SN Computer Science (2021) 2:373
https://doi.org/10.1007/s42979-021-00755-w

ORIGINAL RESEARCH

Tandem Deep Learning Side-Channel Attack on FPGA Implementation of AES

Huanyu Wang[1] · Elena Dubrova[1]

Received: 20 April 2021 / Accepted: 17 June 2021 / Published online: 9 July 2021

Abstract
Side-channel attacks have become a realistic threat to implementations of cryptographic algorithms, especially with the help of deep-learning techniques. The majority of recently demonstrated deep-learning side-channel attacks use a single neural network classifier to extract the secret from implementations of cryptographic algorithms. The potential benefits of combining multiple classifiers using the ensemble learning method have not been fully explored in the side-channel attack's context. In this paper, we propose a tandem approach for the attack in which multiple models are trained on different attack points but are used in parallel to recover the key. Such an approach allows us to considerably reduce (33.5% on average) the number of traces required to recover the key from an FPGA implementation of AES by power analysis. We also show that not all combinations of classifiers improve the attack efficiency.

Keywords Side-channel attack · Deep learning · Tandem model · FPGA · AES

Introduction

Deep-learning Side-Channel Attacks (DL-SCAs) utilize deep-learning models to bypass the theoretical strength of cryptographic algorithms. Many attacks on software implementations of Advanced Encryption Standard (AES) have been demonstrated recently. In [1–3], the effect of changing hyper parameters of deep-learning models for side-channel attacks are investigated. Afterwards, Wu et al. [4] and Rijsdijk et al. [5] provide two different approaches for tuning neural networks' hyperparameters automatically. In [6], a monobit-model technique to improve the attack efficiency is presented. To explore how board diversity affects the attack accuracy of the trained deep-learning models, Wang et al. [7] show to extend which a model trained for one device can lead to successful attacks on another device. To mitigate the effect caused by the board diversity, References [8–10] propose a cross-device approach, which trains models on traces captured from multiple devices. Besides, in [11], the newly proposed federated learning framework [12] is applied to improve the attack efficiency to break an 8-bit ATxmega128D4 microcontroller implementation of AES-128.

However, software implementations of AES are relatively easy to break using side-channel analysis because instructions are computed sequentially [13]. In hardware implementations, computations are performed in parallel. Therefore, DL-SCAs for hardware implementations is inherently more difficult, especially in advanced technologies. Power traces of two well-known public datasets, DPA contest V2 [14] and AES_HD [15], are captured from Xilinx Virtex-5 FPGA series. Many attacks are demonstrated based on these two datasets. In [15], Random Forest (RF) technique requires more than 5000 traces to recover a subkey. Masure et al. [16] investigate the theoretical soundness of Convolutional Neural Networks (CNNs) in the context of side-channel, References [17–19] demonstrated successful attacks on Virtex-5 FPGAs using CNNs. On a lightweight implementation of AES on Artix-7 FPGA [20], a non-profiled attack is able to recover the key with 3700 traces. Apart from FPGA, [21] shows the effectiveness of CNN-based side-channel attacks on ASICs. Table 1 shows a summary of previous attacks on hardware implementations of AES. To the best of authors' knowledge, previous works did not consider the

This article is part of the topical collection "Hardware for AI, Machine Learning and Emerging Electronic Systems" guest edited by Himanshu Thapliyal, Saraju Mohanty and VS Kanchana Bhaaskaran.

✉ Huanyu Wang
huanyu@kth.se

[1] KTH Royal Institute of Technology, Stockholm, Sweden

Table 1 Summary of DL-SCAs on hardware implementations of AES

SCAs	Hardware	Process technology	Number of classifiers	Number of traces required to recover the key
[21]	ASIC	180 nm	Single	≈ 1300
[15]	Virtex-5 FPGA	65 nm		> 2500
[17]				≈ 25000
[18]				≈ 200
[19]				> 2000
[20]	Artix-7 FPGA	28 nm		≈ 3700
This work				251
This work			Multiple	167

potential of combining multiple deep-learning classifiers in DL-SCAs on hardware implementations. When traces are particularly noisy, an ensemble of multiple models is capable of outperforming the single classifier. Also, it is necessary to test models on devices manufactured using advanced technologies.

To address these limitations and to further improve the attack efficiency, we propose a tandem deep-learning side-channel attack. It is inspired by a machine learning meta-algorithm called Adaptive Boosting (AdaBoost) [22], which is a subset of ensemble learning [23]. In AdaBoost, different classifiers (weak classifiers) are trained on the same training set. These weak classifiers are combined to form a boosted classifier (strong classifier). In our approach, several different, separately trained deep-learning models are used in an ensemble and we multiply models' outputs to reduce the generalization error. Since different models usually do not make the same errors on the test set, an ensemble of multiple models is expected to perform better than its members [24]. To reduce the generalization error, we train different classifiers (weak classifiers) on different training sets, which are labeled by different attack points. These weak classifiers are combined to form a boosted classifier (tandem model), which is able to achieve a more efficient attack.

In this paper, we show that while our best single CNN classifier requires 251 traces on average for a successful attack, the number for the tandem model is 167, which is a 33.5% reduction. In summary, our main contributions are as follows:[1]

1. Designing tandem model attacks which uses 33.5% fewer traces on average than single-model DL-SCAs to recover the key. To avoid the overestimation of classification accuracy, we train and test the tandem model on traces captured from different devices.
2. Demonstrating successful attacks against AES-128 implemented on Xilinx Artix-7 FPGAs. The Artix-7 FPGAs are manufactured using 28 nm process technology, which makes the attack particularly difficult.
3. Investigating how different combinations of classifiers can affect the tandem model. We show that utilizing different attack points makes the tandem model more efficient.

Paper organization The rest of the paper is organized as follows. The next section provides background information on software and hardware implementations of AES, and reviews how deep-learning-based side-channel attacks work. The third section reviews existing deep-learning side-channel attacks on two well-known publicly available datasets for hardware implementations of AES. The fourth section shows the equipment used in the experiments. The fifth section explains how different deep-learning models are used in a tandem. The sixth section presents the experimental results. The last section concludes this paper.

Background

In this section, we start by reviewing AES-128 and comparing hardware and software implementations of AES. Afterwards, we review deep-learning techniques and CNN.

AES-128

AES-128 [27] is a symmetric encryption algorithm, which takes a 128-bit block of plaintext and a 128-bit key as inputs. Figure 1 shows the flow of the AES-128 algorithm and three attack points used in our experiments. AES-128 contains 10 rounds in total. Except for the last round, each round has 4 steps: SubBytes, ShiftRows, MixColumns and AddRoundKey. The last round does not mix columns. The SubBytes procedure is a byte-to-byte substitution using a lookup table called *Substitution Box* (SBox).

[1] A preliminary version of this work has appeared in [25].

