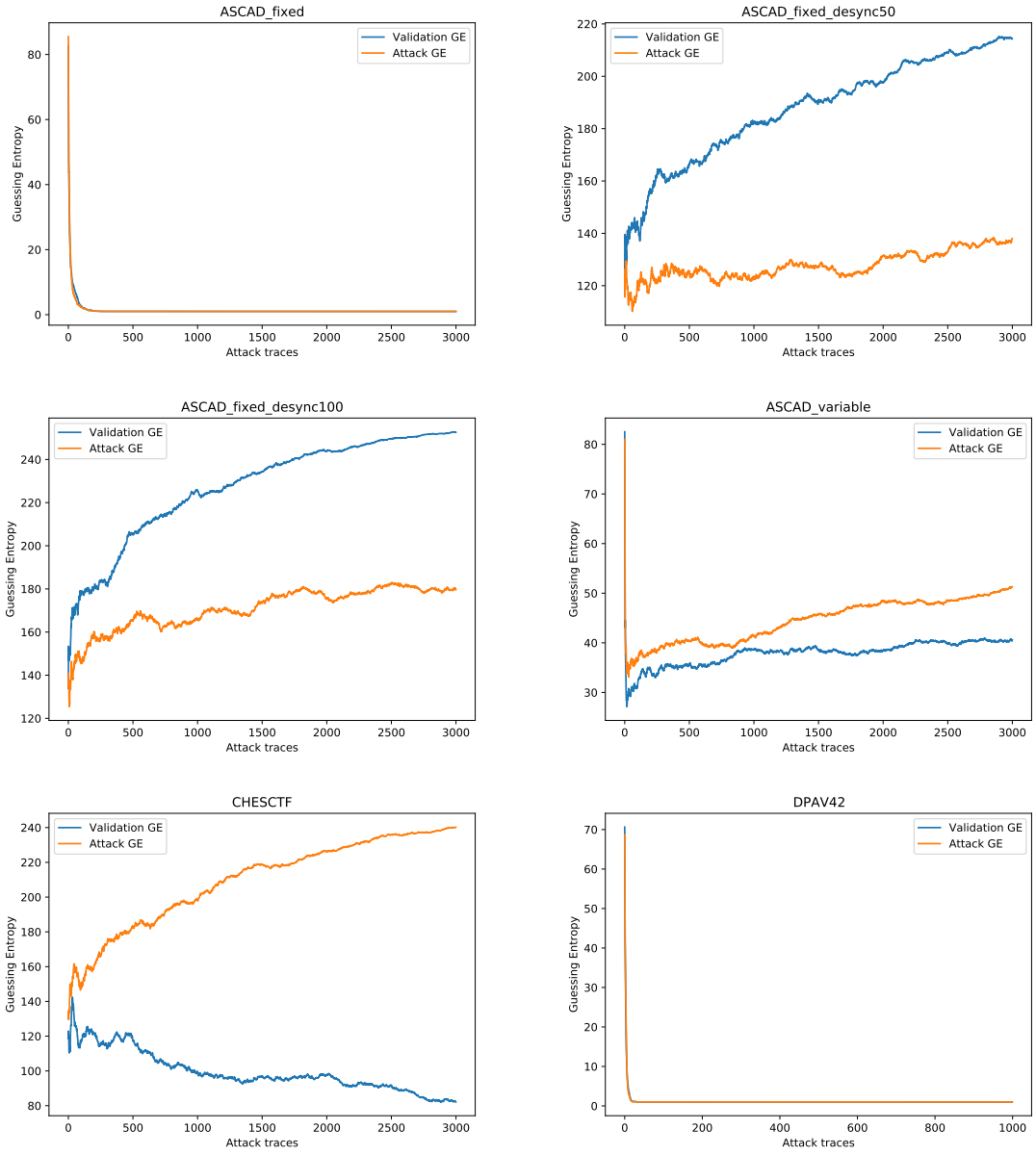

Figure 5: GE baseline results for `No Conv`

sion of the dataset for which the DL-SCA model improved the attack significantly. We can create improved versions of the attacks for the `CNN_Best` model, but the numbers were not stable; therefore, we consider them a small improvement and not a significant one. As a reason for that, we invoke the issue of using a non-deterministic process, being confident that if more experiments are conducted for this kind of architecture on both sides - the baseline and the DL-SCA part - we can see a considerable improvement at the level of the average scores.