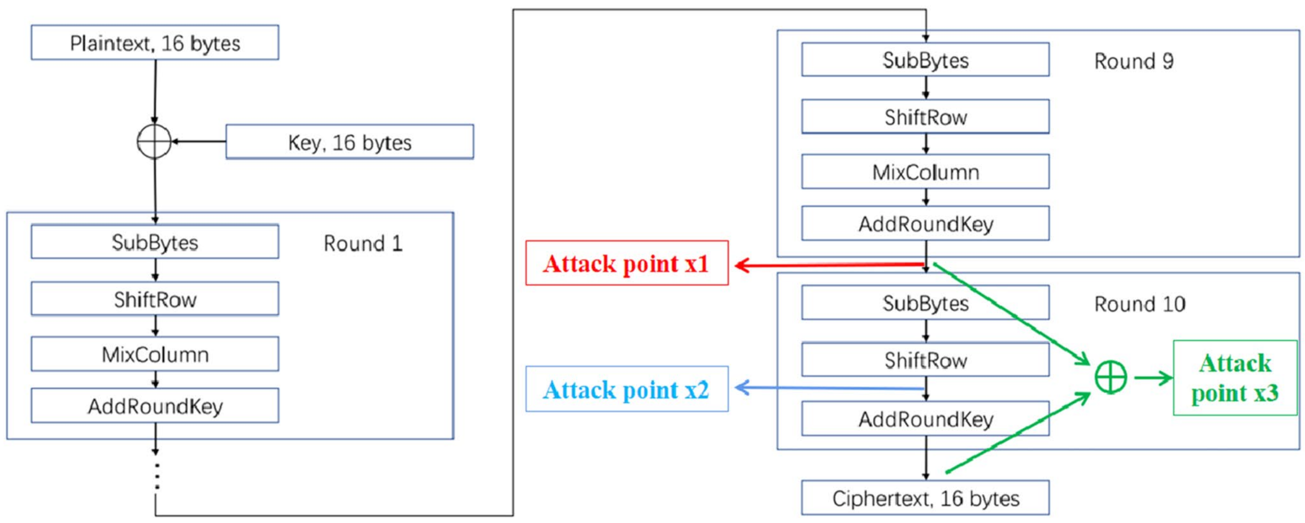


Fig. 1 AES-128 algorithm and three attack points we used to train CNN models [26]

As any block cipher, AES can be used in several modes of operation. In this paper we use *Electronic Codebook* (ECB) mode, in which the message is divided into blocks and each block is encrypted separately.

Hardware vs. Software Implementations of AES

In software implementations of AES, leakage is time-dependent and samples are less noisy since instructions are carried out one by one [18]. This makes deep-learning models easier to learn features from traces. On the other hand, hardware implementations of AES execute instructions in parallel. Therefore, traces captured from hardware implementations overlap features of all subkeys, which makes side-channel analysis inherently more difficult, especially in advanced technology. For example, Fig. 2a, b shows power traces captured from an 8-bit ATxmega128D4 microcontroller and a Xilinx Artix-7 FPGA implementations of AES-128, respectively, during the execution of Sbox operations of the first round of AES. In the trace from the microcontroller, 16 SBox computations are executed sequentially. To recover each key byte, an attacker can build a specialized model on the specific part of the trace. However, FPGAs execute all 16 SBox computations in parallel and a model built for only one subkey needs to handle the overlap caused by other 15 SBox computations. This makes the single-model attack less efficient.

Deep-Learning Side-Channel Attacks

Deep learning [24] is a subset of machine learning which uses neural networks to explore different levels of representative features of data for classification or prediction. Deep-learning models start with simple features and by the layer-by-layer combination continuously explore more complex features. Given the training data and a certain set of parameters, deep-learning models are able to demonstrate some particular tasks such as classification [28].

The aim of DL-SCAs is to use deep-learning models to classify a set of power traces $\mathcal{T} = T_1, T_2, \dots T_m$ based on their labels to derive the secret key, where m is the number of traces and T_i denotes a single trace. The corresponding label of trace T_i is denoted as $l(T_i) \in L$, where $L = \{0, 1, \dots, 255\}$ is the set of intermediate data processed at the attack point. To recover the full 128-bit key K of AES-128, typically a divide-and-conquer strategy is applied in which the key K is divided into 8-bit parts $K_k \in \mathcal{K}$, called subkeys, and the subkeys are recovered independently, for $k \in 1, 2, ..., 16$, where $\mathcal{K} = \{0, 1, \dots, 255\}$.

In most cases, deep-learning side-channel attacks are usually composed of two stages: the profiling stage (Fig. 3a) and the attack stage (Fig. 3b).

At the profiling stage, the attacker first uses the profiling device to encrypt a large number of plaintexts using known keys and captures traces. The model is trained on the labeled traces to learn the correlation between traces and keys. A neural network can be viewed as a mapping $\mathcal{N} : \mathbb{R}^n \to \mathbb{I}^{|L|}$, which maps a trace $T_i \in \mathbb{R}^n$ into a *score* vector $S_i = \mathcal{N}(T) \in \mathbb{I}^{|L|}$ whose elements $s_{i,j}$ represent the probability that label $l(T_i)$ has the value $j \in \{0, 1, ..., 255\}$, where n is the number of data points in T_i.

At the attack stage, the attacker uses the victim device to encrypt a small number of plaintexts and records corresponding traces. Using the trained model to classify traces captured from the victim device, the attacker is able to obtain the corresponding intermediate data and hence derive the subkey. The process mapping from the label to the subkey can be described as a retrieve function $\mathcal{F} : \mathbb{I}^{|L|} \to \mathbb{K}^{|\mathcal{K}|}$.

Fig. 2 Power traces from ATxmega128D4 microcontroller and Artix-7 FPGA implementations of AES

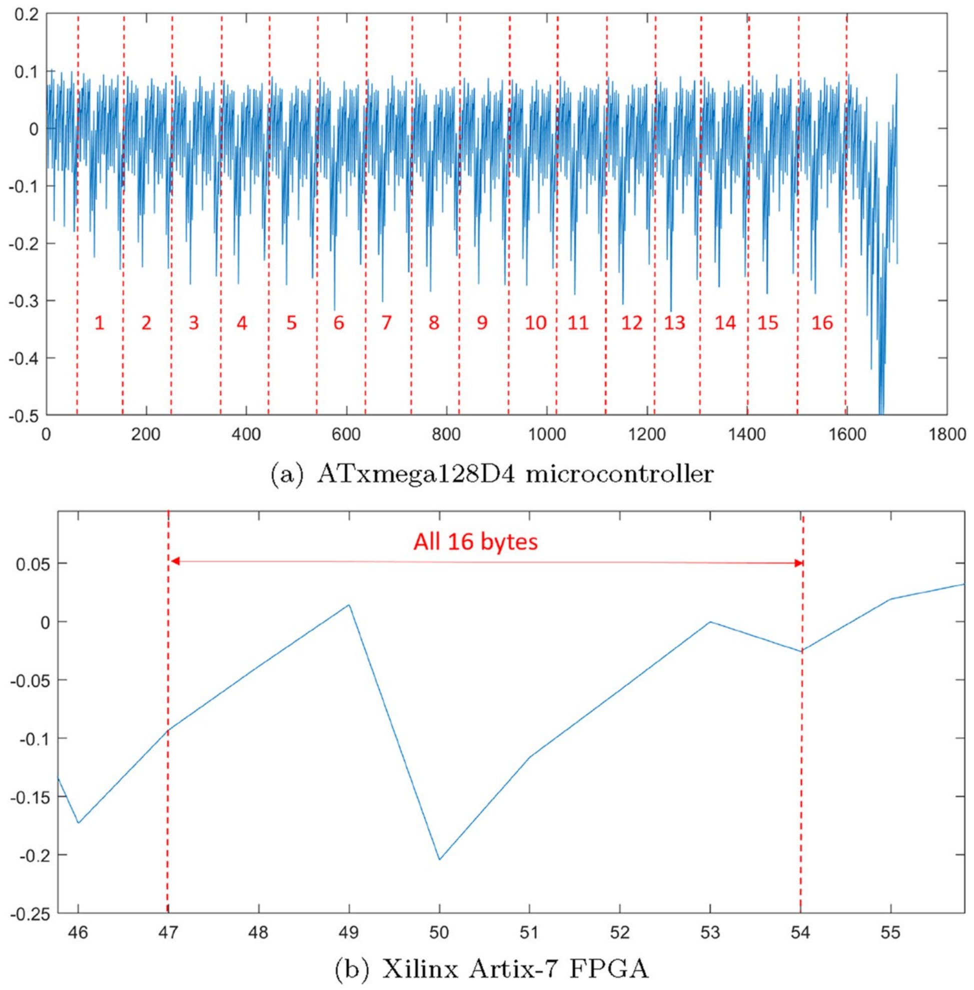
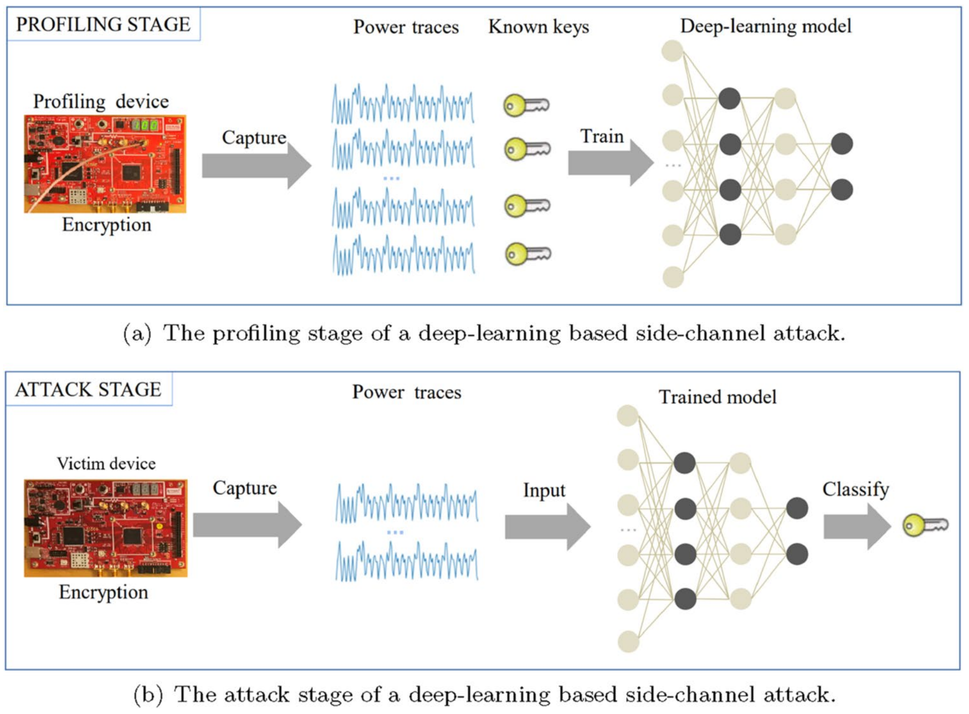


Fig. 3 An overview of how deep-learning-based side-channel attacks work



From the one-to-one mapping process $\mathcal{F}$, a guess vector P_i can be obtained from the score vector S_i. Each $P_i = \{p_{i,0}, p_{i,1}, ..., p_{i,255}\}$ contains the probability $p_{i,j} \in [0,\ 1]$ of each subkey candidate $K_k = j$, where $\sum_{j=0}^{255} p_{i,j} = 1$.

Let $\mathbf{P} = \prod_{i=1}^{m} P_i = \{\tilde{p}_0, \tilde{p}_1, ..., \tilde{p}_{255}\}$ denote the cumulative guess vector, which is an element-wise multiplication for all guess vectors generated by classifying m traces. The attacker can find the subkey $K_k = j$ which has the largest probability in $\mathbf{P}$:

$$K_k = \underset{0 \le j \le 255}{\arg\max}\ \tilde{p}_j.$$

We use K_k^* to denote the real subkey. Once $K_k = K_k^*$, the subkey is recovered successfully.

Since instructions of hardware implementations of AES are executed in parallel, traces captured from FPGAs are particularly noisy. In this scenario, CNN-based side-channel attacks seem to be powerful to handle noisy traces. CNN was originally introduced for image, speech, time series processing and document recognition [29]. The strength of a CNN network is that different network layers can learn features of the input data at different levels. A typical CNN network contains three types of layers: convolutional layers for filtering, pooling layers for down sampling and Fully-Connected (FC) layers for projection. CNNs have been successfully applied to bypass the trace misalignment and to overcome jitter-based countermeasures [30]. CNNs were also used to break protected AES [31–33].

Previous Work

This section reviews some existing deep-learning side-channel attacks on two well-known publicly available datasets for hardware implementations of AES, called DPA v2 and AES_HD.

The DPA Contest v2 [14] was organized by the VLSI research group from the COMELEC department of the Télécom ParisTech french University. The acquisitions have been performed on a SASEBO GII board [38] implementation of AES-128. The board features the Xilinx Virtex-5 [39] LX30/LX50 as the target FPGA for implementation evaluation. For the AES_HD dataset [15], it consists of EM measurements of an unprotected AES-128 implementation on Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board. The implementation is written in VHDL in a round based architecture that takes 11 clock cycles for each encryption. The dataset has 500K traces in total with 500K randomly generated plaintexts and each trace contains 1250 samples.

Deep-learning techniques were first used to assist power analysis in 2013 [40] when a three-layer MLP network was trained to break a Smart Card implementation of AES-128 which contains an 8-bit microcontroller PIC16F84 [41].

Apart from MLPs, Maghrebi et al. [18] investigate how other types of deep-learning models could make the side-channel attacks more efficient based on three different datasets. To break an Virtex-5 FPGA implementation of AES (DPA v2 dataset), the CNN and Autoencoder (AE)-based approaches in [18] require roughly 200 traces on average to recover the secret key. When it comes to the case of template attack, the result becomes about 400 traces.

To pursue the line of works on CNN-based power analysis, Cagli et al. [30] apply the CNN with a data augmentation technique [42] to bypass the trace misalignment and to overcome the jitter-based countermeasures. Cagli et al. [30] first point out that the conventional template attack strategy suffers from the difficulty to deal with the trace misalignment, which forces the attacker to have a critical realignment of the captured traces. Afterwards, they experimentally show that the CNN-based strategy greatly facilitates the attack roadmap since it waives the requirement for trace realignment and precise selection of points of interest.

To further improve the attack efficiency to break hardware implementation of AES, Jin et al. [36] introduce an attention mechanism, which is called Convolutional Block Attention Module (CBAM). Afterwards, they incorporate the proposed CBAM module into their CNN architecture, which helps the model to find the informative points of traces. In their experiments, the enhanced CNN model requires 2100 traces to recover a subkey from an Virtex-5 FPGA implementation of AES (AES_HD dataset).