Moreover, another reason we obtain these results can be based on the capability of the CNNs to analyze the smaller details within an instance so that it can do good classification. Indeed, this raised a problem that was tackled in the paper [6], where it is mentioned that SCA needs many data to improve the attacks. Hence, we are sure that for this kind of model, which consumes much data to gain the desired results, if we can create an environment where we can fetch more instances, we can also obtain promising results for this model.

Presenting these results, we address the question of why DL-SCA models which do not use a CNN architecture can behave well in the attacking environment.

4 Conclusions and Future Work

In conclusion, even if we want to present a new idea that can open more doors in the SCA domain and is a feasible technique for improving the attacks using DL-SCA models, there are still places for development.

As we already mentioned in Section 3.2, for this research, we use 500 features to describe one of the dimensions of the compression traces. Even though we obtain really promising results with this setup, we think another study focusing on other values for the bottleneck layer of the AE can add to our study's important conclusions. In addition, future research can focus on the least amount of features that can be used in the bottleneck layer of the AE, such that we can still obtain improved versions of the attacks. What will be the minimum value of the features which are needed for that? This topic becomes very important if we consider using as small a stor-

Table 3: Comparison with the baseline for the `MLP Basic`

Dataset	Baseline Score			Using AE Score						
				AE Configuration				DL-SCA Scores		
	GE	SR	NT attacking	epochs	learning rate	batch size	mse	GE	SR	NT attacking
`ASCAD_fixed`	1.0	1.0	295	**1000**	**0.0025**	**200**	**0.0588**	**1.0**	**1.0**	**271**
				700	0.001	200	0.0657	1.0	1.0	353
				700	**0.0025**	**400**	**0.0668**	**1.0**	**1.0**	**264**
				1000	0.001	200	0.0681	1.0	1.0	503
				1000	**0.0025**	**400**	**0.0684**	**1.0**	**1.0**	**227**
`ASCAD_fixed_desync50`	141.8	0.0	3000	**1000**	**0.001**	**200**	**0.0598**	**2.62**	**0.49**	**3000**
				1000	**0.0025**	**200**	**0.0619**	**1.27**	**0.83**	**2425**
				1000	**0.001**	**400**	**0.0703**	**5.47**	**0.17**	**3000**
				700	**0.001**	**200**	**0.0705**	**2.43**	**0.41**	**3000**
				700	**0.0025**	**200**	**0.0726**	**3.31**	**0.42**	**3000**
`ASCAD_fixed_desync100`	93.66	0.0	3000	1000	0.0025	200	0.0619	98.3	0.0	3000
				700	**0.0025**	**200**	**0.0601**	**92.94**	**0.0**	**3000**
				1000	0.001	200	0.0670	94.55	0.01	3000
				700	0.001	400	0.0706	179.01	0.0	3000
				1000	**0.001**	**400**	**0.0709**	**38.18**	**0.01**	**3000**
`ASCAD_variable`	1.0	1.0	324	**1000**	**0.001**	**200**	**0.3664**	**1.0**	**1.0**	**57**
				700	**0.001**	**200**	**0.3706**	**1.0**	**1.0**	**51**
				1000	**0.001**	**400**	**0.3843**	**1.0**	**1.0**	**53**
				1000	**0.0025**	**200**	**0.3993**	**1.0**	**1.0**	**52**
				700	**0.001**	**400**	**0.3984**	**1.0**	**1.0**	**49**
`CHESCTF`	89.49	0.0	3000	700	0.001	200	0.0397	93.58	0.0	3000
				1000	0.0025	400	0.0463	120.52	0.0	3000
				500	0.001	200	0.0470	143.48	0.0	3000
				500	0.0025	200	0.0478	181.02	0.0	3000
				1000	**0.0025**	**200**	**0.0499**	**88.88**	**0.0**	**3000**
`DPAV42`	1.0	1.0	52	**700**	**0.0025**	**200**	**0.0326**	**1.0**	**1.0**	**6**
				1000	**0.0025**	**400**	**0.0343**	**1.0**	**1.0**	**5**
				500	**0.0025**	**200**	**0.0347**	**1.0**	**1.0**	**5**
				700	**0.001**	**200**	**0.0404**	**1.0**	**1.0**	**10**
				1000	**0.001**	**200**	**0.0421**	**1.0**	**1.0**	**5**