To explain the role of each hyperparameters of neural networks during the feature selection phase in the side-channel attacks' context, Zaid et al. [37] use three visualization techniques to show the inner-workings of models, which are Weight Visualization [43], Gradient Visualization [44] and Heatmaps [45]. With the help of these visualization approaches, Zaid et al. [37] need 1050 traces to recover a subkey from an Virtex-5 FPGA implementation of AES (AES_HD dataset).

In [6], to train neural networks, each bit of the intermediate data processed at the attack point is used as one label. Thus, when considering a subkey which is a byte, there are 8 labels in total. This technique is presented to overcome the curse of class imbalance since each bit is nearly uniformly distributed compared to the Hamming weight (HW) leakage model. To break an Virtex-5 FPGA implementation of AES (AES_HD dataset), the multi-label model in [6] uses 831 traces to recover a subkey.

Tables 2 and 3 summarize some existing deep-learning side-channel attacks on Virtex-5 FPGA implementations of AES. In this work, we go one step further to focus on an Xilinx Artix-7 FPGA implementation of AES-128. Unlike Virtex-5 FPGAs which are manufactured using 65 nm process technology, Artix-7 FPGAs are manufactured using 28 nm process technology. Advanced manufacturing process

Table 2 Summary of side-channel attacks on DPA v2 dataset

Year	Work	Classifier	Result	Train and test on different devices
2016	[18]	CNN	≈ 200 traces	No
2018	[19]	Naive Bayes	≥ 2K traces	No
2019	[34]	Autoencoder + Template Attack	450 traces	No

Table 3 Summary of side-channel attacks on AES_HD dataset

Year	Work	Classifier	Result	Train and test on different devices
2018	[15]	Pooled Template	≈ 2.5K traces	No
2019	[35]	CNN	≈ 25K traces	No
2019	[6]	CNN	831 traces	No
2020	[36]	CNN	≈ 2.1K traces	No
2020	[37]	CNN	1050 traces	No

technique makes the attack particularly difficult. Besides, all existing works do not take the impact caused by the board diversity into consideration. They train deep-learning models on traces captured from the victim device, which requires an unlimited access to the target. Clearly, this condition is unlikely in a real attack scenario. In our experiments, we train and test deep-learning models on traces captured from different devices to mitigate the effect caused by the board diversity.

Experimental Setup

Figure 4 shows two Xilinx Artix-7 FPGAs manufactured using 28 nm *High-K Metal Gate* (HKMG) process technology. In the sequel, we call these two boards FPGA1 and FPGA2 respectively. They are programmed to the same version of AES-128 in Electronic Codebook (ECB) mode of operation. We use ChipWisperer Lite [46] with a 40 MHz sampling rate for trace capture. In our experiments, FPGA1 is used as the profiling board and FPGA2 is the victim board.

Tandem Deep-Learning Side-Channel Attack

Figure 5 shows an overview of the 3-classifier tandem model. The tandem model aims to combine the classification results of 3 local classifiers to reduce the generalization error. To provide enough diversity, local classifiers are trained on traces labeled by different attack points.

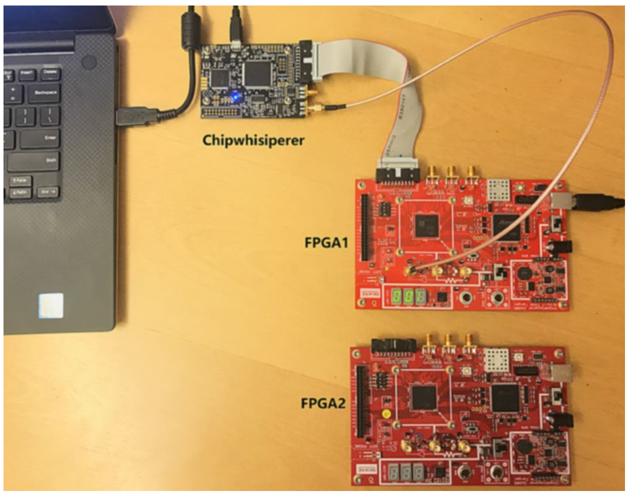


Fig. 4 Two Xilinx Artix-7 FPGAs and ChipWisperer

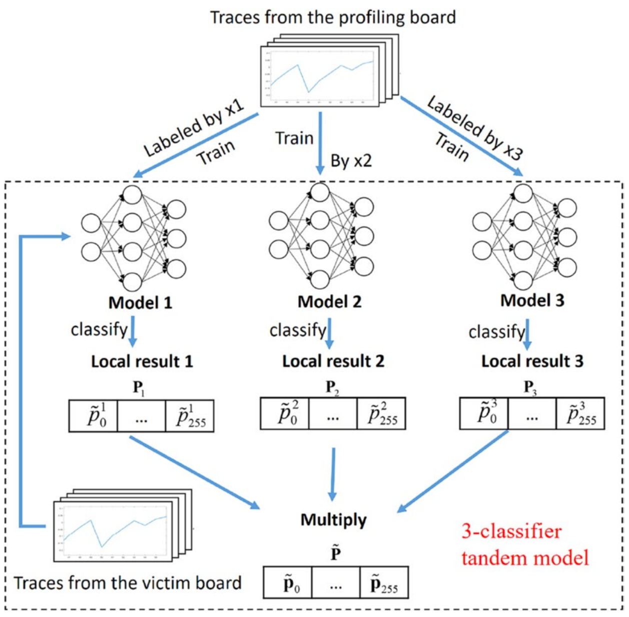


Fig. 5 The 3-Classifier Tandem Model

Attack Point

An attack point is a selected intermediate state which can be used to describe the power consumption. To form the proposed 3-classifier tandem model, three attack points are selected from the last round of AES-128 since the last round does not have the *Mixcolumn* procedure (see Fig. 1). It only has 3 operations: *SubBytes*, *ShiftRows* and *AddRoundKey*. The SubBytes procedure is a byte-to-byte substitution using a lookup table called *Substitution Box* (SBox). We denote these three attack points as x_1, x_2 and x_3:

Table 4 Architecture of CNN classifiers

Layer type	Output shape	Parameter #
Input Layer	(None, 10, 1)	0
Conv1D	(None, 10, 10)	110
Average Pooling	(None, 5, 10)	0
Flatten	(None, 50)	0
Dense 1	(None, 64)	3264
Dense 2	(None, 32)	2080
Dense 3	(None, 32)	1056
Output (Dense)	(None, 256)	8448
Total Parameters: 14,958		

$$\begin{aligned} x_1 &= \mathrm{SBox}^{-1}\left(sft\_row^{-1}(K_k \oplus C_k)\right) \\ x_2 &= K_k \oplus C_k \\ x_3 &= \mathrm{SBox}^{-1}\left(sft\_row^{-1}(K_k \oplus C_k)\right), \oplus C_k \end{aligned}$$

where C_k represents the k_{th} 8-bit ciphertext, $sft\_row^{-1}()$ and $\mathrm{SBox}^{-1}()$ denote the inverse of SubBytes and ShiftRows, respectively. Attack point x_1 is the input of the last round, x_2 is the output of the shift row operation, and x_3 is the XORed value between the input and output of the last round. Note that x_3 represents switching activity, which is known to be the dominant fraction of the total power consumed by a CMOS device.

Model Structure

CNNs have been successfully applied to bypass trace misalignment and to overcome jitter-based countermeasures [30]. Layer structures of our local CNN classifiers are shown in Table 4. We use the *identity* model as the power model, which assumes that the power consumption is proportional to the data processed at the attack point. Three CNN classifiers, referred as classifier 1, 2 and 3, are trained on traces labeled by attack point x_1, x_2 and x_3, respectively. We use *categorical cross-entropy loss* to quantify the classification error and use the *RMSprop* optimizer to tune internal parameters.

Tandem Deep-Learning Model

As shown in Fig. 5, three CNN classifiers are trained on same traces, but labeled by different attack points. To retrieve the subkey K_k from x_1, x_2 and x_3, we define three different retrieve functions $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$:

$$\begin{aligned} K_k = \mathcal{R}_1(x_1) = str\_row\left(\mathrm{SBox}(x_1)\right) \oplus C_k \\ K_k = \mathcal{R}_2(x_2) = x_2 \oplus C_k \\ K_k = \mathcal{R}_3(x_3) = str\_row\left(\mathrm{SBox}(x_3 \oplus C_k)\right) \oplus C_k. \end{aligned}$$

During the attack stage, 3 local classifiers are used to classify traces captured from the victim board individually and obtain their own cumulative guess vectors $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, which represent the classification results of 3 local classifiers. Afterwards, we multiply these classification results and obtain the final guess vector $\tilde{\mathbf{P}} = \mathbf{P}_1 \times \mathbf{P}_2 \times \mathbf{P}_3$ to form the tandem model.

Estimation Metrics

Rank The *rank* of a key K, $Rank(K)$, is the number of keys with a higher probability than K:

$$Rank(K) = \left|\{K' \in \mathcal{K} : \Pr[K|\mathcal{T}] < \Pr[K'|\mathcal{T}]\}\right|$$

Guessing Entropy The *Guessing Entropy* is the expected rank among all possible keys: $GE = \underset{K \in \mathcal{K}}{\mathbb{E}}(Rank(K))$. If subkeys are recovered individually, then the entropy is guessed for each subkey K_k separately and *Partial Guessing Entropy*, PGE, is used as the estimation metric [47].

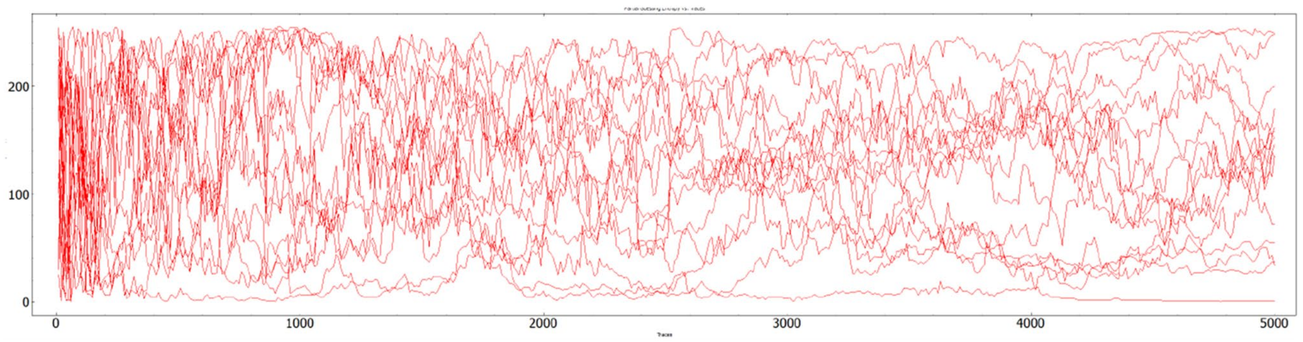
Experimental Results

In this section, we first evaluate how non-profiling attacks such as Correlation Power Analysis (CPA) [48] perform on traces captured from Artix-7 FPGA implementations of AES. Afterwards, we investigate the average number of traces required to recover the key using a single CNN classifier without the tandem approach. Next, we test to which extend the 2-classifier and 3-classifier tandem models can improve the attack efficiency. Afterwards, for completeness, we investigate how the result changes if tandem models are built by combining classifiers trained on the same attack point.

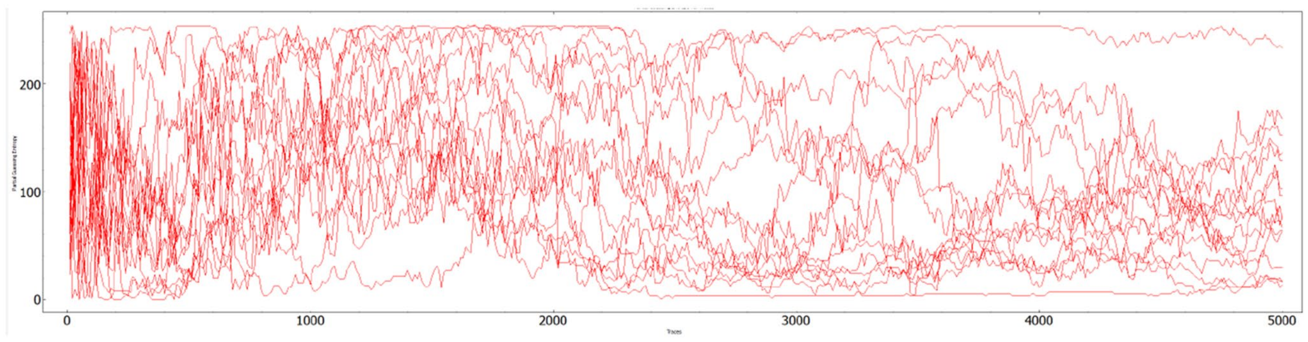
Correlation Power Analysis

To show the reader how non-profiling attacks such as CPA perform on an Artix-7 FPGA implementation of AES, in this section we present CPA results for 5K traces. We use three different attack points with the identify power model. Figure 6a–c shows the correlation results for all 16 subkeys for attack point x_1, x_2 and x_3, respectively.

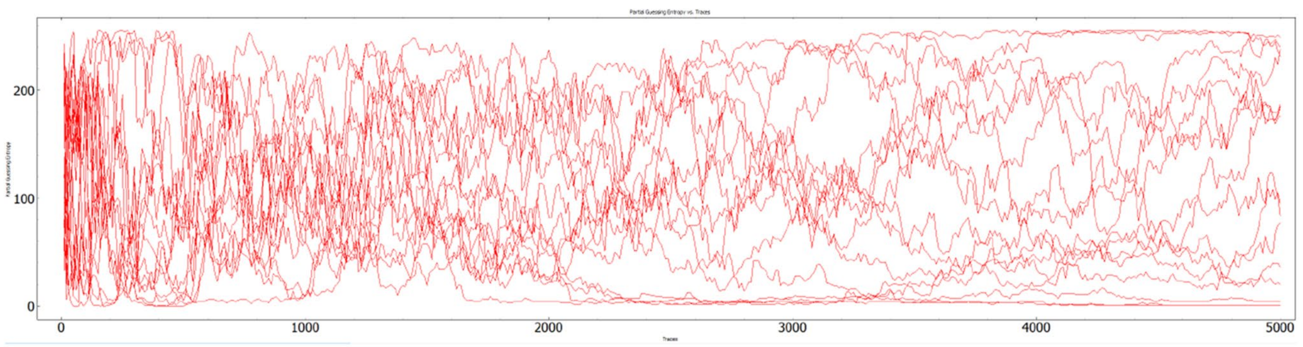
As we can see from the PGE plots in Fig. 6a, b, the CPA cannot recover any subkey for all selected attack points within 5K traces without the key enumeration. The attack point x_3 achieves the best CPA result, in which the minimum rank is 1 and the maximum is 249. Notice that once the rank achieves 0, the key is recovered.