age capacity as possible.

From another perspective, we can also test other types of AE which can do the compression. As an example, we can go for Convolutional AE or Variational AE. However, it is important to mention that for these kinds of AEs, more experiments and data traces are required for obtaining significant-good performances against the baselines.

We can also think there may be a good reason for focusing on a single DL-SCA universal model that can attack all the datasets. However, during this research, as was mentioned in Section 3.2, we have used DL-SCA models, which different scientists propose. Therefore, based on our observations regarding using non-CNN's ML models for this type of attack, there is a gap for a paper where the main focus is to construct a DL-SCA universal model that can break different datasets.

Since this work uses six datasets described in Section 3.1, another curious aspect can be represented if the same results can be obtained for more cryptographic algorithm implementations that use or do not use countermeasures. Thus, further research can be conducted in this direction, and the new results can represent an extension of this study.

Table 4: Comparison with the baseline for the `MLP Best`

Dataset	Baseline Score			Using AE Score						
				AE Configuration				DL-SCA Scores		
	GE	SR	NT attacking	epochs	learning rate	batch size	mse	GE	SR	NT attacking
ASCAD_fixed	1.0	1.0	1437	**1000**	**0.0025**	**200**	**0.0588**	**1.0**	**1.0**	**298**
				700	**0.001**	**200**	**0.0657**	**1.0**	**1.0**	**298**
				700	**0.0025**	**400**	**0.0668**	**1.0**	**1.0**	**295**
				1000	0.001	200	0.0681	1.17	0.88	2042
				1000	**0.0025**	**400**	**0.0684**	**1.0**	**1.0**	**855**
ASCAD_fixed_desync50	72.37	0.0	3000	1000	0.001	200	0.0598	167.93	0.0	3000
				1000	**0.0025**	**200**	**0.0619**	**7.72**	**0.21**	**3000**
				1000	**0.001**	**400**	**0.0703**	**42.36**	**0.01**	**3000**
				700	0.001	200	0.0705	103.1	0.0	3000
				700	0.0025	200	0.0726	163.93	0.0	3000
ASCAD_fixed_desync100	59.10	0.0	3000	1000	0.0025	200	0.0619	98.3	0.0	3000
				700	0.0025	200	0.0601	92.94	0.0	3000
				1000	0.001	200	0.0670	94.55	0.01	3000
				700	0.001	400	0.0706	179.01	0.0	3000
				1000	**0.001**	**400**	**0.0709**	**38.18**	**0.01**	**3000**
ASCAD_variable	37.27	0.0	3000	**1000**	**0.001**	**200**	**0.3664**	**1.0**	**1.0**	**354**
				700	**0.001**	**200**	**0.3706**	**1.0**	**1.0**	**316**
				1000	**0.001**	**400**	**0.3843**	**1.0**	**1.0**	**143**
				1000	**0.0025**	**200**	**0.3993**	**1.0**	**1.0**	**151**
				700	**0.001**	**400**	**0.3984**	**1.0**	**1.0**	**100**
CHESCTF	197.48	0.0	3000	**700**	**0.001**	**200**	**0.0397**	**93.58**	**0.0**	**3000**
				1000	**0.0025**	**400**	**0.0463**	**120.52**	**0.0**	**3000**
				500	**0.001**	**200**	**0.0470**	**153.95**	**0.0**	**3000**
				500	**0.0025**	**200**	**0.0478**	**181.02**	**0.0**	**3000**
				1000	**0.0025**	**200**	**0.0499**	**88.88**	**0.0**	**3000**
DPAV42	136.09	0.0	1000	**700**	**0.0025**	**200**	**0.0326**	**61.41**	**0.02**	**1000**
				1000	0.0025	400	0.0343	143.79	0.01	1000
				500	0.0025	200	0.0347	168.22	0.0	1000
				700	**0.001**	**200**	**0.0404**	**135.07**	**0.01**	**1000**
				1000	0.001	200	0.0421	210.92	0.0	1000