(a) CPA result for all 16 subkeys for attack point x_1



(b) CPA result for all 16 subkeys for attack point x_2



(c) CPA result for all 16 subkeys for attack point x_3

Fig. 6 CPA results for 5K traces captured from FPGA1

Single-Classifier Model

In this section, our experiments are designed to show how many traces are required to recover the key using single-classifier models trained on traces labeled by different attack points. For each attack point, we train classifiers with the learning rate 0.0001, no learning rate decay, no dropout, and the batch size 256. The CNNs are trained using RMSprop optimizer. To select a best number of epochs, for each of the three classifiers we trained 10 models using e epochs, for $e \in \{10, 20, \dots, 100\}$. At each iteration, the model is stored instead of being overwritten. The resulting best numbers of epochs are shown in Table 5.

Classifier 1, 2 and 3 are trained on 1,000K traces captured from FPGA1 labeled by attack point x_1, x_2 and x_3, respectively, with 200K traces randomly set aside for validation. We have two different test sets of the same size, the first one contains 50K traces captured from FPGA1 and another one is from FPGA2. For a single test, 1K

Table 5 The best number of epochs for different classifiers

Model	Attack point	# epochs
Classifier 1	1	60
Classifier 2	2	80
Classifier 3	3	50

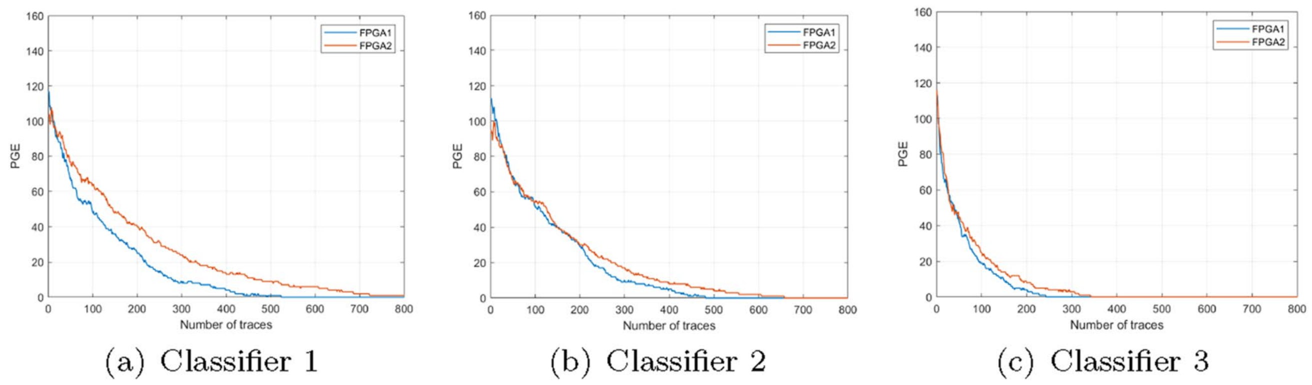


Fig. 7 Average PGE of CNN classifiers tested on traces captured from FPGA1 and FPGA2. To compute the average, 500 tests were performed for each classifier. For each test, 5000 traces were randomly selected from 50K traces

traces are randomly selected to calculate the PGE. For each classifier, we repeat 500 tests to compute the averaged key guess. Following [26], we use *min-max scaling* [49] to map the amplitude of all traces to the interval [0,1]. Given a set of traces $\mathcal{T}$, each trace $T = (\tau_1, \dots, \tau_m) \in \mathbb{R}^m$ of $\mathcal{T}$ is mapped into $T' = (\tau'_1, \dots, \tau'_m)[0,1]^m$ such that, for all $i \in \{1, \dots, m\}$,

$$\tau'_i = \frac{\tau_i - \tau_{\min}}{\tau_{\min} - \tau_{\max}},$$

where $\tau_{\min}$ and $\tau_{\max}$ are the minimum and the maximum data points in T.

Figure 7a–c shows the PGE of classifier 1, 2 and 3 tested on traces captured from FPGA1 and FPGA2 respectively. Classifier 1 is able to recover the key using 524 traces captured from FPGA1, and 815 traces from FPGA2 on average. For classifier 2, the result becomes to 533 and 672 traces. Classifier 3 is the best model which can recover the key using 251 traces captured from FPGA1, and 342 traces from FPGA2. These results are concluded in Table 6. Classifier 3 uses fewer traces to recover the key than other classifiers, which indicates that x_3 is more efficient than other attack points to break both FPGA1 and FPGA2. This is an expected result since classifier 3 uses the attack point defined by the Hamming distance between two states, while classifiers 1 and 2 use the attack points defined by the values of the states themselves (identity power model). It is known that, for hardware implementations, the Hamming distance is a better power model than the identify since the total power consumption is dominated by the dynamic power consumption which, in turn, is determined by the switching activity of logic gates [50]. A larger Hamming distance implies a higher switching activity.

Next, we combine classifiers into a tandem.

2-Classifier Tandem Model

The 2-classifier tandem model is built by combining 2 of 3 CNN classifiers. Figure 8 shows the PGE results and Table 6 shows the average number of traces used by 2-classifier tandem models to break both FPGA1 and FPGA2.

We notice that, except the tandem model built by combining classifiers 1 and 2, all other 2-classifier tandem models use fewer traces than the classifiers they include. Compared to our best single classifier (classifier 3), the tandem model with a combination of classifier 1 and 3 uses 30.7% fewer power traces to recover the key of FPGA1 and 21.1% for FPGA2. Also, the tandem model with a combination of classifier 2 and 3 uses 33.5% of fewer power traces to recover the key of FPGA1 and 11.4% for FPGA2. However, the tandem model which combines classifier 1 and 2 needs to use 10.9% more traces to break FPGA2 than classifier 1, which is the best local classifier of this tandem model. Our explanation is that attack point 1 and 2 do not provide enough diversity.