Table 5: Comparison with the baseline for the `CNN Best`

Dataset	Baseline Score			Using AE Score						
				AE Configuration				DL-SCA Scores		
	GE	SR	NT attacking	epochs	learning rate	batch size	mse	GE	SR	NT attacking
`ASCAD_fixed`	1.0	1.0	1218	1000	0.0025	200	0.0588	4.81	0.3	3000
				700	0.001	200	0.0657	1.16	0.87	2071
				700	0.0025	400	0.0668	1.84	0.71	2939
				1000	0.001	200	0.0681	3.26	0.37	3000
				1000	0.0025	400	0.0684	14.46	0.04	3000
`ASCAD_fixed_desync50`	123.06	0.0	3000	**1000**	**0.001**	**200**	**0.0598**	**75.45**	**0.0**	**3000**
				1000	**0.0025**	**200**	**0.0619**	**21.56**	**0.04**	**3000**
				1000	0.001	400	0.0703	142.47	0.0	3000
				700	**0.001**	**200**	**0.0705**	**63.76**	**0.01**	**3000**
				700	0.0025	200	0.0726	151.8	0.0	3000
`ASCAD_fixed_desync100`	77.52	0.0	3000	1000	0.0025	200	0.0619	84.99	0.0	3000
				700	**0.0025**	**200**	**0.0601**	**30.71**	**0.0**	**3000**
				1000	**0.001**	**200**	**0.0670**	**40.36**	**0.0**	**3000**
				700	**0.001**	**400**	**0.0706**	**19.46**	**0.03**	**3000**
				1000	**0.001**	**400**	**0.0709**	**41.82**	**0.01**	**3000**
`ASCAD_variable`	1.09	0.93	1759	1000	0.001	200	0.3664	15.1	0.04	3000
				70	0.001	200	0.3706	4.55	0.15	3000
				1000	0.001	400	0.3843	2.8	0.41	3000
				1000	0.0025	200	0.3993	1.34	0.83	2310
				700	**0.001**	**400**	**0.3984**	**1.08**	**0.96**	**2046**
`CHESCTF`	190.67	0.0	3000	**700**	**0.001**	**200**	**0.0397**	**189.39**	**0.0**	**3000**
				1000	**0.0025**	**400**	**0.0463**	**102.55**	**0.0**	**3000**
				500	**0.001**	**200**	**0.0470**	**169.83**	**0.0**	**3000**
				500	**0.0025**	**200**	**0.0478**	**135.21**	**0.0**	**3000**
				1000	**0.0025**	**200**	**0.0499**	**101.26**	**0.0**	**3000**
`DPAV42`	134.85	0.0	1000	700	0.0025	200	0.0326	171.45	0.0	1000
				1000	**0.0025**	**400**	**0.0343**	**95.15**	**0.01**	**1000**
				500	**0.0025**	**200**	**0.0347**	**43.48**	**0.04**	**1000**
				700	**0.001**	**200**	**0.0404**	**47.16**	**0.03**	**1000**
				1000	**0.001**	**200**	**0.0421**	**42.36**	**0.01**	**1000**

Table 6: Comparison with the baseline for the `No Conv`

Dataset	Baseline Score			Using AE Score						
				AE Configuration				DL-SCA Scores		
	GE	SR	NT attacking	epochs	learning rate	batch size	mse	GE	SR	NT attacking
`ASCAD_fixed`	1.0	1.0	116	1000	0.0025	200	0.0588	1.0	1.0	164
				700	0.001	200	0.0657	1.0	1.0	128
				700	**0.0025**	**400**	**0.0668**	**1.0**	**1.0**	**114**
				1000	0.001	200	0.0681	1.0	1.0	167
				1000	0.0025	400	0.0684	1.0	1.0	122
`ASCAD_fixed_desync50`	138.05	0.0	3000	**1000**	**0.001**	**200**	**0.0598**	**1.0**	**1.0**	**289**
				1000	**0.0025**	**200**	**0.0619**	**39.42**	**0.0**	**3000**
				1000	**0.001**	**400**	**0.0703**	**1.0**	**1.0**	**272**
				700	**0.001**	**200**	**0.0705**	**1.0**	**1.0**	**932**
				700	**0.0025**	**200**	**0.0726**	**53.07**	**0.0**	**3000**
`ASCAD_fixed_desync100`	179.59	0.0	3000	**1000**	**0.0025**	**200**	**0.0619**	**20.29**	**0.0**	**3000**
				700	**0.0025**	**200**	**0.0601**	**27.36**	**0.0**	**3000**
				1000	**0.001**	**200**	**0.0670**	**74.47**	**0.0**	**3000**
				700	**0.001**	**400**	**0.0706**	**70.2**	**0.0**	**3000**
				1000	**0.001**	**400**	**0.0709**	**133.37**	**0.0**	**3000**
`ASCAD_variable`	51.31	0.0	3000	**1000**	**0.001**	**200**	**0.3664**	**1.0**	**1.0**	**139**
				700	**0.001**	**200**	**0.3706**	**1.0**	**1.0**	**146**
				1000	**0.001**	**400**	**0.3843**	**1.0**	**1.0**	**177**
				1000	**0.0025**	**200**	**0.3993**	**1.0**	**1.0**	**155**
				700	**0.001**	**400**	**0.3984**	**1.0**	**1.0**	**193**
`CHESCTF`	240.26	0.0	3000	**700**	**0.001**	**200**	**0.0397**	**153.46**	**0.0**	**3000**
				1000	**0.0025**	**400**	**0.0463**	**234.02**	**0.0**	**3000**
				500	**0.001**	**200**	**0.0470**	**1.0**	**1.0**	**1007**
				500	**0.0025**	**200**	**0.0478**	**1.1**	**0.94**	**2055**
				1000	**0.0025**	**200**	**0.0499**	**54.84**	**0.01**	**3000**
`DPAV42`	1.0	1.0	14	**700**	**0.0025**	**200**	**0.0326**	**1.0**	**1.0**	**8**
				1000	**0.0025**	**400**	**0.0343**	**1.0**	**1.0**	**8**
				500	0.0025	200	0.0347	1.0	1.0	15
				700	0.001	200	0.0404	1.0	1.0	15
				1000	0.001	200	0.0421	1.0	1.0	16

References

[1] Pinar Akyazi and Touradj Ebrahimi. Learning-based image compression using convolutional autoencoder and wavelet decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, number CONF, 2019.

[2] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[3] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[4] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3146–3154, 2019.

[5] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[6] Naila Mukhtar, Lejla Batina, Stjepan Picek, and Yinan Kong. Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices. In *Cryptographers' Track at the RSA Conference*, pages 297–321. Springer, 2022.

[7] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *Cryptology ePrint Archive*, 2021.

[8] Yash Raut, Tasmai Tiwari, Pooja Pande, and Prachi Thakar. Image compression using convolutional autoencoder. In *ICDSMLA 2019*, pages 221–230. Springer, 2020.

[9] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[10] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 147–168, 2020.