Table 6 Average number of traces to recover target subkey

Model	Attack point	Number of traces FPGA1	Number of traces FPGA2
Classifier 1	1	524	815
Classifier 2	2	533	672
Classifier 3	3	251	342
Classifier 1 and 2	1 and 2	501	745
Classifier 1 and 3	1 and 3	174	270
Classifier 2 and 3	2 and 3	167	303
Classifier 1 and 2 and 3	1 and 2 and 3	172	219

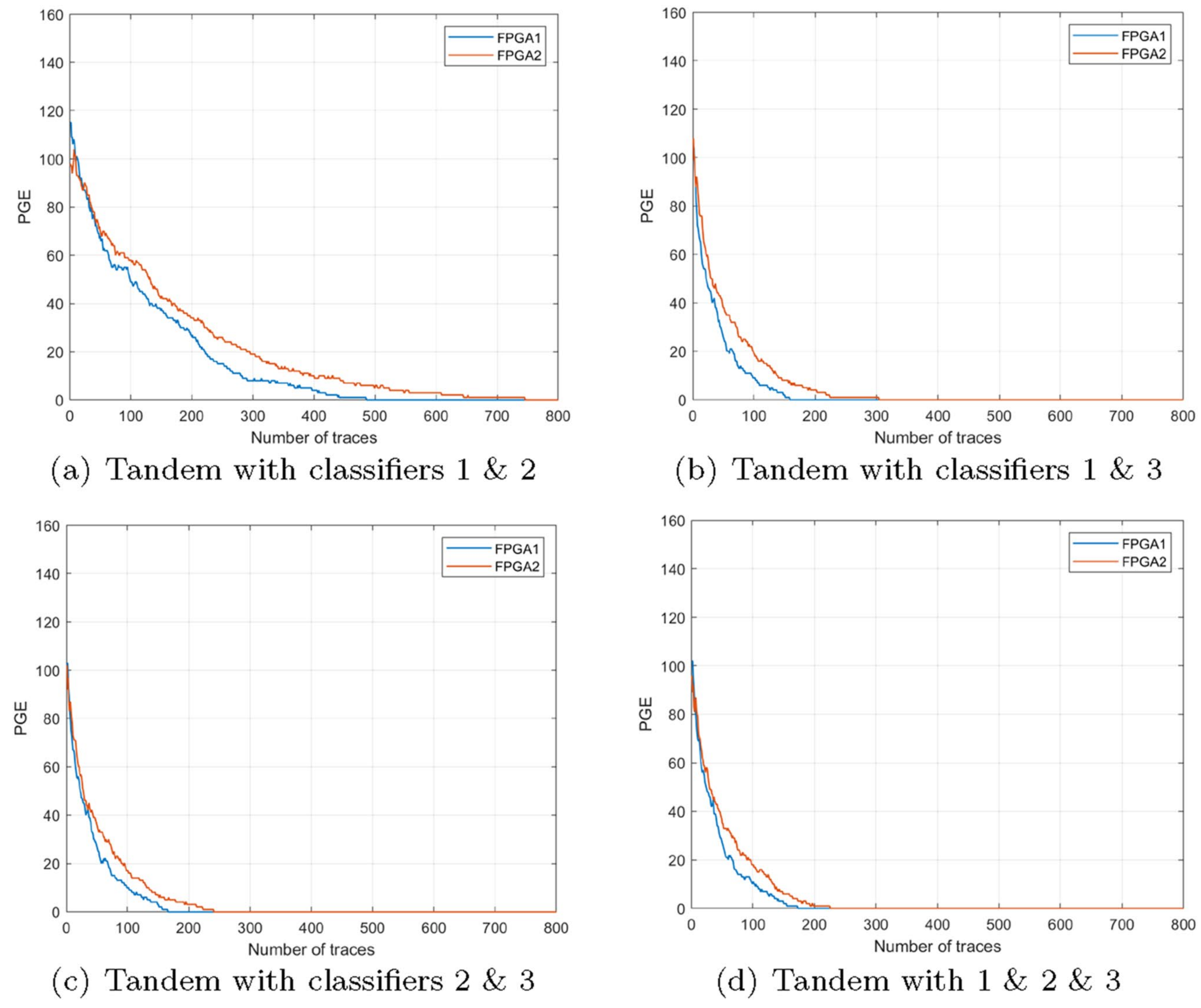


Fig. 8 Average PGE of three 2-classifier and 3-classifier tandem models tested on traces captured from FPGA1 and FPGA2

3-Classifier Tandem Model

Our 3-classifier tandem model is built by combining classifier 1, 2 and 3, which utilizes all available attack points. Figure 8(d) shows the PGE result and Table 6 shows the average number of traces. Compared to the results of 2-classifier models, adding one more classifier indeed improves the attack efficiency when the target device is different from the profiling device. The 3-classifier model uses 172 traces to recover the key of FPGA1 and 219 traces for FPGA2, on average. Compared to our best single classifier, it uses 31.5% fewer traces to break FPGA1 and 36.0% fewer traces for FPGA2.

Table 6 shows that a tandem with a larger number of classifiers seems to be more robust to manufacturing process variations since it needs the fewest number of traces for the attack on the FPGA2. We can also see that, for the FPGA1, all tandems with the classifier based on the attack point 3 achieve comparable results.

Besides, since it is easier to attack the device which was used for training, the experimental results for FPGA1 can be treated as the lower bound on the number of traces required for a successful attack. Attacking another device is more difficult due to manufacturing process variations.

Tandem Model with the Same Attack Point

To verify that it is important to use different attack points to train classifiers for building a tandem model, we further train 4 CNN classifiers on attack point 3 with different number of epochs. Then, we combine them into tandem. Table 7 shows the average number of traces required to recover the

Table 7 Average number of traces to recover the subkey by tandem of multiple classifiers trained on the same attack point

Number of classifiers	1	2	3	4	5
FPGA1	251	246	221	263	249
FPGA2	342	363	347	350	337

key using tandem models with different number of classifiers trained on the same attack point. From Table 7, we can conclude that building tandems from multiple classifiers trained on the same attack point is not a good strategy.

Conclusion

By combining multiple deep-learning classifiers trained on different attack points, the proposed tandem model is able to achieve a more efficient attack on an Artix-7 FPGA implementation of AES. Compared to the conventional single-classifier attack, the tandem model with multiple attack points can significantly improve the attack efficiency. We show that, one of our 2-classifier tandem models is able to use 33.5% fewer traces to break the profiling device (FPGA1). We also show that our 3-classifier tandem model is able to use 31.5% fewer traces to break the victim device (FPGA2). Finally, we show that it is important to use different attack points to build the tandem model.

Acknowledgements This work was supported in part by the research grant 2018-04482 from the Swedish Research Council and by the Vinnova Competence Center for Trustworthy Edge Computing Systems and Applications at KTH Royal Institute of Technology.

Funding Open access funding provided by Royal Institute of Technology.

Declarations

Conflict of Interest Statement On behalf of all the authors, the corresponding author states that there is no conflict of interest.

References

1. Benadjila R, Prouff E, Strullu R, Cagli E, Dumas C. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. In: ANSSI, France & CEA, LETI, MINATEC Campus. Online verfügbar unter, 2018 https://eprint.iacr.org/2018/053.pdf. Accessed 13 Apr 2021.
2. Maghrebi H. Deep learning based side channel attacks in practice. Tech. rep., IACR Cryptology ePrint Archive 2019;578: 2019.
3. Weissbart L. Performance analysis of multilayer perceptron in profiling side-channel analysis. In: International Conference on applied cryptography and network security, 2020; p. 198–216, Springer.
4. Wu L, Perin G, Picek S. I choose you: automated hyperparameter tuning for deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2020/1293, 2020.
5. Rijsdijk J, Wu L, Perin G, Picek S. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2021/071, 2021.
6. Zhang L, Xing X, Fan J, et al. Multilabel deep learning-based side-channel attack. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020;40(6):1207–1216.
7. Wang H, Brisfors M, Forsmark S, Dubrova E. How diversity affects deep-learning side-channel attacks. In: 2019 IEEE Nordic Circuits and Systems Conf. (NORCAS), 2019; p. 1–7.
8. Das D, Golder A, Danial J, Ghosh S, Raychowdhury A, Sen S. X-DeepSCA: cross-device deep learning side channel attack. In: Proc. of the 56th Annual Design Automation Conf. 2019, 2019; p. 134, ACM.
9. Wang H, Forsmark S, Brisfors M, et al. Multi-Source Training Deep-Learning Side-Channel Attacks. In: 2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL). IEEE, 2020:58–63.
10. Golder A, Das D, Danial J, et al. Practical approaches toward deep-learning-based cross-device power side-channel attack. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019;27(12): 2720–2733.
11. Wang H, Dubrova E. Federated learning in side-channel analysis. In International Conference on Information Security and Cryptology. Springer, Cham, 2020:257–272.
12. Yang Q, Liu Y, Cheng Y, Kang Y, Chen T, Yu H. Federated learning. Synth Lect Artif Intell Mach Learn. 2019;13(3):1–207.
13. Sklavos N, Touliou K, Efstathiou C. Exploiting cryptographic architectures over hardware vs. software implementations: advantages and trade-offs. In: Memory, 2006;13: 18.
14. TELECOM T. ParisTech SEN research group. DPA contest v2. 2010. http://www.dpacontest.org/v2/. Accessed 13 Apr 2021.
15. 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. 2018;2019(1):1–29.
16. Masure L, Dumas C, Prouff E A comprehensive study of deep learning for side-channel analysis. In: IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1):348–375.
17. Kim J, Picek S, Heuser A, Bhasin S, Hanjalic A. Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis. In: IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(3):148–179.
18. Maghrebi H, Portigliatti T, Prouff E. Breaking cryptographic implementations using deep learning techniques. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Cham, 2016:3–26.

19. Picek S, Samiotis IP, Kim J, Heuser A, Bhasin S, Legay A. On the performance of convolutional neural networks for side-channel analysis. In: Int. Conf. on security, privacy, and applied crypt. engineering, 2018; p. 157–176, Springer.
20. Ramezanpour K, Ampadu P, Diehl W. SCAUL: Power side-channel analysis with unsupervised learning. 2020. arXiv preprint arXiv:2001.05951.
21. Kubota T, Yoshida K, Shiozaki M, Fujino T. Deep learning side-channel attack against hardware implementations of AES. In: 2019 22nd Euromicro Conf. on digital system design, 2019; p. 261–268, IEEE.
22. Freund Y, Schapire R, Abe N. A short introduction to boosting. J-Jpn Soc Artif Intell. 1999;14(771–780):1612.
23. Opitz D, Maclin R. Popular ensemble methods: an empirical study. J Artif Intell Res. 1999;11:169–98.
24. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press, 2016. http://www.deeplearningbook.org. Accessed 13 April 2021.
25. Wang H, Dubrova E. Tandem deep learning side-channel attack against FPGA implementation of AES. In: 2020 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS). IEEE, 2020:147–150.
26. Wang R, Wang H, Dubrova E. Far field EM side-channel attack on AES using deep learning. In: Proceedings of the 4th ACM Workshop on attacks and solutions in hardware security, 2020; p. 35–44.
27. Daemen J, Rijmen V. The design of Rijndael: AES-the advanced encryption standard. Berlin: Springer Science & Business Media; 2013.
28. Wu Y, Shen K, Chen Z, Wu J. Automatic measurement of fetal cavum septum Pellucidum from ultrasound images using deep attention network. In: 2020 IEEE International Conference on image processing (ICIP), 2020; p. 2511–515.
29. LeCun Y, Bottou L, Bengio Y, Haffner P, et al. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324.
30. Cagli E, Dumas C, Prouff E. Convolutional neural networks with data augmentation against jitter-based countermeasures. In: International Conference on Cryptographic Hardware and Embedded Systems. Springer, Cham, 2017:45–68.
31. Perin G, Ege B, van Woudenberg J. Lowering the bar: deep learning for side channel analysis (white-paper). In: Proc. BlackHat, 2018; p. 1–15.
32. Gilmore R, Hanley N, O'Neill M. Neural network based attack on a masked implementation of AES. In: 2015 IEEE Int. Symp. on hardware oriented security and trust, 2015; p. 106–111, IEEE.
33. Martinasek Z, Dzurenda P, Malina L. Profiling power analysis attack based on MLP in DPA contest V4.2. In: 2016 39th Int. Conf. on telecom. and signal processing, 2016; p. 223–26, IEEE.
34. Yang G, Li H, Ming J, Zhou, Y. Cdae: towards empowering denoising in side-channel analysis. In: International Conference on information and communications security, 2019; p. 269–86, Springer.
35. 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; p. 148–79.
36. Jin M, Zheng M, Hu H, Yu, N. An enhanced convolutional neural network in side-channel attacks and its visualization. 2020. arXiv preprint arXiv:2009.08898.
37. Zaid G, Bossuet L, Habrard A, Venelli A. Methodology for efficient CNN architectures in profiling attacks. IACR Trans Cryptogr Hardw Embed Syst. 2020;2020(1):1–36.
38. Satoh A. Side-channel attack standard evaluation board, sasebo. Project of the AIST–RCIS (Research Center for Information Security), 2011; p. 135, http://www.rcis.aist.go.jp/special/SASEBO, Accessed 9 June 2021.
39. May DS, VF pgas. Virtex-5 FPGA data sheet: DC and switching characteristics. 152(2013):1–65.
40. Martinasek Z, Zeman V. Innovative method of the power analysis. Radioengineering. 2013;22(2):586–94.
41. Wilmshurst T. Designing embedded systems with PIC microcontrollers: principles and applications. Amsterdam: Elsevier; 2006.
42. Wong SC, Gatt A, Stamatescu V, McDonnell MD. Understanding data augmentation for classification: when to warp? In: 2016 International Conference on digital image computing: techniques and applications (DICTA), 2016; p. 1–6, IEEE.
43. Bischof H, Pinz A, Kropatsch WG. Visualization methods for neural networks. In: Proceedings 11th IAPR International Conference on pattern recognition. Vol. II. Conference B: pattern recognition methodology and systems, 1992; p. 581–585, IEEE.
44. Masure L, Dumas C, Prouff E. Gradient visualization for general characterization in profiling attacks. In: International Workshop on constructive side-channel analysis and secure design, 2019; p. 145–167, Springer.
45. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: European Conference on computer vision, 2014; p. 818–833, Springer.
46. O'Flynn C, Chen ZD. Chipwhisperer: an open-source platform for hardware embedded security research In: Int. Work. on Constr. side-channel analysis and secure design, 2014; p. 243–60, Springer.
47. Pahlevanzadeh H, Dofe J, Yu Q. Assessing CPA resistance of AES with different fault tolerance mechanisms In: 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), 2016; p. 661–66, IEEE.
48. Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model In: Int. Workshop on Cryptographic Hardware and Embedded Systems, 2004; p. 16–29, Springer.
49. Juszczak P, Tax D, Duin RP. Feature scaling in support vector data description In: Proc. asci, 2002; p. 95–102, Citeseer.
50. Yu Y, Marranghello F, Teijeira VD, Dubrova E. One-sided countermeasures for side-channel attacks can backfire In: Proceedings of the 11th ACM Conference on security & privacy in wireless and mobile networks, 2018; p. 299–301.